MySQL 4.0 Referenzhandbuch

MySQL 4.0 Referenzhandbuch

Dies ist eine Übersetzung des MySQL-Referenzhandbuchs, das sich auf dev.mysql.com befindet. Das ursprüngliche Referenzhandbuch ist auf Englisch, und diese Übersetzung ist nicht notwendigerweise so aktuell wie die englische Ausgabe. **Das vorliegende deutschsprachige Handbuch behandelt MySQL nur bis zur Version 4.0 (die aktuellste MySQL-Version ist 5.1). Es wird demnächst vom Netz genommen und durch eine aktuelle Version ersetzt. Geplanter Fertigstellungstermin: 30. Juni 2006**

Zusammenfassung

Document generated on: 2009-06-12 (Version: 551)

Inhaltsverzeichnis

Preface

Das ist das Handbuch für das MySQL-Datenbanksystem. Diese Version gehört zur MySQL-Version 5.0.6-beta. Sie finden ein Handbuch zu jeder älteren Version von MySQL in der Binär- oder Quelldistribution der entsprechenden Version.

Kapitel 1. Allgemeine Informationen über MySQL

MySQL ist ein sehr schneller und robuster, Multi-Thread und Multi-User SQL-Datenbank-Server (SQL = Structured Query Language, strukturierte Abfrage-Sprache). Die Einsatzgebiete des MySQL Server liegen in Hochleistungsapplikationen und in der Einbindung in weit verbreitete Massen-Software. `MySQL` ist eine Schutzmarke von `MySQL AB`.

Die MySQL Software steht unter einer `Doppellizenz`. Sie können sie entweder frei im Sinne der `GNU GENERAL PUBLIC LICENSE` (http://www.gnu.org/licenses/) verwenden, oder Sie erwerben eine komerzielle Lizenz, wenn Sie nicht durch die Restriktionen der GPL gebunden sein wollen. See Abschnitt 2.4.4, „MySQL-Lizenzpolitik“.

die MySQL Homepage enthält die letzten Informationen über MySQL.

Die folgende Liste beschreibt nützliche Teile des Handbuchs.

- Informationen zu dem Unternehmen hinter MySQL: Abschnitt 2.3, „Was ist MySQL AB?“.
- Eine Diskussion der Fähigkeiten von MySQL: Abschnitt 2.2.1, „Die wichtigsten Features von MySQL“.
- Installationsanweisungen: Kapitel 3, *Installation von MySQL*.
- Tipps zur Portierung von MySQL auf neue Architekturen oder Betriebssysteme: Anhang E, *Anmerkungen zur Portierung auf andere Systeme*.
- Informationen zum Upgrade von einem Release der Version 3.23: Abschnitt 3.5.1, „Upgrade von 3.23 auf Version 4.0“.
- Informationen zum Upgrade von einem Release der Version 3.22: Abschnitt 3.5.2, „Upgrade von einer Version 3.22 auf 3.23“.
- Einführungs-Tutorial zu MySQL: Kapitel 4, *Einführung in MySQL: Ein MySQL-Tutorial*.
- SQL-Beispiele und Informationen zu Benchmarks befinden sich im Benchmark-Verzeichnis (`sql-bench` in der Distribution).
- Die Geschichte neuer Features und Bugfixes: Anhang D, *MySQL-Änderungsverlauf (Change History)*.
- Eine Liste bekannter Bugs und Feature-Probleme: Abschnitt 2.7.5, „Bekannte Fehler und Design-Unzulänglichkeiten in MySQL“.
- Zukunftspläne: Abschnitt 2.8, „MySQL und die Zukunft (das TODO)“.
- Eine Liste aller Beteiligten, die zu diesem Projekt beitrugen: Anhang C, *Danksagungen*.

WICHTIG:

Berichte zu Fehlern (oft Bugs genannt) sowie Fragen und Bemerkungen sollten an die Mailingliste geschickt werden: `<mysql@lists.mysql.com>`. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“. Das `mysqlbug` Skript sollte benutzt werden, um Fehlerberichte zu erzeugen.

Bei Quelltext-Distributionen liegt das `mysqlbug` Skript im `scripts` Verzeichnis. Bei Binärdistributionen liegt `mysqlbug` im `bin` Verzeichnis. Wenn Sie einen empfindlichen Sicherheits-Bug in MySQL gefunden haben, sollten Sie eine E-Mail an `<security@mysql.com>` schicken.

Kapitel 2. Vorbemerkungen zum deutschen Handbuch

Die Übersetzung einer so umfangreichen technischen Dokumentation wie des MySQL-Referenzhandbuchs ist schon eine besondere Herausforderung, zumindest für jemanden, der seine Zielsprache ernst nimmt:

- In diesem Handbuch wird nicht geupdated, sondern aktualisiert.
- Eine MySQL-Distribution wird nicht gedownloaded, sondern herunter geladen.
- Und Transaktionen werden nicht gerollbackt, sondern zurückgerollt.

Womit wir auch schon bei der besonderen Herausforderung wären: Jeder, der sich mit Transaktionen auskennt, weiß, dass beim Fehlschlagen einer solchen ein Rollback-Befehl ausgeführt wird. Dieses Hauptwort ins Deutsche zu übersetzen, würde zum Verständnis wenig beitragen - im Gegenteil.

Damit bleiben alle technischen Fachbegriffe, die sich so und nicht anders etabliert haben, englisch:

- Ein SQL-Statement wird nicht als "Erklärung in der Strukturierten AbfrageSprache (SAS)" übersetzt.
- Abkürzungen wie TCP/IP werden nicht zu ÜSP/ZP (ÜbertragungsSteuerungsProtokoll/ZwischennetzProtokoll).
- Ein Client bleibt ein Client, und ein Server ein Server.

Die Fallstricke einer Übersetzung stecken allerdings in den Details:

- Jeder SQL-Kenner weiß, was eine "query" ist. In diesem Handbuch ist das eine Anfrage.
- Gibt es Probleme bei der Übermittlung einer Anfrage, kann es sein, dass eine Zeitüberschreitung eintritt. Der Profi hätte wahrscheinlich nach "Timeout" gesucht.
- Manche Dinge sind einfacher: Ein "string" ist eine Zeichenkette (obwohl für Profis vielleicht ungewohnt), ein "hex value" ein hexadezimaler Wert.

Richtig spannend wird die Übersetzung bei Wörtern, die in der deutschen Fachsprache zumeist englisch verwendet werden, obwohl es passende deutsche Entsprechungen gibt:

- Im Hauptspeicher ("RAM") zwischengespeicherte Daten werden auf die Festplatte zurückgeschrieben. Im Englischen heißt das "flushed to disk", und im Deutschen werden die Daten häufig "geflushed".
- Daten werden zwischengespeichert ("gecached").
- Speicher wird zugewiesen. Man kann auch "alloziert" sagen, was dem englischen "allocated" näher kommt.

Alle diese Entsprechungen, bei denen die deutsche Sprache eher in Vergessenheit geraten ist, wurden zweisprachig aufgenommen. Beispiele:

- Alle Daten werden zwischen Anfragen auf die Festplatte zurück geschrieben (flush).
- Aktualisieren Sie (Update), wenn alles in Ordnung ist.
- Auf eine höhere Version von MySQL aktualisieren (Upgrade) ...

Gelegentlich wird auch in diesem Handbuch die "Performance getuned", neue "Features" eines MySQL-"Release" werden beschrieben usw. Anregungen für eine weiter gehende Eindeutschung nimmt der Übersetzer gern entgegen. Insbesondere gilt das auch für Hinweise zur Verkürzung deutscher Ausdrücke. Beispielsweise heißt "case sensitive" (14 Buchstaben) im Handbuch "abhängig von der verwendeten Groß-/Kleinschreibung" (44 Buchstaben).

Letzter Punkt: Die Übersetzung erfolgte in äußerst enger Anlehnung an das englischsprachige Original. Nichts wurde hinzugefügt (ausser diesem Vorwort), geändert oder weggelassen (Ausnahme: die Geschichte der Änderungen (ChangeLog) vor Version 3.23). Es liegt in der Natur der Dinge, dass weder Original noch Übersetzung frei von Fehlern sind (obwohl wir das anstreben). Berichten Sie bitte Übersetzungsfehler, stilistische "Bugs", die das Verständnis beeinträchtigen und sonstige Anmerkungen zur Übersetzung

direkt an:

Stefan Hinz, <stefan@mysql.com>

Berlin, im Februar 2002

Stefan Hinz

2.1. Über dieses Handbuch

Das ist ein Referenzhandbuch. Es enthält keine allgemeinen Anleitungen zu SQL oder relationalen Datenbankkonzepten.

Da die MySQL Datenbank Software eine laufende Weiterentwicklung erfährt, wird das Handbuch regelmäßig aktualisiert. Die jeweils aktuellste Version dieses Handbuchs befindet sich unter http://www.mysql.com/documentation/. Dieses Handbuch ist gegenwärtig verfügbar in Texinfo, als Klartext (plain text), Info, HTML, PostScript und PDF. Das Primärdokument ist die Texinfo-Datei. Die HTML-Version wird automatisch produziert, indem eine modifizierte Version von `texi2html` benutzt wird. Die Klartext- und Info- Versionen werden mit `makeinfo` hergestellt. Die PostScript-Version wird mit `texi2dvi` und `dvips` produziert. Die PDF-Version wird mit `pdftex` hergestellt.

Wenn Sie Schwierigkeiten haben, Informationen zu finden, beachten Sie bitte auch die durchsuchbare PHP Version des Handbuchs unter http://www.mysql.com/doc/.

Wenn Sie Vorschläge für Hinzufügungen oder Korrekturen dieses Handbuchs haben, schicken Sie sie bitte an das Handbuch-Team: Documentation Team.

Dieses Handbuch wurde geschrieben und wird gewartet von David Axmark, Michael (Monty) Widenius, Jeremy Cole, und Paul DuBois. Andere Kontributoren sind unter Anhang C, *Danksagungen* aufgelistet. Die deutsche Übersetzung stammt von Stefan Hinz. Für die Aktualität ist Jan Lehnardt zuständig.

2.1.1. Konventionen in diesem Handbuch

Dieses Handbuch benutzt bestimmte typographische Konventionen:

- `constant`

  Schriftart gleicher Breite (nicht-proportionale Schrift) wird für Befehle und Optionen benutzt, für SQL-Statements, Datenbank-, Tabellen- und Spaltennamen, für C- und PERL-Code und für Umgebungsvariablen. Beispiel: ``Um festzustellen, wie `mysqladmin` funktioniert, rufen Sie den Befehl mit der `--help` Option auf."

- `filename`

  Schriftart gleicher Breite, die von Anführungszeichen umgeben ist, wird für Datei- und Pfadnamen benutzt. Beispiel: ``Die Distribution wird im Verzeichnis `/usr/local/` installiert."

- ‘`c`’

  Schriftart gleicher Breite, die von Anführungszeichen umgeben ist, wird auch benutzt um Zeichenfolgen anzuzeigen. Beispiel: ``Um ein Platzhalterzeichen einzugeben, benutzen Sie das ‘`%`’ Zeichen."

- *italic*

  Kursivschrift wird für Hervorhebungen verwendet, *wie in diesem Beispiel*.

- **boldface**

  Fettschrift wird für Namen von Zugriffsrechten verwendet (zum Beispiel: ``Gewähren Sie das **process** Zugriffsrecht nicht leichtfertig") und gelegentlich, um **besonders starke Hervorhebungen** zu kennzeichnen.

Wenn Befehle gezeigt werden, die durch ein bestimmtes Programm ausgeführt werden sollen, wird dieses Programm durch einen Prompt (Eingabeaufforderung) vor dem Befehl angezeigt. Der `shell>` Prompt zum Beispiel zeigt an, dass Sie den Befehl von Ihrer Login-Shell aus ausführen sollen. `mysql>` zeigt an, dass Sie den Befehl vom `mysql` Client-Programm aus ausführen sollen:

```
shell> geben sie hier ein shell-kommando ein
mysql> geben sie hier ein mysql-kommando ein
```

Shell-Befehle werden mit der Bourne-Shell-Syntax dargestellt. Wenn Sie eine `csh`-Shell benutzen, müssen die Befehle evtl. etwas anders eingegeben werden. Das folgende Beispiel zeigt, wie mit der Bourne-Shell eine Umgebungsvariable gesetzt wird und anschließend ein Befehl abgesetzt wird:

```
shell> VARNAME=wert irgendein_befehl
```

Um `csh` auszuführen, würden Sie folgende Sequenz ausführen:

```
shell> setenv VARNAME wert
shell> irgendein_befehl
```

Oft müssen Datenbank-, Tabellen- und Spaltennamen in konkreten Befehlen ersetzt werden. Um anzuzeigen, dass eine solche Ersetzung notwendig ist, benutzt dieses Handbuch `db_name`, `tbl_name` und `col_name`. Sie könnten zum Beispiel folgendes Statement sehen:

```
mysql> SELECT spalten_name FROM datenbank_name.tabellen_name;
```

Wenn Sie ein ähnliches Statement eingeben wollen, müssen Sie Ihre eigenen Datenbank-, Tabellen- und Spaltennamen eingeben, zum Beispiel wie folgt:

```
mysql> SELECT autor_name FROM bibliothek.autorenliste;
```

SQL-Statements können in Groß- und Kleinschreibung geschrieben werden. Wenn dieses Handbuch SQL-Statements darstellt, wird Großschreibung verwendet, um spezielle Schlüsselwörter in diesem Kontext hervorzuheben. Kleinschreibung wird für den Rest des Statements verwendet. Folgendes könnten Sie im Kontext des `SELECT` Statements sehen:

```
mysql> SELECT count(*) FROM tabellen_name;
```

Im Kontext der `COUNT()` Funktion hingegen könnte dasselbe Statement wie folgt geschrieben werden:

```
mysql> select COUNT(*) from tabellen_name;
```

Wenn keine besondere Hervorhebung beabsichtigt wird, werden alle Schlüsselwörter in Großschreibung dargestellt.

In Syntax-Beschreibungen werden eckige Klammern (‘`[`’ und ‘`]`’) benutzt, um wahlfrei (optionale) Wörter oder Klauseln anzuzeigen:

```
DROP TABLE [IF EXISTS] tabellen_name
```

Wenn ein Syntaxelement aus einer Anzahl von Alternativen besteht, werden die Alternativen durch gerade Striche (‘`|`’) voneinander getrennt. Wenn genau ein Element aus einer Anzahl von Möglichkeiten ausgewählt werden (**kann**), werden die Alternativen mit eckigen Klammern aufgelistet (‘`[`’ und ‘`]`’):

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

Wenn genau ein Element aus einer Anzahl von Möglichkeiten ausgewählt werden **muss**, werden die Alternativen innerhalb geschweifter Klammern aufgelistet (‘`{`’ und ‘`}`’):

```
{DESCRIBE | DESC} tbl_name {col_name | wild}
```

2.2. Was ist MySQL?

MySQL, die populärste Open Source SQL-Datenbank, wird von MySQL AB zur Verfügung gestellt. MySQL AB ist ein kommerzielles Unternehmen, dessen Geschäft darin besteht, Serviceleistungen rund um die MySQL-Datenbank zur Verfügung zu stellen. See Abschnitt 2.3, „Was ist MySQL AB?“.

- MySQL ist ein Datenbank-Managementsystem.

  Eine Datenbank ist eine strukturierte Sammlung von Daten. Das kann alles sein - von einer einfachen Einkaufsliste über eine Bildergalerie bis zu riesigen Informationsmengen in einem Unternehmensnetzwerk. Um Daten zu einer Computer-Datenbank hinzuzufügen, auf sie zuzugreifen und sie zu verarbeiten, benötigen Sie ein Datenbank-Managementsystem wie MySQL. Weil Computer sehr gut darin sind, große Datenmengen zu handhaben, spielt Datenbank-Management eine zentrale Rolle im Computer-Bereich, sowohl bei Anwendungen, die allein laufen (Stand-Alone-Utilities) als auch als Teil anderer Anwendungen.

- MySQL ist ein relationales Datenbank-Managementsystem.

Eine relationale Datenbank speichert Daten in separaten Tabellen, anstatt sie alle in einem einzigen großen Speicherraum unterzubringen. Hierdurch werden hohe Geschwindigkeit und Flexibilität erreicht. Die Tabellen werden durch definierte Beziehungen verbunden (Relationen), was es möglich macht, Daten aus verschiedenen Tabellen auf Nachfrage zu kombinieren. Der SQL-Teil von MySQL steht für "Structured Query Language" (strukturierte Abfragesprache) - die verbreitetste standardisierte Sprache für Datenbankzugriffe.

- MySQL ist Open-Source-Software.

  Open Source bedeutet, dass es für jeden möglich ist, solche Software zu benutzen und zu verändern. Jeder kann MySQL aus dem Internet herunter laden und benutzen, ohne irgend etwas zu bezahlen. Jeder, der daran interessiert ist, kann den Quelltext studieren und den eigenen Bedürfnissen entsprechend verändern. MySQL benutzt die GPL (GNU General Public License) http://www.gnu.org, um festzulegen, was Sie mit der Software tun dürfen und was Sie nicht tun dürfen, abhängig von unterschiedlichen Situationen. Wenn Ihnen die GPL Probleme bereitet oder wenn Sie MySQL in eine kommerzielle Anwendung einbetten müssen, können Sie eine kommerziell lizensierte Version von uns erwerben.

- Warum sollten Sie MySQL benutzen?

  MySQL ist sehr schnell, zuverlässig und leicht zu benutzen. Wenn Sie nach diesen Eigenschaften suchen, sollten Sie MySQL ausprobieren. MySQL besitzt eine ganze Reihe praktischer Features, die in enger Kooperation mit unseren Benutzern entwickelt wurden. Einen Performance-Vergleich zwischen MySQL und einigen anderen Datenbank-Managementsystemen finden Sie auf unserer Benchmark-Seite. See Abschnitt 6.1.4, „Die MySQL-Benchmark-Suite“.

  MySQL wurde ursprünglich entwickelt, um sehr große Datenbanken handhaben zu können, und zwar sehr viel schneller als existierende Lösungen. Es wurde mehrere Jahre in höchst anspruchsvollen Produktionsumgebungen eingesetzt. Heutzutage bietet MySQL eine umfangreiche Reihe sehr nützlicher Funktionen. Connectivity, Geschwindigkeit und Sicherheit machen MySQL äußerst geeignet, um auf Datenbanken über das Internet zuzugreifen.

- Die technischen Features von MySQL

  Weiter führende technische Informationen finden Sie unter Kapitel 7, *MySQL-Sprachreferenz*. MySQL ist ein Client-Server-System, das aus einem multi-thread SQL-Server besteht, der unterschiedliche Backends, verschiedene Client-Programme und -Bibliotheken, Verwaltungswerkzeuge und etliche Programmschnittstellen unterstützt.

  Wir stellen MySQL auch als multi-thread Bibliothek zur Verfügung, die Sie mit Ihren Anwendungen verknüpfen können, um ein kleineres, schnelleres und leichter zu bedienendes Produkt zu erhalten.

- MySQL stellt beigesteuerte (contributed) Software in großer Menge

  zur Verfügung.

  Es ist sehr wahrscheinlich, dass Ihre Lieblingsanwendung oder -sprache bereits MySQL unterstützt.

Offiziell wird MySQL 'Mai Ess Ku Ell' ausgesprochen (nicht 'Mai Siekwel'). Wir vermeiden allerdings, Leute zu korrigieren, die Mai-Siekwel sagen.

Wir fingen ursprünglich mit der Intention an, den `mSQL`-Code zu benutzen, um unsere eigenen Tabellen anzusprechen, wobei wir unsere eigenen schnellen Low-Level-Routinen (ISAM) benutzten. Nach einigem Testen gelangten wir allerdings zur Überzeugung, dass `mSQL` weder schnell noch flexibel genug wäre, um unsere Anforderungen abzudecken. Dies resultierte in einer neuen SQL-Schnittstelle zu unserer Datenbank, allerdings mit fast derselben API-Schnittstelle, wie sie `mSQL` benutzt. Diese API wurde gewählt, weil sie es erlaubte, Code von Drittanbietern einfach zu portieren. Die Entstehung des Namens MySQL ist nicht völlig geklärt. Unser Basis-Verzeichnis und eine große Anzahl unserer Bibliotheken und Werkzeuge hatten immer schon das Präfix ``my'' während mehr als 10 Jahren. Wie auch immer, auch Montys Tochter (einige Jahre jünger) heißt My. Welcher der beiden Umstände MySQL den Namen gab, ist immer noch ein Rätsel, sogar für uns.

2.2.1. Die wichtigsten Features von MySQL

Die folgende Liste beschreibt einige wichtige Charakteristika von MySQL:

- Voll multi-thread unter Benutzung von Kernel-Threads. Das bedeutet, dass Sie sehr einfach mehrere Prozessoren benutzen können, falls verfügbar.
- C-, C++-, Eiffel-, Java-, Perl-, PHP-, Python- und Tcl-APIs. See Kapitel 9, *MySQL-APIs*.
- Läuft auf vielen verschiedenen Plattformen. See Abschnitt 3.2.2, „Betriebssysteme, die von MySQL unterstützt werden“.
- Viele Spaltentypen: vorzeichenbehaftete / vorzeichenlose Ganzzahlen (Integer), die 1, 2, 3, 4 und 8 Byte lang sind, `FLOAT`,

`DOUBLE`, `CHAR`, `VARCHAR`, `TEXT`, `BLOB`, `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, `YEAR`, `SET`, und `ENUM` Typen. See Abschnitt 7.2, „Spaltentypen“.

- Sehr schnelle Joins durch Benutzung eines optimierten Multi-Joins in einem Durchgang (one-sweep multi-join).
- Volle Operator- und Funktionsunterstützung in `SELECT`- und `WHERE`-Teilen von Anfragen. Beispiel:

```
mysql> SELECT CONCAT(vorname, " ", nachname) FROM tabellen_name
           WHERE einkommen/dependents > 10000 AND age > 30;
```

- SQL-Funktionen sind durch eine hoch optimierte Klassenbibliothek implementiert und sollten so schnell sein, wie es geht! Üblicherweise gibt es überhaupt keine Speicherzuordnung (memory allocation) nach der Initialisierung von Anfragen.
- Volle Unterstützung für SQL-`GROUP BY` und `ORDER BY`- Klauseln. Unterstützung für Gruppierungsfunktionen (`COUNT()`, `COUNT(DISTINCT ...)`, `AVG()`, `STD()`, `SUM()`, `MAX()` und `MIN()`).
- Unterstützung für `LEFT OUTER JOIN` und `RIGHT OUTER JOIN` mit ANSI-SQL und ODBC-Syntax.
- Sie können Tabellen aus unterschiedlichen Datenbanken in ein und derselben SQL-Anfrage benutzen (ab Version 3.22).
- Ein System von Zugriffsberechtigungen und Passwörtern, das sehr flexibel und sicher ist, und das Host-basierende Verifizierung erlaubt. Passwörter sind sicher, weil jeder Passwort-Verkehr verschlüsselt wird, wenn Sie sich mit einem Server verbinden.
- ODBC (Open-DataBase-Connectivity) Unterstützung für Win32 (mit Quelltext). Alle ODBC 2.5 Funktionen und viele weitere. Sie können zum Beispiel MS Access benutzen, um sich mit Ihrem MySQL-Server zu verbinden. See Abschnitt 9.3, „MySQL-ODBC-Unterstützung“.
- Sehr schnelle B-tree disk Tabellen mit Index-Kompression.
- Bis zu 32 Indexe pro Tabelle erlaubt. Jeder Index kann aus 1 bis 16 Spalten oder Teilen von Spalten bestehen. Die maximale Indexlänge beträgt 500 Bytes (das ändert sich evtl., wenn MySQL kompiliert wird). Ein Index kann das Präfix eines `CHAR`- oder `VARCHAR`-Felds benutzen.
- Datensätze fester und variabler Länge.
- Im Arbeitsspeicher gehaltene Hash-Tabellen, die als temporäre Tabellen benutzt werden.
- Kann große Datenbanken handhaben. Wir selbst benutzen MySQL bei einigen Datenbanken, die 50 Mio. Datensätze haben und wir kennen Benutzer, die MySQL mit 60.000 Tabellen und etwa 5 Milliarden Zeilen benutzen.
- Alle Spalten können Vorgabewerte (Defaults) haben. Sie können `INSERT` benutzen, um eine Untermenge der Tabellenspalten mit Werten zu bestücken. Diejenigen Spalten, die nicht explizit angesprochen werden, werden auf ihre Vorgabewerte gesetzt.
- Benutzt GNU Automake, Autoconf und Libtool aus Portabilitätsgründen.
- Geschrieben in C und C++. Getestet mit großen Anzahl verschiedener Compiler.
- Ein sehr schnelles Thread-basierendes Speicherzuordnungs-System (memory allocation system).
- Keine Speicherlecks (memory leaks). MySQL wurde mit Purify getestet, einem kommerziellen Werkzeug zur Entdeckung von Speicherlecks.
- Beinhaltet `myisamchk`, ein sehr schnelles Dienstprogramm zur Überprüfung, Optimierung und Reparatur von Tabellen. Die gesamte Funktionalität von `myisamchk` steht auch über die SQL-Schnittstelle zur Verfügung. See Kapitel 5, *MySQL-Datenbankadministration*.
- Volle Unterstützung für mehrere unterschiedliche Zeichensätze, incl. ISO- 8859-1 (Latin1), big5, ujis und weitere. So sind zum Beispiel die skandinavischen Zeichen `a*', `ä' und `ö' in Tabellen- und Spaltennamen erlaubt.
- Alle Daten werden mit dem ausgewählten Zeichensatz gespeichert. Alle Vergleiche für normale Zeichenkettenvergleiche sind unabhängig von Groß- und Kleinschreibung.
- Die Sortierung ist abhängig vom gewählten Zeichensatz (schwedisch als Vorgabe). Das kann beim Start des MySQL-Servers geändert werden. Um beispielsweise eine sehr fortgeschrittene Sortierung zu sehen, sehen Sie sich den tschechischen Sortier-Code an. MySQL unterstützt viele unterschiedliche Zeichensätze, die bei der Kompilierung und während der Laufzeit festgelegt werden können. Der neue Zeichensatz 'latin_de' sorgt für eine korrekte deutsche Sortierreihenfolge.
- Aliase auf Tabellen und Spalten sind erlaubt, wie im SQL92-Standard festgelegt.
- `DELETE`, `INSERT`, `REPLACE` und `UPDATE` geben die Anzahl der Zeilen zurück, die geändert wurden (bzw. betroffen sind).

Es ist statt dessen auch möglich, die Anzahl der übereinstimmenden Zeilen zurückzugeben, indem beim Verbindungsstart zum Server ein entsprechendes Flag gesetzt wird.

- Funktionsnamen kollidieren nicht mit Tabellen- oder Spaltennamen. `ABS` zum Beispiel ist ein gültiger Spaltenname. Die einzige Einschränkung besteht drin, dass in einem Funktionsaufruf keine Leerzeichen zwischen Funktionsname und der öffnenden runden Klammer, die diesem folgt '`(`', erlaubt ist. See Abschnitt 7.1.6, „Ist MySQL pingelig hinsichtlich reservierter Wörter?“.
- Alle MySQL-Programme können mit der `--help` oder `-?` Option aufgerufen werden, um Online-Hilfe zu erhalten.
- Der Server kann Clients Fehlermeldungen in verschiedenen Sprachen zur Verfügung stellen. See Abschnitt 5.6.2, „Nicht englische Fehlermeldungen“.
- Clients können sich mit dem MySQL-Server über TCP/IP Sockets, Unix Sockets (Unix) oder Named Pipes (NT) verbinden.
- Der MySQL-spezifische `SHOW`-Befehl kann benutzt werden, um Informationen über Datenbanken, Tabellen und Indexe zu erhalten. Der `EXPLAIN`-Code kann benutzt werden um festzustellen, wie der Optimierer eine Anfrage auflöst.

2.2.2. Wie stabil ist MySQL?

Dieser Abschnitt beschäftigt sich mit den Fragen ``Wie stabil ist MySQL?" und ``Kann ich mich auf MySQL bei diesem Projekt verlassen?" Wir werden versuchen, einige Dinge klar zu stellen und einige der wichtigeren Fragen zu beantworten, die offensichtlich viele Leute beschäftigen. Dieser Abschnitt wurde aus Informationen zusammen gestellt, die aus der Mailing-Liste gesammelt wurden (die sehr aktiv beim Berichten von Bugs ist).

Bei TcX funktioniert MySQL ohne jegliche Probleme in unseren Projekten seit Mitte 1996. Als MySQL einer breiteren Öffentlichkeit zugänglich gemacht wurde, fiel uns auf, dass es einige Teile von ``ungetestetem Code" gab, die schnell von neuen Benutzern gefunden wurden, die Anfragen machten, die von unseren eigenen abwichen. Seitdem hat jedes neue Release weniger Portabilitätsprobleme als das vorhergehende (obwohl jedes viele neue Features hat).

Jedes Release von MySQL war benutzbar. Probleme gab es nur, wenn Benutzer anfingen, Code aus den ``Grauzonen" zu benutzen. Natürlich wissen Benutzer von ausserhalb nicht, was diese Grauzonen sind, daher versucht dieser Abschnitt, die momentan bekannten aufzuzeigen. Die Beschreibungen hier beziehen sich auf Version 3.23 von MySQL. Alle bekannten und berichteten Bugs werden in der letzten Version behoben, mit Ausnahme der Bugs, die im Bugs-Abschnitt aufgelistet sind, was Dinge sind, die auf das Design zurückzuführen sind. See Abschnitt 2.7.5, „Bekannte Fehler und Design-Unzulänglichkeiten in MySQL“.

MySQL ist in mehrfachen Ebenen (Layers) und verschiedenen unabhängigen Modulen geschrieben. Diese Module sind im Folgenden aufgeführt, wobei angezeigt wird, wie gut getestet jedes von ihnen ist:

- **Der ISAM Tabellen-Handler --- stabil**

  Dieser verwaltet Speicherung und Abfrage aller Daten in MySQL Version 3.22 und früher. In allen Releases von MySQL gab es nicht einen einzigen (berichteten) Bug in diesem Code. Die einzige Möglichkeit, eine zerstörte (korrumpierte) Tabelle zu erhalten, besteht darin, den Server mitten während eines Updates zu killen. Selbst dadurch ist es unwahrscheinlich, dass Daten unwiederbringlich zerstört werden, denn alle Daten werden zwischen Anfragen auf die Festplatte zurück geschrieben (flush). Es hat nicht einen einzigen Bug-Bericht gegeben, in dem von verlorenen Daten aufgrund von MySQL-Bugs berichtet wurde.

- **Der MyISAM Tabellen-Handler --- stabil**

  Dieser wurde in MySQL Version 3.23 hinzu gefügt. Er basiert zum großen Teil auf dem ISAM Tabellen-Code, hat aber eine Menge neuer und sehr nützlicher Features.

- **Der Parser und lexikalische Analysator --- stabil**

  Es hat seit sehr langer Zeit keinen einzigen berichteten Bug in diesem System gegeben.

- **Der C Client-Code --- stabil**

  Keine bekannten Probleme. Im frühen 3.20 Release gab es einige Einschränkungen hinsichtlich der Größe des Sende- / Empfangs-Puffers (buffer size). Ab Version 3.21 ist die Puffergröße jetzt dynamisch, bis zu einem Vorgabewert von 16 M.

- **Standard-Client-Programme --- stabil**

  Dies beinhaltet `mysql`, `mysqladmin`, `mysqlshow`, `mysqldump` und `mysqlimport`.

- **Basis-SQL --- stabil**

  Die grundlegenden SQL-Funktionen, Zeichenketten-Klassen und dynamisches Speicher-Handling. Nicht ein einziger

berichteter Bug in diesem System.

- **Anfragen-Optimierer (Query optimizer) --- stabil**
- **Bereichs-Optimierer (Range optimizer) --- stabil**
- **Join-Optimierer (Join optimizer) --- stabil**
- **Sperren (Locking) --- Gamma**

  Dies ist sehr system-abhängig. Auf einigen Systemen gibt es große Probleme, wenn Standard-Betriebssystem-Sperren verwendet wird (`fcntl()`). In solchen Fällen sollten Sie den MySQL-Daemon mit dem Flag `--skip-locking` laufen lassen. Bekannt ist, dass solche Probleme auf manchen Linux-Systemen vorkommen sowie auf SunOS, wenn NFS- gemountete Dateisysteme verwendet werden.
- **Linux-Threads --- stabil**

  Das hauptsächliche Problem fand sich im `fcntl()`-Aufruf, der durch Benutzung der `--skip-locking`-Option bei `mysqld` behoben werden kann. Einige Leute haben Lockup-Probleme mit Version 0.5 berichtet. Linux-Threads müssen rekompiliert werden, wenn Sie mehr als 1000 gleichzeitige Verbindungen benutzen wollen. Obwohl es möglich ist, so viele Verbindungen mit den vorgabemäßigen Linux-Threads laufen zu lassen (obwohl man nie über 1021 kommen wird), macht das vorgabemäßige Stack-Spacing von 2 MB die Applikation unstabil, und wir konnten einen Coredump reproduzieren, nachdem 1021 Verbindungen im Leerlauf (idle connections) hergestellt wurden. See Abschnitt 3.6.1, „Linux (alle Linux-Versionen)“.
- **Solaris 2.5+ pthreads --- stabil**

  Wir benutzen dies für unsere gesamte Produktionsarbeit.
- **MIT-pthreads (andere Systeme) --- stabil**

  Seit Version 3.20.15 gab es keine berichteten Bugs mehr, und keine bekannten Bugs seit Version 3.20.16. Auf einigen Systemen gibt es ein ``Misfeature", das heißt einige Operationen sind recht langsam (1/20 Sekunde Schlafzyklus zwischen jeder Anfrage). Natürlich können MIT- Threads alles ein bisschen verlangsamen, aber Index-basierende `SELECT`-Statements werden üblicherweise in einem Zeit-Frame ausgeführt, also sollte es kein mutex locking/thread juggling geben.
- **Andere Thread-Implementierungen --- Beta - Gamma**

  Die Ports zu anderen Systemen sind noch sehr neu und können Bugs haben, möglicherweise auch in MySQL, aber in den meisten Fällen in der Thread- Implementierung selbst.
- **`LOAD DATA ...`, `INSERT ... SELECT` --- stabil**

  Einige Leute dachten, hier Bugs gefunden zu haben, aber üblicherweise haben sich diese als Missverständnisse heraus gestellt. Bitte sehen Sie zuerst im Handbuch nach, bevor Sie Bugs berichten!
- **`ALTER TABLE` --- stabil**

  Einige Änderungen in Version 3.22.12.
- **DBD --- stabil**

  Wird jetzt von Jochen Wiedmann gewartet (`<wiedmann@neckar-alb.de>`). Danke!
- **`mysqlaccess` --- stabil**

  Geschrieben und gewartet von Yves Carlier (`<Yves.Carlier@rug.ac.be>`). Danke!
- **`GRANT` --- stabil**

  große Änderungen in MySQL Version 3.22.12.
- **MyODBC (benutzt ODBC SDK 2.5) --- Gamma**

  Scheint mit einigen Programmen gut zu laufen.
- **Replikation -- Beta / Gamma**

  Wir arbeiten noch an der Replikation, also erwarten Sie nicht, dass diese schon felsenfest steht. Auf der anderen Seite benutzen MySQL- Benutzer diese bereits mit guten Resultaten.
- **BDB-Tabellen -- Beta**

Der Berkeley-Datenbank-Code ist sehr stabil, aber wir verbessern immer noch die Schnittstelle zwischen MySQL und BDB-Tabellen, also wird es einige Zeit dauern, bevor dies so gut wie andere Tabellentypen getestet ist.

- **InnoDB-Tabellen -- stabil**

  Der transaktionale Tabellen-Handler `InnoDB` wurde im `MySQL` 3.23-Baum stabil erklärt, ab Version 3.23.49. `InnoDB` wird in großen hochbelasteten Produktionssystemen eingesetzt.

- **Automatische Wiederherstellung von MyISAM-Tabellen - Beta**

  Dies betrifft nur den neuen Code, der beim Öffnen einer Tabelle nachsieht, ob diese korrekt geschlossen wurde und ein automatisches Überprüfen / Reparieren der Tabelle ausführt, falls das nicht der Fall war.

- **MERGE-Tabellen -- Beta / Gamma**

  Die Benutzung von Schlüsseln bei `MERGE`-Tabellen ist noch nicht sehr ausgetestet. Der restliche Teile des `MERGE`-Codes ist recht gut getestet.

- **FULLTEXT -- Beta**

  Textsuche scheint zu funktionieren, wird aber noch nicht viel eingesetzt.

MySQL AB stellt E-Mail-Support für zahlende Kunden bereit, aber die MySQL-Mailingliste bietet üblicher Weise Antworten für die meisten Fragen. Bugs werden meist direkt mit einem Patch behoben; für schwerwiegende Bugs gibt es fast immer ein neues Release.

2.2.3. Wie groß können MySQL-Tabellen sein?

MySQL Version 3.22 hat eine Begrenzung auf 4G bei der Tabellengröße. Mit dem neuen `MyISAM` in MySQL Version 3.23 wurde die maximale Tabellengröße auf 8 Millionen Terabytes (2 ^ 63 bytes) hochgeschraubt.

Beachten Sie allerdings, dass Betriebssysteme ihre eigenen Dateigrößen- Beschränkungen haben. Hier sind einige Beispiele:

Betriebssystem	**Dateigrößen-Beschränkung**
Linux-Intel 32 bit	2G, 4G oder mehr, abhängig von der Linux-Version
Linux-Alpha	8T (?)
Solaris 2.5.1	2G (möglich sind 4G mit Patch)
Solaris 2.6	4G
Solaris 2.7 Intel	4G
Solaris 2.7 ULTRA-SPARC	8T (?)

Auf Linux 2.2 kann man größere Tabellen als 2G benutzen, wenn man den LFS-Patch für das ext2 Dateisystem benutzt. Auf Linux 2.4 existiert zusätzlich ein Patch für das ReiserFS, um Unterstützung für große Dateien zu erhalten.

Letztlich wird die Tabellengröße für MySQL normalerweise durch das Betriebssystem begrenzt.

Vorgabemäßig haben MySQL-Tabellen eine maximale Größe von etwa 4G. Sie können die maximale Tabellengröße für eine Tabelle mit dem `SHOW TABLE STATUS`-Befehl überprüfen oder mit `myisamchk -dv tabellen_name`. See Abschnitt 5.5.5, „`SHOW`-Syntax“.

Wenn Sie größere Tabellen als 4G benötigen (und Ihr Betriebssystem dies unterstützt), sollten Sie die `AVG_ROW_LENGTH`- und `MAX_ROWS`-Parameter benutzen, wenn Sie Ihre Tabelle anlegen. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“. Sie können diese auch später setzen, mit `ALTER TABLE`. See Abschnitt 7.5.4, „`ALTER TABLE`-Syntax“.

Falls auf Ihre große Tabelle nur mit Lesezugriff zugegriffen wird (read-only), können Sie auch `myisampack` benutzen, um mehrere Tabellen zu vereinen (merge) und sie zu einer zu komprimieren. `myisampack` komprimiert eine Tabelle üblicherweise mindestens um 50%, also können Sie effektiv viel größere Tabellen benutzen. See Abschnitt 5.7.4, „myisampack, MySQL-Programm zum Erzeugen komprimierter Nur-Lese-Tabellen“.

Sie können die Dateibegrenzung des Betriebssystems für `MyISAM` Daten-Dateien umgehen, indem Sie die `RAID`-Option benutzen. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.

Eine weitere Lösung kann die MERGE-Bibliothek darstellen, die Ihnen erlaubt, eine Sammlung identischer Tabellen zugleich zu benutzen. See Abschnitt 8.2, „MERGE-Tabellen“.

2.2.4. Jahr-2000-Konformität

MySQL selbst hat keine Probleme mit der Jahr-2000-Konformität:

- MySQL benutzt Unix-Zeitfunktionen und hat keine Probleme mit Datumsangaben bis `2069`. Alle zweistelligen Jahresangaben werden als Angaben zwischen `1970` und `2069`, betrachtet, was bedeutet, dass, wenn Sie `01` in einer Spalte speichern, MySQL dies als `2001` behandelt.
- Alle MySQL Datumsfunktionen sind in einer Datei `sql/time.cc` gespeichert und sehr sorgfältig kodiert, um Jahr-2000-sicher zu sein.
- In MySQL Version 3.22 und später kann der neue Spaltentyp `YEAR` Jahre `0` und von `1901` bis `2155` in 1 Byte speichern und sie mit 2 oder 4 Ziffern anzeigen.

Probleme können Sie bekommen, wenn Sie MySQL mit Applikationen benutzen, die MySQL auf eine Art benutzen, die nicht Jahr-2000-sicher ist. Zum Beispiel speichern oder ändern viele alte Applikationen Jahresangaben, indem sie zweistellige Werte benutzen (was mehrdeutig ist), anstatt vierstellige Werte zu nehmen. Dieses Problem kann durch Applikationen verschlimmert werden, die Werte wie `00` oder `99` als Anzeiger ``fehlender'' Werte benutzen.

Leider sind diese Probleme möglicherweise schwer zu beheben, weil verschiedene Applikationen von unterschiedlichen Programmierern geschrieben sein können, von denen jeder einen anderen Satz von Konventionen und Funktionen benutzt haben kann, was die Handhabung von Datumsangaben betrifft.

Hier ist eine einfache Demonstration, die zeigt, dass MySQL keine Probleme mit Datumsangaben bis zum Jahr 2030 hat:

```
mysql> DROP TABLE IF EXISTS y2k;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE y2k (date date, date_time datetime, time_stamp timestamp);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO y2k VALUES
    -> ("1998-12-31","1998-12-31 23:59:59",19981231235959),
    -> ("1999-01-01","1999-01-01 00:00:00",19990101000000),
    -> ("1999-09-09","1999-09-09 23:59:59",19990909235959),
    -> ("2000-01-01","2000-01-01 00:00:00",20000101000000),
    -> ("2000-02-28","2000-02-28 00:00:00",20000228000000),
    -> ("2000-02-29","2000-02-29 00:00:00",20000229000000),
    -> ("2000-03-01","2000-03-01 00:00:00",20000301000000),
    -> ("2000-12-31","2000-12-31 23:59:59",20001231235959),
    -> ("2001-01-01","2001-01-01 00:00:00",20010101000000),
    -> ("2004-12-31","2004-12-31 23:59:59",20041231235959),
    -> ("2005-01-01","2005-01-01 00:00:00",20050101000000),
    -> ("2030-01-01","2030-01-01 00:00:00",20300101000000),
    -> ("2050-01-01","2050-01-01 00:00:00",20500101000000);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM y2k;
+------------+---------------------+----------------+
| date       | date_time           | time_stamp     |
+------------+---------------------+----------------+
| 1998-12-31 | 1998-12-31 23:59:59 | 19981231235959 |
| 1999-01-01 | 1999-01-01 00:00:00 | 19990101000000 |
| 1999-09-09 | 1999-09-09 23:59:59 | 19990909235959 |
| 2000-01-01 | 2000-01-01 00:00:00 | 20000101000000 |
| 2000-02-28 | 2000-02-28 00:00:00 | 20000228000000 |
| 2000-02-29 | 2000-02-29 00:00:00 | 20000229000000 |
| 2000-03-01 | 2000-03-01 00:00:00 | 20000301000000 |
| 2000-12-31 | 2000-12-31 23:59:59 | 20001231235959 |
| 2001-01-01 | 2001-01-01 00:00:00 | 20010101000000 |
| 2004-12-31 | 2004-12-31 23:59:59 | 20041231235959 |
| 2005-01-01 | 2005-01-01 00:00:00 | 20050101000000 |
| 2030-01-01 | 2030-01-01 00:00:00 | 20300101000000 |
| 2050-01-01 | 2050-01-01 00:00:00 | 00000000000000 |
+------------+---------------------+----------------+
13 rows in set (0.00 sec)
```

Das zeigt, dass die `DATE`- und `DATETIME`-Typen für zukünftige Datumsangaben keine Probleme bereiten werden (sie handhaben Datumsangaben bis zum Jahr 9999).

Der `TIMESTAMP`-Typ, der zur Speicherung der aktuellen Zeit benutzt wird, hat nur einen Bereich bis zu `2030-01-01`. `TIMESTAMP` hat einen Bereich von `1970` bis `2030` auf 32-Bit-Maschinen (vorzeichenbehafteter Wert). Auf 64-Bit-Maschinen handhabt dieser Spaltentyp bis zu `2106` (vorzeichenloser Wert).

Obwohl MySQL Jahr-2000-kompatibel ist, sind Sie dafür verantwortlich, mehrdeutige Eingaben zu vermeiden. Siehe Abschnitt 2.2.4, „Jahr-2000-Konformität“ wegen der Regeln, die MySQL anwendet, wenn mehrdeutige Datumsangaben gemacht werden (Datumsangaben, die zweistellige Jahreswerte verwenden).

2.3. Was ist MySQL AB?

MySQL AB ist das Unternehmen der MySQL Gründer und Hauptentwickler. MySQL AB wurde ursprünglich in Schweden von David Axmark, Allan Larsson und Michael Monty Widenius gegründet.

Alle Entwickler des MySQL Servers sind Angestellte dieses Unternehmens. Wir sind eine virtuelle Firma mit Mitarbeitern, die über die ganze Welt verstreut in aller Herren Länder sitzen. Der Hauptteil unserer Kommunikation untereinander, mit unseren Anwendern, Unterstützern und Partnern wird über das Internet abgewickelt

Wir haben uns der Entwicklung und Verbreitung des MySQL Datenbankservers verschrieben. MySQL hält das Copyright der MySQL Quelltexte, des MySQL Logos und dieses Handbuchs.. See Abschnitt 2.2, „Was ist MySQL?“.

Die MySQL-Kernwerte zeigen unsere Verpflichtung gegenüber MySQL und Open Source.

Wir wollen, dass MySQL folgendes ist:

- Die beste und meist benutzte Datenbank der Welt.
- Verfügbar für alle. Alle sollen sich MySQL leisten können.
- Leicht zu benutzen.
- Kontinuierlich verbessert, trotzdem immer schnell und sicher bleibend.
- Es soll Spass machen, MySQL zu benutzen und zu verbessern.
- Frei von Bugs.

MySQL AB und die Leute von MySQL AB:

- Verbreiten die Open-Source-Philosophie und unterstützen die Open-Source-Community.
- Bemühen sich, gute Bürger zu sein.
- Bevorzugen Partner, die unsere Werte und unsere Geisteshaltung teilen.
- Beantworten Mail und geben Support.
- Sind ein virtuelles Unternehmen, das mit anderen im Netzwerk zusammenarbeitet (networking).
- Arbeiten gegen Software-Patente.

2.3.1. Geschäftsmodell und Dienstleistungen von MySQL AB

Eine der uns häufig gestellten Fragen ist: Wie kann man von etwas leben, das man kostenlos abgibt? Hier ist die Antwort: MySQL AB verdient Geld mit Support, Dienstleistungen, kommerziellen Lizenzen und Lizenzgebühren, das wir dazu verwenden, die Produktentwicklung zu finanzieren und die MySQL-Geschäftsfelder auszubauen.

Unser Unternehmen läuft seit der Gründung profitabel. Im Oktober 2001 akzeptierten wir eine Risikokapitalfinanzierung durch führende skandinavische Investoren und eine Handvoll Business-Angels. Die Investitionen werden genutzt, um unser Geschäftsmodell auf solide Füße zu stellen und eine Grundlage für nachhaltiges Wachstum zu schaffen.

2.3.1.1. Support

MySQL AB gehört den Gründern und Haupt-Entwicklern der MySQL-Datenbank und wird von ihnen betrieben. Die Entwickler fühlen sich verpflichtet, Kunden und anderen Benutzern Support zu bieten, um mit deren Bedürfnissen und Problemen in Kontakt zu bleiben. Unser gesamter Support wird durch qualifizierte Entwickler geleistet. Wirklich schwierige Fragen werden von Michael Monty Widenius beantwortet, der der erste Entwickler des MySQL-Servers ist. See Abschnitt 2.4.1, „Support den MySQL AB anbietet“.

Um Support unterschiedlicher Grade zu bestellen, besuchen Sie bitte die Bestellseite unter https://order.mysql.com/. Wenn Sie nur beschränkten Zugriff auf das Internet haben, setzen Sie sich bitte mit unserem Vertrieb unter <sales@mysql.com> in Verbindung.

2.3.1.2. Training und Zertifizierung

MySQL AB führt Schulungen zu MySQL und verwandten Themen weltweit durch. Wir bieten sowohl offene Kurse als auch In-house-Trainings an, die auf die speziellen Bedürfnisse Ihres Unternehmens zugeschnitten sind. MySQL-Schulungen werden auch durch unsere Partner durchgeführt, die Authorised MySQL Training Center.

Unsere Schulungsmaterialien benutzen dieselben Beispiel-Datenbanken wie unsere Dokumentation und unsere Beispiel-Applikationen und werden ständig aktualisiert, um den Entwicklungsstand der neusten MySQL-Version widerzuspiegeln. Unsere Trainer erhalten Rückhalt vom Entwicklungsteam, um die Trainingsqualität und die kontinuierliche Entwicklung des Schulungsmaterials sicherzustellen. Das stellt auch sicher, dass keine während der Kurse aufgetretenen Fragen unbeantwortet bleiben.

Wenn Sie an unseren Schulungen teilnehmen, können Sie sicher sein, die Ziele zu erreichen, die Sie mit Ihren MySQL-bezogenen Applikationen anstreben. Ausserdem haben Sie folgende Vorteile:

- Sie sparen Zeit.
- Sie verbessern die Performance Ihrer Applikation(en).
- Sie verringern die Notwendigkeit zusätzlicher Hardware, was Kosten spart.
- Sie verbessern Ihre Sicherheit.
- Sie erhöhen die Zufriedenheit Ihrer Kunden und Mitarbeiter.
- Sie bereiten sich auf die MySQL-Zertifizierung vor.

Wenn Sie an unseren Schulungen Interesse als möglicher Teilnehmer oder Trainingspartner haben, besuchen Sie bitte die Seite unter http://www.mysql.com/training/. Wenn Sie nur beschränkten Zugriff auf das Internet haben, setzen Sie sich bitte mit unserem Trainingspersonal unter `<training@mysql.com>` in Verbindung.

Die Veröffentlichung des MySQL-Zertifizierungsprogramms ist für 2002 geplant. Details finden Sie unter http://www.mysql.com/training/certification.html. Wenn Sie stets die neusten Informationen über das MySQL-Zertifizierungsprogramm haben wollen, schicken Sie bitte eine E-Mail an `<certification@mysql.com>`.

2.3.1.3. Beratung

MySQL AB und seine authorisierten Partner bieten Benutzern des MySQL-Servers und denen, die ihn in ihre Software einbetten wollen, Beratungsleistungen, weltweit.

Unsere Berater können Ihnen helfen, Ihre Datenbanken zu entwerfen und zu optimieren, effiziente Anfragen zu konstruieren, Ihre Plattform auf optimale Performance zu tunen, Migrationsprobleme zu lösen, Replikation aufzusetzen, robuste transaktionale Applikationen zu bauen und vieles mehr. Wir helfen auch Kunden dabei, den MySQL-Server für den Großflächigen Einsatz in ihre Produkte und Applikationen einzubauen.

Unsere Berater arbeiten in enger Kooperation mit unserem Entwicklungsteam, was die technische Qualität unserer Dienstleistungen sicherstellt. Beratungsaufgaben erstrecken sich von zweitägigen Power-Start-Sessions bis zu Projekten, die Wochen und Monate dauern. Unsere Kompetenz deckt nicht nur den MySQL-Server ab, sondern auch Programmier- und Skripting-Sprachen wie PHP, Perl und andere.

Wenn Sie an unseren Beratungsleistungen interessiert sind oder ein Consulting-Partner werden wollen, besuchen Sie bitte unsere Seite unter http://www.mysql.com/consulting/.

2.3.1.4. Kommerzielle Lizenzen

Die MySQL-Datenbank wird unter der `GNU General Public License` veröffentlicht (`GPL`). Das bedeutet, dass die MySQL-Software kostenlos unter der GPL benutzt werden darf. Wenn Sie nicht an die GPL-Bedingungen gebunden sein wollen (was in der Folge bedeutet, dass auch Ihre eigenen Applikationen GPL werden), können Sie eine kommerzielle Lizenz für dasselbe Produkt unter https://order.mysql.com/ erwerben.

Weil MySQL AB das Copyright am MySQL-Server besitzt, können wir eine `duale Lizensierung` einsetzen, was heißt, dass dasselbe Produkt sowohl unter der GPL als auch unter einer kommerziellen Lizenz erhältlich ist. Das berührt in keiner Weise die Verpflichtung von MySQL AB gegenüber `Open Source`. Wegen Details, wann eine kommerzielle Lizenz erforderlich ist, sehen Sie bitte unter Abschnitt 2.4.4, „MySQL-Lizenzpolitik“ nach.

Wir verkaufen auch kommerzielle Lizenzen von Open-Source-GPL-Software Dritter. Ein gutes Beispiel hierfür ist der `InnoDB`-Tabellen-Handler, der ACID-Unterstützung, Sperren auf Zeilenebene, Wiederherstellung nach Abstürzen, Multiversionierung, Fremdschlüsselunterstützung und vieles mehr bietet.

2.3.1.5. Partnerprogramme

MySQL AB hat ein weltweites Partnerprogramm, dass Schulungskurse, Support, Beratung, Lösungen, Publikationen plus Weiterverkauf und Vertrieb von MySQL und verwandten Produkten beinhaltet. Partner erscheinen unter http://www.mysql.com/ auf der Website und erhalten das Recht, spezielle Versionen der MySQL-Schutzmarken zu benutzen, um ihre Produkte zu identifizieren und ihr Geschäft voranzutreiben. Wenn Sie interessiert sind, ein MySQL-AB-Partner zu werden, schicken Sie bitte eine E-Mail an <partner@mysql.com>.

Das Wort MySQL und das MySQL-Delphin-Logo sind Schutzmarken von MySQL AB. See Abschnitt 2.4.3, „MySQL-AB-Logos und -Schutzmarken“.

2.3.1.6. Werbung

Die MySQL-Website (http://www.mysql.com/) ist bei Entwicklern und Benutzern beliebt. Im Oktober 2001 bedienten wir 10 Millionen Seitenanfragen (PageViews). Unsere Besucher repräsentieren eine Gruppe, die Kaufentscheidungen und Empfehlungen sowohl für Software als auch für Hardware trifft. 12% unserer Besucher genehmigen Kaufentscheidungen, lediglich 9% sind überhaupt nicht an Kaufentscheidungen beteiligt. Mehr als 65% haben innerhalb des letzten halben Jahres online eingekauft, 70% planen, in den nächsten Monaten einzukaufen. Wenn Sie Interesse haben, Werbebanner auf unserer Website http://www.mysql.com/ zu schalten, setzen Sie sich bitte mit <advertising@mysql.com> in Kontakt.

2.3.1.7. Kontaktinformationen

Die MySQL Website (http://www.mysql.com/) enthält die neusten Informationen über MySQL und MySQL AB.

Für Presseservice und Anfragen aller Art, die in unseren Veröffentlichungen (http://www.mysql.com/news/) nicht behandelt werden, wenden Sie sich bitte an <press@mysql.com>.

Zeitnahe, präzise Antworten auf technische Fragen erhalten Sie, wenn Sie unter order einen unserer Support-Verträge abschließen. MySQL-Support wird von den MySQL-Entwicklern geleistet, weshalb der Standard extrem hoch ist.

Informationen über MySQL Trainig erhalten Sie unter http://www.mysql.com/training/. Wenn Sie einen eingeschränkten Internetzugang haben, kontaktieren Sie bitte unser Trainingspersonal unter <training@mysql.com>. See Abschnitt 2.3.1.2, „Training und Zertifizierung“.

Für Informationen über das MySQL Zertifizierungsprogramm erhalten Sie unter http://www.mysql.com/training/certification.html. Wenn Sie weiterhin über das MySQL Zertifizierungsprogramm informiert werden wollen, schreiben Sie eine E-Mail an <certification@mysql.com>. See Abschnitt 2.3.1.3, „Beratung“.

Kommerzielle Lizenzen können online unter https://order.mysql.com/ abgewickelt werden. Dort finden Sie ausserdem Informationen darüber, wie Sie ihre Bestellung per Fax erledigen können. Wenn Sie Fragen bezüglich der Lizensierung haben, oder Sie ein Angebot über eine größere Lizenzerteilung erhalten wollen, füllen Sie bitte Das Kontaktformular auf unserer Website (http://www.mysql.com/) aus, oder schicken Sie eine E-Mail an <licensing@mysql.com> (für Lizenzfragen) oder an <sales@mysql.com> (für Verkaufsinformationen). See Abschnitt 2.4.4, „MySQL-Lizenzpolitik“.

Wenn Sie daran interessiert sind, ein Werbebanner auf unserer Website (http://www.mysql.com/) zu schalten, schicken Sie bitte eine E-Mail an <advertising@mysql.com>. See Abschnitt 2.3.1.6, „Werbung“.

Wenn Sie ein Unternehmen vertreten, dass an einer Partnerschaft mit MySQL interessiert ist, schicken Sie bitte eine E-Mail an <partner@mysql.com>.

Für weitere Informationen über die MySQL Schutzmarkenbestimmungen, beachten Sie bitte http://www.mysql.com/company/trademark.html oder kontaktieren Sie <trademark@mysql.com>.

See Abschnitt 2.4.3, „MySQL-AB-Logos und -Schutzmarken“.

Wenn Sie an einem der Jobs interessiert sind, die im jobs-Abschnitt aufgeführt sind, schicken Sie bitte eine E-Mail an <jobs@mysql.com>. Bitte senden Sie ihre CV nicht als Anhang an dieser mail mit, sondern fügen Sie sie lieber am Ende ihrer mail als Klartext (plain text) ein.

Allgemeine Diskussionen mit vielen unserer Benutzer können Sie auf den entsprechenden Mailing-Listen führen.

Fehlerberichte (Auch Bugreporte genannt), sowie Fragen und Kommentare, sollten an die Mailingliste <mysql@lists.mysql.com> gehen. Wenn Sie ein empfindliches Sicherheitsloch im MySQL Server gefunden haben, sollten

Sie eine E-Mail an <security@mysql.com> schreiben. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.

Wenn Sie Benchmarkergebnis haben, die wir veröffentlichen können, kontaktieren Sie uns unter <benchmarks@mysql.com>.

Wenn Sie Vorschläge für Hinzufügungen oder Korrekturen dieses Handbuchs haben, schicken Sie sie bitte an das Handbuch-Team: Documentation Team.

Fragen zur Arbeitsweise oder zu Inhalten unserer Website(http://www.mysql.com/) stellen Sie bitte an

<webmaster@mysql.com>.

Fragen über das MySQL Portal (http://www.mysql.com/portal/) können an <portals@mysql.com> geschickt werden.

Die Datenschutzbestimmungen von MySQL AB können Sie unter http://www.mysql.com/company/privacy.html einsehen. Für irgendwelche Fragen darüber, wenden Sie sich bitte an <privacy@mysql.com>.

Allgemeine Informationsanfragen schicken Sie bitte an <info@mysql.com>.

2.4. MySQL Support and Lizensierung

Dieser Abschnitt beschreibt die MySQL Support und Lizensierungsvereinbarungen

2.4.1. Support den MySQL AB anbietet

Wir versuchen, technischen Support in einem breiten und umfassenden Blickwinkel zu sehen. Fast jedes Problem im Zusammenhang mit MySQL-Software ist für uns wichtig, wenn es für Sie wichtig ist. Typischerweise suchen Kunden Hilfe dabei, wie man unterschiedliche Befehle und Dienstprogramme zum Funktionieren bringt, wie Performance-Flaschenhälse beseitigt werden können, wie man beschädigte Systeme repariert, wie sich Betriebssysteme oder Netzwerkkonfigurationen auf MySQL auswirken, wie man Datensicherung und Wiederherstellung optimal konfiguriert, wie man APIs benutzt usw. Unser Support deckt nur den MySQL-Server und unsere eigenen Dienstprogramme ab, nicht Produkte Dritter, die auf den MySQL-Server zugreifen, obwohl wir auch hierbei versuchen, zu helfen wo wir können.

Detaillierte Informationen über unsere unterschiedlichen Support-Optionen finden Sie auf https://order.mysql.com/, wo auch Support-Verträge online bestellt werden können. Wenn Sie nur beschränkten Zugriff auf das Internet haben, setzen Sie sich mit unserem Vertrieb unter <sales@mysql.com> in Verbindung.

2.4.2. Copyrights und Lizenzen, die von MySQL verwendet werden.

MySQL AB besitzt das Copyright des MySQL-Quellcodes, die MySQL-Logos und (registrierten) Schutzmarken sowie dieses Handbuch. See Abschnitt 2.3, „Was ist MySQL AB?“. Es gibt einige verschiedene Lizenzen, die für MySQL-Distributionen relevant sind:

1. Der gesamte MySQL-spezifische Quelltext des Servers, die mysqlclient-Bibliothek und der Client sowie die GNU-readline-Bibliothek unterliegen der GNU General Public License. See Anhang H, *GNU GENERAL PUBLIC LICENSE*. Der Text dieser Lizenz befindet sich in der Datei COPYING in allen MySQL-Distributionen.
2. Die GNU-getopt-Bibliothek unterliegt der GNU Lesser General Public License. Siehe http://www.fsf.org/licenses/.
3. Einige Teile des Quelltextes (die regexp-Bibliothek) werden von einem Copyright in Berkeley-Art abgedeckt.
4. Ältere Versionen von (3.22 und früher) stehen unter einer strikteren Lizenz (http://www.mysql.com/products/mypl.html). Bitte beachten sie die Dokumentation der speziellen Version für weitere Informationen.
5. Das Handbuch steht *nicht* unter einer GPL-artigen Lizenz. Die Benutzung des Handbuchs unterliegt den folgenden Bestimmungen.

   - Die Konvertierung in andere Formate ist erlaubt, der Inhalt jedoch darf auf keinen Fall geändert oder bearbeitet werden.
   - Sie können eine gedruckte Version für Ihren Privatgebrauch erstellen.
   - Für alle anderen Zwecke, wie den Verkauf von gedruckten Kopien oder die Verwendung (auch in Teilen) des Handbuchs in anderen Veröffentlichungen, ist eine vorherige Vereinbarung mit MySQL AB erforderlich.

   Bitte senden Sie eine E-Mail an Documentation Team für weitere Informationen oder wenn Sie daran interessiert sind, eine Übersetzung zu erstellen.

Für Informationen darüber, wie die MySQL-Lizenzen in der Praxis angewendet werden, beachten Sie bitte Abschnitt 2.4.4, „MySQL-Lizenzpolitik“. Siehe auch Abschnitt 2.4.3, „MySQL-AB-Logos und -Schutzmarken“.

2.4.2.1. Verwendung des MySQL Servers unter einer kommerziellen Lizenz

Internet Service Provider (ISP) hosten oft MySQL-Server für ihre Kunden. Aufgrund der GPL-Lizenz ist hierfür keine Lizensierung erforderlich.

Auf der anderen Seite ermutigen wir Leute, ISPs zu benutzen, die MySQL-Support haben, und das wird ihnen Vertrauen geben, dass ihr ISP im Falle von Problemen mit ihrer MySQL-Installation helfen wird, das Problem zu lösen (in manchen Fällen mit der Hilfe des MySQL-Entwicklungsteams).

Alle ISPs, die auf dem neuesten Stand der Dinge bleiben wollen, sollten sich in die `announce`-Mailing-Liste eintragen, um auf der Hut zu sein vor schwerwiegenden Problemen, die für ihre MySQL-Installationen relevant sein könnten.

Beachten Sie bitte, dass ein ISP ohne MySQL-Lizenz seinen Kunden zumindest Lesezugriff auf den Quelltext der MySQL-Installation geben sollte, damit die Kunden feststellen können, dass diese korrekt gepatcht ist.

2.4.2.2. Einen Webserver betreiben, der MySQL benutzt

Wenn Sie MySQL in Verbindung mit einem Webserver unter Unix betreiben, brauchen Sie nicht für eine Lizenz zu bezahlen.

Das gilt selbst dann, wenn Sie einen kommerziellen Webserver betreiben, der MySQL benutzt, weil Sie nicht selbst eine eingebettete MySQL-Version verkaufen. Dennoch bitten wir Sie, in einem solchen Fall MySQL-Support zu kaufen, weil MySQL Ihrem Unternehmen hilft.

2.4.3. MySQL-AB-Logos und -Schutzmarken

Viele MySQL-Datenbankbenutzer wollen auf Ihren Websites, ihren Büchern und Packungsprodukten das MySQL-AB-Delphin-Logo zeigen. Wir begrüßen das und ermuntern dazu, weisen aber darauf hin, dass das Wort `MySQL` und das MySQL-Delphin-Logo Schutzmarken von MySQL AB sind und nur so benutzt werden dürfen, wie in unserer Schutzmarken-Richtlinie unter http://www.mysql.com/company/trademark.html festgelegt.

2.4.3.1. Das Original-MySQL-Logo

Das MySQL-Delphin-Logo wurde von der finnischen Werbeagentur Priority im Jahr 2001 entworfen. Der Delphin wurde als passendes Symbol für die MySQL-Datenbank gewählt, weil er schlau, schnell und schlank ist und mühelos durch die Daten-Ozeane navigiert. Ausserdem mögen wir Delphine.

Das Original-MySQL-Logo darf nur von Repräsentanten von MySQL AB und von Personen benutzt werden, die eine schriftliche Erlaubnis hierfür haben.

2.4.3.2. MySQL-Logos die ohne schriftliche Erlaubnis benutzt werden dürfen

Wir haben einen Satz spezieller Logos für *vorbehaltliche Benutzung* angelegt, die von unserer Website unter http://www.mysql.com/press/logos.html herunter geladen werden können und von Dritten auf ihren Websites ohne schriftliche Erlaubnis von MySQL AB benutzt werden dürfen. Der Gebrauch dieser Logos ist - wie der Name anzeigt - nicht völlig uneingeschränkt, sondern unterliegt unseren Schutzmarken-Richtlinien, die Sie auf unserer Website finden. Sie sollten diese Richtlinien lesen, wenn Sie planen, die Logos zu benutzen. Die Anforderungen sind im Wesentlichen:

- Benutzen Sie das gewünschte Logo von der http://www.mysql.com/- Site. Sie dürfen die Größe nach Ihren Bedürfnissen anpassen, aber keine Farben oder das Design ändern noch die Grafik in sonstiger Form verändern.
- Heben Sie hervor, dass Sie - und nicht MySQL AB - der Betreiber und Eigner der Site ist, auf der die MySQL-Schutzmarke gezeigt wird.
- Sie dürfen die Schutzmarke nicht auf eine Weise benutzen, die MySQL AB oder dem Wert der MySQL-AB-Schutzmarken schadet. Wir behalten uns das Recht vor, das Recht zur Benutzung der MySQL-AB-Schutzmarke zu widerrufen.
- Wenn Sie die Schutzmarke auf einer Website benutzen, machen Sie sie anklickbar, wobei direkt nach http://www.mysql.com/ verlinkt wird.
- Wenn Sie die MySQL-Datenbank unter GPL in einer Applikation benutzen, muss Ihre Applikation (i) Open Source sein, (ii) in der Lage sein, sich mit einem MySQL-Server zu verbinden.

Setzen Sie sich unter `<trademark@mysql.com>` mit uns in Verbindung, um wegen spezieller Arrangements anzufragen, die Ihren Bedürfnissen entsprechen.

2.4.3.3. Wann Sie eine Erlaubnis für die Benutzung des MySQL-Logos benötigen

In folgenden Fällen benötigen Sie eine schriftliche Erlaubnis von MySQL AB, bevor Sie die MySQL-Logos benutzen:

- Wenn Sie irgend ein MySQL-AB-Logo irgendwo ausser auf Ihrer Website zeigen.
- Wenn Sie irgend ein MySQL-AB-Logo ausser den oben erwähnten Logos zur *vorbehaltlichen Benutzung* auf Websites oder anderswo anzeigen.

Aus rechtlichen und kommerziellen Gründen müssen wir die Benutzung der MySQL-Schutzmarken auf Produkten, Büchern usw. beobachten. Üblicherweise verlangen wir eine Gebühr für das Anzeigen von MySQL-AB-Logos auf kommerziellen Produkten, weil wir der Meinung sind, das es vertretbar ist, dass einige der Erlöse für die Weiterentwicklung der MySQL- Datenbank zurückfließen.

2.4.3.4. MySQL-AB-Partnerschafts-Logos

MySQL-Partnerschafts-Logos dürfen nur von Unternehmen und Personen benutzt werden, die eine schriftliche Partnerschaftsvereinbarung mit MySQL AB haben. Partnerschaften beinhalten eine Zertifizierung als MySQL-Trainer oder -Berater. Sehen Sie bitte unter

Abschnitt 2.3.1.5, „Partnerprogramme“ nach.

2.4.4. MySQL-Lizenzpolitik

Die formalen Bedingungen der GPL-Lizenz stehen unter Anhang H, *GNU GENERAL PUBLIC LICENSE*. Im Wesentlichen ist unsere Lizenzpolitik und die Interpretation der GPL wie folgt:

Beachten Sie bitte, dass ältere Versionen von MySQL immer noch einer strengeren Lizenz unterliegen. Sehen Sie in der Dokumentation der betreffenden Version wegen entsprechender Informationen nach. Wenn Sie eine kommerzielle Lizenz benötigen, weil die GPL-Lizenz nicht zu den Anforderungen Ihrer Applikation passt, können Sie eine Lizenz unter https://order.mysql.com/ kaufen.

Für normalen internen Gebrauch kostet MySQL nichts. Sie brauchen uns nichts zu bezahlen, wenn Sie nicht wollen.

Eine Lizenz wird benötigt:

- Wenn Sie ein Programm, das nicht freie Software ist, mit Code des MySQL-Servers oder der Client-Programme verbinden, die den GPL-Copyrights unterliegen. Das ist zum Beispiel der Fall, wenn Sie MySQL als eingebetteten Server (Embedded Server) in Ihren Applikationen benutzen, oder wenn Sie dem MySQL-Server Erweiterungen hinzufügen, die nicht freie Software sind. In diesen Fällen würden Ihre Applikation bzw. Ihr Code ebenfalls GPL werden, weil die GPL in solchen Fällen wie ein Virus wirkt. Sie können dieses Problem vermeiden, wenn Sie den MySQL-Server mit einer kommerziellen Lizenz von MySQL AB erwerben. Siehe http://www.gnu.org/copyleft/gpl-faq.html.
- Wenn Sie eine kommerzielle Applikation haben, die NUR mit MySQL funktioniert, und wenn Sie die Applikation zusammen mit dem MySQL-Server ausliefern. Wir betrachten so etwas als Einbindung, selbst wenn es über das Netzwerk geschieht.
- Wenn Sie eine Distribution von MySQL besitzen und nicht den Quelltext für Ihre Kopie des MySQL-Servers zur Verfügung stellen, so wie es in der GPL-Lizenz festgelegt ist.

In Situationen, wo eine MySQL-Lizenz benötigt wird, brauchen Sie eine Lizenz pro Maschine, auf der der MySQL-Server läuft. Eine Mehrprozessor-Maschine zählt jedoch als eine einzelne Maschine, und es gibt keine Beschränkung hinsichtlich der Anzahl von MySQL-Servern, die auf einer Maschine laufen, oder hinsichtlich der Anzahl von Clients, die zur gleichen Zeit mit einem Server verbunden sind, der auf dieser Maschine läuft!

Falls Sie nicht sicher sind, ob für Ihre spezielle Benutzung von MySQL eine Lizenz erforderlich ist, lesen Sie diesen Abschnitt bitte nochmals, bevor Sie uns kontaktieren. See Abschnitt 2.3.1.7, „Kontaktinformationen“.

Wenn Sie eine MySQL-Lizenz benötigen, ist die Bezahlung am einfachsten, wenn Sie das Lizenzformular auf dem Secure-Server von MySQL unter https://order.mysql.com/ benutzen.

2.4.4.1. Benutzung des Worts `MySQL` in Druckmaterialien oder

Präsentationen

MySQL AB begrüßt Verweise auf die MySQL-Datenbank, aber das Wort `MySQL` ist eine Schutzmarke von MySQL AB. Deshalb müssen Sie der ersten oder deutlichsten Erwähnung des Worts `MySQL` das Schutzmarken-Symbol `TM` hinzufügen, und wo angebracht deutlich machen, dass `MySQL` eine Schutzmarke von MySQL AB ist. Details entnehmen Sie bitte unserer Schutzmarken-Richtlinie unter http://www.mysql.com/company/trademark.html.

2.4.4.2. Benutzung des Worts `MySQL` in Unternehmens- und

Produktnamen

Die Benutzung des Worts `MySQL` in Produkt- und Unternehmensnamen oder in Internet-Domänen-Namen ist nur mit vorheriger schriftlicher Erlaubnis durch MySQL AB gestattet.

2.5. MySQL 4.0 kurz und bündig

Dateline: 16. Oktober 2001, Uppsala, Schweden

Lange durch MySQL AB angekündigt und lange von unseren Benutzern erwartet: Der MySQL-Server 4.0 ist jetzt in der Alpha-Version zum Herunterladen von http://www.mysql.com/ und unseren Mirrors verfügbar.

Die neuen Haupt-Features des MySQL-Servers 4.0 sind eng mit unserem bestehenden Geschäft und den Community-Nutzern verzahnt. Durch ihn wird die MySQL-Datenbank-Software als Lösung für geschäftskritische Schwerlast-Datenbanksysteme verbessert. Weitere neue Features zielen auf die Benutzer eingebetteter Datenbanken.

2.5.1. Schritt für Schritt

Das Erscheinen des MySQL-Servers 4.0 wird in mehreren Schritten erfolgen, wobei die erste Version 4.0.0 genannt wird und bereits die meisten neuen Features enthält. Zusätzliche Features werden in die Versionen 4.0.1, 4.0.2 usw. eingebaut, höchstwahrscheinlich innerhalb weniger Monate. MySQL 4.0 wird als Beta gekennzeichnet. In MySQL 4.1 werden dann weitere neue Features hinzugefügt. Es wird angestrebt, das Alpha-Release Anfang 2002 herauszubringen.

2.5.2. Für den sofortigen Entwicklungseinsatz

Es wird nicht empfohlen, Produktionssysteme auf den MySQL-Server 4.0 umzustellen, bis dieser in der Beta-Version veröffentlicht wird. Selbst das anfängliche Release hat unsere ausgiebigen Tests ohne jegliche Fehler durchlaufen, auf allen Plattformen, auf denen wir testen. Wegen der großen Zahl neuer Features empfehlen wir daher den MySQL-Server selbst in der Alpha-Version für Entwicklungsarbeiten, wobei in Betracht gezogen werden kann, dass der MySQL-Server 4.0 das Stadium "stabil" erreichen wird, bevor Applikationen hiermit veröffentlicht werden, die jetzt im Entwicklungsstadium sind.

2.5.3. Eingebettetes MySQL

`libmysqld` macht den MySQL-Server für einen erheblich ausgedehnten Bereich von Applikationen geeignet. Wenn man die eingebettete MySQL- Server-Bibliothek benutzt, kann man den MySQL-Server in unterschiedlichste Applikationen und elektronische Geräte einbetten, bei denen der Endbenutzer keinerlei Ahnung davon hat, dass ihnen eine Datenbank unterlegt ist. Der eingebettete MySQL-Server ist ideal für Benutzung hinter den Kulissen in Internet-Geräten, öffentlichen Kiosken, schlüsselfertigen Hardware-/Software-Einheiten, Hochlast-Internet- Servern oder Datenbanken, die auf CD-ROM vertrieben werden.

Viele Benutzer von eingebettetem MySQL können von der *dualen Lizensierung* der MySQL-Software profitieren. Neben der GPL-Lizenz sind auch kommerzielle Lizenzen für diejenigen verfügbar, die nicht an die GPL gebunden sein wollen. Die eingebettete MySQL-Bibliothek benutzt dieselbe Schnittstelle wie die normale Client-Bibliothek und ist daher angenehm und leicht zu benutzen. See Abschnitt 9.4.9, „libmysqld, die eingebettete MySQL-Server-Bibliothek“.

2.5.4. Weitere ab MySQL 4.0.0 verfügbare Features

- Version 4.0 erhöht die *Geschwindigkeit des MySQL-Servers* in einigen Bereichen noch weiter, zum Beispiel bei Massen-`INSERT`s, beim Suchen auf komprimierten Indexen, der Erzeugung von `FULLTEXT`-Indexen oder auch bei `COUNT(DISTINCT)`.
- Der Tabellen-Handler `InnoDB` wird jetzt als Feature des standardmäßigen MySQL-Servers angeboten und enthält vollständige Unterstützung für `Transaktionen` und `Sperren auf Zeilenebene`.
- Der MySQL-Server 4.0 unterstützt sichere Kommunikation zwischen Client und Server, wodurch die Sicherheit gegen böswilliges Eindringen und unbefugten Zugriff erheblich erhöht wird. Bei Web-Applikationen, die ein Grundpfeiler der MySQL-Benutzung sind, konnten Web-Entwickler immer schon SSL verwenden, um den Verkehr zwischen Endbenutzer-Browser und der Web-Applikation zu sichern, sei sie nun in PHP, Perl, ASP oder mit irgend einem anderen Web-Entwicklungswerkzeug geschrieben. Der Verkehr zwischen dem Entwicklungswerkzeug und dem mysqld-Serverprozess konnte bislang aber nur dadurch gesichert werden, dass die Prozesse auf Computern innerhalb derselben Firewall residierten. Ab MySQL-Server 4.0 kann der *mysqld*-Server-Daemon-Prozess selbst `Secure Sockets Layer` (`SSL`) benutzen, was ihn in die Lage versetzt, eine sichere Datenübertragung zwischen einer MySQL-Datenbank und beispielsweise einer Windows-Applikation ausserhalb der Firewall aufzubauen.
- Unsere deutschen, österreichischen und schweizerischen Benutzer werden bemerken, dass es einen neuen Zeichensatz latin_de gibt, der die *deutsche Sortierreihenfolge* beinhaltet, indem deutsche Umlaute in derselben Sortierung erscheinen wie bei deutschen Telefonbüchern üblich.
- Zu den Features, die die `Migration` von anderen Datenbanksystemen zum MySQL-Server erleichtern, gehören `TRUNCATE TABLE` (wie in Oracle) und `IDENTITY` a1s Synonym für automatisch hochgezählte Schlüssel (wie in Sybase). viele Benutzer werden sich auch darüber freuen, dass der MySQL-Server jetzt das `UNION`-Statement unterstützt, ein lang erwartetes Standard-SQL-Feature.

- Bei der Erstellung neuer Features für neue Benutzer haben wir die Gemeinschaft treuer Benutzer nicht vergessen. Es gibt jetzt Multi- Tabellen-`DELETE`-Statements. Durch das Hinzufügen von Unterstützung für `symbolisches Verknüpfen` von `MyISAM` auf Tabellenebene (und nicht nicht - wie bisher - auf Datenbankebene), sowie durch das vorgabemäßige Anschalten der Verknüpfungen unter Windows hoffen wir zeigen zu können, dass wir Verbesserungsvorschläge ernst nehmen. Funktionen wie `SQL_CALC_FOUND_ROWS` und `FOUND_ROWS()` ermöglichen herauszufinden, wie viele Zeilen eine Anfrage ohne eine `LIMIT`-Klausel zurückgegeben hätte.

2.5.5. Zukünftige Features in MySQL 4.0

Für die kommenden Releases des MySQL-Servers 4.0 (4.0.1, 4.0.2 usw.) können Sie folgende Features erwarten, die noch in der Entwicklung sind:

- Benutzer des MySQL-Servers für geschäftskritische Hochlast-Anwendungen werden die Ergänzungen unseres Replikationssystems und unsere Online- "Hot"-Datensicherung begrüßen. Spätere Versionen von 4.0 werden `absturzsichere Replikation` beinhalten, die es bereits in Version 4.0.0 gibt, sowie den `LOAD DATA FROM MASTER`-Befehl, der bald das Aufsetzen von Slaves automatisieren wird. `online backup` wird das Hinzufügen eines neuen Replikations-Slaves erleichtern, ohne dass man den Master herunterfahren muss, und es gibt auf Systemen mit vielen Aktualisierungen nur geringe Geschwindigkeitseinbussen.
- Als Bequemlichkeits-Feature für Datenbank-Administratoren wird hinzugefügt, dass mysqld-Parameter (Startoptionen) bald ohne das Herunterfahren des Servers gesetzt werden können.
- Die neuen Eigenschaften des MySQL-Servers 4.0 für die Volltext- (`FULLTEXT`)-Suche ermöglichen die `FULLTEXT`-Indexierung großer Texte sowohl mit binärer wie auch mit natürlichsprachiger Suchlogik. Benutzer können minimale Wortlängen anpassen und ihre eigenen Stopp-Wort-Listen in jeder menschlichen Sprache festlegen, wodurch gänzlich neue Applikationen ermöglicht werden, die auf dem MySQL-Server aufbauen.
- Viele Applikationen mit starkem Lesezugriff werden durch die noch weiter erhöhte Geschwindigkeit des neu geschriebenen `Schlüssel-Caches` profitieren.
- Viele Entwickler wird auch die `MySQL-Befehlshilfe` im Client freuen.

2.5.6. MySQL 4.1, das folgende Entwicklungs-Release

Intern wird durch das neue .frm-Dateiformat für Tabellendefinitionen in MySQL-Server 4.0 die Grundlage für neue Features in MySQL-Server 4.1 gelegt, beispielsweise `verschachtelte Unterabfragen`, `gespeicherte Prozeduren` und `Fremdschlüssel- Integritätsregeln`, die ganz oben auf der Wunschliste vieler unserer Kunden stehen. Daneben werden auch einfachere Erweiterungen wie Multi- Tabellen-`UPDATE`-Statements hinzugefügt.

Nach diesen Ergänzungen werden Kritiker des MySQL-Datenbankservers es noch schwerer haben, auf Schwächen des MySQL-Datenbank-Managementsystems hinzuweisen. MySQL, das seit langem für seine Stabilität, Geschwindigkeit und Einfachheit der Benutzung bekannt ist, wird dann den Anforderungen sehr anspruchsvoller Käufer genügen.

2.6. MySQL-Informationsquellen

2.6.1. MySQL-Portale

Die MySQL-Portale (http://www.mysql.com/portal/) auf unserer Website bieten ein breites Spektrum MySQL-bezogener Informationen und Links. Sie sind so aufgebaut, dass Sie leicht die Dinge finden, die Sie interessieren.

Sie können sich als Benutzer registrieren. In diesem Fall können Sie alle Dinge in den Portalen kommentieren und bewerten und auch selbst Dinge beisteuern. Bei der Registrierung können Sie auch angeben, ob und - wenn ja - welche Newsletter aus welchen Kategorien Sie beziehen wollen.

Einige der momentanen MySQL-Portal-Kategorien:

- Bücher Hier finden Sie alle möglichen MySQL- oder Computer-bezogenen Bücher, die Sie kommentieren, bewerten oder kaufen können. Während dieses Handbuch (insbesondere die Online-Version) immer noch der richtige Platz für aktuellste technische Informationen ist, ist sein vorrangiges Ziel, alles zu enthalten, was man über das MySQL-Datenbanksystem wissen kann. Manchmal ist es nett, ein gebundenes Buch zu haben, dass man im Bett oder auf Reisen lesen kann. Wenn Sie ein Buch über die angegebenen Hyperlinks kaufen, tragen Sie zur Entwicklung der MySQL-Software bei.
- Entwicklung Dieses Portal hat Links auf Seiten, die den MySQL-Server für unterschiedliche Zwecke benutzen, mit einer Beschreibung jeder Site. Diese Informationen können Ihnen eine gute Vorstellung davon geben, wer MySQL-

Datenbank-Software benutzt und wie der MySQL-Server ihre Anforderungen erfüllt. Teile Sie uns auch *Ihre* Site oder Erfolgsgeschichte mit!

- Software Hier finden Sie eine Vielzahl von Applikationen und Wrappern, die den MySQL-Server benutzen, die Sie auch herunter laden können.
- Distributionen Hier finden Sie die verschiedenen Linux-Distributionen und weitere Software-Pakete, die die MySQL-Software enthalten.
- Berater Hier finden Sie Informationen über MySQL-Berater.
- Partner Hier finden Sie alle MySQL-Partner.

2.6.2. MySQL-Mailing-Listen

Dieser Abschnitt führt Sie in die MySQL-Mailing-Listen ein und zeigt einige Richtlinien und ihre Benutzung auf.

2.6.2.1. Die MySQL-Mailing-Listen

Um die MySQL-Haupt-Mailing-Liste zu abonnieren, schicken Sie eine Nachricht an die E-Mail-Adresse `<mysql-subscribe@lists.mysql.com>`.

Um sich aus der MySQL-Haupt-Mailing-Liste auszutragen, schicken Sie eine Nachricht an die E-Mail-Adresse `<mysql-unsubscribe@lists.mysql.com>`.

Von Bedeutung ist nur die Adresse, unter der Sie Ihre Nachrichten abschicken. Betreffzeile und Text der Nachricht werden ignoriert.

Wenn Ihre Antwortadresse nicht gültig ist, können Sie Ihre Adresse explizit angeben. Fügen Sie einen Bindestrich zum Abonnement- oder Abmelde-Kommando hinzu, gefolgt von Ihrer Adresse, wobei das '`@`'-Zeichen in Ihrer Adresse durch '`=`' ersetzt wird. Um sich zum Beispiel mit `your_name@host.domain` einzutragen, schicken Sie eine Nachricht an `mysql-subscribe-your_name=host.domain@lists.mysql.com`.

Mails an `<mysql-subscribe@lists.mysql.com>` oder `<mysql-unsubscribe@lists.mysql.com>` werden automatisch vom ezmlm Mailing-Listen-Prozessor bearbeitet. Informationen über ezmlm sind auf The ezmlm Website verfügbar.

Um eine Nachricht an die Liste selbst zu schicken, schicken Sie eine Mail an `mysql@lists.mysql.com`. Schicken aber bitte *keine* Mail an `<mysql@lists.mysql.com>`, die das Abonnieren oder Austragen betrifft, denn Mails an diese Adresse werden automatisch an tausende anderer Benutzer verteilt.

Wenn Ihre lokale Site viele Abonnenten für `<mysql@lists.mysql.com>` hat, sollten Sie evtl. eine lokale Mailing-Liste einrichten, so dass Nachrichten, die von `lists.mysql.com` an Ihre Site gesandt werden, an die lokale Liste verteilt werden. In solchen Fällen wenden Sie sich bitte an Ihre Systemadministrator, um zur lokalen Mailing-Liste hinzugefügt oder aus ihr gelöscht zu werden.

Wenn Sie wollen, dass der Traffic einer Mailing-Liste in eine separate Mailbox Ihres E-Mail-Programms geleitet wird, setzen Sie einen Filter, der auf die E-Mail-Header (Kopfdaten) reagiert. Sie können dazu entweder den `List-ID:`- oder den `Delivered-To:`-Header benutzen, um die Listennachrichten zu erkennen.

Die folgenden MySQL-Mailing-Listen existieren:

- `<announce-subscribe@lists.mysql.com> announce`

  Diese Liste kündigt neue Versionen von MySQL und verwandter Programme an. Sie hat geringen Traffic; alle MySQL-Benutzer sollten sie abonnieren.

- `<mysql-subscribe@lists.mysql.com> mysql`

  Die Hauptliste für allgemeine MySQL-Diskussionen. Bitte beachten Sie, dass bestimmte Themen besser in spezialisierteren Listen diskutiert werden. Wenn Sie an die falsche Liste posten, erhalten Sie vielleicht keine Antwort!

- `<mysql-digest-subscribe@lists.mysql.com> mysql-digest`

  Die `mysql`-Liste in Digest-Form (zusammengefasst). Anstelle individueller Nachrichten wird einmal pro Tag eine große Mail mit allen Nachrichten dieses Tages geschickt.

- `<bugs-subscribe@lists.mysql.com> bugs`

An diese Liste sollte Sie ausschließlich komplette, wiederholbare Bug-Berichte schicken, indem Sie das `mysqlbug`-Skript benutzen. (Wenn Sie unter Windows arbeiten, sollten Sie eine Beschreibung des Betriebssystems und der MySQL-Version hinzufügen.) Vorzugsweise sollten Sie den Problemfall mit der letzten stabilen oder Entwicklungs-Version von MySQL testen, bevor Sie den Bericht posten! Jeder sollte in der Lage sein, den Bug zu wiederholen, indem einfach `mysql test < Skript` auf den beigefügten Testfall angewandt wird. Alle Bugs, die auf dieser Liste gepostet werden, werden im nächsten MySQL-Release behoben oder dokumentiert! Wenn nur kleinere Code-Änderungen betroffen sind, werden wir zusätzlich ein Patch bereitstellen, das das Problem behebt.

- `<bugs-digest-subscribe@lists.mysql.com> bugs-digest`

  Die Digest-Version (zusammengefasst) der `bugs`-Liste.

- `<internals-subscribe@lists.mysql.com> internals`

  Eine Liste für Leute, die am MySQL-Code arbeiten. Auf dieser Liste kann man auch die MySQL-Entwicklung diskutieren und Patches posten.

- `<internals-digest-subscribe@lists.mysql.com> internals-digest`

  Die Digest-Version (zusammengefasst) der `internals`-Liste.

- `<java-subscribe@lists.mysql.com> java`

  Diskussionen über MySQL und Java, hauptsächlich über JDBC-Treiber.

- `<java-digest-subscribe@lists.mysql.com> java-digest`

  Eine `java`-Liste.

- `<win32-subscribe@lists.mysql.com> win32`

  Alles betreffend MySQL auf Microsoft-Betriebssystemen wie Win95, Win98, NT, XP, und Win2000.

- `<win32-digest-subscribe@lists.mysql.com> win32-digest`

  Die Digest-Version (zusammengefasst) der `win32`-Liste.

- `<myodbc-subscribe@lists.mysql.com> myodbc`

  Alles betreffend ODBC-Verbindungen zu MySQL.

- `<myodbc-digest-subscribe@lists.mysql.com> myodbc-digest`

  Die Digest-Version (zusammengefasst) der `myodbc`-Liste.

- `<plusplus-subscribe@lists.mysql.com> plusplus`

  Alles, was das Programmieren mit der C++-API von MySQL betrifft.

- `<plusplus-digest-subscribe@lists.mysql.com> plusplus-digest`

  Die Digest-Version (zusammengefasst) der `plusplus`-Liste.

- `<msql-mysql-modules-subscribe@lists.mysql.com> msql-mysql-modules`

  Eine Liste zur Perl-Unterstützung in MySQL. msql-mysql-modules

- `<msql-mysql-modules-digest-subscribe@lists.mysql.com> msql-mysql-modules-digest`

  Die Digest-Version (zusammengefasst) der `msql-mysql-modules`-Liste.

Alle Listen abonnieren Sie - und tragen sich wieder aus - auf dieselbe Art wie oben beschrieben. Tragen Sie in Ihre Mail zum Abonnieren oder Austragen die entsprechende Mailing-Liste ein anstelle von `mysql`. Um sich zum Beispiel für die `myodbc`-Liste einzutragen, schicken Sie eine Nachricht an `<myodbc-subscribe@lists.mysql.com>` oder `<myodbc-unsubscribe@lists.mysql.com>`.

Wenn Sie keine Antwort auf Ihre Fragen von der Mailing-Liste erhalten, ist eine Option, für den Support von MySQL AB zu bezahlen, was Sie in direkten Kontakt mit den MySQL-Entwicklern bringt. See Abschnitt 2.4.1, „Support den MySQL AB anbietet“.

Die folgende Tabelle listet einige Mailing-Listen in anderen Sprachen als englisch auf. Beachten Sie, dass diese nicht von MySQL AB unterhalten werden. Daher können wir nicht für die Qualität dieser Listen garantieren.

- `<mysql-france-subscribe@yahoogroups.com> Eine französische Mailing-Liste`, `<list@tinc.net> Eine koreanische Mailing-Liste`

  Schicken Sie eine E-Mail mit dem Betreff `subscribe mysql your@email.address` an diese Liste.

- `<mysql-de-request@lists.4t2.com> Eine deutsche Mailing-Liste`

  Schicken Sie eine E-Mail mit dem Betreff `subscribe mysql-de your@email.address` an diese Liste. Informationen über diese Liste finden Sie unter http://www.4t2.com/mysql.

- `<mysql-br-request@listas.linkway.com.br> Eine portugiesische Mailing-Liste.`

  Schicken Sie eine E-Mail mit dem Betreff `subscribe mysql-br your@email.address` an diese Liste.

- `<mysql-alta@elistas.net> Eine spanische Mailing-Liste.`

  Schicken Sie eine E-Mail mit dem Betreff `subscribe mysql your@email.address` an diese Liste.

2.6.2.2. Wie man Fragen stellt oder Bugs berichtet

Bevor Sie einen Bug berichten oder eine Frage stellen, tun Sie bitte folgendes:

- Suchen Sie im MySQL-Online-Handbuch: http://www.mysql.com/documentation/manual.php Wir bemühen uns, das Handbuch aktuell zu halten, indem wir es häufig mit Lösungen für neu bekannt gewordene Probleme aktualisieren!
- Durchsuchen Sie die MySQL-Mailing-Listen-Archive: http://www.mysql.com/documentation/
- Sie können ausserdem http://www.mysql.com/search.html benutzen, um alle Webseiten zu durchsuchen (inklusive des Handbuchs), die unter http://www.mysql.com/ zu finden sind.

Wenn Sie weder im Handbuch noch in den Archiven eine Antwort finden können, versuchen Sie es mit Ihrem lokalen MySQL-Experten. Wenn Sie immer noch keine Antwort auf Ihre Frage finden, lesen Sie den nächsten Abschnitt über die Mailing-Listen unter `<mysql@lists.mysql.com>`.

2.6.2.3. Wie man Bugs oder Probleme berichtet

Einen guten Bug-Bericht zu schreiben braucht Geduld, aber es gleich beim ersten Mal richtig zu machen spart Ihnen und uns Zeit. Ein guter Bug-Bericht enthält einen kompletten Testfall für den Bug, der es sehr wahrscheinlich macht, dass wir ihn im nächsten Release beheben. Dieser Abschnitt hilft Ihnen, Ihren Bericht korrekt zu schreiben, damit Sie Ihre Zeit nicht damit verschwenden, etwas zu schreiben, was uns wenig oder gar nicht weiterhilft.

Wir ermutigen jeden, das `mysqlbug`-Skript zu benutzen, um einen Bug-Bericht anzufertigen (oder einen Bericht über irgendein anderes Problem), falls das möglich ist. Der `mysqlbug` findet sich im `Skripts`-Verzeichnis der Quelldistribution, bzw. im `bin`-Verzeichnis der Binärdistribution, im Verzeichnis unterhalb Ihres MySQL-Installationsverzeichnisses. Falls es Ihnen nicht möglich ist, `mysqlbug` zu benutzen, sollten Sie trotzdem alle notwendigen Informationen mitliefern, die in diesem Abschnitt aufgeführt sind.

Das `mysqlbug`-Skript hilft Ihnen, einen Bericht zu erstellen, der viele der folgenden Informationen automatisch einschließt, aber falls etwas Wichtiges fehlt, fügen Sie es bitte Ihrer Nachricht hinzu! Bitte lesen Sie diesen Abschnitt sorgfältig und stellen Sie sicher, dass alle hier beschriebenen Informationen in Ihrem Bericht enthalten sind.

Für gewöhnlich sollten Sie Ihren Bug-Bericht und Probleme an `<mysql@lists.mysql.com>` schicken. Wenn Sie einen Testfall erzeugen können, der den Bug klar demonstriert, sollten Sie ihn an die `<bugs@lists.mysql.com>`-Liste schicken. Beachten Sie, dass Sie nur einen kompletten, nachvollziehbaren Bug-Bericht an diese Liste schicken sollten, indem Sie das `mysqlbug`-Skript benutzen. Falls Sie unter Windows arbeiten, sollten Sie eine Beschreibung des Betriebssystems und der MySQL-Version hinzufügen. Vorzugsweise sollten Sie den Problemfall mit der letzten stabilen oder Entwicklungs-Version von MySQL testen, bevor Sie den Bericht posten! Jeder sollte in der Lage sein, den Bug zu wiederholen, indem einfach `mysqltest < Skript` auf den beigefügten Testfall angewandt wird. Alle Bugs, die auf dieser Liste gepostet werden, werden im nächsten MySQL-Release behoben oder dokumentiert! Wenn nur kleinere Code-Änderungen betroffen sind, werden wir zusätzlich ein Patch bereitstellen, das das Problem behebt.

Denken Sie daran, dass es immer möglich ist, auf eine Nachricht zu antworten, die zu viele Informationen enthält, aber nicht immer auf eine, die zu wenige Informationen enthält. Oft lassen Leute Fakten aus, weil sie denken, die Ursache eines Probleme zu kennen

und annehmen, dass einige Details nicht von Wichtigkeit sind. Ein gutes Prinzip ist folgendes: Falls Sie im Zweifel sind, ob Sie etwas Bestimmtes mitteilen sollten, teilen Sie es mit! Es ist tausendmal schneller und weniger ärgerlich, ein paar Zeilen mehr in Ihrem Bericht zu schreiben, als gezwungen zu sein, noch einmal zu fragen und auf die Antwort zu warten, weil Sie beim ersten Mal nicht genug Informationen geliefert haben.

Die häufigste Fehler besteht darin, dass Leute die Versionsnummer der MySQL-Distribution, die sie benutzen nicht angeben, oder vergessen anzugeben, auf welcher Plattform sie MySQL installiert haben (inklusive der Betriebssystem-Version). Diese Informationen sind äußerst relevant, und in 99 von 100 Fällen ist der Bug-Bericht ohne sie nutzlos! Sehr oft erhalten wir Fragen wie 'Warum funktioniert das bei mir nicht?', nur um herauszufinden, dass das beschriebene Feature nicht in der benutzten MySQL-Version implementiert war, oder dass der Bug, der im Bericht beschrieben wurde, bereits in einer neueren MySQL-Version behoben wurde. Manchmal ist der Fehler plattformabhängig; in solchen Fällen ist es praktisch unmöglich, irgend etwas zu beheben, ohne das Betriebssystem und die Versionsnummer des Betriebssystems zu kennen.

Denken Sie auch daran, Informationen über Ihren Compiler einzuschließen, falls sie MySQL selbst kompilieren. Oft finden Leute Fehler in Compilern und denken, dass das Problem MySQL-bezogen ist. Die meisten Compiler werden permanent weiter entwickelt und werden von Version zu Version besser. Um festzustellen, ob ein Problem von Ihrem Compiler abhängt oder nicht, müssen wir wissen, welcher Compiler benutzt wird. Beachten Sie, dass jedes Compiler-Problem als Bug-Bericht betrachtet und deshalb entsprechend berichtet werden sollte.

Es ist äußerst hilfreich, wenn eine gute Beschreibung des Probleme in Ihrem Bug-Bericht eingeschlossen ist, das heißt ein gutes Beispiel aller Dinge, die Sie getan haben, die zu dem Problem führten, sowie das Problem selbst. Die besten Bug-Berichte sind diejenigen, die ein komplettes Beispiel zeigen, wie man den Bug oder das Problem reproduzieren kann. See Abschnitt E.1.6, „Einen Testfall herstellen, wenn Sie Tabellenbeschädigung feststellen“.

Wenn ein Programm eine Fehlermeldung produziert, ist es sehr wichtig, diese in Ihren Bericht einzuschließen! Wenn wir in den Archiven der Programme suchen, ist es besser, wenn die Fehlernachricht exakt mit derjenigen übereinstimmt, die das Programm produziert. (Sogar Groß-/Kleinschreibung sollte berücksichtigt werden!) Sie sollten nie versuchen, sich daran zu erinnern, was die Fehlernachricht war; stattdessen sollten Sie die gesamte Nachricht per Kopieren und Einfügen in Ihrem Bericht unterbringen!

Wenn Sie ein Problem mit MyODBC haben, sollten Sie versuchen, eine MyODBC-Trace-Datei zu erstellen. See Abschnitt 9.3.7, „Probleme mit MyODBC berichten“.

Bitten denken Sie daran, dass viele Leute, die Ihren Bericht lesen, dabei ein 80-Spalten-Anzeigegerät benutzen. Wenn Sie Berichte oder Beispiele erzeugen, indem Sie das `mysql`-Kommandozeilen-Werkzeug benutzen, sollten Sie deshalb die `--vertical`-Option (oder den `\G`-Statement-Begrenzer) für Ausgaben benutzen, die ansonsten die verfügbare Anzeigebreite überschreiten würden (zum Beispiel beim `EXPLAIN SELECT`-Statement; siehe dazu das Beispiel weiter unten). Bitte schließen Sie folgende Informationen in Ihren Bug-Bericht ein:

- Die Versionsnummer der MySQL-Distribution, die Sie benutzen (zum Beispiel MySQL Version 3.23.22). Sie finden heraus, welche Version Sie benutzen, indem Sie `mysqladmin version` eingeben. `mysqladmin` findet sich im `bin`-Verzeichnis unterhalb Ihres MySQL-Installationsverzeichnisses.
- Hersteller und Modell der Maschine, auf der Sie arbeiten.
- Name und Version des Betriebssystems. Bei den meisten Betriebssystemen läßt sich diese Information herausfinden, indem man das Unix-Kommando `uname -a` ausführt.
- Manchmal ist die Größe des Arbeitsspeichers (real und virtuell) relevant. Im Zweifelsfall schließen Sie diese Werte ein.
- Wenn Sie eine Quelldistribution von MySQL benutzen, werden Name und Versionsnummer des Compilers benötigt. Wenn Sie eine Binärdistribution haben, geben Sie den Namen der Distribution an.
- Wenn das Problem während der Kompilation auftritt, schließen Sie die exakte Fehlermeldung (bzw. -meldungen) ein und zusätzlich ein paar Zeilen des Kontextes um den problembehafteten Code herum in der Datei, wo der Fehler auftrat.
- Falls `mysqld` abstürzt, sollten Sie auch die Anfrage (Query) mitteilen, die `mysqld` zum Absturz brachte. Gewöhnlich können Sie das herausfinden, indem Sie `mysqld` mit angeschaltetem Logging laufen lassen. See Abschnitt E.1.5, „Log-Dateien benutzen, um Gründe für Fehler in mysqld zu finden“.
- Falls irgend eine Datenbanktabelle mit dem Problem zu tun hat, schließen Sie die Ausgabe von `mysqldump --no-data db_name tbl_name1 tbl_name2 ...` ein. Das ist sehr leicht zu bewerkstelligen und eine sehr hilfreiche Möglichkeit, Informationen über jegliche Tabelle in einer Datenbank zu erhalten, die uns hilft, eine Situation herzustellen, die mit derjenigen übereinstimmt, die Sie haben.
- Bei Bugs, die sich auf Geschwindigkeitsprobleme beziehen, oder bei Problemen mit `SELECT`-Statements, sollten Sie immer die Ausgabe von `EXPLAIN SELECT ...` einschließen, und zumindest die Anzahl der Zeilen, die das `SELECT`-Statement produziert. Je mehr Informationen Sie uns über Ihre Situation geben, desto wahrscheinlicher ist es, dass Ihnen jemand helfen kann! Das folgende Beispiel ist ein sehr gutes Beispiel eines Bug-Berichts (es sollte natürlich mit dem `mysqlbug`-Skript berichtet werden):

Beispiel unter Benutzung des `mysql`-Kommandozeilen-Werkzeugs (achten Sie auf die Benutzung des `\G`-Statement-Begrenzers für Statements, deren Ausgabebreite ansonsten die von 80-Zeilen-Ausgabegeräten überschreiten würde):

```
mysql> SHOW VARIABLES;
mysql> SHOW COLUMNS FROM ...\G
       <Ausgabe von SHOW COLUMNS>
mysql> EXPLAIN SELECT ...\G
       <Ausgabe von EXPLAIN>
mysql> FLUSH STATUS;
mysql> SELECT ...;
       <Eine Kurzfassung der Ausgabe von SELECT,
       inclusive der Zeit, die die Anfrage benötigte>
mysql> SHOW STATUS;
       <Ausgabe von SHOW STATUS>
```

- Wenn ein Problem auftritt, während **mysqld** läuft, legen Sie nach Möglichkeit ein Eingabeskript bei, das die Anomalie reproduziert. Dieses Skript sollte alle notwendigen Quelltextdateien beinhalten. Je exakter das Skript Ihre Situation reproduzieren kann, desto besser. Wenn Sie einen wiederholbaren Testfall erstellen können, sollten Sie ihn an `<bugs@lists.mysql.com>` schicken, damit er mit hoher Priorität behandelt wird!

  Falls Sie kein Skript zur Verfügung stellen können, sollten Sie zumindest die Ausgaben von `mysqladmin variables extended-status processlist` in Ihrer Mail mitschicken, um einige Informationen darüber zu geben, wie Ihr System arbeitet!

- Falls Sie keinen Testfall mit ein paar Zeilen produzieren können oder falls Ihre Tabelle zu Groß ist, um an die Mailing-Liste geschickt zu werden (mehr als 10 Zeilen), sollten Sie mit `mysqldump` einen Dump Ihrer Tabellen machen und eine `README`-Datei erzeugen, die Ihr Problem beschreibt.

  Erzeugen Sie ein komprimiertes Archiv Ihrer Dateien, indem Sie `tar` und `gzip` oder `zip` benutzen, und benutzen Sie `ftp`, um das Archiv nach ftp://Support.mysql.com/pub/mysql/secret/ zu transferieren. Schicken Sie danach eine kurze Beschreibung des Probleme an `<bugs@lists.mysql.com>`.

- Wenn Sie glauben, dass MySQL auf eine Anfrage hin merkwürdige Ergebnisse liefert, fügen Sie nicht nur das Ergebnis bei, sondern auch, wie das Ergebnis Ihrer Meinung nach aussehen sollte, sowie eine Erklärung, wie Sie zu dieser Meinung gelangt sind.

- Wenn Sie ein Beispiel Ihres Problems schildern, ist es besser, die Variablen-, Tabellen- etc. Namen zu verwenden, die in Ihrer aktuellen Situation existieren, anstatt sich neue Namen auszudenken. Das Problem könnte nämlich etwas mit dem Namen der Variablen oder Tabelle zu tun haben! Diese Fälle sind zwar selten, aber hier sollte man lieber auf Nummer sicher gehen. Letztlich sollte es für Sie auch leichter sein, ein Beispiel zur Verfügung zu stellen, das Ihre tatsächliche Situation schildert, und es ist in jedem Fall besser für uns. Falls Sie mit Daten arbeiten, die Sie keinen anderen zeigen wollen, können Sie `ftp` benutzen, um die Daten nach ftp://Support.mysql.com/pub/mysql/secret/ zu transferieren. Falls die Daten streng geheim sind und Sie sie nicht einmal uns zeigen wollen, legen Sie bitte ein Beispiel mit anderen Namen an, betrachten Sie dies aber bitte als allerletzte Möglichkeit.

- Fügen Sie alle Optionen ein, die den relevanten Programmen übergeben wurden, falls möglich. Geben Sie zum Beispiel die Optionen an, die Sie benutzt haben, als Sie den `mysqld`-Daemon gestartet haben, und die Sie für Client-Programme wie `mysql` benutzen, sowie diejenigen, die Sie für die Konfiguration des `configure`-Skripts nehmen, denn diesen sind oft der Schlüssel für Antworten und deshalb äußerst relevant! Es ist immer eine gute Idee, sie in jedem Fall anzugeben! Wenn Sie Module wie Perl oder PHP benutzen, fügen Sie bitte die Versionszahl von diesen mit ein.

- Wenn sich Ihre Frage auf das Berechtigungssystem (Zugriffsberechtigungen auf den Datenbank-Server) bezieht, fügen Sie bitte die Ausgabe von `mysqlaccess`, die Ausgabe von `mysqladmin reload` und alle Fehlermeldungen, die Sie erhalten, wenn Sie versuchen, sich zu verbinden, bei! Wenn Sie Ihre Zugriffsberechtigungen testen, sollten Sie zunächst `mysqlaccess` ausführen. Führen Sie danach `mysqladmin reload version` aus und versuchen Sie dann, sich mit dem Programm zu verbinden, das Probleme macht. `mysqlaccess` liegt im `bin`-Verzeichnis unter Ihrem MySQL-Installationsverzeichnis. MySQL-Installationsverzeichnis.

- Wenn Sie einen Patch für ein Bug haben, ist das gut, aber nehmen Sie bitte nicht an, dass der Patch alles ist, was wir brauchen. Gehen Sie auch nicht davon aus, dass wir den Patch benutzen werden, wenn Sie nicht auch einige notwendige Informationen mitschicken, zum Beispiel Testfälle, die den Bug zeigen, der durch Ihren Patch behoben wird. Möglicherweise finden wir Probleme, die Ihr Patch verursacht, oder wir verstehen ihn überhaupt nicht. Wenn das der Fall ist, können wir ihn nicht benutzen.

  Wenn wir nicht genau feststellen können, wofür der Patch gedacht ist, werden wir ihn nicht benutzen. In diesen Fällen werden uns Testfälle weiter helfen. Zeigen Sie darin auf, dass der Patch all die Situationen bewältigt, die eintreten können. Falls wir einen Grenzfall finden (sogar, wenn es ein seltener ist), bei dem der Patch nicht funktioniert, ist er vielleicht nutzlos.

- Vermutungen, worin der Bug besteht, warum er auftritt oder wovon er abhängt, sind meist falsch. Selbst das MySQL-Team kann solche Dinge nicht erraten, sondern muss einen Debugger benutzen, um den wahren Grund des Bugs feststellen zu können.

- Geben Sie in Ihrer Mail zu erkennen, dass Sie das Referenzhandbuch gelesen und die Mail-Archive durchgesehen haben, damit andere wissen, dass Sie versucht haben, das Problem selbst zu lösen.

- Wenn Sie einen `parse error` erhalten, überprüfen Sie bitte genau Ihre Syntax! Wenn Sie nichts Falsches darin finden können, ist es sehr wahrscheinlich, dass Ihre aktuelle Version von MySQL die Anfrage, die Sie formuliert haben, nicht unterstützt. Wenn Sie die aktuelle Version benutzen und das Handbuch unter http://www.mysql.com/documentation/manual.php die Syntax, die Sie benutzen, nicht beschreibt, unterstützt MySQL Ihre Anfrage nicht. In diesem Fall bleibt Ihnen nur, die Syntax entweder selbst zu implementieren oder per E-Mail an `<mysql-licensing@mysql.com>` nach einem Angebot für die Implementation anzufragen!

  Wenn das Handbuch die Syntax, die Sie benutzen, beschreibt, Sie aber eine ältere Version von MySQL benutzen, sollten Sie in der MySQL-Änderungsgeschichte (Change History) nachsehen, wann die Syntax implementiert wurde. In diesem Fall haben Sie die Möglichkeit, ein Upgrade auf eine neuere Version von MySQL vorzunehmen. See Anhang D, *MySQL-Änderungsverlauf (Change History)*.

- Wenn Sie ein Problem in der Art haben, dass Ihre Daten anscheinend beschädigt sind oder Sie Fehlermeldungen bekommen, wenn Sie auf eine bestimmte Tabelle zugreifen, sollten Sie zunächst Ihre Tabellen überprüfen und anschließend reparieren, indem Sie `myisamchk` oder `CHECK TABLE` und `REPAIR TABLE` benutzen. See Kapitel 5, *MySQL-Datenbankadministration*.

- Wenn Sie oft beschädigte Tabellen erhalten, sollten Sie versuchen herauszufinden, wann und warum das geschieht! In diesem Fall kann die `mysql-data-directory/'hostname'.err`-Datei einige Informationen darüber enthalten, was geschehen ist. See Abschnitt 5.9.1, „Die Fehler-Log-Datei“. Bitte fügen Sie jede relevante Information aus dieser Datei in Ihren Bug-Bericht ein! Normalerweise sollte `mysqld` **NIE** eine Tabelle zerstören, ausser wenn der Server mitten während eines Updates gekillt wurde! Wenn Sie den Grund für den Absturz von `mysqld` herausfinden können, ist es sehr viel einfacher für uns, Ihnen eine Lösung des Problems an die Hand zu geben! See Abschnitt A.1, „Wie man feststellt, was Probleme verursacht“.

- Falls möglich, sollten Sie die aktuellste Version von MySQL herunter laden, installieren und überprüfen, ob das Ihr Problem löst. Alle Versionen von MySQL werden gründlich getestet und sollten ohne Probleme funktionieren! Wir halten uns daran, alles so abwärtskompatibel wie möglich zu machen. Daher sollte es Ihnen möglich sein, innerhalb von Minuten die MySQL-Version auszutauschen! See Abschnitt 3.2.3, „Welche MySQL-Version Sie benutzen sollten“.

Wenn Sie ein Support-Kunde sind, schicken Sie bitte den Bug-Bericht an `<mysql-Support@mysql.com>`, damit dieser eine höhere Priorität in der Bearbeitung erfährt. Schicken Sie ihn gleichzeitig an die entsprechende Mailing-Liste, um zu sehen, ob schon jemand anderes das selbe Problem hatte (und vielleicht gelöst hat).

Informationen zu Bug-Berichten siehe **MyODBC** und Abschnitt 9.3.4, „Wie Sie Probleme mit MyODBC berichten“.

Lösungen für häufig auftretende Probleme siehe See Anhang A, *Probleme und häufige Fehler*.

Wenn Ihnen Antworten individuell zugesandt werden und nicht an die Mailing-Liste, wird es als gute Etikette betrachtet, die Antworten zusammenzufassen und die Zusammenfassung an die Mailing-Liste zu schicken, damit andere von den Antworten profitieren können, die Ihnen geholfen haben, Ihr Problem zu lösen!

2.6.2.4. Richtlinien für die Beantwortung von Fragen auf der Mailing-Liste

Wenn Sie davon ausgehen, dass Ihre Antwort auf breites Interesse stößt, sollten Sie an die Mailing-Liste posten, statt direkt der Person zu antworten, die die Frage stellte. Versuchen Sie, Ihre Antwort so allgemein zu halten, dass auch andere als der ursprünglich Fragende von Ihrer Antwort profitieren können. Wenn Sie an die Liste posten, stellen Sie bitte sicher, dass Ihre Antwort kein Duplikat einer vorhergehenden Antwort ist.

Versuchen Sie, den wesentlichen Teil der Frage in Ihrer Antwort zusammenzufassen. Fühlen Sie sich nicht verpflichtet, die gesamte ursprüngliche Nachricht zu zitieren.

Bitte schicken Sie Ihre Mailnachrichten nicht im HTML-Format! Viele Benutzer lesen Nachrichten mit nicht HTML-fähigen Anwendungen!

2.7. Wie Standard-kompatibel ist MySQL?

Dieser Abschnitt beschreibt, wie sich MySQL zum ANSI SQL-Standard verhält. MySQL hat viele Erweiterungen zum ANSI SQL-Standard, und hier steht, welche das sind und wie man sie benutzt. Hier finden Sie auch Informationen über Funktionalität, die MySQL fehlt, und wie man mit diesen Unterschieden umgeht.

2.7.1. An welche Standards hält sich MySQL?

Entry-Level-SQL92. ODBC-Levels 0-2.

Wir beabsichtigen ANSI SQL99 vollständig zu unterstützen. Dies wollen wir jedoch keinesfalls auf Kosten von Geschwindigkeit

oder Codequalität erreichen.

2.7.2. MySQL im ANSI-Modus laufen lassen

Wenn Sie `mysqld` mit der `--ansi`-Option starten, ändert sich folgendes Verhalten von MySQL:

- `||` ist Zeichenketten-Verkettung (Konkatenation) anstelle von `OR`.
- Sie können eine beliebige Anzahl von Leerzeichen zwischen Funktionnamen und '`(`' eingeben. Das führt zwangsläufig dazu, dass alle Funktionsnamen als reservierte Wörter behandelt werden.
- '`"`' ist dann ein Quotierungsbezeichner (wie das MySQL- '`` ` ``'-Anführungszeichen) und kein Zeichen, dass einen String einschließt.
- `REAL` wird zu einem Synonym für `FLOAT` anstelle eines Synonyms für `DOUBLE`.
- Der Standard-Isolationslevel für Transaktionen ist `SERIALIZABLE`. See Abschnitt 7.7.3, „`SET TRANSACTION`-Syntax“.

Das ist dasselbe, als würde man `--sql-mode=REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,SERIALIZE,ONLY_FULL_GROUP_BY` benutzen.

2.7.3. MySQL-Erweiterungen zu ANSI SQL92

MySQL beinhaltet einige Erweiterungen, die Sie in anderen SQL-Datenbanken wahrscheinlich nicht finden werden. Passen Sie auf, wenn Sie diese benutzen, denn Ihr Code ist dann nicht mehr kompatibel mit anderen SQL-Servern. In einigen Fällen können Sie Code schreiben, der MySQL-Erweiterungen enthält und dennoch portabel ist, indem Sie Kommentare in der Form `/*! ... */` benutzen. In diesem Fall wird MySQL den Code innerhalb des Kommentars parsen und ausführen wie jedes andere MySQL-Statement, aber andere SQL-Server werden die Erweiterungen ignorieren. Zum Beispiel:

```
SELECT /*! STRAIGHT_JOIN */ col_name FROM tabelle1, tabelle2 WHERE ...
```

Wenn Sie hinter `'!'` die Versionsnummer angeben, wird die Syntax nur ausgeführt, wenn die MySQL-Version gleich oder neuer als die benutzte Versionsnummer ist:

```
CREATE /*!32302 TEMPORARY */ TABLE (a int);
```

Je höher bedeutet, wenn Sie Version 3.23.02 oder neuer haben, wird MySQL das `TEMPORARY`-Schlüsselwort benutzen.

MySQL-Erweiterungen sind:

- Die Feldtypen `MEDIUMINT`, `SET`, `ENUM` und die unterschiedlichen `BLOB`- und `TEXT`-Typen.
- Die Feldattribute `AUTO_INCREMENT`, `BINARY`, `NULL`, `UNSIGNED` und `ZEROFILL`.
- Alle Zeichenkettenvergleiche achten vorgabemäßig nicht auf Groß-/Kleinschreibung, wobei die Sortierreihenfolge vom aktuell verwendeten Zeichensatz abhängig ist (ISO-8859-1 Latin1 als Vorgabe). Wenn Sie das nicht wollen, sollten Sie Ihre Spalten mit dem `BINARY`-Attribut deklarieren oder den `BINARY`-Cast benutzen, der dafür sorgt, dass Vergleiche mit der ASCII-Sortierung durchgeführt werden, die der MySQL-Server-Host benutzt.
- MySQL legt jede Datenbank als Verzeichnis unterhalb des MySQL-data- Verzeichnisses an und Tabellen innerhalb einer Datenbank als Dateien in dem Datenbank-Verzeichnis.

  Das hat ein paar Auswirkungen:

  - Bei Datenbanknamen und Tabellennamen wird auf Unterschiede in der Groß-/Kleinschreibung geachtet, wenn das Betriebssystem auf Groß-/Kleinschreibung achtet (wie auf den meisten Unix-Systemen). See Abschnitt A.5.1, „Groß-/Kleinschreibung beim Suchen“.
  - Datenbank-, Tabellen-, Index-, Spalten- oder Alias-Namen dürfen mit einer Ziffer beginnen (aber nicht ausschließlich aus Ziffern bestehen).
  - Sie können Standard-Kommandos des Betriebssystems benutzen, um Tabellen zu sichern (Datensicherung), umzubenennen, zu verschieben, zu löschen und zu kopieren. Um zum Beispiel eine Tabelle umzubenennen, benennen Sie die Dateien `.MYD`,

`.MYI` und `.frm` um, die der Tabelle entsprechen.

- In SQL-Statements können Sie Tabellen aus verschiedenen Datenbanken mit der `db_name.tbl_name`-Syntax ansprechen. Einige SQL-Syntax stellen dieselbe Funktionalität zur Verfügung, nennen dies aber `User space`. MySQL unterstützt keine Tablespaces wie in folgendem Beispiel: `create tabelle ralph.meine_tabelle...IN mein_tablespace`.
- `LIKE` ist für numerische Spalten erlaubt.
- Die Benutzung von `INTO OUTFILE` und `STRAIGHT_JOIN` in einem `SELECT`-Statement. See Abschnitt 7.4.1, „`SELECT`-Syntax“.
- Die Option `SQL_SMALL_RESULT` in einem `SELECT`-Statement.
- `EXPLAIN SELECT`, um eine Beschreibung zu erhalten, wie Tabellen verknüpft werden (Join).
- Die Benutzung von Index-Namen, Indexen auf ein Prefix eines Feldes, und die Benutzung von `INDEX` oder `KEY` in einem `CREATE TABLE`- Statement. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.
- Die Benutzung von `TEMPORARY` oder `IF NOT EXISTS` mit `CREATE TABLE`.
- Die Benutzung von `COUNT(DISTINCT list)`, wobei 'list' mehr als ein Element ist.
- Die Benutzung von `CHANGE spalten_name`, `DROP spalten_name`, oder `DROP INDEX`, `IGNORE` oder `RENAME` in einem `ALTER TABLE`-Statement. See Abschnitt 7.5.4, „`ALTER TABLE`-Syntax“.
- Die Benutzung von `RENAME TABLE`. See Abschnitt 7.5.5, „`RENAME TABLE`-Syntax“.
- Die Benutzung mehrfacher `ADD`-, `ALTER`-, `DROP`-, oder `CHANGE`-Klauseln in einem `ALTER TABLE` Statement.
- Die Benutzung von `DROP TABLE` mit the keywords `IF EXISTS`.
- Sie können mehrere Tabellen löschen mit einem einzigen `DROP TABLE`-Statement.
- Die `LIMIT`-Klausel des `DELETE`-Statements.
- Die `DELAYED`-Klausel der `INSERT`- und `REPLACE`- Statements.
- Die `LOW_PRIORITY`-Klausel der `INSERT`-, `REPLACE`-, `DELETE`- und `UPDATE`-Statements.
- Die Benutzung von `LOAD DATA INFILE`. In vielen Fällen ist diese Syntax kompatibel mit Oracles `LOAD DATA INFILE`. See Abschnitt 7.4.9, „`LOAD DATA INFILE`-Syntax“.
- Die `ANALYZE TABLE`-, `CHECK TABLE`-, `OPTIMIZE TABLE`- und `REPAIR TABLE`-Statements.
- Das `SHOW`-Statement. See Abschnitt 5.5.5, „`SHOW`-Syntax“.
- Zeichenketten dürfen sowohl durch ‘`"`’ als auch durch ‘`'`’ eingeschlossen werden, nicht nur durch ‘`'`’.
- Die Benutzung des Escape(‘`\`’)Zeichens.
- Das `SET OPTION`-Statement. See Abschnitt 6.5.6, „`SET`-Syntax“.
- Sie müssen nicht alle ausgewählten Spalten im `GROUP BY`-Teil nennen. Hierdurch ergibt sich eine bessere Performance für einige sehr spezifische, aber recht gewöhnliche Anfragen.
- Man kann `ASC` und `DESC` bei `GROUP BY` spezifizieren.
- Um es Benutzern leichter zu machen, die von anderen SQL-Umgebungen kommen, unterstützt MySQL Aliase für viele Funktionen. Zum Beispiel unterstützen alle Zeichenketten-Funktionen sowohl die ANSI-SQL-Syntax als auch die ODBC-Syntax.
- MySQL kennt die Operatoren `||` und `&&`, die logisches Oder und logisches Und bedeuten, wie in der Programmiersprache C. In MySQL sind `||` und `OR` Synonyme, wie auch `&&` und `AND`. Aufgrund dieser freundlichen Syntax unterstützt MySQL nicht den ANSI-SQL-`||`-Operator für Zeichenketten-Verkettung (Konkatenation); benutzen Sie statt dessen `CONCAT()`. Weil `CONCAT()` eine beliebige Anzahl von Argumenten entgegennimmt, ist es leicht, die Benutzung des `||`-Operators zu MySQL zu konvertieren.
- `CREATE DATABASE` oder `DROP DATABASE`. See Abschnitt 7.5.1, „`CREATE DATABASE`-Syntax“.
- Der `%`-Operator ist ein Synonym für `MOD()`. Das heißt `N % M` ist äquivalent zu `MOD(N,M)`. `%` wird für C-Programmierer und für Kompatibilität mit PostgreSQL unterstützt.

- Die =-, <>-, <=-, <-, >=-, >-, <<-, >>-, <=>-, AND-, OR- oder LIKE-Operatoren können in Spaltenvergleichen links von FROM in SELECT Statements benutzt werden. Beispiel:

```
mysql> SELECT spalte1=1 AND spalte2=2 FROM tabelle_name;
```

- Die LAST_INSERT_ID()-Funktion. See Abschnitt 9.4.3.30, „mysql_insert_id()“.
- Die REGEXP- und NOT REGEXP-Operatoren für erweiterte reguläre Ausdrücke.
- CONCAT() oder CHAR() mit einem Argument oder mehr als zwei Argumenten. (In MySQL können diese Funktionen jede beliebige Anzahl von Argumenten entgegennehmen.)
- Die Funktionen BIT_COUNT(), CASE, ELT(), FROM_DAYS(), FORMAT(), IF(), PASSWORD(), ENCRYPT(), md5(), ENCODE(), DECODE(), PERIOD_ADD(), PERIOD_DIFF(), TO_DAYS() oder WEEKDAY().
- Die Benutzung von TRIM(), um Teile von Zeichenketten zu entfernen. ANSI SQL unterstützt nur die Entfernung einzelner Zeichen.
- Die GROUP BY-Funktionen STD(), BIT_OR() und BIT_AND().
- Die Benutzung von REPLACE anstelle von DELETE + INSERT. See Abschnitt 7.4.8, „REPLACE-Syntax“.
- Das FLUSH flush_option-Statement.
- Die Möglichkeit, Variablen in einem Statement mit := zu setzen:

```
SELECT @a:=SUM(total),@b=COUNT(*),@a/@b AS avg FROM test_tabelle;
SELECT @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;
```

2.7.4. MySQL-Unterschiede im Vergleich zu ANSI SQL92

Wir versuchen möglichst, dass MySQL dem ANSI-SQL-Standard und dem ODBC-SQL-Standard folgt, aber in einigen Fällen macht MySQL Dinge auf andere Weise:

- -- ist nur dann ein Kommentar, wenn darauf Whitespace folgt. See Abschnitt 2.7.4.8, „'--' als Beginn eines Kommentars“.
- Bei VARCHAR-Spalten werden Leerzeichen am Ende entfernt, wenn der Wert gespeichert wird. See Abschnitt 2.7.5, „Bekannte Fehler und Design-Unzulänglichkeiten in MySQL“.
- In einigen Fällen ändern sich CHAR-Spalten automatisch (silent) in VARCHAR-Spalten. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“.
- Zugriffsrechte für eine Tabelle werden nicht automatisch widerrufen, wenn Sie eine Tabelle löschen. Sie müssen explizit ein REVOKE-Statement absetzen, um die Zugriffsrechte für eine Tabelle zu widerrufen. See Abschnitt 5.3.1, „GRANT- und REVOKE-Syntax“.
- NULL AND FALSE werden zu NULL ausgewertet und nicht zu FALSE. Der Grund hierfür liegt darin, dass wir meinen, dass es keine gute Idee ist, eine Menge von Sonderkonditionen für diesen Fall auswerten zu müssen.

2.7.4.1. Sub-Selects

MySQL unterstützt momentan nur Sub-Selects der Form INSERT ... SELECT ... und REPLACE ... SELECT .... In anderen Zusammenhängen können Sie allerdings die Funktion IN() benutzen.

In vielen Fällen können Sie Ihre Anfragen ohne Sub-Selects schreiben:

```
SELECT * FROM tabelle1 WHERE id IN (SELECT id FROM tabelle2);
```

Das kann wie folgt umgeschrieben werden:

```
SELECT tabelle1.* FROM tabelle1,tabelle2 WHERE tabelle1.id=tabelle2.id;
```

Die Anfragen:

```
SELECT * FROM tabelle1 WHERE id NOT IN (SELECT id FROM tabelle2);
SELECT * FROM tabelle1 WHERE NOT EXISTS (SELECT id FROM tabelle2 where tabelle1.id=tabelle2.id);
```

Können wie folgt umgeschrieben werden:

```
SELECT tabelle1.* FROM tabelle1 LEFT JOIN tabelle2 ON tabelle1.id=tabelle2.id where tabelle2.id IS NULL
```

Für kompliziertere Unteranfragen (Subqueries) können Sie oft temporäre Tabelle anlegen, die die Unteranfrage enthalten. In einigen Fällen wird diese Option allerdings nicht funktionieren. Am häufigsten treten diese Fälle mit `DELETE`-Statements auf, wofür Standard-SQL keine Verknüpfungen (Joins) unterstützt. Für solche Situationen sind zwei Optionen verfügbar, solange MySQL noch keine Unteranfragen unterstützt.

Die erste Option besteht darin, eine prozedurale Programmiersprache (wie PHP oder Perl) zu benutzen, um eine `SELECT`-Anfrage zu erhalten, die die Primärschlüssel enthält, die benötigt werden, um die entsprechenden Datensätze zu löschen, und dann diese Werte zu benutzen, um das `DELETE`-Statement zu formulieren (`DELETE FROM ... WHERE ... IN (key1, key2, ...)`).

Die zweite Option besteht darin, interaktives SQL zu benutzen, um automatisch eine Reihe von `DELETE`-Statements zu formulieren, indem die MySQL-Erweiterung `CONCAT()` benutzt wird (anstelle des Standard-Operators `||`). Beispiel:

```
SELECT CONCAT('DELETE FROM tabelle1 WHERE pkid = ', tabelle1.pkid, ';')
  FROM tabelle1, tabelle2
 WHERE tabelle1.spalte1 = tabelle2.spalte2;
```

Sie können diese Anfrage in eine Skriptdatei schreiben und deren Eingabe an den Kommandozeilen-Interpreter `mysql` leiten und von dort die Ausgabe zurück an eine zweite Instanz des Interpreters:

```
prompt> mysql --skip-column-names meine_db < mein_skript.sql | mysql meine_db
```

MySQL 4.0 unterstützt das Löschen aus mehreren Tabellen (multi-table deletes), was benutzt werden kann, um effizient Zeilen zu löschen, basierend auf den Informationen aus einer Tabelle oder sogar aus mehreren Tabellen zur gleichen Zeit.

2.7.4.2. `SELECT INTO TABLE`

MySQL unterstützt noch nicht die Oracle-SQL-Erweiterung `SELECT ... INTO TABLE ...`. MySQL unterstützt statt dessen die ANSI-SQL-Syntax `INSERT INTO ... SELECT ...`, die im Prinzip dasselbe ist. See Abschnitt 7.4.3.1, „`INSERT ... SELECT`-Syntax“.

```
INSERT INTO tabelle_temp2 (fldID) SELECT tabelle_temp1.fldOrder_ID FROM tabelle_temp1 WHERE
tabelle_temp1.fldOrder_ID > 100;
```

Alternativ können Sie `SELECT INTO OUTFILE...` oder `CREATE TABLE ... SELECT` benutzen, um Ihre Probleme zu lösen.

2.7.4.3. Transaktionen

Weil MySQL heutzutage Transaktionen unterstützt, gelten die folgenden Erörterungen nur, wenn Sie nur Tabellentypen benutzen, die nicht transaktionssicher sind. See Abschnitt 7.7.1, „`BEGIN/COMMIT/ROLLBACK`-Syntax“.

Oft wird von neugierigen oder kritischen Leuten gefragt: ``Warum ist MySQL keine transaktionale Datenbank?" oder ``Warum unterstützt MySQL keine Transaktionen?"

MySQL hat sich bewusst entschieden, andere Paradigmen für die Datenintegrität zu unterstützen: ``atomische Operationen." Es entspricht unserer Denkweise und unserer Erfahrung, dass atomische Operationen gleiche oder bessere Integrität bei wesentlich besserer Performance gewährleisten. Nichtsdestotrotz schätzen und verstehen wir das transaktionale Datenbank-Paradigma und planen, im Verlauf der nächsten Releases transaktionssichere Tabellen einzuführen, auf der Basis der Transaktionssicherheit pro einzelner Tabelle. Wir werden unseren Benutzern die Entscheidung überlassen, ob Sie in ihren Applikationen den Geschwindigkeitsvorteil atomischer Operationen benötigen oder die transaktionalen Features.

Wie benutzt man die Features von MySQL, um rigorose Integrität beizubehalten, und wie sind diese Features im Vergleich mit dem transaktionalen Paradigma zu bewerten?

Zunächst ist es nach dem transaktionalen Paradigma bequemer, mit Transaktionen zu arbeiten, wenn Ihre Applikationen auf eine Weise geschrieben sind, dass sie in kritischen Situationen ``rollback" anstelle von ``commit" aufrufen. Darüber hinaus stellen Transaktionen sicher, dass unbeendete Updates oder zerstörende Aktivitäten nicht an die Datenbank abgesetzt werden; der Server hat die Gelegenheit, ein automatisches Rollback durchzuführen, wodurch Ihre Datenbank gerettet wird.

In fast allen Fällen erlaubt Ihnen MySQL, potentiellen Problemen vorzubauen, indem einfache Überprüfungen eingebaut und einfache Skripte laufen gelassen werden, die die Datenbanken auf Inkonsistenzen prüfen und automatisch reparieren oder Warnmeldungen ausgeben, wenn so etwas passiert. Beachten Sie auch, dass allein durch die Benutzung der MySQL-Logdatei oder durch das Hinzufügen einer speziellen Logdatei Tabellen perfekt repariert werden können, ohne dass ein Verlust an Datenintegrität eintritt.

Darüber hinaus können fatale transaktionale Updates so umgeschrieben werden, dass sie atomisch sind. In der Tat gehen wir so weit zu sagen, dass alle Integritätsprobleme, die Transaktionen lösen, mit `LOCK TABLES` oder atomischen Update durchgeführt werden können, was sicherstellt, dass Sie nie einen automatischen Abbruch von der Datenbank bekommen, was ein gewöhnliches Problem transaktionaler Datenbanken darstellt.

Nicht einmal Transaktionen können jeden Verlust verhindern, wenn der Server abstürzt. In solchen Fällen können sogar transaktionale Systeme Daten verlieren. Der Unterschied zwischen unterschiedlichen Systemen besteht einzig darin, wie kurz die Zeitverzögerung ist, in der Daten verloren gehen könnten. Kein System ist 100%-ig sicher, sondern lediglich ``sicher genug''. Selbst von Oracle, ansonsten als das sicherste aller transaktionalen Datenbanken berühmt, wird berichtet, dass es manchmal in solchen Situationen Daten verliert.

Um mit MySQL auf der sicheren Seite zu sein, brauchen Sie lediglich Datensicherungen und angeschaltetes Update-Logging. Damit können Sie in jeder denkbaren Situation genau wie mit jeder beliebigen transaktionalen Datenbank Daten wiederherstellen. Natürlich ist es immer eine gute Idee, Datensicherungen zu haben, unabhängig von der verwendeten Datenbank.

Das transaktionale Paradigma hat seine Vor- und Nachteile. Viele Benutzer und Applikationsentwickler verlassen sich auf die Einfachheit, mit der sie um Probleme herum Code schreiben können, dort wo anscheinend ein Abbruch erfolgt ist, oder wo es notwendig ist, haben sie womöglich ein bisschen mehr Arbeit mit MySQL, weil sie anders denken oder mehr schreiben müssen. Wenn Ihnen atomische Operationen neu sind oder Sie vertrauter mit Transaktionen sind (oder Sie sich damit besser fühlen), kommen Sie nicht gleich zur Schlussfolgerung, dass sich MySQL nicht mit diesen Überlegungen beschäftigt hat. Zuverlässigkeit und Integrität stehen für uns absolut im Vordergrund. Aktuelle Schätzungen gehen davon aus, dass zur Zeit mehr als eine Million `mysqld`-Server laufen, von denen viele in Produktionsumgebungen eingesetzt werden. Wir hören sehr, sehr selten von Benutzern, die irgendwelche Daten verloren haben, und in fast allen Fällen sind Benutzerfehler im Spiel. Das ist unserer Meinung nach der beste Beweis für die Stabilität und Zuverlässigkeit von MySQL.

Im übrigen lassen die aktuellen Features von MySQL Zuverlässigkeit und Integrität auf Transaktionsebene oder besser zu, wenn in bestimmten Situationen Integrität von höchster Wichtigkeit ist. Wenn Sie Tabellen mit `LOCK TABLES` sperren, werden alle Updates angehalten, bis jegliche Integritätsprüfungen durchgeführt sind. Wenn Sie nur eine Lesesperre (Read Lock) machen (im Gegensatz zu einer Schreibsperre - Write Lock), werden Lese- und Einfügeoperationen noch zugelassen. Die neu eingefügten Datensätze können von nicht Clients gesehen werden, die eine `READ`-Sperre haben, bis sie ihre Lesesperre aufheben. Mit `INSERT DELAYED` können Sie Einfügeoperationen in eine lokale Warteschlange (Local Queue) stellen, solange, bis die Sperren aufgehoben sind, ohne dass der Client warten muss, bis die Einfügeoperationen abgeschlossen sind. See Abschnitt 7.4.4, „`INSERT DELAYED`-Syntax“.

``Atomisch'', so wie wir es meinen, ist nichts Magisches. Es bedeutet nur, dass Sie sicher sein können, dass kein anderer Benutzer mit irgendeinem laufenden Update in Konflikt kommen kann, und dass es nie ein automatisches Rollback geben kann (was bei transaktionsbasierenden Systemen vorkommen kann, wenn Sie nicht sehr vorsichtig sind). MySQL garantiert auch, dass es keine schmutzigen Lesezugriffe (Dirty Reads) gibt. Sie finden einige Beispiele, wie man atomische Updates schreibt, im Abschnitt über Commits und Rollbacks.

Wir haben reichlich über Integrität und Performance nachgedacht und glauben, dass unser atomisches Paradigma sowohl Zuverlässigkeit als auch extrem hohe Performance gewährleistet, und zwar drei- bis fünfmal schneller, als es die schnellste und optimal eingestellte transaktionale Datenbank schafft. Wir haben Transaktionen nicht deshalb heraus gelassen, weil sie schwer zu machen sind. Der Hauptgrund für die Entscheidung für atomische Operationen gegen Transaktionen liegt darin, dass wir dadurch viele Geschwindigkeitsoptimierungen machen konnten, die auf andere Art nicht möglich gewesen wären.

Viele unserer Benutzer, für die Geschwindigkeit das Wichtigste ist, haben keinerlei Bedenken hinsichtlich Transaktionen. Für sie sind Transaktionen kein Thema. Diejenigen Benutzer, die Sorgen mit Transaktionen haben oder sich darüber wundern, dass MySQL diese nicht unterstützt, gibt es eine ``MySQL-Art'', die wir weiter oben beschrieben haben. Denjenigen, denen Sicherheit wichtiger als Geschwindigkeit ist, empfehlen wir die Benutzung von `BDB`- oder `InnoDB`-Tabellen für alle kritischen Daten. See Kapitel 8, *MySQL-Tabellentypen*.

Ein letzter Hinweis: Wir arbeiten zur Zeit an einem sicheren Replikationsschema, vom dem wir glauben, dass es besser als jedes kommerzielle Replikationssystem ist, das wir kennen. Dieses System wird mit dem atomischen, nicht-transaktionalen Paradigma mit höchster Zuverlässigkeit laufen. Bleiben Sie dran!

2.7.4.4. Gespeicherte Prozeduren und Trigger

Eine gespeicherte Prozedur ist ein Satz von SQL-Kommandos, die kompiliert und auf dem Server gespeichert werden können. Wenn dies einmal geschehen ist, müssen Clients nicht mehr die gesamte Anfrage absetzen, sondern können sich auf die gespeicherte Prozedur beziehen. Hiermit wird bessere Performance erreicht, den die Anfrage muss nur einmal geparst werden, und es muss weniger Information zwischen Client und Server ausgetauscht werden. Man kann sogar die konzeptionelle Ebene steigern, indem man Bibliotheken von Funktionen auf dem Server bereit hält.

Ein Trigger ist eine gespeicherte Prozedur, die aufgerufen wird, wenn ein bestimmtes Ereignis eintritt. Beispielsweise kann man eine gespeicherte Prozedur installieren, die jedes Mal ausgeführt wird, wenn ein Datensatz aus einer Transaktionstabelle gelöscht wird, und die automatisch den dazu gehörigen Kunden aus einer Kundentabelle löscht, wenn alle seine Transaktionen gelöscht wurden.

Für ein späteres Release ist geplant, dass MySQL gespeicherte Prozeduren handhaben kann, aber ohne Trigger. Trigger

verlangsamen üblicherweise alles, sogar Anfragen, für die sie nicht benötigt werden.

Um festzustellen, ab wann MySQL gespeicherte Prozeduren bekommen wird, siehe auch Abschnitt 2.8, „MySQL und die Zukunft (das TODO)“.

2.7.4.5. Fremdschlüssel

Ab MySQL-Serverversion 3.23.44 unterstützen `InnoDB`-Tabellen die Prüfung auf Fremdschlüsselbeschränkungen wie `CASCADE`, `ON DELETE` und `ON UPDATE`.

Bei anderen Tabellentypen parst der MySQL-Server die `FOREIGN KEY`-Syntax in `CREATE TABLE`-Befehlen lediglich, benutzt oder speichert diese Informationen jedoch nicht.

Beachten Sie, dass Fremdschlüssel in SQL nicht dazu benutzt werden, um Tabellen zu verknüpfen, sondern hauptsächlich, um die referentielle Integrität zu überprüfen (Fremdschlüssel-Restriktionen). Wenn Sie durch ein `SELECT`-Statement Ergebnisse aus mehreren Tabellen erhalten wollen, tun Sie dies, indem Sie Tabellen verknüpfen (Join):

```
SELECT * FROM table1,table2 WHERE table1.id = table2.id;
```

See Abschnitt 7.4.1.1, „`JOIN`-Syntax“. See Abschnitt 4.5.6, „Wie Fremdschlüssel (Foreign Keys) verwendet werden“.

Wenn Fremdschlüssel (`FOREIGN KEY`s) als Beschränkung benutzt werden, müssen sie nicht verwendet werden, wenn eine Applikation Zeilen in der korrekten Reihenfolge in `MyISAM`-Tabellen einfügt.

Bei `MyISAM`-Tabellen können Sie das Problem, dass `ON DELETE...` nicht implementiert ist, dadurch umgehen, dass Sie das entsprechende `DELETE`-Statement einer Applikation hinzufügen, wenn Sie Datensätze aus einer Tabelle löschen, die Fremdschlüssel hat. In der Praxis ist das genauso schnell (in einigen Fällen schneller) und wesentlich portabler, als wenn Sie Fremdschlüssel benutzen würden.

Beim MySQL-Server 4.0 können Sie das Löschen aus mehreren Tabellen verwenden (Multi-Table Delete), um mit einem einzigen Befehl Zeilen aus vielen Tabellen zu löschen. See Abschnitt 7.4.6, „`DELETE`-Syntax“.

Die `FOREIGN KEY`-Syntax ohne `ON DELETE ...` wird oft von ODBC-Applikationen verwendet, um automatische `WHERE`-Klauseln zu erzeugen.

Die `FOREIGN KEY`-Implementation wird künftig so erweitert werden, dass die Informationen in der Tabellendefinitionsdatei gespeichert werden und von `mysqldump` und ODBC abgerufen werden können. Für einen späteren Zeitpunkt ist vorgesehen, auch für `MyISAM`-Tabellen Fremdschlüsselbeschränkungen hinzuzufügen.

Denken Sie daran, dass Fremdschlüssel oft falsch eingesetzt werden, was zu schweren Problemen führen kann. Selbst wenn sie korrekt verwendet werden, sind sie keine Zauberlösung für das Problem der referenziellen Integrität, doch können sie solche Dinge erleichtern.

Einige Vorteile der Erzwingung von Fremdschlüsselbeschränkungen sind:

- Korrekte Konzeption der Beziehungen vorausgesetzt erschweren es Fremdschlüsselbeschränkungen einem Programmierer, in eine Datenbank Inkonsistenzen einzuführen.
- Kaskadierende Updates und Deletes können den Applikationscode vereinfachen.
- Korrekt konzipierte Fremdschlüsselregeln erleichtern das Dokumentieren von Beziehungen zwischen Tabellen.

Nachteile:

- Fehler, die beim Konzipieren von Fremdschlüsselbeziehungen leicht vorkommen, können schwerwiegende Probleme verursachen. Beispiele hierfür sind zirkuläre Regeln oder die falsche Kombination kaskadierender Deletes.
- Eine korrekt geschriebene Applikation stellt bereits intern sicher, dass sie Beschränkungen zur Wahrung der referenziellen Integrität nicht verletzt, bevor sie eine Anfrage absetzt. Daher führen zusätzliche Überprüfungen auf Datenbankebene dazu, dass die Performanz einer Applikation sinkt.
- Häufig kommt es auch vor, dass ein DBA eine solchermaßen komplexe Topologie von Beziehungen anlegt, dass es schwierig oder gar unmöglich wird, die einzelnen Tabellen zu sichern oder wiederherzustellen.

2.7.4.6. Warum wir Fremdschlüssel nicht implementiert haben

Die Informationen auf dieser Seite befinden sich in Überarbeitung. Ab MySQL-Serverversion 3.23.44 unterstützen `InnoDB`-Tabellen die Prüfung auf Fremdschlüsselbeschränkungen wie `CASCADE`, `ON DELETE` und `ON UPDATE`.

2.7.4.7. Sichten (Views)

MySQL unterstützt noch keine Sichten. Die Implementierung von Views ist aber für Version 5.1 geplant.

Sichten sind äußerst nützlich, um Benutzern Zugang zu einem Satz von Beziehungen wie zu einer einzigen Tabelle zu gewähren (Lesezugriff). Viele SQL-Datenbanken lassen es nicht zu, dass irgend welche Zeilen in einer Sicht aktualisiert werden (Update). Statt dessen müssen die einzelnen Tabellen aktualisiert werden.

Weil MySQL meist in Applikationen und in Web-Systemen eingesetzt wird, wo der Applikationsprogrammierer volle Kontrolle über die Datenbankbenutzung hat, sehen die meisten unserer Benutzer Sichten als nicht sehr wichtig an.

In MySQL werden Sichten nicht benötigt, um den Zugriff auf Spalten zu beschränken, weil MySQL ein sehr ausgefeiltes System der Zugriffsberechtigungen hat. See Abschnitt 5.2, „Allgemeine Sicherheitsthemen und das MySQL-Zugriffsberechtigungssystem“.

2.7.4.8. '`--`' als Beginn eines Kommentars

Einige andere SQL-Datenbanken benutzen '`--`', um Kommentare zu beginnen. MySQL benutzt ‘`#`’ als Anfangszeichen, wenn auch das `mysql`-Kommandozeilen-Werkzeug alle Zeilen entfernt, die mit '`--`' anfangen. Sie können in MySQL auch Kommentare im C-Stil verwenden `/* Das ist ein Kommentar */`. See Abschnitt 7.1.5, „Kommentar-Syntax“.

MySQL ab Version 3.23.3 unterstützt Kommentare, die mit '`--`' beginnen, allerdings nur, wenn der Kommentarbeginn von einem Leerzeichen gefolgt wird. Der Grund liegt darin, dass dieser degenerierte Kommentar-Stil eine Menge Probleme mit automatisch generierten SQL-Anfragen verursacht, die Ähnliches wie den folgenden Code benutzen, wo automatisch der Wert einer Zahlung für `!zahlung!` eingefügt wird:

```
UPDATE tabelle_name SET kredit=kredit-!zahlung!
```

Was, glauben Sie, passiert, wenn der Wert von `zahlung` negativ wird?

Weil `1--1` in SQL zulässig ist, sind wir der Meinung, dass es furchtbar ist, dass '`--`' den Anfang eines Kommentars bedeutet.

In MySQL ab Version 3.23 können Sie allerdings folgendes benutzen: `1-- Das ist ein Kommentar`

Die folgenden Erörterungen treffen nur zu, wenn Sie eine MySQL-Version vor 3.23 laufen lassen:

Wenn Sie ein SQL-Programm in einer Textdatei haben, das '`--`'-Kommentare enthält, sollten Sie folgendes benutzen:

```
shell> replace " --" " #" < text-datei-mit-merkwuerigen-kommentaren.sql \
        | mysql datenbank
```

anstelle des üblichen:

```
shell> mysql datenbank < text-datei-mit-merkwuerdigen-kommentaren.sql
```

Sie können auch die Kommandodatei ``direkt" editieren, um die '`--`'-Kommentare zu ‘`#`’-Kommentaren zu machen:

```
shell> replace " --" " #" -- text-datei-mit-merkwuerdigen-kommentaren.sql
```

Machen Sie die Änderungen mit folgendem Befehl rückgängig:

```
shell> replace " #" " --" -- text-datei-mit-merkwuerdigen-kommentaren.sql
```

2.7.5. Bekannte Fehler und Design-Unzulänglichkeiten in MySQL

Die folgenden Probleme sind bekannt. Ihre Behebung hat eine sehr hohe Priorität:

- `ANALYZE TABLE` kann eine BDB-Tabelle in manchen Fällen unbenutzbar machen, bis `mysqld` neu gestartet wird. Wenn so etwas passiert, stehen Fehlermeldungen wie die folgende in der MySQL-Fehler-Datei (Error File):

  ```
  001207 22:07:56  bdb:  log_flush: LSN past current end-of-log
  ```

- Führen Sie mit einer `BDB`-Tabelle nicht `ALTER TABLE` aus, wenn Sie mit dieser noch nicht abgeschlossene Mehrfach-Statement-Transaktionen durchführen. (Die Transaktion wird wahrscheinlich ignoriert.)

- `ANALYZE TABLE`, `OPTIMIZE TABLE` und `REPAIR TABLE` können Probleme bei Tabellen verursachen, für die `INSERT DELAYED` benutzt wird.
- Wenn Sie `LOCK TABLE ..` und `FLUSH TABLES ..` benutzen, können Sie nicht sicher sein, dass bei der fraglichen Tabelle keine halb abgeschlossenen Transaktionen im Gange sind.
- BDB-Tabellen lassen sich etwas langsam öffnen. Wenn Sie viele BDB-Tabellen in einer Datenbank haben, kann es sehr lange dauern, bis Sie den `mysql`-Client für diese Datenbank benutzen können, wenn Sie die `-A`-Option oder `rehash` benutzen. Das macht sich speziell dann bemerkbar, wenn Sie einen große Tabellen-Cache benutzen.
- Das momentane Replikationsprotokoll kann nicht mit `LOAD DATA INFILE` und mit Zeilenbegrenzungszeichen (line terminator characters) umgehen, die mehr als 1 Zeichen enthalten.

Folgende Probleme sind bekannt und werden zu gegebener Zeit behoben:

- Momentan funktioniert `MATCH` nur bei `SELECT`-Statements.
- Wenn Sie `SET CHARACTER SET` benutzen, können Sie keine landesspezifischen (nationalen) Zeichen für Datenbank-, Tabellen- und Spaltennamen verwenden (also z. B. kein ä, ö, ü).
- `DELETE FROM merge_table` ohne `WHERE` löscht nur die Zuordnung (das Mapping) für die Tabelle, nicht alles in der zugeordneten (gemappten) Tabelle.
- Sie können den Server nicht in ein anderes Verzeichnis bauen, wenn Sie MIT-Pthreads verwenden. Weil dies Änderungen an MIT-Pthreads bedingen würde, werden wir dieses Problem wahrscheinlich nicht beheben.
- `BLOB`-Werte können nicht ``zuverlässig'' in `GROUP BY`-, `ORDER BY` oder `DISTINCT`-Klauseln benutzt werden. In diesen Fällen werden bei Vergleichen nur die ersten `max_sort_length` Bytes (Vorgabewert 1024) von `BLOB`s benutzt. Die Voreinstellung kann mit der `-O max_sort_length`-Option für `mysqld` geändert werden. In den meisten Fällen können Sie als Workaround eine Teilzeichenkette (Substring) verwenden: `SELECT DISTINCT LEFT(blob,2048) FROM tabelle`.
- Berechnungen werden mit `BIGINT` oder `DOUBLE` durchgeführt (beide sind normalerweise 64 Bits lang). Es hängt von der verwendeten Funktion ab, welche Genauigkeit man erhält. Als allgemeine Regel gilt, dass Bit-Funktionen mit `BIGINT`-Genauigkeit, `IF` und `ELT()` mit `BIGINT`- oder `DOUBLE`-Genauigkeit und der Rest mit `DOUBLE`-Genauigkeit durchgeführt werden. Man sollte vermeiden, vorzeichenlose Werte, die größer als 63 Bits sind (9223372036854775807), zu verwenden, ausser für Bit-Felder! MySQL 4.0 bietet eine bessere `BIGINT`-Handhabung als MySQL 3.23.
- Bei allen Zeichenketten-Spalten ausser bei `BLOB`- und `TEXT`-Spalten werden Leerzeichen am Ende automatisch entfernt, wenn sie abgerufen werden. Bei `CHAR`-Typen ist das okay und kann gemäß ANSI-SQL92 als ein Feature betrachtet werden. Der Bug besteht darin, dass in MySQL auch `VARCHAR`-Spalten auf dieselbe Art behandelt werden.
- Pro Tabelle können höchstens 255 `ENUM`- und `SET`-Spalten verwendet werden.
- `safe_mysqld` leitet alle Nachrichten von `mysqld` in die `mysqld`-Logdatei um. Ein Problem ergibt sich, wenn Sie `mysqladmin refresh` benutzen, um die Logdatei zu schließen und wieder zu öffnen. In diesem Fall werden `stdout` und `stderr` immer noch in die alte Logdatei geleitet. Wenn Sie `--log` umfangreich benutzen, sollten Sie `safe_mysqld` editieren, um in `'hostname'.err` anstelle von `'hostname'.log` zu loggen, damit Sie den Speicherplatz für das alte Log leicht neu belegen können, indem Sie das alte Log löschen und `mysqladmin refresh` ausführen.
- Im `UPDATE`-Statement, werden Spalten von links nach rechts aktualisiert (Update). Wenn Sie sich auf eine aktualisierte Spalte beziehen, erhalten Sie den aktualisierten Werte anstelle des ursprünglichen Werts. Beispiel:

  ```
  mysql> UPDATE tabelle SET KEY=KEY+1,KEY=KEY+1;
  ```

  Dieses Statement aktualisiert `KEY` mit `2` anstelle von `1`.
- Sie können temporäre Tabellen nicht öfter als einmal innerhalb derselbe Anfrage benutzen. Das Folgende zum Beispiel funktioniert nicht:

  ```
  select * from temporaere_tabelle, temporaere_tabelle as t2;
  ```

- `RENAME` funktioniert nicht bei `TEMPORARY`-Tabellen.
- Unter Umständen behandelt der Optimierer (Optimizer) `DISTINCT` unterschiedlich, je nachdem, ob Sie 'versteckte' Spalten in einem Join benutzen oder nicht. In einem Join werden versteckte Spalten als Teil des Ergebnisses gezählt (selbst wenn sie nicht angezeigt werden), während versteckte Spalten in normalen Anfragen nicht an einem `DISTINCT`-Vergleich teilnehmen. Zukünftig werden wir dieses Verhalten wahrscheinlich ändern, so dass versteckte Spalten nie verglichen werden, wenn

`DISTINCT` ausgeführt wird.

Hierfür ein Beispiel:

```
SELECT DISTINCT mp3id FROM band_downloads WHERE userid = 9 ORDER BY id
DESC;
```

und

```
SELECT DISTINCT band_downloads.mp3id, FROM band_downloads,band_mp3
WHERE band_downloads.userid = 9 AND band_mp3.id = band_downloads.mp3id
ORDER BY band_downloads.id DESC;
```

Im zweiten Fall bekommen Sie in MySQL 3.23.x möglicherweise zwei identische Zeilen in der Ergebnismenge (weil die versteckten 'id'-Spalten unterschiedlich sein können).

Beachten Sie, dass dies nur für Anfragen zutrifft, bei denen die ORDER BY-Spalten nicht im Ergebnis enthalten sind. ANSI-SQL erlaubt dies nicht

- Weil MySQL es zuläßt, mit Tabellentypen zu arbeiten, die keine Transaktionen unterstützen (und folglich Daten nicht per `rollback` in den vorherigen Zustand bringen können), verhalten sich einige Dinge in MySQL etwas anderes als in anderen SQL-Servern. Das kann manchmal etwas ungünstig sein, weil Spaltenwerte in der Applikation überprüft werden müssen. Auf der anderen Seite erhalten Sie dadurch eine nette Geschwindigkeitssteigerung, weil es MySQL gestattet, einige Optimierungen vorzunehmen, die ansonsten sehr schwer durchzuführen sein würden.

  Wenn Sie eine Spalte auf einen nicht zulässigen Wert setzen, speichert MySQL, statt ein Rollback durchzuführen, den `besten möglichen Wert` in der Spalte:

  - Wenn Sie versuchen, in einer numerischen Spalte einen Wert ausserhalb des Wertebereichs zu speichern, speichert MySQL statt dessen den kleinsten oder größten möglichen Wert.
  - Wenn Sie versuchen, eine Zeichenkette, die nicht mit einer Zahl beginnt, in einer numerischen Spalte zu speichern, speichert MySQL 0.
  - Wenn Sie versuchen, `NULL` in einer Spalte zu speichern, die keine `NULL`-Werte zuläßt, speichert MySQL 0 oder `''` (leere Zeichenkette). (Man kann dieses Verhalten jedoch mit der -DDONT_USE_DEFAULT_FIELDS-Kompilierungs-Option ändern.)
  - MySQL läßt zu, dass einige falsche Datumswerte in `DATE`- und `DATETIME`-Spalten gespeichert werden (wie 2000-02-31 oder 2000-02-00). Wenn das Datum völlig falsch ist, speichert MySQL den speziellen Datumswert 0000-00-00 in der Spalte.
  - Wenn Sie `enum` auf einen nicht unterstützten Wert setzen, wird es auf den Fehlerwert 'leere Zeichenkette' oder (bei numerischen Werten) auf 0 gesetzt.
- Wenn Sie `PROCEDURE` auf eine Anfrage ausführen, die eine leere Ergebnismenge liefert, kann es in einigen Fällen vorkommen, dass `PROCEDURE` die Spalten nicht umwandelt.
- Wenn Sie eine Tabelle vom Typ `MERGE` anlegen, wird nicht überprüft, ob die zugrunde liegenden Tabellen von einem kompatiblen Typ sind.
- MySQL kann bislang nicht mit `NaN`-, `-Inf`- und `Inf`-Werten in Doubles umgehen. Wenn Sie diese benutzen, gibt es Probleme, wenn Daten importiert oder exportiert werden. Als Zwischenlösung sollten Sie `NaN` in `NULL` umwandeln (falls möglich) und `-Inf` und `Inf` in den kleinsten bzw. größten möglichen Wert.
- Negative Zahlen in der `LIMIT`-Klausel werden als große positive Zahlen behandelt.
- Wenn Sie `ALTER TABLE` benutzen, um einen `UNIQUE`-Index zu einer Tabelle hinzuzufügen, die in einer `MERGE`-Tabelle benutzt wird, und dann `ALTER TABLE` benutzen, um der `MERGE`-Tabelle einen normalen Index hinzuzufügen, weicht die Reihenfolge der Schlüssel für die Tabellen ab. Das liegt daran, dass `ALTER TABLE` `UNIQUE`-Schlüssel vor normalen Schlüsseln einfügt, um doppelte Schlüssel so früh wie möglich erkennen zu können.

Folgende bekannte Bugs gibt es in früheren Versionen von MySQL:

- Man kann einen hängenden Thread erhalten, wenn man `DROP TABLE` auf eine Tabelle ausführt, die zu vielen Tabellen gehört, die mit `LOCK TABLES` gesperrt sind.
- In folgenden Fällen können Sie einen Core Dump erhalten:

  - Die Routine für verzögertes Einfügen (Delayed Insert Handler) hat noch nie ausgeführte Einfügeoperationen (Pending Inserts) auf eine Tabelle.
  - `LOCK tabelle` mit `WRITE`
  - `FLUSH TABLES`
- Vor MySQL-Version 3.23.2 kann ein `UPDATE` fehlschlagen, dass einen Schlüssel mit einer `WHERE`-Klausel auf denselben Schlüssel aktualisiert, weil der Schlüssel benutzt wurde, um nach Datensätzen zu suchen, und dieselbe Zeile mehrfach gefunden wurde:

  ```
  UPDATE tabelle SET KEY=KEY+1 WHERE KEY > 100;
  ```

  Ein Workaround besteht in der Benutzung von:

  ```
  mysql> UPDATE tabelle SET KEY=KEY+1 WHERE KEY+0 > 100;
  ```

  Das funktioniert, weil MySQL auf Ausdrücke (Expressions) in der `WHERE`-Klausel keine Indizes benutzt.
- Vor MySQL-Version 3.23 wurden alle numerischen Typen als Festkomma-Felder behandelt. Das bedeutet, dass Sie festlegen müssen, wie viele Dezimalstellen ein Fließkomma-Feld haben soll. Alle Werte wurden mit der korrekten Anzahl von Dezimalstellen zurückgegeben.

Was Plattform-spezifische Bugs angeht, sehen Sie bitte im Abschnitt über Kompilieren und Portieren nach.

2.8. MySQL und die Zukunft (das TODO)

Dieser Anhang listet die Features auf, für die wir eine Implementierung in MySQL geplant haben.

Alles auf dieser Liste gibt nur ungefähr die Reihenfolge wieder, in der es gemacht werden wird. Wenn Sie die Prioritäten beeinflussen wollen, registrieren Sie bitte eine Lizenz oder unterstützen Sie uns und teilen uns mit, was Sie schneller gemacht haben wollen. See Abschnitt 2.4.4, „MySQL-Lizenzpolitik“.

Geplant ist, dass wir in Zukunft den kompletten ANSI-SQL99-Standard unterstützen, aber mit einer Menge nützlicher Erweiterungen. Die Herausforderung liegt darin, dass durchzuführen, ohne Geschwindigkeitsvorteile zu opfern oder den Code zu kompromittieren.

2.8.1. Dinge, die in Version 4.0 enthalten sein sollten

Wir haben uns der Entwicklung von MySQL Version 4.0 zugewandt. Die meisten grundsätzlichen Dinge, die wir in Version 4.0 haben wollen, sind bereits gemacht. Das Ziel ist, den Rest der folgenden Features schnell einzubauen und dann zur Entwicklung von MySQL 4.1 überzugehen.

See Abschnitt 2.5, „MySQL 4.0 kurz und bündig“.

Der News-Abschnitt für 4.0 beinhaltet eine Liste der Features, die wir bereits im 4.0-Baum implementiert haben. See Abschnitt D.1, „Änderungen in Release 4.0.x (Entwicklung; Alpha)“.

- Benutzern erlauben, die Startoptionen zu ändern, ohne den Server herunter fahren zu müssen.
- Störsichere Replikation.
- Mehr Funktionen für die Volltextsuche. See Abschnitt 7.8.3, „Neue Features der Volltextsuche in MySQL 4.0“.
- Neuer Schlüssel-Cache
- Neues Dateiformat für die Tabellendefinition (`.frm`-Dateien). Das versetzt uns in die Lage, nicht irgendwann keine Bits mehr zu haben, wenn wir weitere Tabellenoptionen hinzufügen. Es wird nach wie vor möglich sein, in 4.0 das alte `.frm`-Dateiformat zu benutzen. Alle neu erzeugten Tabellen werden jedoch das neue Format benutzen.

  Das neue Dateiformat versetzt uns in die Lage, neue Spaltentypen, mehr Optionen für Schlüssel und `FOREIGN KEY`-Support hinzuzufügen.
- Die Replikation sollte mit `RAND()` und Benutzer-Variablen `@var` funktionieren.
- Online-Datensicherung mit sehr geringen Performance-Einbussen. Das Online-Backup wird das Hinzufügen eines neuen

Replikations-Slaves erleichtern, ohne dass man den Master herunter fahren muss.

- Es zulassen, dass `DELETE` auf `MyISAM`-Tabellen den Datensatz-Cache benutzt. Um das zu tun, müssen wir den Thread-Cache für Datensätze aktualisieren, wenn wir die `.MYD`-Datei aktualisieren.
- Zeichensatz-Festlegungen (Casts) und Syntax für die Handhabung mehrerer Zeichensätze.
- Hilfe für alle Befehle des Clients.
- Sichere Verbindungen (mit SSL).
- `SHOW COLUMNS FROM tabelle` (der vom `mysql`-Client benutzt für die Erweiterung von Spaltennamen benutzt wird) sollte nicht die Tabelle öffnen, sondern nur die Definitionsdatei. Das wird weniger Speicher beanspruchen und sehr viel schneller sein.
- Bei der Benutzung von `SET CHARACTER SET` sollten wir die gesamte Anfrage übersetzen und nicht nur Zeichenketten. Das würde Benutzern ermöglichen, landesspezifische Zeichen auch in Datenbank-, Tabellen- und Spaltenamen zu benutzen.
- Hinzufügen einer portablen Schnittstelle zu `gethostbyaddr_r()`, damit wir `ip_to_hostname()` davon abhalten können, andere Threads zu blockieren, während es DNS-Lookups durchführt.
- Hinzufügen der `record_in_range()`-Methode zu `MERGE`-Tabellen, um den richtigen Index auswählen zu können, wenn es viele gibt, aus denen ausgewählt werden kann. Wir sollten auch die info-Schnittstelle erweitern, um die Schlüsselverteilung für jeden Index zu erhalten, wenn `analyze` über alle Unter-Tabellen läuft.
- `SET SQL_DEFAULT_TABLE_TYPE=[MyISAM | INNODB | BDB | HEAP]`.

2.8.2. Dinge, die in naher Zukunft erledigt werden müssen

- Unteranfragen (Subqueries). `select id from t where grp in (select grp from g where u > 100)`
- Atomische Multi-Tabellen-Updates, zum Beispiel `update items,month set items.price=month.price where items.id=month.id;`
- Abgeleitete Tabellen (Derived Tables).

```
select a.col1, b.col2 from (select max(col1) as col1 from root_table ) a,
other_table b where a.col1=b.col1
```

  Das könnte erreicht werden, indem für die Dauer der Anfrage automatisch temporäre Tabellen für die abgeleiteten Tabellen erzeugt werden.
- Hinzufügen eines `PREPARE` von Statements und Senden von Parametern an `mysqld`.
- Erweiterung des Client-Server-Protokolls, um Warnungen (Warnings) zu unterstützen.
- Hinzufügen von Optionen zum Client-Server-Protokoll, um Fortschrittsanzeigen für lange laufende Kommandos zu erhalten.
- Hinzufügen von Datenbank und echtem Tabellennamen (im Falle von Alias) zur MYSQL_FIELD-Struktur.
- Nicht mehr als die festgelegte Anzahl von Threads zulassen, um MyISAM recover zeitgleich laufen zu lassen.
- `INSERT ... SELECT` ändern, um optional konkurrierende Inserts zu benutzen.
- `RENAME DATABASE` implementieren. Damit das sicher für alle Tabellen-Handler funktioniert, sollte es wie folgt laufen:
  - Neue Datenbank anlegen.
  - Für jede Tabelle ein Umbenennen der Tabelle zu einer anderen Datenbank durchführen, wie wir es schon mit dem `RENAME`-Befehl machen.
  - Alte Datenbank löschen.
- Die Original-Feldtypen zurückgeben, wenn `SELECT MIN(column)... GROUP BY` ausgeführt wird.
- Mehrfache Ergebnismengen (Multiple Result Sets).
- Änderung des Protokolls, um Binärübertragung von Werten zu ermöglichen. Um das effizient zu machen, müssen wir eine API hinzufügen, die Bindung (Binding) von Variablen erlaubt.

`mysqld`.

- Es soll möglich sein, `long_query_time` mit einer Auflösung in Mikrosekunden festzulegen.
- Hinzufügen eines konfigurierbaren Prompts zum `mysql`-Kommandozeilen-Werkzeug, mit Optionen wie Datenbank in Benutzung, Zeit und Datum ...
- Hinzufügen von Bereichsüberprüfung (Range Checking) zu `MERGE`-Tabellen.
- `myisampack`-Code in den Server einlinken.
- Portierung von MySQL auf BeOS.
- Portierung von MySQL-Clients auf LynxOS.
- Hinzufügen eines temporären Schlüssel-Puffer-Caches während `INSERT/DELETE/UPDATE`, um den vorherigen Zustand elegant wiederherstellen zu können, wenn der Index voll wird.
- Wenn ein `ALTER TABLE` auf eine Tabelle durchgeführt wird, die per Symlink auf einer anderen Festplatte ist, temporäre Tabellen auf dieser Festplatte erzeugen.
- Implementierung eines `DATE/DATETIME`-Typs, der Zeitzonen-Informationen sauber handhabt, damit der Umgang mit Datumswerten in verschiedenen Zeitzonen leichter wird.
- FreeBSD- und MIT-pThreads; nehmen schlafende Threads CPU in Anspruch?
- Prüfen, ob gesperrte Threads CPU beanspruchen.
- Configure reparieren, so dass man alle Bibliotheken (wie `MyISAM`) ohne Threads kompilieren kann.
- Hinzufügen einer Option, um regelmäßig die Schlüsselseiten (Key Pages) für Tabellen mit verzögerten Schlüsseln (Delayed Keys) zu löschen (flush), wenn Sie eine Weile nicht in Gebrauch waren.
- Verknüpfungen (Join) auf Teile des Schlüssels zulassen (Optimierungsthema).
- `INSERT SQL_CONCURRENT` und `mysqld --concurrent-insert` sollen ein konkurrierendes Insert am Ende der Datei machen, falls die Datei lese-gesperrt ist.
- `FOREIGN`-Key-Festlegungen in der `.frm`-Datei speichern.
- Kaskadierendes Löschen (`DELETE`)
- Serverseitige Cursor.
- Prüfen, ob `lockd` mit modernen Linux-Kernels funktioniert; wenn nicht, müssen wir `lockd` überarbeiten! Um das zu testen, startet man `mysqld` mit `--enable-locking` und läßt die verschiedenen fork* test suits laufen. Sie sollten keine Fehler produzieren, wenn `lockd` funktioniert.
- SQL-Variablen in `LIMIT` zulassen, wie `LIMIT @a,@b`.
- Aktualisierung von Variablen in `UPDATE`-Statements zulassen, zum Beispiel: `UPDATE TABLE foo SET @a=a+b,a=@a, b=@a+c`
- Wenn Benutzervariablen aktualisiert werden, so ändern, dass man sie mit `GROUP BY` benutzen kann wie in folgendem Beispiel: `SELECT id, @a:=count(*), sum(sum_col)/@a FROM tabelle GROUP BY id`.
- Keine automatischen `DEFAULT`-Werte zu Spalten hinzufügen. Fehler ausgeben, wenn ein `INSERT` benutzt wird, dass keine Spalte enthält, die keinen `DEFAULT`-Wert hat.
- Caching von Anfragen und Ergebnissen. Das sollte als separates Modul gemacht werden, das jede Anfrage prüft. Falls diese Anfrage im Cache ist, soll das Cache-Ergebnis zurückgegeben werden. Wenn man eine Tabelle aktualisiert, sollte man so wenige Anfragen wie möglich aus dem Cache entfernen. Das sollte eine große Geschwindigkeitssteigerung auf Maschinen geben, die viel RAM haben und wo Anfragen of wiederholt werden (wie WWW-Applikationen). Eine Idee wäre, nur Anfrage des Typs `SELECT CACHED ...` zu cachen.
- `libmysql.c` überarbeiten, damit zwei `mysql_query()`-Befehle in einer Zeile stehen können, ohne dass Ergebnisse gelesen werden oder man eine nette Fehlermeldung erhält, wenn man das tut.
- Optimierung des `BIT`-Typs, so dass er 1 Bit aufnimmt (momentan nimmt `BIT` 1 Zeichen auf).
- Prüfen, warum MIT-pThreads `ctime()` auf einigen FreeBSD-Systemen nicht funktioniert.

- Hinzufügen einer `IMAGE`-Option zu `LOAD DATA INFILE`, damit `TIMESTAMP`- und `AUTO_INCREMENT`-Felder nicht aktualisiert werden.
- `LOAD DATE INFILE.. UPDATE`-Syntax hinzufügen.
  - Wenn Daten bei Tabellen mit Primärschlüssel den Primärschlüssel enthalten, werden Einträge, die zu diesem Primärschlüssel passen, vom Rest der Spalten aktualisert. Spalten, die im herein kommenden Datenstrom NICHT enthalten sind, werden jedoch nicht berührt.
  - Bei Tabellen mit Primärschlüsseln, wo im herein kommenden Datenstrom ein Teil des Schlüssels fehlt, oder wenn kein Primärschlüssel eingegeben wird, wird die Eingabe so behandelt wie jetzt schon `LOAD DATA INFILE ... REPLACE INTO`.
- `LOAD DATA INFILE` soll auch folgende Syntax verstehen:

```
LOAD DATA INFILE 'datei.txt' INTO TABLE tabelle
TEXT_FIELDS (text_feld1, text_feld2, text_feld3)
SET tabelle_feld1=concatenate(text_feld1, text_feld2), tabelle_feld3=23
IGNORE text_feld3

Das kann benutzt werden, um zusätzliche Spalten in der Textdatei zu
überspringen oder um Spalten basierend auf Ausdrücken in den gelesenen
Daten zu aktualisieren ...
```

- `LOAD DATA INFILE 'datei' INTO TABLE 'tabelle' ERRORS TO err_tabelle` Das würde bewirken, dass alle Fehler und Warnungen in der err_tabelle mitgeschrieben werden. Diese Tabelle hätte etwa folgende Struktur:

```
zeile_nummer     - Zeilennummer in der Datendatei
fehler_nachricht - die Fehler-/Warnungs-Nachricht
und vielleicht


daten_zeile      - die Zeilennummer der Datendatei
```

- Hinzufügen von echter `VARCHAR`-Unterstützung (gibt es schon in MyISAM).
- Automatische Ausgabe von `mysql` an Netscape.
- `LOCK DATABASES`. (mit vielerlei Optionen)
- Ändern wie Sortierung Speicher alloziert, um bessere Speicherausnutzung zu erhalten.
- `DECIMAL`- und `NUMERIC`-Typen können keine exponentiellen Zahlen lesen; `Field_decimal::store(const char *from,uint len)` muss neu kodiert werden, um das zu beheben.
- `mysql.cc` überarbeiten, damit weniger `malloc()`-Aufrufe durchgeführt werden, wenn Feldnamen gehasht werden.
- Funktionen: ADD_TO_SET(wert,set) und REMOVE_FROM_SET(wert,set)
- Benutzung von `t1 JOIN t2 ON ...` und `t1 JOIN t2 USING ...` hinzufügen. Momentan kann man diese Syntax nur mit `LEFT JOIN` benutzen.
- Volle Unterstützung für `unsigned long long`-Typen hinzufügen.
- Viele weitere Variablen für `show status`. Zähler für: `INSERT`-/`DELETE`-/`UPDATE`-Statements. Gelesene und aktualisierte Datensätze. Select auf 1 Tabelle und Selects mit Joins. Durchschnittliche Anzahl von Tabellen in Selects. Anzahl von `ORDER BY`- und `GROUP BY`-Anfragen.
- Wenn man `mysql` mitten in einer Anfrage abbricht, sollte man eine neue Verbindung herstellen und die alte, laufende Anfrage killen. Alternativ könnte man den Versuch unternehmen, so etwas im Server zu entdecken.
- Eine Handler-Schnittstelle für Tabelleninformation hinzufügen, damit man sie als Systemtabelle benutzen kann. Das wäre ein bisschen langsam, wenn man Informationen über alle Tabellen abfragt, aber sehr flexibel. `SHOW INFO FROM tabelle` für Basisinformationen über Tabellen sollte implementiert werden.
- Unterstützung für UNICODE hinzufügen.
- `NATURAL JOIN` und `UNION JOIN`.
- Anfragen wie `select a from crash_me left join crash_me2 using (a)` zulassen; in diesem Fall wird angenommen, dass a aus der crash_me-Tabelle kommt.
- Überarbeitung, damit `ON` und `USING` mit dem `JOIN`-Verknüpfungstyp funktioniert.

- Oracle-mäßiges `CONNECT BY PRIOR ...`, um hierarchische Strukturen zu durchsuchen.
- `mysqladmin copy datenbank neue_datenbank`. -- Erfordert, dass `mysqld` der COPY-Befehl hinzugefügt wird.
- Prozessliste sollte die Anzahl von Anfragen pro Thread zeigen.
- `SHOW HOSTS` zur Informationsausgaben über den Hostnamen-Cache.
- Format von `DATETIME` ändern, um Bruchteile von Sekunden zu speichern.
- Alle fehlenden ANSI92- und ODBC 3.0-Typen hinzufügen.
- Für berechnete Spalten Tabellennamen von leerer Zeichenkette zu `NULL` ändern.
- 'Item_copy_string' nicht auf numerische Werte anwenden, um Zahl->Zeichenkette->Zahl-Umwandlung zu vermeiden, im Falle von: `SELECT COUNT(*)*(id+0) FROM tabelle GROUP BY id`
- Benutzung der neuen GNU-regexp-Bibliothek anstelle der aktuellen ermöglichen (die GNU-Bibliothek sollte viel schneller sein als die alte).
- `ALTER TABLE` sollte nicht mehr Clients abbrechen, die `INSERT DELAYED` ausführen.
- So überarbeiten, dass, wenn Spalten, auf die in einer `UPDATE`-Klausel verwiesen wird, die alten Werte enthalten, bevor das Update begonnen wird.
- `myisamchk`, `REPAIR` und `OPTIMIZE TABLE` sollten in der Lage sein, mit Fällen umzugehen, wo die Daten und / oder Indexdateien symbolische Links sind.
- Simulation von `pread()`/`pwrite()` auf Windows einarbeiten, um konkurrierende Inserts zu ermöglichen.
- Ein Logdatei-Analyzer, aus dem Informationen herausgefiltert (geparst) werden können, welche Tabellen am häufigsten angesprochen werden, wie oft Verknüpfungen (Joins) mit mehreren Tabellen ausgeführt werden usw. Es sollte Benutzern helfen, Bereiche oder Dinge im Tabellenentwurf zu erkennen, die optimiert werden können, um sehr viel effizientere Anfragen auszuführen.
- Add `SUM(DISTINCT)`
- `ANY()`-,`EVERY()`- und `SOME()`-Gruppierungsfunktionen hinzufügen. In ANSI-SQL funktionieren diese auf boolsche Spalten, aber wir können sie so erweitern, dass sie mit beliebigen Spalten / Ausdrücken funktionieren, indem wir folgendes anwenden: wert == 0 -> FALSE und wert <> 0 -> TRUE.
- So überarbeiten, dass `MAX(column)` vom selben Typ ist wie der Spaltentyp.

```
create tabelle t1 (a DATE);
insert into t1 values (now());
create tabelle t2 select max(a) von t1;
show columns from t2;
```

- Eine nette Syntax für ein Statement entwickeln, dass auf eine Zeile ein `UPDATE` ausführt, wenn sie existiert, und eine neue Zeile einfügt (`INSERT`), wenn sie nicht existiert (so wie `REPLACE` bei `INSERT` / `DELETE` funktioniert).

2.8.3. Dinge die irgendwann gemacht werden müssen

- Funktion implementieren: `get_changed_tables(timeout,table1,table2,...)`
- Lesen durch Tabellen so ändern, das memmap benutzt wird, falls möglich. Momentan benutzen nur komprimierte Tabellen memmap.
- Ein neues Zugriffsrecht '**Show_priv**' für `SHOW`-Befehle hinzufügen.
- Den automatischen Zeitstempel-Code netter machen. Zeitstempel zum Update-Log hinzufügen mit `SET TIMESTAMP=#;`
- An manchen Stellen read/write mutex benutzen, um mehr Geschwindigkeit zu erhalten.
- Volle Unterstützung von Fremdschlüsseln. Wahrscheinlich wird man zuerst einmal eine prozedurale Sprache implementieren wollen.
- Einfache Sichten (Views; zunächst auf eine Tabelle, später auf jeden beliebigen Ausdruck).

- Automatisches Schließen einiger Tabellen, wenn eine Tabelle, eine temporäre Tabelle oder eine temporäre Datei einen Fehler 23 bekommt (nicht genug offene Dateien).
- Wenn ein Feld=# gefunden wird, alle Vorkommen von Feld auf # setzen. Momentan wird das nur in einigen einfachen Fällen gemacht.
- Alle konstanten Ausdrücke mit berechneten Ausdrücken austauschen, falls möglich.
- schlüssel = ausdruck optimieren. Momentan wird nur schlüssel = feld oder schlüssel = konstante optimiert.
- Einige der Copy-Funktionen verbinden, um netter Code zu erhalten.
- `sql_yacc.yy` in einen Inline-Parser umändern, um die Größe zu reduzieren und bessere Fehlermeldungen zu erhalten (5 Tage).
- Den Parser so ändern, dass er nur eine Regel pro unterschiedlicher Anzahl von Argumenten in Funktionen benutzt.
- Die Benutzung von vollen Berechnungsnamen (full calculation names) im ORDER-Teil (order part). (Für ACCESS97)
- `UNION`, `MINUS`, `INTERSECT` und `FULL OUTER JOIN`. (Momentan wird nur `LEFT OUTER JOIN` unterstützt.)
- `UNIQUE` bei Feldern zulassen, die `NULL` sein können.
- `SQL_OPTION MAX_SELECT_TIME=#` um einer Anfrage eine Zeitbeschränkung zu setzen.
- Make the update log to a Datenbank. Update soll in eine Datenbank loggen.
- Negative `LIMIT`-Parameter, um Daten vom Ende abrufen zu können.
- Alarm around client connect/read/write Funktionen.
- Bitte beachten sie die Änderungen in `safe_mysqld`: Nach FSSTND (woran sich Debian versucht zu halten) sollten PID-Dateien als `/var/run/<progname>.pid` angelegt werden und Log-Datei in `/var/log`. Es wäre nett, wenn man "DATADIR" in die erste Deklaration von "pidfile" und "log" packen könnte, damit die Unterbringung dieser Dateien mit einem einzigen Statement geändert werden könnte.
- Einem Client erlauben, Mitloggen anzufordern.
- Benutzung von `zlib()` für `gzip`-te Dateien in `LOAD DATA INFILE` zulassen.
- Sortieren und Gruppieren von `BLOB`-Spalten in Ordnung bringen (teilweise bereits gelöst).
- Gespeicherte Prozeduren. Wird aktuell nicht als sehr wichtig erachtet, weil gespeicherte Prozeduren noch nicht sehr standardisiert sind. Ein weiteres Problem besteht darin, dass es echte gespeicherte Prozeduren dem Optimierer viel schwerer machen und dass in vielen Fällen das Ergebnis langsamer sein wird als vorher. Auf der anderen Seite werden wir versuchen, eine einfache (atomische) Update-Sprache hinzuzufügen, die benutzt werden kann, um Schleifen und ähnliches im MySQL-Server zu schreiben.
- So ändern, dass Semaphore benutzt werden, wenn Threads gezählt werden. Man sollte zuerst eine Semaphor-Bibliothek zu MIT-pThreads implementieren.
- Keinen neuen `AUTO_INCREMENT`-Wert zuweisen, wenn eine Spalte auf 0 gesetzt wird. Statt dessen `NULL` setzen.
- Volle Unterstützung von Verknüpfungen (`JOIN`) mit Klammern.
- Als Alternative für einen Thread pro Verbindung einen Pool von Threads verwalten, der die Anfragen handhabt.
- Einem gestatten, mehr als eine Sperre (Lock) mit `GET_LOCK` zu erhalten. Wenn man das tut, muss man die möglichen Deadlocks handhaben, die diese Änderung einführen wird.

Zeitangaben stehen für den Umfang der Arbeit, nicht für echte Zeit.

2.8.4. Ein paar Dinge, für deren Umsetzung wir keine Pläne haben

- Nichts; auf lange Sicht planen wir, voll ANSI-92- / ANSI-99-kompatibel zu sein.

Kapitel 3. Installation von MySQL

Dieses Kapitel beschreibt, woher man MySQL bezieht und wie man MySQL installiert:

- Eine Liste der Site, von denen Sie MySQL beziehen können, finden Sie unter Abschnitt 3.2.1, „Wie man MySQL erhält“.
- Um festzustellen, welche Plattformen unterstützt werden, siehe Abschnitt 3.2.2, „Betriebssysteme, die von MySQL unterstützt werden“. Beachten Sie bitte, dass nicht alle unterstützten Systeme gleich gut sind, um MySQL laufen zu lassen. Auf einigen läuft es sehr viel robuster und effizienter als auf anderen - siehe Abschnitt 3.2.2, „Betriebssysteme, die von MySQL unterstützt werden“ für Details.
- Mehrere Versionen von MySQL sind sowohl als Binär- als auch als Quellcode-Distributionen erhältlich. Wir stellen auch öffentlichen Zugriff auf unseren aktuellen Quellcode-Baum für diejenigen zur Verfügung, die die aktuellsten Entwicklungen sehen und uns helfen wollen, neuen Code zu testen. Um festzustellen, welche Version und welche Art von Distribution Sie benutzen sollten, siehe Abschnitt 3.2.3, „Welche MySQL-Version Sie benutzen sollten“. Im Zweifelsfall benutzen Sie die Binärdistribution.
- Installationsanleitungen für Binär- und Quelldistributionen sind beschrieben in Abschnitt 3.2.6, „MySQL-Binärdistributionen, die von MySQL AB kompiliert wurden“ und Abschnitt 3.3, „Installation der Quelldistribution“. Jede Anleitung enthält einen Abschnitt über System-spezifische Probleme, denen Sie begegnen können.
- Prozeduren, die nach der Installation durchgeführt werden sollen / müssen, finden Sie unter Abschnitt 3.4, „Einstellungen und Tests nach der Installation“. Diese Prozeduren gelten, egal ob Sie MySQL von einer Binär- oder einer Quellcode-Distribution installieren.

3.1. Schnelle Standard-Installation von MySQL

3.1.1. MySQL auf Linux installieren

Die empfohlene Vorgehensweise für die Installation von MySQL auf Linux ist die Benutzung einer RPM-Datei. Die MySQL-RPMs werden aktuell auf einer RedHat-Version 6.2 gebaut, sollten aber auch auf anderen Linux-Versionen funktionieren, die `rpm` unterstützen und `glibc` benutzen.

Wenn Sie Probleme mit einer RPM-Datei haben, wenn Sie beispielsweise den Fehler ``Sorry, the host 'xxxx' could not be looked up`" erhalten, sehen Sie bitte unter Abschnitt 3.1.1, „MySQL auf Linux installieren“ nach.

Die RPM-Dateien, die Sie benutzen sollten, sind:

- `MySQL-VERSION.i386.rpm`

  Der MySQL-Server. Sie brauchen diese, es sei denn, Sie wollen sich lediglich mit einem MySQL-Server verbinden, der auf einer anderen Maschine läuft.

- `MySQL-client-VERSION.i386.rpm`

  Die Standard-MySQL-Client-Programme. Dieses Paket sollten Sie wohl immer installieren.

- `MySQL-bench-VERSION.i386.rpm`

  Tests und Benchmarks. Erfordert Perl und msql-mysql-modules RPMs.

- `MySQL-devel-VERSION.i386.rpm`

  Bibliotheken und Include-Dateien, die benötigt werden, wenn Sie andere MySQL-Clients kompilieren wollen, beispielsweise Perl-Module.

- `MySQL-VERSION.src.rpm`

  Dieses Paket enthält den Quelltext für alle obigen Pakete. Es kann auch dazu benutzt werden, um RPMs für andere Architekturen zu bauen (zum Beispiel für Alpha oder SPARC).

Um alle Dateien in einem RPM-Paket zu sehen, geben Sie folgendes ein:

```
shell> rpm -qpl MySQL-VERSION.i386.rpm
```

Um eine minimale Standard-Installation durchzuführen, geben Sie folgendes ein:

```
shell> rpm -i MySQL-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Um nur das Client-Paket zu installieren, geben Sie folgendes ein:

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

Das RPM legt Dateien in `/var/lib/mysql` ab. Ausserdem erzeugt das RPM die entsprechenden Einträge in `/etc/rc.d/`, um den Server beim Booten automatisch zu starten. (Falls Sie bereits vorher eine Installation durchgeführt haben, bedeutet das, dass Sie eine Kopie Ihrer vorher installierten MySQL-Startdateien machen sollten, falls Sie darin Änderungen vorgenommen haben, damit Sie diese Änderungen nicht verlieren.)

Nach der Installation der RPM-Datei(en) sollte der `mysqld`-Daemon laufen und Sie sollten jetzt in der Lage sein, mit der Benutzung von MySQL zu beginnen. See Abschnitt 3.4, „Einstellungen und Tests nach der Installation“.

Wenn etwas schief geht, finden Sie weitere Informationen im Kapitel über die Binär-Installationen. See Abschnitt 3.2.6, „MySQL-Binärdistributionen, die von MySQL AB kompiliert wurden“.

3.1.2. Installation von MySQL unter Windows

Der MySQL-Server für Windows ist in zwei Distributionstypen erhältlich:

1. Die Binärdistribution enthält ein Setup-Programm, das alles Benötigte installiert, so dass Sie den Server sofort starten können.
2. Die Quelldistribution enthält den gesamten Code und Unterstützungsdateien, um die ausführbaren Dateien unter Benutzung des VC++-6.0-Kompilers zu bauen. See Abschnitt 3.3.7, „Windows-Quelldistribution“.

Im Allgemeinen sollten Sie die Binärdistribution benutzen.

Sie benötigen folgendes:

- Ein Windows-32-Bit-Betriebssystem der Familien Windows 9x, ME, NT oder Windows 2000. Die NT-Familie gestattet, den MySQL-Server als Systemdienst laufen zu lassen. See Abschnitt 3.6.2.2, „MySQL auf Windows NT oder Windows 2000 starten“.

  Wenn Sie Tabellen benutzen, die größer als 4 GB sind, sollten Sie MySQL auf NTFS oder einem neueren Dateisystem installieren. Vergessen Sie bei der Erzeugung der Tabellen nicht, `MAX_ROWS` und `AVG_ROW_LENGTH` zu benutzen. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.

- TCP/IP-Protokollunterstützung.
- Die MySQL-Binär- oder Quelldistribution für Windows kann von http://www.mysql.com/downloads/ herunter geladen werden.

  Hinweis: Die Distributionsdateien werden in einem komprimierten Format zur Verfügung gestellt. Wir empfehlen die Benutzung eines FTP-Clients, der in der Lage ist, abgebrochene FTP-Downloads wieder aufzunehmen (resume).

- Ein `ZIP`-Programm, um die Distributionsdatei zu entpacken.
- Genug Platz auf der Festplatte, um die Datenbanken entsprechend Ihren Anforderungen zu entpacken, zu installieren und zu erzeugen.
- Wenn Sie planen, sich über `ODBC` mit dem MySQL-Server zu verbinden, benötigen Sie zusätzlich den `MyODBC`-Treiber. See Abschnitt 9.3, „MySQL-ODBC-Unterstützung“.

3.1.2.1. Binärdateien installieren

1. Wenn Sie auf einem NT- oder Windows-2000-Server arbeiten, melden Sie sich als Benutzer mit Administrationsrechten an.
2. Wenn Sie ein Upgrade einer früheren MySQL-Installation durchführen, müssen Sie den Server anhalten. Wenn Sie den Server als Systemdienst laufen lassen, geben Sie ein:

   ```
   C:\> NET STOP MySQL
   ```

Ansonsten geben Sie folgendes ein:

```
C:\mysql\bin> mysqladmin -u root shutdown
```

3. Auf NT-/Windows-2000-Maschinen müssen Sie auch den Systemdienst entfernen, wenn Sie die ausführbare Datei des Servers (z. B. -max or -nt) austauschen wollen:

```
C:\mysql\bin> mysqld-max-nt --remove
```

4. Entpacken Sie die Distributionsdatei in ein temporäres Verzeichnis.

5. Starten Sie `setup.exe`, um den Installationsprozess zu beginnen. Wenn Sie in ein anderes Verzeichnis als das vorgabemäßige (`c:\mysql`) installieren wollen, legen Sie mit der Schaltfläche `Durchsuchen` das gewünschte Verzeichnis fest.

6. Beenden Sie den Installationsprozess.

Seit MySQL 3.23.38 enthält die Windows-Distribution sowohl die normalen als auch die **MySQL-Max**-Binärdateien. Der wichtigste Vorteil der Benutzung der normalen `mysqld.exe`-Binärdatei liegt darin, dass sie etwas schneller ist und weniger Ressourcen belegt.

Hier ist eine Liste der unterschiedlichen MySQL-Server, die Sie benutzen können:

`mysqld`	Kompiliert mit komplettem Debugging und automatischer Überprüfung der Speicherzuordnung (memory allocation), symbolischen Links, InnoDB- und BDB-Tabellen.
`mysqld-opt`	Optimierte Binärdistribution ohne Unterstützung von Transaktionstabellen.
`mysqld-nt`	Optimierte Binärdatei für NT mit Unterstützung von Named Pipes. Man kann diese Version auf Windows 98 laufen lassen, aber in diesem Fall werden keine Named Pipes angelegt und man muss TCP/IP installiert haben.
`mysqld-max`	Optimierte Binärdistribution mit Unterstützung symbolischer Links, InnoDB und BDB-Tabellen.
`mysqld-max-nt`	Wie `mysqld-max`, aber mit Unterstützung von Named Pipes kompiliert.

Alle genannten Binärdistributionen sind für den Pentium Pro Prozessor optimiert, sollten aber auf jedem Intel-Prozessor >= 386 laufen.

ACHTUNG: Wenn Sie InnoDB-Tabellen benutzen wollen, müssen Sie bestimmte Start-Optionen in Ihrer `my.ini`-Datei festlegen! See Abschnitt 8.5.2, „Mit InnoDB anfangen - Optionen“.

3.2. Allgemeine Installationsthemen

3.2.1. Wie man MySQL erhält

Sehen Sie wegen Informationen zur aktuellen Version und für Download-Anweisungen auf MySQL home page nach.

Unser Haupt-Mirror-Server für den Download ist hier:

http://mirrors.sunsite.dk/mysql/

Wenn Sie Interesse haben, eine MySQL-Mirror-Site beizusteuern, können Sie anonymes rsync mit `rsync://sunsite.dk/ftp/mirrors/mysql/` machen. Schicken Sie bitte eine E-Mail an `<webmaster@mysql.com>` und geben Sie uns Bescheid, wo Ihr Mirror liegt, damit wir ihn der unten stehenden Liste hinzufügen können.

Wenn Sie Probleme beim Download von unserer Hauptseite aus haben, probieren Sie eine der unten stehenden Mirror-Sites.

Geben Sie bitte `<webmaster@mysql.com>` Bescheid, wenn Sie auf schlechte oder veraltete Mirror-Sites stoßen.

3.2.2. Betriebssysteme, die von MySQL unterstützt werden

Wir benutzen GNU Autoconf, daher ist es möglich, MySQL auf alle modernen Betriebssysteme zu portieren, auf denen Posix-Threads und ein C++-Kompiler funktionieren. (Um nur den Client-Code zu kompilieren, wir lediglich ein C++-Kompiler benötigt.) Wir benutzen und entwickeln die Software selbst hauptsächlich auf Sun Solaris (Versionen 2.5 - 2.7) und SuSE Linux Version 7.x.

Beachten Sie, dass die native Thread-Unterstützung für viele Betriebssysteme nur mit den neuesten Versionen funktioniert. Es

wurde berichtet, dass MySQL erfolgreich auf folgenden Betriebssystemen / Thread-Paket-Kombinationen kompiliert wurde:

- AIX 4.x mit nativen Threads. See Abschnitt 3.6.6.4, „Anmerkungen zu IBM-AIX“.
- Amiga.
- BSDI 2.x mit enthaltenem MIT-pThreads-Paket. See Abschnitt 3.6.4.6, „Anmerkungen zu BSD/OS“.
- BSDI 3.0, 3.1 und 4.x mit nativen Threads. See Abschnitt 3.6.4.6, „Anmerkungen zu BSD/OS“.
- DEC Unix 4.x mit nativen Threads. See Abschnitt 3.6.6.6, „Anmerkungen zu Alpha-DEC-UNIX (Tru64)“.
- FreeBSD 2.x mit enthaltenem MIT-pThreads-Paket. See Abschnitt 3.6.4.1, „Anmerkungen zu FreeBSD“.
- FreeBSD 3.x und 4.x mit nativen Threads. See Abschnitt 3.6.4.1, „Anmerkungen zu FreeBSD“.
- HP-UX 10.20 mit enthaltenem MIT-pThreads-Paket. See Abschnitt 3.6.6.2, „Anmerkungen zu HP-UX Version 10.20“.
- HP-UX 11.x mit nativen Threads. See Abschnitt 3.6.6.3, „Anmerkungen zu HP-UX Version 11.x“.
- Linux 2.0+ mit LinuxThreads 0.7.1+ oder `glibc` 2.0.7+. See Abschnitt 3.6.1, „Linux (alle Linux-Versionen)“.
- Mac OS X Server. See Abschnitt 3.6.5, „Anmerkungen zu Mac OS X“.
- NetBSD 1.3/1.4 Intel und NetBSD 1.3 Alpha (benötigt GNU make). See Abschnitt 3.6.4.2, „Anmerkungen zu NetBSD“.
- OpenBSD > 2.5 mit nativen Threads. OpenBSD < 2.5 mit enthaltenem MIT-pThreads-Paket. See Abschnitt 3.6.4.3, „Anmerkungen zu OpenBSD“.
- OS/2 Warp 3, FixPack 29 und OS/2 Warp 4, FixPack 4. See Abschnitt 3.6.7, „Anmerkungen zu OS/2“.
- SGI Irix 6.x mit nativen Threads. See Abschnitt 3.6.6.8, „Anmerkungen zu SGI Irix“.
- Solaris 2.5 und höher mit nativen Threads auf SPARC und x86. See Abschnitt 3.6.3, „Anmerkungen zu Solaris“.
- SunOS 4.x mit enthaltenem MIT-pThreads-Paket. See Abschnitt 3.6.3, „Anmerkungen zu Solaris“.
- Caldera (SCO) OpenServer mit einem aktuellen Port des FSU-PThreads-Pakets. See Abschnitt 3.6.6.9, „Anmerkungen zu Caldera“.
- Caldera (SCO) UnixWare 7.0.1. See Abschnitt 3.6.6.10, „Anmerkungen zu Caldera Unixware Version 7.0“.
- Tru64 Unix
- Windows 95, Windows 98, NT und Windows 2000. See Abschnitt 3.6.2, „Anmerkungen zu Windows“.

Beachten Sie, dass nicht alle Plattformen gleichermaßen gut geeignet sind, um MySQL laufen zu lassen. Wie gut eine bestimmte Plattform für hohe Last und geschäftskritische Anwendungen geeignet ist, hängt von folgenden Faktoren ab:

- Allgemeine Stabilität der Thread-Bibliothek. Eine Plattform mag in anderer Hinsicht einen exzellenten Ruf haben, aber wenn die Thread-Bibliothek unstabil ist, die von MySQL aufgerufen wird, läuft MySQL nur so stabil wie die Thread-Bibliothek, selbst wenn alles Sonstige perfekt ist.
- Fähigkeit des Kernels und / oder der Thread-Bibliothek, die Vorteile von **SMP** auf Mehrprozessor-Systemen wahrzunehmen. Mit anderen Worten: Wenn ein Prozess einen Thread anlegen, sollte es für diesen Thread möglich sein, auf anderen Prozessoren zu laufen als der Original-Prozess.
- Fähigkeit des Kernels und / oder der Thread-Bibliothek, viele Threads laufen zu lassen, die häufig einen Mutex über eine kurze, kritische Region anlegen / lösen können ohne exzessive Kontext-Umschaltungen. Mit anderen Worten: Wenn die Implementation von `pThread_mutex_lock()` zu sehr darauf bedacht ist, CPU zu erlangen, wird das MySQL gewaltig schmerzen. Wenn man sich dieser Tatsache nicht bewusst ist, machen zusätzliche Prozessoren MySQL in der Tat langsamer.
- Allgemeine Stabilität und Performance des Dateisystems.
- Fähigkeit des Dateisystems, überhaupt mit großen Dateien umgehend zu können, und zwar effizient, wenn Ihre Tabellen Groß sind.
- Unser Grad von Erfahrung, hier bei MySQL AB, mit der Plattform. Wenn wir eine Plattform gut kennen, setzen wir plattformspezifische Optimierungen / Verbesserungen (Fixes) ein, die zur Kompilierzeit aktiv werden. Darüber hinaus können

wir Sie beraten, wie Sie Ihr System optimal für MySQL konfigurieren.

- Umfang des Testens ähnlicher Konfigurationen, das wir intern durchgeführt haben.
- Anzahl von Benutzern, die MySQL auf dieser Plattform erfolgreich mit ähnlichen Konfigurationen haben laufen lassen. Wenn diese Zahl Groß ist, ist die Wahrscheinlichkeit viel geringer, plattformspezifische Überraschungen zu erleben.

Nach den genannten Kriterien sind die besten Plattformen für MySQL bislang x86 mit SuSE Linux 7.1, 2.4 Kernel und ReiserFS (oder jede ähnliche Linux-Distribution) und Sparc mit Solaris 2.7 oder 2.8. FreeBSD kommt als drittes, aber wir hoffen wirklich, dass es zur Spitze aufschließt, sobald erst einmal die Thread-Bibliothek verbessert ist. Wir hoffen auch, dass wir alle anderen Plattformen, auf denen MySQL kompiliert werden kann und korrekt läuft, die aber nicht ganz denselben Grad an Stabilität und Performance aufweisen, in die Spitzenkategorie aufnehmen können. Das erfordert von unserer Seite aus einige Kooperationsbemühungen mit den Entwicklern der Betriebssystem-Bibliothek-Komponenten, von denen MySQL abhängt. Wenn Sie Interesse daran haben, eine dieser Komponenten zu verbessern und in der Lage sind, ihre Entwicklung zu beeinflussen, und detailliertere Informationen darüber brauchen, was MySQL benötigt, um besser zu laufen, schicken Sie eine E-Mail an <internals@lists.mysql.com>.

Beachten Sie bitte auch, dass der obige Vergleich nichts darüber aussagen will, dass ein Betriebssystem allgemein besser oder schlechter als ein anderes sei. Wir reden hier über die Auswahl eines bestimmten Betriebssystems für einen ganz bestimmten Zweck - nämlich, MySQL laufen zu lassen, und vergleichen die Betriebssysteme nur in dieser Hinsicht. Folglich wäre das Ergebnis dieses Vergleichs ein anderes, wenn wir weitere Belange berücksichtigen würden. In manchen Fällen liegt der Grund, warum ein Betriebssystem besser als ein anderes geeignet ist, schlicht darin, dass wir auf dieser speziellen Plattform mehr Tests und Optimierungen durchgeführt haben. Wir stellen hier nur unsere Beobachtungen dar, um Ihnen bei der Entscheidung zu helfen, auf welcher Plattform Sie MySQL benutzen sollten.

3.2.3. Welche MySQL-Version Sie benutzen sollten

Zunächst müssen Sie entscheiden, ob Sie das letzte Entwicklungs-Release oder das letzte stabile Release benutzen wollen:

- Normalerweise, wenn Sie MySQL zum ersten Mal benutzen, oder wenn Sie versuchen, MySQL auf ein System zu portieren, für das es keine Binärdistribution gibt, empfehlen wir, das stabile (stable) Release zu nehmen. Beachten Sie, dass alle MySQL-Releases mit den MySQL-Benchmarks und einer umfassenden Test-Suite getestet sind, bevor das Release heraus gegeben wird.
- Wenn Sie ein altes System laufen lassen und es aktualisieren möchten, aber nicht riskieren wollen, dass ein Update nicht reibungslos klappt, sollten Sie zur aktuellsten Version des Zweiges aktualisieren, den Sie benutzen (bei dem nur die letzte Versionsnummer neuer ist als Ihre, also z. B. von 3.23.36 auf 3.23.44, wenn 3.23.44 die neueste Version des Zweigs ist). Wir haben uns innerhalb der Versions-Zweige bemüht, nur schwere Fehler zu beseitigen und kleine, relativ sichere Änderungen zu machen.

Als nächstes müssen Sie entscheiden, ob Sie eine Quelldistribution oder eine Binärdistribution nehmen wollen. In den meisten Fällen ist es ratsam, eine Binärdistribution zu nehmen, wenn eine für Ihre Plattform existiert, weil sich diese im Allgemeinen leichter installieren läßt als eine Quelldistribution.

In folgenden Fällen fahren Sie mit einer Quellinstallation wahrscheinlich besser:

- Wenn Sie MySQL an einer ganz bestimmten Stelle installieren wollen. (Die Standard-Binärdistributionen sind an jeder Stelle lauffähig, aber vielleicht wollen Sie noch mehr Flexibilität haben.)
- Um unterschiedlichen Bedürfnissen von Benutzern entgegen zu kommen, stellen wir zwei unterschiedliche Binärversionen zur Verfügung: Eine, die mit den nicht transaktionalen Tabellen-Handlern kompiliert ist (eine kleine, schnelle Binärdatei), sowie eine, die mit den wichtigsten erweiterten Optionen wie transaktionssicheren Tabellen kompiliert ist. Beide Versionen sind aus derselben Quelldistribution kompiliert. Alle nativen `MySQL`-Clients können sich mit beiden MySQL-Versionen verbinden.

  Die erweiterte MySQL-Binärdistribution ist mit dem `-max`-Suffix gekennzeichnet und ist mit denselben Optionen konfiguriert wie `mysqld-max`. See Abschnitt 5.7.5, „mysqld-max, ein erweiterter mysqld-Server“.

  Wenn Sie das `MySQL-Max`-RPM benutzen wollen, müssen Sie zuerst das Standard-`MySQL`-RPM installieren.
- Wenn Sie `mysqld` mit einigen zusätzlichen Features konfigurieren wollen, die NICHT in den Standard-Binärdistributionen enthalten sind. Hier ist eine Liste der gebräuchlichsten Zusatzoptionen, die Sie vielleicht nutzen wollen:
  - `--with-berkeley-db`
  - `--with-innodb`
  - `--with-raid`

  - `--with-libwrap`
  - `--with-named-z-lib (ist in einigen Binärdateien enthalten)`
  - `--with-debug[=full]`

- Die vorgabemäßige Binärdistribution wird normalerweise mit Unterstützung für alle Zeichensätze kompiliert und sollte auf einer Vielzahl von Prozessoren derselben Prozessorfamilie laufen.

  Wenn Sie einen schnelleren MySQL-Server erhalten wollen, können Sie ihn erneut kompilieren und nur die Zeichensätze benutzen, die Sie brauchen. Sie können auch einen besseren Kompiler (wie `pgcc`) oder andere Kompilieroptionen benutzen, die besser auf Ihren Prozessor optimiert sind.

- Wenn Sie einen Bug gefunden und dem MySQL-Entwicklungsteam mitgeteilt haben, werden Sie wahrscheinlich einen Patch erhalten, den Sie mit der Quelldistribution verwenden müssen, um den Bug zu beheben.

- Wenn Sie den C- und C++-Code lesen (und / oder ändern) wollen, aus dem MySQL besteht, müssten Sie eine Quelldistribution laden. Der Quellcode ist immer das "letzte Handbuch". Quelldistributionen enthalten auch mehr Tests und Beispiele als Binärdistributionen.

Das MySQL Benennungsschema benutzt Release-Nummern, die aus drei Zahlen und einem Suffix bestehen. Ein Release-Name wie `mysql-3.21.17-beta` zum Beispiel wird wie folgt interpretiert:

- Die erste Zahl (`3`) beschreibt das Dateiformat. Alle Version-3-Releases haben dasselbe Dateiformat.
- Die zweite Zahl (`21`) ist die Release-Ebene (Level). Normalerweise kann man hier zwischen zweien auswählen. Einer ist der stabile Zweig des Releases (aktuell `23`), der andere ist der Entwicklungs-Zweig (aktuell `4.0`). Normalerweise sind beide stabil, aber die Entwicklungsversion kann Macken oder fehlende Dokumentation neuer Features haben oder sich auf einigen Systemen nicht kompilieren lassen.
- Die dritte Zahl (`17`) ist die Versionsnummer innerhalb der Release-Ebene. Diese wird für jede neue Distribution hochgezählt. Üblicherweise werden Sie die neueste Version der Release-Ebene einsetzen wollen, die Sie gewählt haben.
- Das Suffix (`beta`) zeigt den Stabilitätsgrad des Releases an. Mögliche Suffixe sind:
  - `alpha` zeigt an, dass das Release größere Abschnitte von neuem Code enthält, der noch nicht zu 100% getestet wurde. Bekannte Bugs (üblicherweise gibt es keine) sind im News-Abschnitt dokumentiert. See Anhang D, *MySQL-Änderungsverlauf (Change History)*. In den meisten Alpha-Releases gibt es neue Befehle und Erweiterungen. Bei einem Alpha-Release können durch aktive Weiterentwicklung größere Code-Änderungen vorkommen, aber alles wird getestet, bevor ein Release veröffentlicht wird. Es sollte in keinem MySQL-Release bekannte Bugs geben.
  - `beta` bedeutet, dass jeglicher neue Code getestet wurde. Es wurden keine neuen Features hinzugefügt, die bei altem Code Probleme verursachen könnten. Es sollte keine bekannten Bugs geben. Eine Version wird von Alpha auf Beta gesetzt, wenn innerhalb der Alpha-Version mindestens einen Monat lang keine schweren Fehler mehr berichtet wurden. Wir planen für eine solche Version dann keine neuen Features mehr, die einen alten Befehl unzuverlässiger machen könnten.
  - `gamma` ist eine Beta-Version, die eine ganze Weile draussen war und offensichtlich gut funktioniert. Nur kleinere Problembehebungen wurden hinzugefügt. So etwas nennen viele andere Unternehmen ein Release.
  - Wenn eine Version kein Suffix besitzt, bedeutet das, dass diese Version schon eine ganze Weile auf vielen unterschiedlichen Sites eingesetzt wird, wobei keine Bugs ausser plattformspezifischen Bugs berichtet wurden. Für ein solches Release werden nur kritische Fehlerbehebungen durchgeführt. So etwas nennen wir ein stabiles Release.

Alle Versionen von MySQL laufen durch unsere Standard-Tests und -Benchmarks, um sicherzustellen, dass man sie relativ sicher benutzen kann. Weil die Standard-Tests im Laufe der Zeit erweitert werden, um auf alle früher gefundenen Bugs zu prüfen, wird die Test-Suite immer besser.

Beachten Sie, dass alle Releases mindestens wie folgt getestet wurden:

- Mit der internen Test-Suite

  Diese ist Teil unseres Produktionssystems für einen Kunden. Sie besitzt viele Tabellen mit Hunderten Megabytes an Daten.

- Mit der MySQL-Benchmark-Suite

  Diese läßt eine Reihe gebräuchlicher Anfragen laufen. Das ist zusätzlich ein Test darauf, ob die letzten Optimierungen den

Code tatsächlich schneller gemacht haben. See Abschnitt 6.1.4, „Die MySQL-Benchmark-Suite“.

- Mit dem `crash-me`-Test

  Dieser Test versucht festzustellen, welche Features die Datenbank unterstützt und was ihre Fähigkeiten und Beschränkungen sind. See Abschnitt 6.1.4, „Die MySQL-Benchmark-Suite“.

Ein weiterer Test besteht darin, dass wir die neueste MySQL-Version in unserer internen Entwicklungsumgebung einsetzen, mindestens auf einer Maschine. Wir arbeiten hierbei mit mehr als 100 Gigabytes an Daten.

3.2.4. Installationslayouts

Dieser Abschnitt beschreibt das vorgabemäßige Layout der Verzeichnisse, die durch die Installation von Binär- und Quelldistributionen angelegt werden.

Eine Binärdistribution wird installiert, indem sie an die Installationsstelle entpackt wird, die Sie auswählen (typischer Weise `/usr/local/mysql`). Die Installation erstellt folgende Verzeichnisse an dieser Stelle:

Verzeichnis	Verzeichnisinhalt
`bin`	Client-Programme und der `mysqld`-Server
`data`	Log-Dateien, Datenbanken
`include`	Include-(Header)-Dateien
`lib`	Bibliotheken
`scripts`	`mysql_install_db`
`share/mysql`	Dateien mit Fehlernachrichten
`sql-bench`	Benchmarks

Eine Quelldistribution wird installiert, nachdem Sie sie konfiguriert und kompiliert haben. Vorgabemäßig werden Dateien unter `/usr/local` installiert, und zwar in den folgenden Unterverzeichnissen:

Verzeichnis	Verzeichnisinhalt
`bin`	Client-Programme und -Skripte
`include/mysql`	Include-(Header)-Dateien
`info`	Dokumentation im Info-Format
`lib/mysql`	Bibliotheken
`libexec`	Der `mysqld`-Server
`share/mysql`	Dateien mit Fehlernachrichten
`sql-bench`	Benchmarks und `crash-me`-Test
`var`	Datenbanken und Log-Dateien

Innerhalb eines Installationsverzeichnisses weicht das Layout einer Quellinstallation von dem einer Binärinstallation wie folgt ab:

- Der `mysqld`-Server wird in das `libexec`-Verzeichnis installiert und nicht in das `bin`-Verzeichnis.
- Das Daten-Verzeichnis ist `var` und nicht `data`.
- `mysql_install_db` wird in das `/usr/local/bin` Verzeichnis installiert und nicht in `/usr/local/mysql/Skripts`.
- Die Header-Datei und Bibliotheksverzeichnisse sind `include/mysql` und `lib/mysql` und nicht `include` und `lib`.

Sie können Ihre eigene Binärinstallation aus einer kompilierten Quelldistribution erzeugen, indem Sie das Skript `Skripts/make_binary_Distribution` ausführen.

3.2.5. Wann und wie Updates veröffentlicht werden

MySQL entwickelt sich ziemlich schnell hier bei MySQL AB und wir wollen, dass andere MySQL-Benutzer daran teilhaben. Wir versuchen, immer dann ein neues Release heraus zu bringen, wenn wir sehr nützliche Features haben, für die offensichtlich ein Bedarf besteht.

Auch versuchen wir, unseren Benutzern zu helfen, wenn Sie nach Features anfragen, die einfach zu implementieren sind. Wir notieren, was unsere lizensierten Nutzer haben wollen, und insbesondere, was unsere Benutzer mit erweitertem E-Mail-Support haben wollen, und versuchen ihnen, eben das zu bieten.

Niemand muss einen neuen Release herunter laden. Im News-Abschnitt steht stets, ob das neue Release etwas beinhaltet, was Sie wirklich brauchen. See Anhang D, *MySQL-Änderungsverlauf (Change History)*.

Wenn wir MySQL aktualisieren, fahren wir folgende Politik:

- Bei kleineren Updates wird die letzte Zahl (von rechts) in der Versionsnummer herauf gezählt (Minor Release). Wenn es größere neue Features gibt oder kleinere Inkompatibilitäten mit vorherigen Versionen, wird die zweite Zahl der Versionsnummer herauf gezählt (Major Release). Wenn sich das Dateiformat ändert, wird die erste Zahl herauf gezählt.
- have to do with "small bugs" => minor releases? Stable tested releases are meant to appear about 1-2 times a year, but if small bugs are found, a release mit only bug fixes will be released. Als stabil getestete Releases sollten etwa ein- bis zweimal im Jahr erscheinen, aber wenn kleinere Fehler gefunden werden, wird nur ein Release mit Bug-Fixes heraus gegeben.
- Funktionierende Releases sollten etwa alle 1 bis 8 Wochen erscheinen.
- Binärdistributionen für einige Plattformen werden von uns für größere Releases (Major) heraus gegeben. Andere Leute stellen vielleicht auch Binärdistributionen für andere Systeme her, aber nicht so häufig.
- Patches stellen wir üblicherweise zur Verfügung, sobald wir kleinere Bugs ausfindig gemacht und behoben haben.
- Für nicht kritische, aber störende Bugs machen wir Patches verfügbar, wenn sie uns zugesandt werden. Ansonsten kombinieren wir mehrere davon in einem größeren Patch.
- Wenn durch unglückliche Umstände ein Release einen schweren Fehler enthält, erstellen wir sobald wie möglich ein neues Release. Das würden wir auch gern bei anderen Unternehmen so sehen.

The current stable release ist Version 3.23; We have already moved active Entwicklung to Version 4.0. Bugs will still be fixed in the stable version. We don't believe in a complete freeze, as this also leaves out bug fixes und things that ``must be done.'' ``Somewhat frozen'' means that we may add small things that ``almost surely will not affect anything that's already working.''

3.2.6. MySQL-Binärdistributionen, die von MySQL AB kompiliert wurden

Als Service stellen wir bei MySQL AB einen Satz von Binärdistributionen von MySQL zur Verfügung, die auf unserer Site kompiliert wurden oder auf Sites von Kunden, die uns freundlicherweise Zugang zu Ihren Maschinen gewährt haben.

Diese Distributionen werden mit `Skripts/make_binary_distribution` erzeugt und mit folgenden Kompilern und Optionen konfiguriert:

- SunOS 4.1.4 2 sun4c mit `gcc` 2.7.2.1

```
CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors" ./configure -
-prefix=/usr/local/mysql --disable-shared --with-extra-charsets=complex -
-enable-assembler
```

- SunOS 5.5.1 (und höher) sun4u mit `egcs` 1.0.3a oder 2.90.27 oder gcc 2.95.2 und neuer

```
CC=gcc CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -
fno-rtti" ./configure --prefix=/usr/local/mysql --with-low-memory -
-with-extra-charsets=complex --enable-assembler
```

- SunOS 5.6 i86pc mit `gcc` 2.8.1

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql --with-low-memory -
-with-extra-charsets=complex
```

- Linux 2.0.33 i386 mit `pgcc` 2.90.29 (`egcs` 1.0.3a)

```
CFLAGS="-O3 -mpentium -mstack-align-double" CXX=gcc CXXFLAGS="-O3 -mpentium -
mstack-align-double -felide-constructors -fno-exceptions -fno-rtti" ./configure -
-prefix=/usr/local/mysql --enable-assembler --with-mysqld-ldflags=-all-static -
```

```
-with-extra-charsets=complex
```

- Linux 2.2.x mit x686 mit `gcc` 2.95.2

```
CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro -felide-constructors -
fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql --enable-assembler -
-with-mysqld-ldflags=-all-static --disable-shared --with-extra-charset=complex
```

- SCO 3.2v5.0.4 i386 mit `gcc` 2.7-95q4

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql -
-with-extra-charsets=complex
```

- AIX 2 4 mit `gcc` 2.7.2.2

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql -
-with-extra-charsets=complex
```

- OSF1 V4.0 564 alpha mit `gcc` 2.8.1

```
CC=gcc CFLAGS=-O CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql -
-with-low-memory --with-extra-charsets=complex
```

- Irix 6.3 IP32 mit `gcc` 2.8.0

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql -
-with-extra-charsets=complex
```

- BSDI BSD/OS 3.1 i386 mit `gcc` 2.7.2.1

```
CC=gcc CXX=gcc CXXFLAGS=-O ./configure --prefix=/usr/local/mysql -
-with-extra-charsets=complex
```

- BSDI BSD/OS 2.1 i386 mit `gcc` 2.7.2

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql -
-with-extra-charsets=complex
```

Wenn jemand optimalere Optionen für die obigen Konfigurationen hat, können diese jederzeit der Entwickler-Mailing-Liste unter `<internals@lists.mysql.com>` mitgeteilt werden.

RPM-Distributionen von MySQL-Version 3.22 wurden durch Benutzer beigesteuert. Ab Version 3.22 werden die RPMs von uns bei MySQL AB erzeugt.

Wenn Sie eine Debug-Version von MySQL kompilieren wollen, müssen Sie den oben genannten Kompilierzeilen `--with-debug` oder `--with-debug=full` hinzufügen und jegliche `-fomit-frame-pointer`-Optionen entfernen.

3.3. Installation der Quelldistribution

Bevor Sie mit der Quellinstallation fortfahren, sehen Sie nach, ob eine Binärdistribution für Ihre Plattform verfügbar ist, die so wie Sie wollen funktioniert. Wir geben uns viel Mühe, die Binärdistributionen mit den bestmöglichen Optionen zu bauen.

Sie benötigen folgende Werkzeuge, um MySQL aus der Quelldistribution zu bauen und zu installieren:

- GNU `gunzip`, um die Distribution zu entpacken.
- Ein vernünftiges `tar`, um die Distribution zu entpacken. Von GNU `tar` ist bekannt, dass es funktioniert. Sun `tar` ist dafür bekannt, dass es Probleme verursacht.
- Einen funktionierenden ANSI-C++-Kompiler. `gcc` >= 2.95.2, `egcs` >= 1.0.2 oder `egcs 2.91.66`, SGI C++ und SunPro C++ sind einige der Kompiler, von denen bekannt ist, dass sie funktionieren. `libg++` wird nicht benötigt, wenn Sie `gcc` benutzen. `gcc` 2.7.x hat einen Bug, der es verunmöglicht, einige perfekt der vorgeschriebenen Form entsprechende C++-Dateien zu kompilieren, zum Beispiel `sql/sql_base.cc`. Wenn Sie nur `gcc` 2.7.x zur Verfügung haben, müssen Sie Ihren `gcc` aktualisieren, um MySQL kompilieren zu können. `gcc` 2.8.1 ist ebenfalls für Probleme auf einigen Plattformen bekannt, daher sollten Sie auch diesen vermeiden, wenn Sie einen neueren Kompiler für diese Plattform zur Verfügung haben.

  `gcc` >= 2.95.2 wird für das Kompilieren von MySQL-Versionen 3.23.x empfohlen.
- Ein gutes `make`-Programm. GNU `make` wird stets empfohlen und ist manchmal erforderlich. Wenn Sie Probleme bekommen,

empfehlen wir, es mit GNU `make` 3.75 oder neuer zu versuchen.

Wenn Sie eine aktuelle Version von **gcc** verwenden (aktuell genug, um die `-fno-exceptions`-Option zu verstehen), ist es **SEHR WICHTIG**, dass Sie diese Option benutzen. Ansonsten könnte es sein, dass Sie eine Binärdatei kompilieren, die zu zufälligen Zeitpunkten abstürzt. Wir empfehlen zusätzlich, dass Sie `-felide-contructors` und `-fno-rtti` zusammen mit `-fno-exceptions` benutzen. Im Zweifel gehen Sie wie folgt vor:

```
CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysq
```

Für die meisten Systeme werden Sie dadurch eine schnelle, stabile Binärinstallation erhalten.

Wenn Sie Probleme bekommen, **BITTE BENUTZEN SIE IMMER `mysqlbug`** zum Fragenstellen die Liste `<mysql@lists.mysql.com>`. Selbst wenn das Problem kein Bug ist, sammelt `mysqlbug` Systeminformationen, die anderen helfen werden, Ihr Problem zu lösen. Wenn Sie `mysqlbug` nicht benutzen, verringern Sie die Möglichkeit, eine Lösung Ihres Problems zu bekommen! `mysqlbug` finden Sie im `scripts`-Verzeichnis, nachdem Sie die Distribution entpackt haben. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.

3.3.1. Schnellinstallation, Überblick

Die grundlegenden Befehle, die Sie ausführen müssen, um eine MySQL-Quelldistribution zu installieren, sind:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> gunzip < mysql-VERSION.tar.gz | tar -xvf -
shell> cd mysql-VERSION
shell> ./configure --prefix=/usr/local/mysql
shell> make
shell> make install
shell> scripts/mysql_install_db
shell> chown -R root  /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
```

Wenn Sie Unterstützung für InnoDB-Tabellen haben wollen, sollten Sie die Datei `/etc/my.cnf` editieren und die ‘#’-Zeichen vor den Parametern entfernen, der mit `innodb_...` beginnen. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“. See Abschnitt 8.5.2, „Mit InnoDB anfangen - Optionen“.

Wenn Sie mit einem Quell-RPM anfangen, gehen Sie wie folgt vor:

```
shell> rpm --rebuild MySQL-VERSION.src.rpm
```

Das erzeugt ein Binär-RPM, das Sie installieren können.

Sie können neue Benutzer hinzufügen, indem Sie das `bin/mysql_setpermission`-Skript benutzen, falls Sie die `DBI`- und `Msql-Mysql-modules`-Perl-Module installieren.

Eine detailliertere Beschreibung folgt.

Um eine Quelldistribution zu installieren, führen Sie die unten stehenden Schritte aus und gehen dann weiter zu Abschnitt 3.4, „Einstellungen und Tests nach der Installation“, um die Schritte nach der Installation und ein paar Tests durchzuführen.

1. Wählen Sie das Verzeichnis, in dem Sie die Distribution entpacken wollen, und wechseln Sie dort hinein.

2. Holen Sie sich eine Distributionsdatei von einer der Sites, die unter Abschnitt 3.2.1, „Wie man MySQL erhält“ aufgelistet sind.

3. Wenn Sie Berkeley-DB-Tabellen mit MySQL verwenden wollen, müssen Sie sich eine gepatchte Version des Berkeley-DB-Quellcodes besorgen. Bitte lesen Sie das Kapitel über Berkeley-DB-Tabellen, bevor Sie fortfahren. See Abschnitt 8.6, „BDB- oder Berkeley_db-Tabellen“.

   MySQL-Quelldistributionen stehen als komprimierte `tar`-Archive zur Verfügung und haben Namen wie `mysql-VERSION.tar.gz`, wobei `VERSION` eine Zahl ist, wie 5.0.6-beta.

4. Fügen Sie einen Benutzer (User) und eine Gruppe (Group) hinzu, unter dem / der `mysqld` laufen soll:

   ```
   shell> groupadd mysql
   shell> useradd -g mysql mysql
   ```

Diese Befehle fügen den Benutzer `mysql` und die Gruppe `mysql` hinzu. Die Syntax für `useradd` und `groupadd` kann sich auf unterschiedlichen Unix-Systemen geringfügig unterscheiden. Die Befehle können `adduser` und `addgroup` heißen. Wenn Sie wollen, können Sie Benutzer und Gruppe auch anders nennen als `mysql`.

5. Entpacken Sie die Distribution ins aktuelle Verzeichnis:

```
shell> gunzip < /pfad/zu/mysql-VERSION.tar.gz | tar xvf -
```

Dieser Befehl erzeugt ein Verzeichnis namens `mysql-VERSION`.

6. Wechseln Sie in das oberste Verzeichnis der entpackten Distribution:

```
shell> cd mysql-VERSION
```

Beachten Sie, dass Sie aktuell MySQL aus diesem obersten Verzeichnis konfigurieren und bauen müssen. Sie können MySQL nicht in ein anderes Verzeichnis bauen.

7. Konfigurieren Sie das Release und kompilieren Sie alles:

```
shell> ./configure --prefix=/usr/local/mysql
shell> make
```

Wenn Sie `configure` laufen lassen, können Sie dabei einige Optionen angeben. Geben Sie `./configure --help` ein, um eine Liste von Optionen zu erhalten. Abschnitt 3.3.3, „Typische `configure`-Optionen“ erörtert einige der nützlicheren Optionen.

Wenn `configure` fehlschlägt und Sie sich wegen Hilfe an `<mysql@lists.mysql.com>` wenden, geben Sie bitte alle Zeilen aus `config.log` an, von denen Sie annehmen, dass sie bei der Problembehebung hilfreich sein könnten. Fügen Sie auch die letzten Zeilen der Ausgabe von `configure` hinzu, wenn `configure` abbricht. Schicken Sie den Bug-Bericht ein, indem Sie das `mysqlbug`-Skript benutzen. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.

Wenn das Kompilieren fehlschlägt, sehen Sie unter Abschnitt 3.3.5, „Probleme beim Kompilieren?“ nach, was bei einer Reihe geläufiger Probleme hilft.

8. Installieren Sie alles:

```
shell> make install
```

Eventuell müssen Sie diesen Befehl als `root` ausführen.

9. Erzeugen Sie die MySQL-Berechtigungstabellen (Grant Tables, nur notwendig, wenn Sie MySQL noch nie vorher installiert haben):

```
shell> scripts/mysql_install_db
```

Beachten Sie, dass bei MySQL-Versionen vor Version 3.22.10 der MySQL-Server startet, wenn Sie `mysql_install_db` laufen lassen. Das gilt für neuere Versionen nicht mehr!

10. Ändern Sie den Besitzer der Binärdateien zu `root` und den Besitzer des Daten-Verzeichnisses zu dem Benutzer, unter dem Sie `mysqld` laufen lassen wollen:

```
shell> chown -R root  /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql
```

Der erste Befehl ändert die `owner`-Attribute der Dateien auf den Benutzer `root`, der zweite ändert die `owner`-Attribute des Daten-Verzeichnisses auf den Benutzer `mysql` und der dritte ändert die `group`-Attribute auf die Gruppe `mysql`.

11. Wenn Sie die Unterstützung für die Perl-`DBI`/`DBD`-Schnittstelle hinzufügen wollen, sehen Sie unter Abschnitt 9.2, „MySQL-Perl-API“ nach.

12. Wenn Sie wollen, dass MySQL automatisch startet, wenn Sie Ihre Maschine hoch fahren, kopieren Sie `support-files/mysql.server` an die Stelle, wo Ihr System seine Startdateien hat. Weitere Informationen finden Sie im `support-files/mysql.server`-Skript selbst sowie unter Abschnitt 3.4.3, „MySQL automatisch starten und anhalten“.

Nachdem alles installiert wurde, sollten Sie Ihre Distribution initialisieren und testen:

```
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
```

Wenn dieser Befehl sofort mit `mysqld daemon ended` fehlschlägt, finden Sie einige Informationen dazu in der Datei `mysql-Daten-Verzeichnis/'hostname'.err`. Der wahrscheinliche Grund ist der, dass bereits ein anderer `mysqld`-Server läuft. See Abschnitt 5.1.4, „Viele MySQL-Server auf derselben Maschine laufen lassen“.

See Abschnitt 3.4, „Einstellungen und Tests nach der Installation“.

3.3.2. Wie man Patches anwendet

Manchmal erscheinen Patches auf der Mailing-Liste oder werden auf Patches-Bereich auf der MySQL-Website eingestellt.

Um einen Patch aus der Mailing-Liste anzuwenden, speichern Sie die Nachricht, in der der Patch enthalten ist, in eine Datei. Wechseln Sie dann ins oberste Verzeichnis Ihres MySQL-Source-Trees und geben Sie folgende Befehle ein:

```
shell> patch -p1 < patch-datei-name
shell> rm config.cache
shell> make clean
```

Patches von der FTP-Site werden als Klartextdateien (Plain Text) oder als mit `gzip` komprimierte Dateien distribuiert. Ein Klartext-Patch wenden Sie genau so an, wie oben für die Patches von der Mailing-Liste beschrieben. Um ein komprimiertes Patch anzuwenden, wechseln Sie ins oberste Verzeichnis Ihres MySQL-Source-Trees und geben Sie folgende Befehle ein:

```
shell> gunzip < patch-datei-name.gz | patch -p1
shell> rm config.cache
shell> make clean
```

Nachdem Sie einen Patch angewendet haben, folgen Sie den Anweisungen für eine normale Installation vom Quellcode, indem Sie mit dem Schritt `./configure` anfangen. Nach dem Schritt `make install`, starten Sie den MySQL-Server neu.

Es kann sein, dass Sie jeden laufenden Server anhalten müssen, bevor Sie `make install` laufen lassen können. (Das machen Sie mit `mysqladmin shutdown`.) Einige Systeme lassen es nicht zu, dass eine neue Programmversion installiert wird, wenn diese eine Version ersetzt, die momentan ausgeführt wird.

3.3.3. Typische `configure`-Optionen

Das `configure`-Skript gibt Ihnen in großem Umfang Kontrolle über die Konfigurationsmöglichkeiten Ihrer MySQL-Distribution. Typischerweise machen Sie das unter Verwendung der Optionen auf der `configure`-Kommandozeile. Sie können ausserdem `configure` beeinflussen, indem Sie bestimmte Umgebungsvariablen benutzen. See Anhang F, *Umgebungsvariablen*. Um eine Liste der Optionen zu erhalten, die `configure` unterstützt, geben Sie folgendes ein:

```
shell> ./configure --help
```

Einige der gebräuchlicheren `configure`-Optionen sind im Folgenden beschrieben:

- Um nur die MySQL-Client Bibliotheken und Client-Programme und nicht den Server zu kompilieren, benutzen Sie die `--ohne-server`-Option:

  ```
  shell> ./configure --without-server
  ```

  Wenn Sie keinen C++-Kompiler haben, können Sie `mysql` nicht kompilieren (MySQL ist das einzige Client-Programm, das C++ erfordert). In diesem Fall können Sie den Code in `configure` entfernen, der auf den C++-Kompiler testet, und dann `./configure` mit der `--without-server`-Option eingeben. Dieser Kompilierschritt wird nach wie vor versuchen, `mysql` zu bauen, aber Sie können alle Warnungen zu `mysql.cc` ignorieren. (Wenn `make` anhält, versuchen Sie `make -k`, um ihm mitzuteilen, dass es mit dem Rest des Builds fortfahren soll, auch wenn Fehler auftreten.)

- Wenn Sie nicht wollen, dass Ihre Log-Dateien und Datenbankverzeichnisse unter `/usr/local/var` liegen, benutzen Sie ein `configure`-Kommando wie folgendes:

  ```
  shell> ./configure --prefix=/usr/local/mysql
  shell> ./configure --prefix=/usr/local \
             --localstatedir=/usr/local/mysql/data
  ```

  Der erste Befehl ändert das Installationspräfix, so dass alles unter `/usr/local/mysql` statt unter `/usr/local` installiert wird. Der zweite Befehl bewahrt das vorgabemäßige Installationspräfix, aber überschreibt die vorgabemäßige Stelle für Datenbankverzeichnisse (normalerweise `/usr/local/var`) und ändert sie zu `/usr/local/mysql/data`.

- Wenn Sie Unix benutzen und wollen, dass der MySQL-Socket an anderer Stelle liegt als vorgabemäßig (normalerweise im Verzeichnis `/tmp` oder `/var/run`), benutzen Sie ein `configure`-Kommando wie folgendes:

```
shell> ./configure --with-unix-socket-path=/usr/local/mysql/tmp/mysql.sock
```

Beachten Sie, dass die angegebene Datei mit einem absoluten Pfadnamen angegeben werden muss! Sie können den Speicherort von `mysql.sock` auch später noch ändern, indem Sie die MySQL Optionsdateien benutzen. See Abschnitt A.4.5, „Wie Sie die MySQL-Socket-Datei `/tmp/mysql.sock` schützen oder ändern“.

- Wenn Sie statisch gelinkte Programme kompilieren wollen (um zum Beispiel eine Binärdistribution zu machen, mehr Geschwindigkeit zu erhalten oder Probleme mit RedHat-Linux-Distributionen zu umgehen (Workaround)), geben Sie `configure` wie folgt ein:

```
shell> ./configure --with-client-ldflags=-all-static \
           --with-mysqld-ldflags=-all-static
```

- Wenn Sie `gcc` benutzen und `libg++` oder `libstdc++` nicht installiert haben, können Sie `configure` mitteilen, `gcc` als Ihren C++-Kompiler zu benutzen:

```
shell> CC=gcc CXX=gcc ./configure
```

Wenn Sie `gcc` als C++-Kompiler benutzen, versucht dieser nicht, `libg++` oder `libstdc++` zu linken.

Hier sind einige gebräuchliche Umgebungsvariablen, die man in Abhängigkeit vom verwendeten Kompiler setzen kann:

gcc 2.7.2.1	CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors"
egcs 1.0.3a	CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti"
gcc 2.95.2	CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro -felide-constructors -fno-exceptions -fno-rtti"
pgcc 2.90.29 oder newer	CFLAGS="-O3 -mpentiumpro -mstack-align-double" CXX=gcc CXXFLAGS="-O3 -mpentiumpro -mstack-align-double -felide-constructors -fno-exceptions -fno-rtti"

In den meisten Fällen erhalten Sie eine ziemlich optimale MySQL-Binärdatei, indem Sie die Optionen von weiter oben nutzen und die folgenden Optionen zur Konfigurationszeile hinzufügen:

```
--prefix=/usr/local/mysql --enable-assembler --with-mysqld-ldflags=-all-static
```

Die komplette Konfigurationszeile würde also etwa wie folgt aussehen (für alle aktuellen gcc-Versionen):

```
CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro -felide-constructors -fno-exceptions -fno-rtti" ./conf
```

Die Binärdistributionen, die wir auf der MySQL-Website unter http://www.mysql.com zur Verfügung stellen, sind allesamt mit voller Optimierung kompiliert und sollten daher für die meisten Benutzer perfekt sein. See Abschnitt 3.2.6, „MySQL-Binärdistributionen, die von MySQL AB kompiliert wurden“. Einiges können Sie noch fein justieren, um noch schnellere Binärdistributionen zu erhalten, aber das ist nur etwas für fortgeschrittene Benutzer. See Abschnitt 6.5.3, „Wie Kompilieren und Linken die Geschwindigkeit von MySQL beeinflusst“.

Wenn der Build fehlschlägt und Fehler produziert, die aussagen, dass Ihr Kompiler oder Linker nicht in der Lage ist, die gemeinsam benutzte (shared) Bibliothek `libmysqlclient.so.#` (‘#’ ist eine Versionsnummer) zu erzeugen, können Sie dieses Problem umgehen, indem Sie die `--disable-shared`-Option von `configure` benutzen. In diesem Fall baut `configure` keine gemeinsam benutzte `libmysqlclient.so.#`-Bibliothek.

- Sie können MySQL so konfigurieren, dass keine `DEFAULT`-Spaltenwerte für Nicht-`NULL`-Spalten benutzt werden (also Spalten, bei denen nicht zulässig ist, dass sie `NULL` sind). Das führt dazu, dass `INSERT`-Statements einen Fehler erzeugen, ausser wenn ausdrücklich Werte für Spalten angegeben werden, die einen Nicht-`NULL`-Werte verlangen. Um die Benutzung von Vorgabewerten zu unterdrücken, geben Sie `configure` wie folgt ein:

```
shell> CXXFLAGS=-DDONT_USE_DEFAULT_FIELDS ./configure
```

- Als Vorgabe benutzt MySQL den Zeichensatz ISO-8859-1 (Latin1). Um diesen Vorgabesatz zu ändern, benutzen Sie die `--with-charset`-Option:

```
shell> ./configure --with-charset=CHARSET
```

`CHARSET` kann einer der folgenden sein: `big5`, `cp1251`, `cp1257`, `czech`, `danish`, `dec8`, `dos`, `euc_kr`, `gb2312`, `gbk`, `german1`, `hebrew`, `hp8`, `hungarian`, `koi8_ru`, `koi8_ukr`, `latin1`, `latin2`, `sjis`, `swe7`, `tis620`, `ujis`, `usa7` oder `win1251ukr`. See Abschnitt 5.6.1, „Der für Daten und Sortieren benutzte Zeichensatz“.

Wenn Sie Zeichen zwischen Server und Client konvertieren wollen, sollten Sie sich den `SET OPTION CHARACTER SET`-Befehl ansehen. See Abschnitt 6.5.6, „SET-Syntax“.

Achtung: Wenn Sie Zeichensätze ändern, nachdem Sie irgend welche Tabellen angelegt haben, müssen Sie `myisamchk -r -q` über jede Tabelle laufen lassen, denn ansonsten könnten Ihre Indexe falsch sortiert werden. (Das kann passieren, wenn Sie MySQL installieren, ein paar Tabellen erzeugen und danach MySQL rekonfigurieren, so dass es einen anderen Zeichensatz benutzt, und dann neu installieren.)

Mit der Option `--with-extra-charset=LIST` können Sie zusätzliche Zeichensätze definieren, die in den Server einkompiliert werden sollen.

Hierbei ist `LIST` entweder eine Liste eines Zeichensatzes, die durch Leerzeichen getrennt ist, oder `complex`, um alle Zeichen einzuschließen, die nicht dynamisch geladen werden können, oder `all`, um alle Zeichensätze in die Binärdateien einzuschließen.

- Um MySQL mit Debug-Code zu konfigurieren, benutzen Sie die `--with-debug`-Option:

```
shell> ./configure --with-debug
```

  Das bewirkt, dass eine sichere Speicherzuweisung (Memory Allocator) eingeschlossen wird, die einige Fehler finden kann und die Ausgaben liefert, was passiert ist. See Abschnitt E.1, „Einen MySQL-Server debuggen“.

- Wenn Ihre Client-Programme Threads benutzen, müssen Sie zusätzlich eine Thread-sichere Version der MySQL-Client-Bibliothek mit der `--enable-Thread-safe-client`-configure-Option kompilieren. Hierdurch wird eine `libmysqlclient_r`-Bibliothek angelegt, mit der Sie Ihre threaded Applikationen linken können. See Abschnitt 9.4.8, „Wie man einen threaded Client herstellt“.

- Optionen, die zu bestimmten Systemen gehören, finden sich im systemspezifischen Abschnitt dieses Handbuchs. See Abschnitt 3.2.2, „Betriebssysteme, die von MySQL unterstützt werden“.

3.3.4. Installation vom Entwicklungs-Source-Tree

VORSICHT: Sie sollten diesen Abschnitt nur lesen, wenn Sie daran interessiert sind, uns beim Testen von neuem Code zu helfen. Wenn Sie nur wollen, dass MySQL auf Ihrem System läuft, sollten Sie eine Standard-Distribution wählen (entweder eine Quell- oder eine Binärdistribution).

Um unseren aktuellsten Entwicklungs-Source-Tree zu bekommen, folgen Sie diesen Anweisungen:

1. Laden Sie **BitKeeper** von http://www.bitmover.com/cgi-bin/download.cgi herunter. Sie benötigen **Bitkeeper** 2.0 oder neuer, um auf unser Repository zuzugreifen.

2. Folgen Sie den Anweisungen, um BitKeeper zu installieren.

3. Nachdem **BitKeeper** installiert ist, benutzen Sie diesen Befehl, um den MySQL-3.23-Branch zu klonen:

```
shell> bk clone bk://mysql.bkbits.net/mysql-3.23 mysql-3.23
```

   Um den 4.0-Branch zu klonen, benutzen Sie statt dessen diesen Befehl:

```
shell> bk clone bk://mysql.bkbits.net/mysql-4.0 mysql-4.0
```

   Um den 4.1-Branch zu klonen, benutzen Sie statt dessen diesen Befehl:

```
shell> bk clone bk://mysql.bkbits.net/mysql-4.1 mysql-4.1
```

   Um den 5.0-Branch zu klonen, benutzen Sie statt dessen diesen Befehl:

```
shell> bk clone bk://mysql.bkbits.net/mysql-5.0 mysql-5.0
```

   Das erstmalige Herunterladen des Source-Trees kann eine Weile dauern, abhängig von Ihrer Verbindungsgeschwindigkeit. Bitte Geduld.

4. Sie brauchen GNU `autoconf`, `automake`, `libtool` und `m4`, um die nächsten Befehle auszuführen. Wenn Sie in diesem Stadium seltsame Fehler erhalten, überprüfen Sie bitte, ob Sie wirklich `libtool` installiert haben!

```
shell> cd mysql
shell> bk -r edit
```

```
shell> aclocal; autoheader; autoconf;  automake;
shell> ./configure  # Geben Sie hier Ihre Lieblingsoptionen an
shell> make
```

Eine Sammlung unserer Standard-configure-Skripts befindet sich im `BUILD/` Unterverzeichnis. Wenn Sie faul sind, können Sie `BUILD/compile-pentium-debug` benutzen. Um für unterschiedliche Architekturen zu kompilieren, ändern Sie das Skript ab und entfernen die Flags, die Pentium-spezifisch sind.

5. Wenn der Build fertig ist, lassen Sie `make install` laufen. Seien Sie damit vorsichtig auf Produktionsmaschinen, denn dieser Befehl kann Ihre Live-Release-Installation überschreiben! Wenn Sie eine weitere Installation von MySQL haben, empfehlen wir, dass Sie `./configure` mit anderen Werten für die `prefix`-, `tcp-port`- und `unix-socket-path`-Optionen ausführen als die, die für Ihren Produktionsserver benutzt werden.

6. Spielen Sie reichlich mit Ihrer neuen Installation herum und versuchen Sie, die neuen Features zum Absturz zu bringen. Fangen Sie an, indem Sie `make test` laufen lassen. See Abschnitt 10.3.2, „MySQL-Test-Suite“.

7. Wenn Sie bis zum `make`-Stadium gekommen sind und die Distribution sich nicht kompilieren läßt, berichten Sie das bitte an `<bugs@lists.mysql.com>`. Wenn Sie die letzten Versionen der erforderlichen GNU-Werkzeuge installiert haben und sie abstürzen, wenn Sie versuchen, Ihre Konfigurationsdateien zu verarbeiten, berichten Sie das bitte ebenfalls. Wenn Sie jedoch `aclocal` und einen `Befehl nicht gefunden`-Fehler erhalten, berichten Sie diesen nicht. Stellen Sie statt dessen sicher, dass alle notwendigen Werkzeuge installiert sind und dass Ihre `PATH`-Variable korrekt gesetzt ist, damit Ihre Shell diese finden kann.

8. Nach der erstmaligen `bk clone`-Operation, um den Source-Tree zu erhalten, sollten Sie in regelmäßigen Abständen `bk pull` laufen lassen, um Aktualisierungen zu erhalten.

9. Sie erhalten die Änderungen-Geschichte (Change History) des Trees mit allen Diffs, indem Sie `bk sccstool` benutzen. Wenn Sie seltsame Diffs sehen oder Code, zu dem Sie Fragen haben, zögern Sie nicht, uns eine E-Mail an `<internals@lists.mysql.com>` zu schicken. Auch wenn Sie meinen, eine bessere Idee zu haben, wie etwas gemacht werden sollte, schicken Sie uns eine E-Mail an dieselbe Adresse, mit einem Patch. `bk diffs` erzeugt ein Patch für Sie, nachdem Sie Änderungen am Quellcode durchgeführt haben. Wenn Sie keine Zeit haben, Ihre Idee zu kodieren, schicken Sie einfach eine Beschreibung.

10. **BitKeeper** hat ein nettes Hilfe-Dienstprogramm, auf das Sie über `bk helptool` zugreifen können.

3.3.5. Probleme beim Kompilieren?

Alle MySQL-Programme lassen sich sauber ohne Warnungen auf Solaris mit `gcc` kompilieren. Auf anderen Systemen können Warnungen wegen Unterschieden in System-Include-Dateien auftreten. Siehe Abschnitt 3.3.6, „Anmerkungen zu MIT-pThreads“ wegen Warnungen, die auftreten können, wenn Sie MIT-pThreads verwenden. Wegen anderer Probleme sehen Sie bitte in der unten stehenden Liste nach.

Die Lösung für viele Probleme beinhaltet Rekonfigurieren. Wenn Sie rekonfigurieren müssen, beachten Sie Folgendes:

- Wenn `configure` laufen gelassen wird, nachdem es schon einmal lief, benutzt es möglicherweise Informationen, die bei vorherigen Aufrufen gesammelt wurden. Diese Information wird in der Datei `config.cache` gespeichert. Wenn `configure` startet, sucht es diese Datei und liest ihren Inhalt, wenn sie existiert, unter der Annahme, dass diese Information immer noch stimmt. Diese Annahme ist falsch, wenn Sie rekonfigurieren.

- Immer, wenn Sie `configure` laufen lassen, müssen Sie auch `make` laufen lassen, um erneut zu kompilieren. Sie werden jedoch einige alte Objektdateien vorheriger Builds entfernen wollen, denn diese wurden mit anderen Konfigurationsoptionen kompiliert.

Um zu verhindern, dass alte Konfigurationsinformationen oder Objektdateien benutzt werden, geben Sie vor dem erneuten Aufruf von `configure` folgende Befehle ein:

```
shell> rm config.cache
shell> make clean
```

Alternativ können Sie auch `make distclean` laufen lassen.

Die unten stehende Liste beschreibt einige der Probleme, die beim Kompilieren von MySQL am häufigsten auftreten:

- Wenn Sie Probleme beim Kompilieren von `sql_yacc.cc` erhalten, die den unten gezeigten ähneln, haben Sie wahrscheinlich keinen Arbeitsspeicher oder Swap-Platz (Auslagerungsdatei) mehr.

```
Internal compiler error: Programm cc1plus got fatal signal 11
```

```
  oder
Out of virtual memory
  oder
Virtual memory exhausted
```

Das Problem liegt darin, dass `gcc` riesige Mengen von Arbeitsspeicher benötigt, um `sql_yacc.cc` mit Inline-Funktionen zu kompilieren. Versuchen Sie, `configure` mit der `--with-low-memory`-Option auszuführen:

```
shell> ./configure --with-low-memory
```

Diese Option veranlasst, dass `-fno-inline` zur Kompilierzeile hinzugefügt wird, wenn Sie `gcc` benutzen, bzw. `-O0`, wenn Sie etwas anderes benutzen. Sie sollten die `--with-low-memory`-Option selbst dann benutzen, wenn Sie glauben, so viel Arbeitsspeicher und Swap-Platz zu haben, dass Ihnen diese unmöglich ausgehen können. Das Problem wurde selbst auf Systemen mit großzügiger Hardware-Ausstattung beobachtet, und die `--with-low-memory`-Option behebt es üblicherweise.

- Vorgabemäßig sucht `configure` `c++` als Kompiler-Namen aus und GNU `c++` linkt mit `-lg++`. Wenn Sie `gcc` benutzen, kann dieses Verhalten Probleme bei Konfigurationen wie dieser verursachen:

```
configure: error: installation oder configuration problem:
c++ compiler cannot create executables.
```

Eventuell stoßen Sie beim Kompilieren auch auf Probleme, die mit `g++`, `libg++` oder `libstdc++` zu tun haben.

Eine Ursache dieser Probleme liegt darin, dass Sie kein `g++` haben dürfen, oder Sie dürfen `g++` haben, aber nicht `libg++` oder `libstdc++`. Schauen Sie in die `config.log`-Datei! Sie sollten die genaue Ursache enthalten, warum Ihr C++-Kompiler nicht funktioniert! Um dieses Problem zu umgehen, können Sie `gcc` als Ihren C++-Kompiler benutzen. Versuchen Sie, die Umgebungsvariable `CXX` auf `"gcc -O3"` zu setzen. Beispiel:

```
shell> CXX="gcc -O3" ./configure
```

Das funktioniert, weil `gcc` C++-Quellen genau so gut wie `g++` kompiliert, aber vorgabemäßig weder `libg++` noch `libstdc++` linkt.

Eine andere Möglichkeit, das Problem zu beheben, besteht natürlich darin, `g++`, `libg++` und `libstdc++` zu installieren.

- Wenn Ihr Kompilieren mit Fehlern wie dem folgenden fehlschlägt, müssen Sie Ihre Version von `make` auf GNU `make` aktualisieren:

```
making all in mit-pThreads
make: Fatal error in reader: Makefile, line 18:
Badly formed macro assignment
  oder
make:Datei `Makefile' line 18: Must be a separator (:
  oder
pThread.h: No such file or directory
```

Von Solaris und FreeBSD ist bekannt, dass sie problembehaftete `make`-Programme haben.

GNU `make` Version 3.75 funktioniert bekanntermaßen.

- Wenn Sie Flags definieren wollen, die von Ihrem C- oder C++-Kompiler benutzt werden, fügen Sie die Flags den `CFLAGS`- und `CXXFLAGS`-Umgebungsvariablen hinzu. Sie können auf diese Weise auch die Kompilernamen festlegen, indem Sie `CC` und `CXX` benutzen. Beispiel:

```
shell> CC=gcc
shell> CFLAGS=-O3
shell> CXX=gcc
shell> CXXFLAGS=-O3
shell> export CC CFLAGS CXX CXXFLAGS
```

Siehe Abschnitt 3.2.6, „MySQL-Binärdistributionen, die von MySQL AB kompiliert wurden“: Eine Liste von Flag-Definitionen, die sich auf verschiedenen Systemen als nützlich erwiesen haben.

- Wenn Sie einen Fehler wie den folgenden erhalten, müssen Sie Ihren `gcc`-Kompiler aktualisieren:

```
client/libmysql.c:273: parse error before `__attribute__'
```

`gcc` 2.8.1 funktioniert bekanntermaßen, aber wir empfehlen statt dessen `gcc` 2.95.2 oder `egcs` 1.0.3a.

- Wenn Sie Fehler wie die unten stehenden erhalten, wenn Sie `mysqld` kompilieren, hat `configure` den Typ des letzten

Arguments für `accept()`, `getsockname()` oder `getpeername()` nicht korrekt erkannt:

```
cxx: Error: mysqld.cc, line 645: In this statement, the referenced
     type of the pointer value "&length" is "unsigned long", which
     is not compatible with "int".
new_sock = accept(sock, (struct sockaddr *)&cAddr, &length);
```

Um das zu beheben, editieren Sie die `config.h`-Datei (die von `configure` angelegt wird). Suchen Sie nach folgenden Zeilen:

```
/* Define as the base type of the last arg to accept */
#define SOCKET_SIZE_TYPE XXX
```

Ändern Sie `XXX` zu `size_t` oder `int`, abhängig von Ihrem Betriebssystem. (Beachten Sie, dass Sie das jedes Mal tun müssen, wenn Sie `configure` laufen lassen, weil `configure` die Datei `config.h` neu erzeugt.)

- Die `sql_yacc.cc`-Datei wird von `sql_yacc.yy` erzeugt. Normalerweise muss der Build-Prozess keine `sql_yacc.cc` erzeugen, weil MySQL schon mit einer fertig erzeugten Kopie daher kommt. Wenn Sie sie jedoch neu erzeugen müssen, könnten Sie folgenden Fehler erhalten:

  ```
  "sql_yacc.yy", line xxx fatal: default action causes potential...
  ```

  Das ist ein Indiz dafür, dass Ihre Version von `yacc` fehlerhaft ist. Sie müssen statt dessen wahrscheinlich `bison` (die GNU-Version von `yacc`) installieren und benutzen.

- Wenn Sie `mysqld` oder einen MySQL-Client debuggen wollen, lassen Sie `configure` mit der `--with-debug`-Option laufen. Kompilieren Sie danach neu und linken Sie Ihre Clients mit der neuen Client-Bibliothek. See Abschnitt E.2, „Einen MySQL-Client debuggen“.

3.3.6. Anmerkungen zu MIT-pThreads

Dieser Abschnitt beschreibt einige der Themen im Zusammenhang mit MIT-pThreads.

Beachten Sie, dass Sie auf Linux KEINE MIT-pThreads benutzen, sondern statt dessen LinuxThreads installieren sollten! See Abschnitt 3.6.1, „Linux (alle Linux-Versionen)“.

Wenn Ihr System keine native Thread-Unterstützung bietet, müssen Sie MySQL unter Verwendung des MIT-pThread-Pakets bauen. Das betrifft ältere FreeBSD-Systeme, SunOS 4.x, Solaris 2.4 und früher und einige andere. See Abschnitt 3.2.2, „Betriebssysteme, die von MySQL unterstützt werden“.

- Auf den meisten Systemen können Sie die Benutzung von erzwingen, indem Sie `configure` mit der `--with-mit-Threads`-Option laufen lassen:

  ```
  shell> ./configure --with-mit-threads
  ```

  Wenn Sie MIT-pThreads benutzen, wird das Bauen (Building) in ein Nicht-Quellcode-Verzeichnis nicht unterstützt, weil wir die Änderungen an diesem Code minimal halten wollen.

- Die Überprüfungen, die festlegen, ob MIT-pThreads benutzt werden sollten oder nicht, finden nur in dem Teil des Konfigurationsprozesses statt, der mit dem Server-Code zu tun hat. Wenn Sie die Distribution mit `--without-server` konfigurieren, um nicht den Client-Code zu bauen, wissen die Clients nicht, ob sie MIT-pThreads benutzen sollen oder nicht und werden vorgabemäßig Unix-Socket-Verbindungen benutzen. Weil Unix-Sockets unter MIT-pThreads nicht laufen, heißt das, dass Sie `-h` oder `--host` benutzen müssen, wenn Sie Client-Programme laufen lassen.

- Wenn MySQL so kompiliert wird, dass es MIT-pThreads benutzt, wird System-Sperren (System Locking) vorgabemäßig aus Performance-Gründen ausgeschaltet. Mit der `--use-locking`-Option können Sie dem Server mitteilen, System-Sperren zu benutzen.

- Manchmal schlägt der pThread-`bind()`-Befehl fehl und bindet nicht an ein Socket, ohne jede Fehlermeldung (zumindest auf Solaris). Als Ergebnis schlagen alle Verbindungen zum Server fehl. Beispiel:

  ```
  shell> mysqladmin version
  mysqladmin: connect to server at '' failed;
  error: 'Can't connect to mysql server on localhost (146)'
  ```

  Die Lösung besteht darin, den `mysqld`-Server zu killen und neu zu starten. Uns ist das nur dann passiert, wenn wir den Server gezwungen haben, herunter zu fahren und sofort danach einen Neustart durchgeführt haben.

- Bei MIT-pThreads läßt sich der `sleep()`-Systemaufruf nicht mit `SIGINT` (break) unterbrechen. Das merken Sie nur, wenn Sie `mysqladmin --sleep` ausführen. Sie müssen dann warten, bis der `sleep()`-Aufruf beendet wurde, bevor die Unterbrechungsanforderung (Interrupt) bedient wird und der Prozess anhält.
- Wenn Sie linken, erhalten Sie möglicherweise Warnmeldungen wie diese (zumindest auf Solaris). Sie können sie ignorieren:

```
ld: warning: symbol `_iob' hat differing sizes:
    (file /my/local/pThreads/lib/libpThread.a(findfp.o) value=0x4;
file /usr/lib/libc.so value=0x140);
    /my/local/pThreads/lib/libpThread.a(findfp.o) definition taken
ld: warning: symbol `__iob' hat differing sizes:
    (file /my/local/pThreads/lib/libpThread.a(findfp.o) value=0x4;
file /usr/lib/libc.so value=0x140);
    /my/local/pThreads/lib/libpThread.a(findfp.o) definition taken
```

- Einige weitere Warnungen können ebenfalls ignoriert werden:

```
implicit declaration of function `int strtoll(...)'
implicit declaration of function `int strtoul(...)'
```

- Wir haben es bislang nicht geschafft, `readline` mit MIT-pThreads zum Laufen zu bringen. (Das wird zwar nicht benötigt, mag aber für einige interessant sein.)

3.3.6.1. Vorbereitung der Windows-Umgebung

3.3.7. Windows-Quelldistribution

Sie benötigen folgendes:

- VC++-6.0-Kompiler (aktualisiert mit Service-Pack 4 oder 5 und dem Präprozessor-Paket). Das Präprozessor-Paket wird für den Makro-Assembler benötigt. Weitere Details finden Sie unter: http://msdn.microsoft.com/vstudio/sp/vs6sp5/faq.asp.
- Die MySQL-Quelldistribution für Windows, die von http://www.mysql.com/downloads/ herunter geladen werden kann.

MySQL bauen

1. Erzeugen Sie ein Arbeitsverzeichnis (z. B. workdir).
2. Entpacken Sie die Quelldistribution in dieses Verzeichnis.
3. Starten Sie den VC++-6.0-Kompiler.
4. Wählen Sie im `File`-Menü `Open Workspace`.
5. Öffnen Sie den `mysql.dsw`-Workspace, den Sie im Arbeitsverzeichnis finden.
6. Wählen Sie im `Build`-Menü das `Set Active Configuration`- Menü.
7. Wählen Sie `mysqld - Win32 Debug` und klicken Sie auf OK.
8. Drücken Sie `F7`, um mit dem Bauen des Debug-Servers, der Bibliotheken und einiger Client-Applikationen zu beginnen.
9. Wenn das Kompilieren beendet ist, kopieren Sie die Bibliotheken und die ausführbaren Dateien in ein separates Verzeichnis.
10. Kompilieren Sie die Release-Versionen, die Sie haben wollen, auf dieselbe Art.
11. Erzeugen Sie das Verzeichnis für die MySQL-Dateien, z. B. `c:\mysql`.
12. Kopieren Sie aus dem Arbeitsverzeichnis folgende Verzeichnisse in das c:\mysql-Verzeichnis:
    - Data
    - Docs
    - Share
13. Erzeugen Sie das Verzeichnis `c:\mysql\bin` und kopieren Sie alle Server und Clients, die Sie vorher kompiliert haben, hinein.

14. Wenn Sie wollen, können Sie auch das `lib`-Verzeichnis erzeugen und die vorher kompilierten Bibliotheken hinein kopieren.

15. Führen Sie mit Visual Studio ein Clean durch.

Konfigurieren und starten Sie den Server auf dieselbe Weise wie bei der Windows-Binärdistribution. See Abschnitt 3.3.6.1, „Vorbereitung der Windows-Umgebung“.

3.4. Einstellungen und Tests nach der Installation

Wenn Sie MySQL erst einmal installiert haben (aus einer Binär- oder einer Quelldistribution), müssen Sie die Berechtigungstabellen (Grant Tables) initialisieren, den Server starten und sicherstellen, dass der Server korrekt funktioniert. Eventuell wollen Sie auch einrichten, dass der Server automatisch gestartet und angehalten wird, wenn Ihr System startet oder herunter gefahren wird.

Normalerweise installieren Sie die Berechtigungstabellen und starten den Server wie folgt: Bei der Installation einer Quelldistribution:

```
shell> ./scripts/mysql_install_db
shell> cd mysql_installations_verzeichnis
shell> ./bin/safe_mysqld --user=mysql &
```

Bei einer Binärdistribution (nicht RPM- oder pkg-Pakete) tun Sie folgendes:

```
shell> cd mysql_installations_verzeichnis
shell> ./bin/mysql_install_db
shell> ./bin/safe_mysqld --user=mysql &
```

Das legt die `mysql`-Datenbank an, die alle Zugriffsrechte auf Datenbanken enthält, die `test`-Datenbank, die Sie benutzen können, um MySQL zu testen und zusätzlich Berechtigungseinträge für den Benutzer, der `mysql_install_db` ausführt sowie einen `root`-Benutzer (ohne Passworte!). Durch den letzten Befehl wird der `mysqld`-Server gestartet.

`mysql_install_db` überschreibt keine alten Berechtigungstabellen, deshalb sollte es unter allen Umständen sicher sein. Wenn Sie die `test`-Datenbank nicht haben wollen, können Sie sie mit `mysqladmin -u root drop test` entfernen.

Am einfachsten läßt sich das Durchtesten vom obersten Verzeichnis der MySQL-Distribution durchführen. Bei einer Binärdistribution ist das Ihr Installationsverzeichnis (üblicherweise etwas wie `/usr/local/mysql`). Bei einer Quelldistribution ist es das Hauptverzeichnis Ihres MySQL-Source-Trees.

In den unten dargestellten Befehlen dieses Abschnitts und der folgenden Unterabschnitte ist `BINDIR` der Pfad zu dem Speicherort, wo Programme wie `mysqladmin` und `safe_mysqld` installiert sind. Bei einer Binärdistribution ist das `bin`-Verzeichnis innerhalb der Distribution. Bei einer Quelldistribution ist `BINDIR` wahrscheinlich `/usr/local/bin`, es sei denn, Sie haben ein anderes Installationsverzeichnis als `/usr/local` angegeben, als Sie `configure` laufen ließen. `EXECDIR` ist der Speicherort, in dem der `mysqld`-Server installiert ist. Bei einer Binärdistribution ist das derselbe wie `BINDIR`. Bei einer Quelldistribution ist `EXECDIR` wahrscheinlich `/usr/local/libexec`.

Das Durchtesten wird im Folgenden detailliert beschrieben.

1. Falls notwendig, starten Sie den `mysqld`-Server und richten die anfänglichen MySQL-Berechtigungstabellen ein, die alle Zugriffsrechte enthalten, die festlegen, wie sich Benutzer mit dem Server verbinden dürfen. Das wird normalerweise mit dem `mysql_install_db`-Skript gemacht:

   ```
   shell> scripts/mysql_install_db
   ```

   Typischerweise müssen Sie `mysql_install_db` nur laufen lassen, wenn Sie MySQL zum ersten Mal installieren. Wenn Sie eine existierende Installation aktualisieren (Update), können Sie deshalb diesen Schritt überspringen. (`mysql_install_db` ist jedoch ziemlich sicher und aktualisiert keine bereits existierenden Tabellen, daher können Sie im Zweifel immer `mysql_install_db` laufen lassen.)

   `mysql_install_db` erzeugt sechs Tabellen (`user`, `db`, `host`, `tables_priv`, `columns_priv` und `func`) in der `mysql`-Datenbank. Eine Beschreibung der anfänglichen Zugriffsrechte wird in Abschnitt 5.2.5, „Wie das Berechtigungssystem funktioniert“ festgelegt. Kurz gesagt erlauben diese Zugriffsrechte dem MySQL-Benutzer `root`, alles zu tun, und jedem, Datenbanken anzulegen oder zu benutzen, deren Name `'test'` ist oder mit `'test_'` beginnt.

   Wenn Sie die Zugriffsberechtigungstabellen (Grant Tables) nicht einrichten, wird folgender Fehler in der Logdatei erscheinen, wenn Sie den Server starten:

   ```
   mysqld: Can't find file: 'host.frm'
   ```

Dasselbe kann auch bei einer MySQL-Binärdistribution passieren, wenn Sie MySQL nicht mit exakt `./bin/safe_mysqld` starten! See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.

Eventuell müssen Sie `mysql_install_db` als `root` laufen lassen. Wenn Sie wollen, können Sie jedoch den MySQL-Server als unprivilegierter (non-`root`)-Benutzer laufen lassen, vorausgesetzt, dieser Benutzer darf Dateien im Datenbank-Verzeichnis lesen und schreiben. Anweisungen, wie Sie MySQL als unprivilegierter Benutzer laufen lassen können, finden Sie in Abschnitt 5.3.3, „Wann Berechtigungsänderungen wirksam werden“.

Wenn Sie Probleme mit `mysql_install_db` bekommen, sehen Sie bitte unter Abschnitt 3.4.1, „Probleme mit `mysql_install_db`“ nach.

Es gibt eine Reihe von Alternativen zum Laufenlassen des `mysql_install_db`-Skripts, was mit der MySQL-Distribution mitgeliefert wird:

- Sie können `mysql_install_db` editieren, bevor Sie es laufen lassen, um die anfänglichen Zugriffsrechte zu ändern, die in die Rechtetabellen installiert werden. Das ist nützlich, wenn Sie MySQL auf einer großen Zahl von Maschinen mit denselben Zugriffsrechten installieren wollen. In diesem Fall müssen Sie wahrscheinlich nur ein paar zusätzliche `INSERT`-Statements für die `mysql.user`- und `mysql.db`-Tabellen hinzufügen!
- Wenn Sie Dinge in den Berechtigungstabellen ändern wollen, nachdem diese installiert wurden, lassen Sie `mysql_install_db` laufen und geben dann den Befehl `mysql -u root mysql` ein, um sich als MySQL-`root`-Benutzer mit den Berechtigungstabellen zu verbinden. Danach können Sie SQL-Statements eingeben, um die Tabellen direkt zu verändern.
- Es ist möglich, die Berechtigungstabellen komplett neu zu erzeugen, nachdem Sie angelegt wurden. Das werden Sie zum Beispiel tun wollen, wenn Sie die Tabellen bereits angelegt haben, Sie nun aber neu anlegen wollen, weil Sie `mysql_install_db` editiert haben.

Zu weiteren Informationen über diese Alternativen siehe Abschnitt 5.2, „Allgemeine Sicherheitsthemen und das MySQL-Zugriffsberechtigungssystem“.

2. Starten Sie den MySQL-Server wie folgt:

```
shell> cd mysql_installations_verzeichnis
shell> bin/safe_mysqld &
```

Wenn Sie Probleme haben, den Server zu starten, sehen Sie unter Abschnitt 3.4.2, „Probleme mit dem Start des MySQL-Servers“ nach.

3. Benutzen Sie `mysqladmin`, um sicherzustellen, dass der Server läuft. Die folgenden Befehle sind ein einfacher Test, um zu überprüfen, ob der Server läuft und auf Verbindungen reagiert:

```
shell> BINDIR/mysqladmin version
shell> BINDIR/mysqladmin variables
```

Die Ausgabe von `mysqladmin version` kann geringfügig variieren, abhängig von Ihrer Plattform und der Version von MySQL, sollte aber etwa wie folgt aussehen:

```
shell> BINDIR/mysqladmin version
mysqladmin  Ver 8.14 Distrib 3.23.32, for linux on i586
Copyright (C) 2000 MySQL AB & MySQL Finnland AB & TCX DataKonsult AB
This software comes mit ABSOLUTELY NO WARRANTY. This ist free software,
und you are welcome to modify und redistribute it under the GPL license

Server version          3.23.32-debug
Protokoll version        10
Connection              Localhost via Unix socket
TCP port                3306
UNIX socket             /tmp/mysql.sock
Uptime:                 16 sec

Threads: 1  Questions: 9  Slow queries: 0  Opens: 7  Flush tables: 2  Open tables: 0 Queries per second avg: 0.000
```

Um ein Gefühl dafür zu bekommen, was Sie sonst noch mit `BINDIR/mysqladmin` tun können, rufen Sie es mit der `--help`-Option auf.

4. Stellen Sie sicher, dass Sie den Server herunter fahren können:

```
shell> BINDIR/mysqladmin -u root shutdown
```

5. Stellen Sie sicher, dass Sie den Server erneut starten können. Tun Sie das unter Benutzung von `safe_mysqld` oder indem Sie `mysqld` direkt aufrufen. Beispiel:

```
shell> BINDIR/safe_mysqld --log &
```

Wenn `safe_mysqld` fehlschlägt, versuchen Sie, es vom MySQL-Installationsverzeichnis aus zu starten (falls Sie noch nicht dort sind). Wenn das nicht funktioniert, sehen Sie unter see Abschnitt 3.4.2, „Probleme mit dem Start des MySQL-Servers“ nach.

6. Lassen Sie ein paar einfache Tests ablaufen um sicherzustellen, dass der Server funktioniert. Die Ausgabe sollte ähnlich der folgenden sein:

```
shell> BINDIR/mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
+-----------+

shell> BINDIR/mysqlshow mysql
Datenbank: mysql
+--------------+
|    Tables    |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+

shell> BINDIR/mysql -e "select host,db,user from db" mysql
+------+--------+------+
| host | db     | user |
+------+--------+------+
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

Zusätzlich gibt es eine Benchmark-Suite im `sql-bench`-Verzeichnis (unterhalb des MySQL-Installationsverzeichnisses), die Sie benutzen können, um die Leistungsdaten von MySQL auf verschiedenen Plattformen zu vergleichen. Das `sql-bench/Results`-Verzeichnis enthält die Ergebnisse vieler Testläufe mit verschiedenen Datenbanken und Plattformen. Um alle Tests durchzuführen, geben Sie folgende Befehle ein:

```
shell> cd sql-bench
shell> run-all-tests
```

Wenn Sie kein `sql-bench`-Verzeichnis haben, benutzen Sie wahrscheinlich ein RPM für eine Binärdistribution. (Quelldistributions-RPMs beinhalten das Benchmark-Verzeichnis.) In diesem Fall müssen Sie die Benchmark-Suite zuerst installieren, bevor Sie sie benutzen können. Ab MySQL Version 3.22 gibt es Benchmark-RPM-Dateien, die `mysql-bench-VERSION-i386.rpm` benannt sind, die Benchmark-Code und Daten enthalten.

Wenn Sie eine Quelldistribution haben, können Sie auch die Tests im `tests`-Unterverzeichnis ausführen. Um beispielsweise `auto_increment.tst` auszuführen, geben Sie folgendes ein:

```
shell> BINDIR/mysql -vvf test < ./tests/auto_increment.tst
```

Die Ergebnisse stehen dann in der `./tests/auto_increment.res`-Datei.

3.4.1. Probleme mit `mysql_install_db`

Der Zweck des `mysql_install_db`-Skripts ist, neue MySQL-Berechtigungstabellen zu erzeugen. Es betrifft keine anderen Daten! Es tut nichts, wenn Sie bereits MySQL-Berechtigungstabellen installiert haben!

Wenn Sie Ihre Berechtigungstabellen neu erzeugen wollen, sollten Sie den `mysqld`-Server herunter fahren, falls er läuft, und dann etwas Ähnliches wie folgendes tun:

```
mv mysql-data-verzeichnis/mysql mysql-data-verzeichnis/mysql-old
mysql_install_db
```

Dieser Abschnitt listet Probleme auf, denen Sie vielleicht begegnen, wenn Sie `mysql_install_db` laufen lassen:

- **`mysql_install_db` installiert die Berechtigungstabellen nicht.**

  Eventuell stellen Sie fest, dass `mysql_install_db` bei der Installations der Berechtigungstabellen fehlschlägt und mit

folgenden Meldungen endet:

```
starting mysqld daemon with databases from XXXXXX
mysql daemon ended
```

In diesem Fall sollten Sie einen gründlichen Blick in die Log-Datei werfen! Diese sollte sich im Verzeichnis `XXXXXX` befinden, das in der Fehlermeldung ausgegeben wird, und sollte anzeigen, warum `mysqld` nicht startete. Wenn Sie nicht verstehen, was passiert ist, schicken Sie einen Bug-Bericht inklusive Log. Benutzen Sie hierfür `mysqlbug`! See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.

- **Es läuft bereits ein `mysqld`-Daemon.**

  In diesem Fall müssen Sie wahrscheinlich `mysql_install_db` überhaupt nicht starten. Sie müssen `mysql_install_db` nur einmal starten, und zwar, wenn Sie MySQL zum ersten Mal installieren.

- **Die Installation eines zweiten `mysqld`-Daemons schlägt fehl,**

  wenn bereits ein Daemon läuft.

  Das kann vorkommen, wenn Sie bereits eine existierende MySQL-Installation haben, aber eine neue Installation an einem anderen Speicherort unterbringen wollen (zum Beispiel für Testzwecke, oder vielleicht wollen Sie auch einfach zwei Installationen zugleich laufen lassen. Im Allgemeinen ist der Grund für das Problem, wenn Sie versuchen, den zweiten Server laufen zu lassen, dass der zweite Server versucht, denselben Socket und Port wie der alte zu benutzen. In diesem Fall erhalten Sie als Fehlermeldung: `Can't start server: Bind on TCP/IP port: Address already in use` oder `Can't start server : Bind on unix socket...`. See Abschnitt 5.1.4, „Viele MySQL-Server auf derselben Maschine laufen lassen“.

- **Sie haben keinen Schreibzugriff auf `/tmp`.**

  Wenn Sie keinen Schreibzugriff haben, um eine Socket-Datei am vorgabemäßigen Ort anzulegen (in `/tmp`) oder keine Berechtigung, um temporäre Dateien in `/tmp` anzulegen, erhalten Sie einen Fehler, wenn Sie `mysql_install_db` laufen lassen oder starten oder wenn Sie `mysqld` benutzen.

  So können Sie einen anderen Socket und ein anderes temporäres Verzeichnis festlegen:

```
shell> TMPDIR=/irgendein_temporaeres_verzeichnis/
shell> MYSQL_UNIX_PORT=/irgendein_temporaeres_verzeichnis/mysqld.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

  See Abschnitt A.4.5, „Wie Sie die MySQL-Socket-Datei `/tmp/mysql.sock` schützen oder ändern“.

  `irgendein_temporaeres_verzeichnis` sollte der Pfad zu einem Verzeichnis sein, für das Sie Schreibberechtigung haben. See Anhang F, *Umgebungsvariablen*.

  Danach sollten Sie in der Lage sein, `mysql_install_db` laufen zu lassen und den Server zu starten, und zwar mit folgenden Befehlen:

```
shell> scripts/mysql_install_db
shell> BINDIR/safe_mysqld &
```

- **`mysqld` stürzt sofort ab**

  Wenn Sie RedHat Version 5.0 mit einer Version von `glibc` laufen lassen, die älter als 2.0.7-5 ist, sollten Sie sicherstellen, dass Sie alle `glibc`-Patches installiert haben! Darüber gibt es jede Menge Informationen in den MySQL-Mail-Archiven. Links zu den Mail-Archiven finden Sie online unter http://www.mysql.com/documentation/. Siehe auch Abschnitt 3.6.1, „Linux (alle Linux-Versionen)“.

  Sie können `mysqld` auch manuell starten, dabei die `--skip-grant-tables`-Option benutzen und dann die Berechtigungsinformationen selbst mit `mysql` eintragen:

```
shell> BINDIR/safe_mysqld --skip-grant-tables &
shell> BINDIR/mysql -u root mysql
```

  Von `mysql` aus geben Sie die SQL-Befehle ein, die in `mysql_install_db` stehen. Stellen Sie sicher, dass Sie danach `mysqladmin flush-privileges` oder `mysqladmin reload` laufen lassen, um dem Server mitzuteilen, die Berechtigungstabellen neu zu laden.

3.4.2. Probleme mit dem Start des MySQL-Servers

Wenn Sie Tabellen einsetzen werden, die Transaktionen unterstützen (InnoDB, BDB), sollten Sie zuerst eine my.cnf-Datei anlegen und die Startoptionen für die Tabellentypen setzen, die Sie einsetzen wollen. See Kapitel 8, *MySQL-Tabellentypen*.

Im allgemeinen starten Sie den `mysqld`-Server auf eine der drei folgenden Weisen:

- Indem Sie `mysql.server` aufrufen. Dieses Skript wird hauptsächlich beim Systemstart und -herunterfahren eingesetzt. Es wird ausführlicher in Abschnitt 3.4.3, „MySQL automatisch starten und anhalten“ beschrieben.
- Indem Sie `safe_mysqld` aufrufen. Dieses Skript versucht die korrekten Optionen für `mysqld` festzustellen und läßt den Server dann mit diesen Optionen laufen. See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.
- Auf Windows NT sollten Sie `mysqld` wie folgt als Systemdienst starten:

  ```
  bin\mysqld-nt --install              # MySQL als Systemdienst installieren
  ```

  Jetzt können Sie `mysqld` wie folgt starten / anhalten:

  ```
  NET START mysql
  NET STOP mysql
  ```

  Beachten Sie, dass Sie in diesem Fall keine weiteren Optionen für `mysqld` benutzen können!

  Sie können den Systemdienst wie folgt entfernen:

  ```
  bin\mysqld-nt --remove               # MySQL als Systemdienst entfernen
  ```

- Indem Sie `mysqld` direkt aufrufen.

Wenn der `mysqld`-Daemon hoch fährt, wechselt er in das Daten-Verzeichnis. Dort erwartet er, Log-Dateien und die (process ID)-Datei schreiben zu können. Ebenfalls erwartet er dort, Datenbanken zu finden.

Der Speicherort des Daten-Verzeichnisses wird zum Zeitpunkt des Kompilierens der Distribution fest verdrahtet. Wenn `mysqld` jedoch erwartet, das Daten-Verzeichnis irgendwo sonst als an der Stelle zu finden, wo es auf Ihrem System tatsächlich ist, funktioniert er nicht richtig. Wenn Sie Probleme mit fehlerhaften Pfaden haben, können Sie durch den Aufruf von `mysqld` mit der `--help`-Option herausfinden, welche Optionen `mysqld` erlaubt und was die vorgabemäßigen Pfad-Einstellung sind. Sie können die Vorgaben überschreiben, indem Sie die korrekten Pfadnamen als Kommandozeilen-Argumente für `mysqld` festlegen. (Diese Optionen können auch bei `safe_mysqld` benutzt werden.)

Normalerweise sollte es lediglich nötig sein, `mysqld` das Basis-Verzeichnis mitzuteilen, wo MySQL installiert ist. Das können Sie mit der Option `--basedir` machen. Zusätzlich können Sie `--help` benutzen, um die Auswirkung der Pfadänderungsoptionen zu überprüfen (beachten Sie, dass `--help` die letzte Option des `mysqld`-Befehls wein *muss*. Beispiel:

```
shell> EXECDIR/mysqld --basedir=/usr/local --help
```

Wenn Sie die Pfadeinstellungen erst einmal festgelegt haben, die Sie wollen, starten Sie den Server ohne die `--help`-Option.

Mit welcher Methode auch immer Sie den Server starten: Wenn er nicht korrekt hoch fährt, untersuchen Sie die Log-Datei, um zu sehen, ob Sie den Grund dafür herausfinden können. Log-Dateien liegen im Daten-Verzeichnis (typischerweise `/usr/local/mysql/data` bei einer Binärdistribution, `/usr/local/var` bei einer Quelldistribution und `\mysql\data\mysql.err` unter Windows). Suchen Sie im Daten-Verzeichnis nach Dateien mit Namen der Form `host_name.err` und `host_name.log`, wobei `host_name` der Name Ihres Server-Hosts ist. Sehen Sie in den letzten paar Zeilen dieser Dateien nach:

```
shell> tail host_name.err
shell> tail host_name.log
```

Wenn Sie etwas wie das Folgende in der Log-Datei finden:

```
000729 14:50:10  bdb:  Recovery function for LSN 1 27595 failed
000729 14:50:10  bdb:  warning: ./test/t1.db: No such file or directory
000729 14:50:10  Can't init databases
```

Das bedeutet, dass Sie `mysqld` nicht mit `--bdb-no-recover` gestartet haben und Berkeley DB findet, dass etwas mit seinen Log-Dateien nicht in Ordnung ist, als es versuchte, Ihre Datenbanken wiederherzustellen. Um weitermachen zu können, sollten Sie alle alten Berkeley-DB-Log-Dateien aus dem Datenbankverzeichnis an eine andere Stelle verschieben, wo Sie sie später untersuchen können. Die Log-Dateien sind wie `log.0000000001` benannt, wobei die Nummer im Zeitablauf hochgezählt wird.

Wenn Sie `mysqld` mit BDB-Tabellenunterstützung laufen lassen und `mysqld` beim Start einen Speicherauszug (Core Dump) liefert, könnte das an Problemen mit den BDB-Wiederherstellungs-Logs liegen. In diesem Fall können Sie versuchen, `mysqld` mit `--bdb-no-recover` zu starten. Wenn das hilft, sollten Sie danach alle `log.*`-Dateien aus dem Daten-Verzeichnis entfernen und versuchen, `mysqld` erneut zu starten.

Wenn Sie folgenden Fehler bekommen, bedeutet das, dass ein anderes Programm (oder ein anderer `mysqld`-Server) bereits den TCP/IP-Port oder -Socket benutzt, den `mysqld` versucht zu benutzen:

```
Can't start server: Bind on TCP/IP-Port: Address already in use
  oder
Can't start server : Bind on unix socket...
```

Benutzen Sie `ps`, um sicherzustellen, dass kein weiterer `mysqld`-Server läuft. Wenn Sie keinen weiteren Server finden, können Sie den Befehl `telnet ihr-host-name tcp-ip-port-nummer` eingeben und mehrere Male `EINGABE` drücken. Wenn Sie keine Fehlermeldung wie `telnet: Unable to connect to remote host: Connection refused` erhalten, benutzt irgend etwas anderes den TCP/IP-Port, den `mysqld` versucht zu benutzen. Siehe Abschnitt 3.4.1, „Probleme mit `mysql_install_db`“ und Abschnitt 5.1.4, „Viele MySQL-Server auf derselben Maschine laufen lassen“.

Wenn `mysqld` gerade läuft, können Sie herausfinden, welche Pfadeinstellungen er benutzt, indem Sie folgenden Befehl ausführen:

```
shell> mysqladmin variables
```

oder

```
shell> mysqladmin -h 'ihr-host-name' variables
```

Wenn `safe_mysqld` hoch den Server hoch fährt, Sie sich aber nicht mit ihm verbinden können, stellen Sie sicher, dass Sie einen Eintrag wie den folgenden in `/etc/hosts` haben:

```
127.0.0.1       localhost
```

Dieses Problem tritt nur auf Systemen auf, die keine funktionierende Thread-Bibliothek besitzen, und für die MySQL so konfiguriert werden muss, dass es MIT-pThreads benutzt.

Wenn Sie es nicht schaffen, `mysqld` zu starten, können Sie versuchen, eine Trace-Datei anzulegen, um das Problem zu finden. See Abschnitt E.1.2, „Trace-Dateien erzeugen“.

Wenn Sie InnoDB-Tabellen benutzen, sehen Sie bei den InnoDB-spezifischen Startoptionen nach. See Abschnitt 8.5.2, „Mit InnoDB anfangen - Optionen“.

Wenn Sie BDB-(Berkeley DB)-Tabellen benutzen, sollten Sie sich mit den verschiedenen Startoptionen von BDB vertraut machen. See Abschnitt 8.6.3, „BDB-Startoptionen“.

3.4.3. MySQL automatisch starten und anhalten

Die `mysql.server`- und `safe_mysqld`-Skripte können benutzt werden, um den Server automatisch beim Hochfahren des Systems zu starten. `mysql.server` kann ebenfalls dazu benutzt werden, den Server anzuhalten.

Das `mysql.server`-Skript kann benutzt werden, um den Server zu starten oder anzuhalten, indem man es mit den `start`- oder `stop`-Argumenten aufruft:

```
shell> mysql.server start
shell> mysql.server stop
```

`mysql.server` liegt im `share/mysql`-Verzeichnis unterhalb des MySQL-Installationsverzeichnisses oder im `support-files`-Verzeichnis des MySQL-Source-Trees.

Bevor `mysql.server` den Server startet, wechselt es in das MySQL-Installationsverzeichnis. Dann ruft es `safe_mysqld` auf. Eventuell müssen Sie `mysql.server` editieren, wenn Sie eine Binärdistribution haben, die Sie an eine nicht stardardmäßige Stelle installiert haben. Ändern Sie es so ab, dass es in das richtige Verzeichnis wechselt (`cd`), bevor es `safe_mysqld` startet. Wenn Sie wollen, dass der Server unter einem bestimmten Benutzer läuft, fügen Sie eine entsprechende `user`-Zeile zur `/etc/my.cnf`-Datei hinzu, so wie weiter unten in diesem Abschnitt dargestellt.

`mysql.server stop` hält den Server an, indem es ihm ein Signal sendet. Sie können den Server auch automatisch herunter fahren, indem Sie `mysqladmin shutdown` ausführen.

Wenn Sie möchten, können Sie diese Start- und Stop-Befehle an den entsprechenden Stellen Ihrer `/etc/rc*`-Dateien einfügen, wenn Sie MySQL für Produktions-Applikationen benutzen. Beachten Sie, wenn Sie `mysql.server` editieren und dann gelegentlich MySQL aktualisieren (Update), dass dann Ihre geänderte Version überschrieben wird. Daher sollten Sie eine Kopie

Ihrer editierten Version machen, die Sie erneut installieren können.

Wenn Ihr System `/etc/rc.local` benutzt, um externe Skripte zu starten, sollten Sie folgendes anhängen:

```
/bin/sh -c 'cd /usr/local/mysql ; ./bin/safe_mysqld --user=mysql &'
```

Sie können Optionen für `mysql.server` in einer globalen `/etc/my.cnf`-Datei hinzufügen. Eine typische `/etc/my.cnf`-Datei sieht wie folgt aus:

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql

[mysql.server]
basedir=/usr/local/mysql
```

Das `mysql.server`-Skript kennt folgende Optionen: `datadir`, `basedir` und `pid-file`.

Folgende Tabelle zeigt, welche Optionsgruppen jedes der Startskripts aus den Optionsdateien liest:

Skript	Optionsgruppen
`mysqld`	`mysqld` und `server`
`mysql.server`	`mysql.server`, `mysqld`, und `server`
`safe_mysqld`	`mysql.server`, `mysqld`, und `server`

See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

3.5. MySQL aktualisieren (Upgrade / Downgrade)

Sie können die MySQL-form- und data-Dateien jederzeit für verschiedene Versionen auf derselben Architektur benutzen, solange Sie dieselbe Grundversion von MySQL haben. Die aktuelle Grundversion ist 3. Wenn Sie den Zeichensatz ändern, während MySQL läuft (was auch die Sortierreihenfolge betreffen kann), müssen Sie `myisamchk -r -q` auf alle Tabellen ausführen. Ansonsten könnte es sein, dass Ihre Indexe nicht korrekt sortiert werden.

Wenn Sie vor neuen Versionen zurück schrecken, können Sie Ihren alten `mysqld` zu etwas wie `mysqld`-'alte-versions-nummer' umbenennen. Wenn Ihr neuer `mysqld` dann etwas Unerwartetes tut, können Sie ihn einfach anhalten und mit Ihrem alten `mysqld` neu starten!

Wenn Sie ein Upgrade vornehmen, sollte Sie natürlich Ihre alten Datenbanken sichern.

Wenn Sie nach einem Upgrade auf Probleme mit neu kompilierten Client-Programmen stoßen, zum Beispiel `Commands out of sync` oder unerwartete Speicherauszüge (Core Dumps), haben sie wahrscheinlich einen alten Header oder eine alte Bibliotheksdatei benutzt, als Sie die Programme kompilierten. In diesem Fall sollten Sie das Datum Ihrer `mysql.h`-Datei und `libmysqlclient.a`-Bibliothek überprüfen, um sicherzustellen, dass sie aus der neuen MySQL-Distribution stammten. Wenn nicht, kompilieren sie Ihre Programme bitte neu!

Wenn Sie Probleme der Art erhalten, dass Ihr neuer `mysqld`-Server nicht startet oder dass Sie sich nicht ohne Passwort verbinden können, überprüfen Sie, ob Sie nicht etwa noch die alte `my.cnf`-Datei Ihrer alten Installation haben! Sie können das mit `program-name --print-defaults` tun. Wenn es irgend etwas anderes als den Programmnamen ausgibt, haben Sie eine aktive `my.cnf`-Datei, die sich auf die Dinge auswirkt!

Es ist eine gute Idee, die `Msql-Mysql-modules`-Distribution neu zu bauen und neu zu installieren, wann immer Sie ein neues Release von MySQL installieren, speziell dann, wenn Sie Symptome wie die bemerken, dass alle Ihre `DBI`-Skripte mit Core-Dumps abbrechen, nachdem Sie MySQL aktualisiert haben.

3.5.1. Upgrade von 3.23 auf Version 4.0

Sie können Ihre alten data-Dateien ohne jede Änderung mit Version 4.0 benutzen. Wenn Sie Ihre Daten eines MySQL-4.0-Servers für einen älteren Server verwenden wollen, müssen Sie `mysqldump` benutzen.

Alte Clients sollen mit einem Server Version 4.0 ohne jedes Problem funktionieren.

Die folgende Liste stellt dar, auf was Sie aufpassen müssen, wenn Sie auf Version 4.0 aktualisieren (Upgrade):

- `safe_mysqld` wurde zu `mysqld_safe` umbenannt.

- Die alten C-API-Funktionen `mysql_drop_db`, `mysql_create_db` und `mysql_connect` werden nicht mehr unterstützt, es sei denn, MySQL wird mit `USE_OLD_FUNCTIONS` kompiliert.
- Sie sollten `TRUNCATE TABLE` benutzen, wenn Sie alle Zeilen aus einer Tabelle löschen wollen und Ihnen egal ist, wie viele Zeilen gelöscht wurden. (`TRUNCATE TABLE` ist schneller als `DELETE FROM tabelle`).
- Sie bekommen einen Fehler, wenn Sie ein aktives `LOCK TABLES` oder eine aktive Transaktion am Laufen haben, wenn Sie versuchen, `TRUNCATE TABLE` oder `DROP DATABASE` auszuführen.
- Sie sollten Ganzzahl-(Integer)-Werte in BIGINT-Spalten benutzen (anstelle von Zeichenketten wie in MySQL 3.23). Man kann immer noch Zeichenketten benutzen, aber die Benutzung von Ganzzahlen ist viel effizienter.
- Das Format von `SHOW OPEN TABLE` hat sich geändert.
- Multithreaded Clients sollten `mysql_thread_init()` und `mysql_thread_end()` benutzen. See Abschnitt 9.4.8, „Wie man einen threaded Client herstellt“.

3.5.2. Upgrade von einer Version 3.22 auf 3.23

MySQL-Version 3.23 unterstützt Tabellen des neuen `MyISAM`-Typs und des alten `ISAM`-Typs. Sie müssen Ihre alten Tabellen nicht konvertieren, um sie mit Version 3.23 einsetzen zu können. Vorgabemäßig werden alle neuen Tabellen mit dem Typ `MyISAM` angelegt (es sei denn, Sie starten `mysqld` mit der `--default-table-type=isam`-Option). Sie können eine `ISAM`-Tabelle zu einer `MyISAM`-Tabelle mit `ALTER TABLE tabelle TYPE=MyISAM` konvertieren oder mit dem Perl-Skript `mysql_convert_table_format`.

Clients der Versionen 3.22 und 3.21 funktionieren ohne jedes Problem mit einem Server der Version 3.23.

Die folgende Liste stellt dar, auf was Sie aufpassen müssen, wenn Sie auf Version 3.23 aktualisieren (Upgrade):

- Alle Tabellen, die den `tis620`-Zeichensatz benutzen, müssen mit `myisamchk -r` oder `REPAIR TABLE` in Ordnung gebracht werden.
- Wenn Sie ein `DROP DATABASE` auf eine mit symbolischem Link verknüpfte Datenbank ausführen, werden sowohl der symbolische Links als auch die Datenbank gelöscht. (Das war in Version 3.22 nicht der Fall, weil configure den `readlink`-Systemaufruf nicht erkannte).
- `OPTIMIZE TABLE` funktioniert jetzt nur bei **MyISAM**-Tabellen. Bei anderen Tabellentypen können Sie `ALTER TABLE` benutzen, um die Tabelle zu optimieren. Während der Ausführung von `OPTIMIZE TABLE` wird die Tabelle jetzt vor dem Zugriff anderer Threads gesperrt.
- Der MySQL-Client `mysql` wir jetzt vorgabemäßig mit der Option `--no-named-commands (-g)` gestartet. Diese Option kann mit `--enable-named-commands (-G)` abgeschaltet werden. Dies kann ein paar Inkompatibilitätsprobleme verursachen, zum Beispiel in SQL-Skripten, die benannte (named) Befehle ohne ein Semikolon! Befehle im Langformat dagegen funktionieren noch auf der ersten Zeile.

  some cases, für Beispiel in SQL Skripts that use named Befehle ohne a semicolon! Long format Befehle still work from the first line.
- If you are using the `german` character sort order, you must repair all your Tabellen mit `isamchk -r`, as we have made some changes in the sort order!
- The default return type of `IF` will now depend on both arguments und not only the first argument.
- `AUTO_INCREMENT` funktioniert nicht mit negativen Zahlen. Der Grund liegt darin, dass negative Zahlen beim Übergang von -1 auf 0 Probleme verursachen. `AUTO_INCREMENT` wird jetzt bei MyISAM-Tabellen auf einem niedrigeren Level gehandhabt und ist viel schneller als vorher. Bei MyISAM-Tabellen werden alte Zahlen auch nicht mehr wieder benutzt, selbst wenn Sie einige Zeilen aus der Tabelle löschen.
- `CASE`, `DELAYED`, `ELSE`, `END`, `FULLTEXT`, `INNER`, `RIGHT`, `THEN` und `WHEN` sind jetzt reservierte Wörter.
- `FLOAT(X)` ist jetzt ein echter Fließkomma-Typ und kein Wert mit einer festen Anzahl von Dezimalstellen.
- Wenn Sie `DECIMAL(length,dec)` deklarieren, beinhaltet das Längen-Argument nicht mehr den Platz für das Vorzeichen oder den Dezimalpunkt.
- Eine `TIME`-Zeichenkette muss jetzt von einem der folgenden Formate sein: `[[[DAYS] [H]H:]MM:]SS[.bruchteil]` oder `[[[[[H]H]H]H]MM]SS[.bruchteil]`
- `LIKE` vergleicht jetzt Zeichenketten unter Verwendung derselben Vergleichsregeln wie `'='`. Wenn Sie das alte Verhalten

benötigen, können Sie MySQL mit dem `CXXFLAGS=-DLIKE_CMP_TOUPPER`-Flag kompilieren.

- `REGEXP` arbeitet jetzt bei normalen (nicht binären) Zeichenketten unabhängig von der Groß-/Kleinschreibung.
- Wenn Sie Tabellen prüfen / reparieren, sollten Sie `CHECK TABLE` oder `myisamchk` für `MyISAM`-Tabellen (`.MYI`) benutzen und `isamchk` für ISAM-Tabellen (`.ISM`).
- Wenn Sie wollen, dass `mysqldump`-Dateien zwischen MySQL-Version 3.22 und Version 3.23 kompatibel sind, sollten Sie nicht die `--opt`- oder `--full`-Option für `mysqldump` benutzen.
- Überprüfen Sie Ihre Aufrufe von `DATE_FORMAT()` und stellen Sie sicher, dass vor jedem Formatierungszeichen ein '`%`' steht. (Spätere MySQL-Versionen 3.22 ließen diese Syntax zu.)
- `mysql_fetch_fields_direct` ist jetzt eine Funktion (es war ein Makro) und gibt einen Zeiger auf `MYSQL_FIELD` anstelle eines `MYSQL_FIELD` zurück.
- `mysql_num_fields()` kann nicht mehr für ein `MYSQL*`-Objekt benutzt werden (es ist jetzt eine Funktion, die `MYSQL_RES*` als Argument nimmt. Sie sollten jetzt statt dessen `mysql_field_count()` benutzen.
- In MySQL-Version 3.22 war die Ausgabe von `SELECT DISTINCT ...` fast immer sortiert. In Version 3.23 müssen Sie `GROUP BY` oder `ORDER BY` benutzen, um eine sortierte Ausgabe zu erhalten.
- `SUM()` gibt jetzt `NULL` zurück statt 0, wenn es keine überein stimmenden Zeilen gibt. Das ist in Übereinstimmung mit ANSI-SQL.
- Ein `AND` oder `OR` mit `NULL`-Werten gibt jetzt `NULL` anstelle von 0 zurück. Das betrifft hauptsächlich Anfragen, die `NOT` bei einem `AND/OR`-Ausdruck wie `NOT NULL` = `NULL` benutzen. `LPAD()` und `RPAD()` kürzen die Ergebnis-Zeichenkette, wenn sie länger als das Längen-Argument ist.

3.5.3. Upgrade von Version 3.21 auf Version 3.22

Nichts, was die Kompatibilität betrifft, hat sich zwischen Version 3.21 und 3.22 geändert. Die einzige Falle ist die, dass neue Tabellen, die unter Verwendung des `DATE`-Typs erzeugt werden, die neue Art der Datenspeicherung benutzen. Diese neuen Felder kann man daher nicht von einer alten Version von `mysqld` ansprechen.

Nachdem Sie MySQL-Version 3.22 installiert haben, starten Sie den neuen Server und lassen dann das `mysql_fix_privilege_tables`-Skript laufen. Dieses fügt die neuen Zugriffsberechtigungen ein, die Sie benötigen, um den `GRANT`-Befehl zu benutzen. Wenn Sie das vergessen, erhalten Sie ein `Access denied`, wenn Sie versuchen, `ALTER TABLE`, `CREATE INDEX` oder `DROP INDEX` zu benutzen. Wenn Ihr MySQL-Root ein Passwort benötigt, müssen Sie dieses als Argument zu `mysql_fix_privilege_tables` angeben.

Die C-API-Schnittstelle für `mysql_real_connect()` hat sich geändert. Wenn Sie ein altes Client-Programm haben, das diese Funktion aufruft, müssen Sie eine `0` als neues `db`-Argument einfügen (oder den Client neu kodieren, so dass er das `db`-Element für schnellere Verbindungen benutzt). Zusätzlich müssen Sie `mysql_init()` aufrufen, bevor Sie `mysql_real_connect()` aufrufen! Diese Änderung wurde durchgeführt, damit die neue `mysql_options()`-Funktion in der `MYSQL`-Handler-Struktur Optionen speichern kann.

The `mysqld`-Variable `key_buffer` wurde umbenannt in `key_buffer_size`, Sie können aber in Ihren Startdateien immer noch den alten Namen verwenden.

3.5.4. Upgrade von Version 3.20 auf Version 3.21

Wenn Sie eine Version benutzen, die älter als Version 3.20.28 ist, und auf Version 3.21 umstellen wollen, müssen Sie folgendes tun:

Sie können den `mysqld`-Server Version 3.21 mit `safe_mysqld --old-protocol` starten, um ihn mit Clients aus einer Distribution Version 3.20 zu benutzen. In diesem Fall gibt die neue Client-Funktion `mysql_errno()` überhaupt keine Server-Fehler zurück, nur `CR_UNKNOWN_ERROR` (funktioniert aber bei Client-Fehlern), und der Server benutzt die alte `password()`-Überprüfung statt der neuen.

Wenn Sie die `--old-protocol`-Option **NICHT** für `mysqld` benutzen, müssen Sie folgende Änderungen durchführen:

- Jeder Client-Code muss neu kompiliert werden. Wenn Sie ODBC benutzen, müssen Sie die neuen **MyODBC**-2.x-Treiber verwenden.
- Sie müssen das Skript `Skripts/add_long_password` laufen lassen, um das `Password`-Feld in der `mysql.user`-Tabelle zu `CHAR(16)` zu ändern.

- Alle Passwörter müssen in der `mysql.user`-Tabelle neu zugewiesen werden (um 62-Bit- statt 31-Bit-Passwörter zu erhalten).
- Das Tabellenformat hat sich nicht geändert, daher müssen Sie keinerlei Tabellen konvertieren.

MySQL-Version 3.20.28 und höher kann das neue `user`-Tabellenformat handhaben, ohne sich auf Clients auszuwirken. Wenn Sie eine MySQL-Version vor Version 3.20.28 haben, funktionieren Passwörter damit nicht mehr, wenn Sie die `user`-Tabelle konvertieren. Um auf Nummer Sicher zu gehen, sollten Sie mindestens auf Version 3.20.28 aktualisieren und erst dann auf Version 3.21.

Der neue Client-Code funktioniert bei einem 3.20.x `mysqld`-Server. Wenn Sie daher Probleme mit 3.21.x bekommen, können Sie den alten 3.20.x-Server benutzen, ohne die Clients neu kompilieren zu müssen.

Wenn Sie nicht die `--old-protocol`-Option für `mysqld` benutzen, werden alte Clients folgende Fehlermeldung ausgeben:

```
ERROR: Protocol mismatch. Server Version = 10 Client Version = 9
```

Die neue Perl-`DBI`/`DBD`-Schnittstelle unterstützt auch die alte `mysqlperl`-Schnittstelle. Die einzige Änderung, die Sie machen müssen, wenn Sie `mysqlperl` benutzen, ist, die Argumente für die `connect()`-Funktion zu ändern. Die neuen Argumente sind: `host`, `database`, `user` und `password` (die `user`- und `password`-Argumente haben die Plätze getauscht. See Abschnitt 9.2.2, „Die DBI-Schnittstelle“.

Folgende Änderungen können Anfragen in alten Applikationen betreffen:

- `HAVING` muss jetzt vor einer möglichen `ORDER BY`-Klausel spezifiziert werden.
- Die Parameter für `LOCATE()` wurden getauscht.
- Es gibt einige neue reservierte Wörter. Die wichtigsten sind `DATE`, `TIME` und `TIMESTAMP`.

3.5.5. Upgrade auf eine andere Architektur

Wenn Sie MySQL-Version 3.23 benutzen, können Sie die `.frm`-, `.MYI`- und `.MYD`-Dateien zwischen verschiedenen Architekturen kopieren, die dasselbe Fließkomma-Format unterstützen. (MySQL kümmert sich um eventuelle Byte-Tausch-Belange.)

Die MySQL-`ISAM`-Daten und Index-Dateien (`.ISD` und `*.ISM`, je nachdem) sind Architektur-abhängig und in manchen Fällen Betriebssystem-abhängig. Wenn Sie Ihre Applikationen auf eine andere Maschine mit einer unterschiedlichen Architektur oder einem anderen Betriebssystem verlagern wollen, wollten Sie nicht einfach eine Datenbank verschieben, indem Sie deren Dateien auf die andere Maschine kopieren. Benutzen Sie statt dessen `mysqldump`.

Vorgabemäßig erzeugt `mysqldump` eine Datei mit SQL-Statements. Sie können diese Datei auf die andere Maschine übertragen und Sie als Eingabe für den `mysql`-Client benutzen.

`mysqldump --help` zeigt Ihnen, welche Optionen verfügbar sind. Wenn Sie die Daten mit einer neueren Version von MySQL benutzen werden, sollten Sie `mysqldump --opt` mit der neueren Version benutzen, um einen schnellen, kompakten Dump zu erhalten.

Die einfachste (wenngleich nicht schnellste) Art, eine Datenbank von einer Maschine auf eine andere zu bringen, ist, die folgenden Befehle auf der Maschine auszuführen, auf der die Datenbank liegt:

```
shell> mysqladmin -h 'anderer hostname' create db_name
shell> mysqldump --opt db_name \
       | mysql -h 'anderer hostname' db_name
```

Wenn Sie eine Datenbank von einer entfernten Maschine über ein langsames Netzwerk kopieren wollen, können Sie folgendes benutzen:

```
shell> mysqladmin create db_name
shell> mysqldump -h 'anderer hostname' --opt --compress db_name \
       | mysql db_name
```

Sie können das Ergebnis auch in einer Datei speichern, diese Datei auf die Zielmaschine übertragen und dort in die Datenbank laden. Sie können zum Beispiel wie folgt die Datenbank in eine Datei auf der Quellmaschine ausgeben (dumpen):

```
shell> mysqldump --quick db_name | gzip > db_name.inhalte.gz
```

(Die in diesem Beispiel erzeugte Datei ist komprimiert.) Übertragen Sie die Datei, die die Datenbankinhalte enthält, auf die

Zielmaschine und geben Sie dort diese Befehle ein:

```
shell> mysqladmin create db_name
shell> gunzip < db_name.inhalte.gz | mysql db_name
```

Sie können auch `mysqldump` und `mysqlimport` benutzen, um den Datenbank-Transfer zu bewerkstelligen. Das ist bei großen Tabellen wesentlich schneller als die Benutzung von `mysqldump`. In den unten dargestellten Befehlen repräsentiert `DUMPDIR` den vollen Pfadnamen des Verzeichnisses, das Sie benutzen, um die Ausgabe von `mysqldump` zu speichern.

Legen Sie zunächst das Verzeichnis für die Ausgabe-Dateien an und geben Sie die Datenbank aus (Dump):

```
shell> mkdir DUMPDIR
shell> mysqldump --tab=DUMPDIR db_name
```

Übertragen Sie dann die Dateien des `DUMPDIR`-Verzeichnisses in ein entsprechendes Verzeichnis auf der Zielmaschine und laden Sie dort die Dateien in MySQL:

```
shell> mysqladmin create db_name           # Datenbank erzeugen
shell> cat DUMPDIR/*.sql | mysql db_name   # Tabellen in der Datenbank erzeugen
shell> mysqlimport db_name DUMPDIR/*.txt   # Daten in die Tabellen laden
```

Vergessen Sie auch nicht, die `mysql`-Datenbank zu kopieren, den dort befinden Sie die Berechtigungstabellen (`user`, `db`, `host`). Eventuell müssen Sie die Befehle als MySQL-`root`-Benutzer auf der neuen Maschine eingeben, um die `mysql`-Datenbank angelegt zu bekommen.

Nachdem Sie die `mysql`-Datenbank auf die neue Maschine kopiert haben, führen Sie `mysqladmin flush-privileges` aus, damit der Server die Berechtigungsinformationen neu einliest.

3.6. Betriebssystem-spezifische Anmerkungen

3.6.1. Linux (alle Linux-Versionen)

Die Anmerkungen weiter unten, die **glibc** betreffen, gelten nur dann, wenn Sie MySQL selbst bauen. Wenn Sie Linux auf einer x86-Maschine fahren, ist es in den meisten Fällen wesentlich besser, einfach unsere Binärdateien zu benutzen. Wir linken unsere Binärdateien an die am besten gepatchte Version von **glibc**, die wir bieten können, und mit den besten Kompiler-Optionen, wobei wir versuchen, MySQL für Hochlast-Server geeignet zu machen. Wenn Sie also den Text unten lesen und sich nicht sicher sind, was Sie tun sollen, sollten Sie zunächst unsere Binärdateien ausprobieren, um zu sehen, ob diese Ihren Anforderungen entsprechen. Kümmern Sie sich nur dann um einen eigenen Build, wenn Sie feststellen, dass unsere Binärdateien nicht gut genug sind. In diesem Fall wären wir für einen Hinweis dazu dankbar, damit wir beim nächsten Mal eine bessere Binärdatei bauen können. Für eine typische Benutzung, selbst bei einer großen Zahl gleichzeitiger Verbindungen und / oder Tabellen, die größer als 2 GB sind, sind unsere Binärdateien in den meisten Fällen die beste Wahl.

MySQL benutzt auf Linux LinuxThreads. Wenn Sie eine alte Linux-Version benutzen, die keine `glibc2` hat, müssen Sie LinuxThreads installieren, bevor Sie MySQL kompilieren. Sie erhalten LinuxThreads unter http://www.mysql.com/downloads/os-linux.html.

ACHTUNG: Wir haben einige seltsame Probleme bei Linux 2.2.14 und MySQL auf SMP-Systemen festgestellt. Wenn Sie ein SMP-System haben, empfehlen wir, so schnell wie möglich auf Linux 2.4 zu aktualisieren (Upgrade)! Dadurch wird Ihr System ausserdem schneller und stabiler!

Beachten Sie, dass `glibc`-Versionen vor und einschließlich Version 2.1.1 einen schweren Fehler im `pThread_mutex_timedwait`-Handling haben, was benutzt wird, wenn Sie `INSERT DELAYED` verwenden. Wir empfehlen, vor einem Upgrade der glibc `INSERT DELAYED` nicht zu verwenden.

Wenn Sie planen, mehr als 1000 gleichzeitige Verbindungen zu haben, müssen Sie einige Änderungen an LinuxThreads vornehmen, es neu kompilieren und mit der neuen `libpThread.a` linken. Setzen Sie `PTHREAD_THREADS_MAX` in `sysdeps/unix/sysv/linux/bits/local_lim.h` auf 4096 herauf und setzen Sie `STACK_SIZE` in `linuxThreads/internals.h` auf 256 KB herunter. Die Pfade sind relativ zum Wurzelverzeichnis von `glibc`. Beachten Sie, dass MySQL bei etwa 600 bis 1000 Verbindungen nicht stabil läuft, wenn `STACK_SIZE` auf den Vorgabewert von 2 MB gesetzt wird.

Wenn Sie Probleme damit bekommen, dass MySQL nicht genug Dateien oder Verbindungen öffnen kann, haben Sie möglicherweise Linux nicht so konfiguriert, dass es genug Dateien handhaben kann.

In Linux 2.2 und Folgenden können Sie die Anzahl der allozierten Datei-Handler herausbekommen, wenn Sie folgendes eingeben:

```
cat /proc/sys/fs/file-max
cat /proc/sys/fs/dquot-max
cat /proc/sys/fs/super-max
```

Wenn Sie mehr als 16M Speicher haben, sollten Sie etwas Ähnliches wie folgendes in Ihr Boot-Skript (`/etc/rc/boot.local` auf SuSE) eintragen:

```
echo 65536 > /proc/sys/fs/file-max
echo 8192 > /proc/sys/fs/dquot-max
echo 1024 > /proc/sys/fs/super-max
```

Das können Sie auch von der Kommandozeile aus als Root eingeben, aber in diesem Fall werden die alten Beschränkungen wieder benutzt, wenn Sie Ihren Computer neu starten.

Zusätzlich sollten Sie in /etc/my.cnf einfügen:

```
[safe_mysqld]
open-files-limit=8192
```

Das sollte MySQL erlauben, bis zu 8192 Verbindungen und Dateien zu erzeugen.

Die `STACK_SIZE`-Konstante in LinuxThreads steuert das Spacing von Thread-Stacks im Adressraum. Sie muss Groß genug sein, damit reichlich Platz für den Stack jedes individuellen Threads bleibt, aber klein genug, um den Stack irgend eines Threads davon abzuhalten, mit den globalen `mysqld`-Daten zu kollidieren. Wie wir durch Experimentieren heraus fanden, unmappt die Linux-Implementation von `mmap()` erfolgreich eine bereits gemappte Region, wenn Sie sie anweisen, eine Adresse auszumappen, die bereits in Benutzung ist, wobei sie alle Daten der gesamten Seite auf Null setzt, statt einen Fehler zurück zu geben. Daher beruht die Sicherheit von `mysqld` oder jeder anderen Thread-Applikation auf dem "Gentleman"-Verhalten des Codes, der Threads erzeugt. Der Benutzer muss Vorkehrungen treffen, die sicherstellen, dass die Anzahl laufender Threads jederzeit ausreichend gering ist, damit die Thread-Stacks sich vom globalen Heap fernhalten. Bei `mysqld` sollten Sie dieses "Gentleman"-Verhalten forcieren, indem Sie einen vernünftigen Wert für die the `max_connections`-Variable setzen.

Wenn Sie MySQL selbst bauen und sich nicht mit dem Patchen von LinuxThreads herum plagen wollen, sollten Sie `max_connections` auf einen Wert nicht größer als 500 setzen. Dieser Wert sollte sogar noch kleiner sein, wenn Sie einen großen Schlüsselpuffer (Key Buffer), große Heap-Tabellen oder andere Dinge haben, die `mysqld` dazu bringen könnten, eine Menge Speicher zu allozieren, oder wenn Sie einen 2.2-Kernel mit einem 2GB-Patch fahren. Wenn Sie unsere Binärdateien oder RPM-Versionen 3.23.23 oder später benutzen, können Sie `max_connections` sicher auf 1500 setzen, unter der Annahme, dass es keine großen Schlüsselpuffer oder Heap-Tabellen mit vielen Daten gibt. Je mehr Sie `STACK_SIZE` in LinuxThreads reduzieren können, desto mehr können Sie sicher Threads erzeugen. Wir empfehlen einen Wert zwischen 128K und 256K.

Wenn Sie viele gleichzeitige Verbindungen benutzen, bekommen Sie vielleicht Probleme durch ein "Feature" im 2.2-Kernel, der einen Prozess dafür bestraft, dass er sich aufspaltet (fork) oder einen Kindprozess klont, um einen Fork-Bombenangriff (Fork Bomb Attack) zu verhindern. Das bringt MySQL dazu, nicht so gut zu skalieren, wenn Sie die Anzahl gleichzeitiger Clients erhöhen. Wir konnten beobachten, dass sich das auf Einprozessor-Systemen mit sehr langsamer Thread-Erzeugung bemerkbar macht, was sich darin zeigt, dass es sehr lange dauern kann, sich mit MySQL zu verbinden (bis zu einer Minute), und genau so lange, um es herunter zu fahren. Auf Multiprozessor-Systemen haben wir einen allmählichen Abfall der Anfrage-Geschwindigkeit beobachtet, wenn die Anzahl der Clients zunimmt. Im Verlauf der Suche nach einer Lösung haben wir von einem unserer Benutzer einen Kernel-Patch erhalten, von dem dieser sagt, dass er auf seiner Site eine beträchtliche Rolle spielt. Der Patch ist hier verfügbar (http://www.mysql.com/Downloads/Patches/linux-fork.patch). Inzwischen haben wir recht ausführliche Tests dieses Patchs sowohl auf Entwicklungs- als auch auf Produktionssystemen gemacht. Er hat die Performance von `MySQL` erheblich verbessert, ohne irgend welche Probleme zu verursachen, und wir empfehlen ihn jetzt denjenigen unserer Benutzer, die immer noch Hochlast-Server auf 2.2-Kerneln fahren. Dieses Problem wurde im 2.4-Kernel behoben. Wenn Sie daher nicht zufrieden mit der momentanen Performance Ihres Systems sind, ist es wahrscheinlich einfacher, auf 2.4 zu aktualisieren, statt den 2.2-Kernel zu patchen, was zusätzlich zur Behebung dieses Fairness-Bugs auch noch Multiprozessor-Systemen einen netten Schub gibt.

Wir haben MySQL auf dem 2.4-Kernel auf einer Zweiprozessor-Maschine getestet und haben festgestellt, dass MySQL VIEL bessere Leistungsdaten bringt - es gab praktisch keine Verlangsamung bei Anfragen bis ganz herauf zu 1000 Clients, und der Skalierungsfaktor von MySQL (berechnet als Verhältnis von maximalem Durchsatz zum Durchsatz mit 1 Client) war 100%. Ähnliches haben wir auf einer Vierprozessor-Maschine beobachtet - praktisch keine Verlangsamung, während die Anzahl der Clients bis auf 1000 stieg sowie ein Skalierungsfaktor von 300%. Für einen unter Hochlast fahrenden Multiprozessor-Server empfehlen wir daher ausdrücklich den 2.4-Kernel. Weiter haben wir festgestellt, dass es essentiell wichtig ist, den `mysqld`-Prozess auf dem 2.4-Kernel mit der höchstmöglichen Priorität laufen zu lassen, um maximale Performance zu erreichen. Das kann dadurch erreicht werden, dass man den `renice -20 $$`-Befehl zu `safe_mysqld` hinzufügt. Bei unseren Tests auf der Vierprozessor-Maschine ergab die Erhöhung der Priorität eine 60%-ige Steigerung des Durchsatzes bei 400 Clients.

Wir sind derzeit dabei, mehr Informationen über die Performance von `MySQL` auf dem 2.4-Kernel auf 4-Weg- und 8-Weg-Systemen zu bekommen. Wenn Sie Zugang zu einem solchen System haben und einige Benchmarks gemacht haben, schicken Sie bitte eine Mail mit den Ergebnissen an Documentation Team - wir werden Sie dem Handbuch hinzufügen.

Es gibt eine weitere Sache, die die Performance von MySQL stark beeinträchtigt, besonders auf SMP-Systemen. Die Implementation von mutex in LinuxThreads in **glibc-2.1** ist sehr schlecht für Programme mit vielen Threads, die den mutex nur für kurze Zeit behalten. Wenn Sie MySQL mit unveränderten **LinuxThreads** linken, führt ironischerweise auf einem SMP-System in manchen Fällen das Entfernen von Prozessoren zu einer Leistungssteigerung von MySQL. Für **glibc 2.1.3** haben wir ein Patch bereit gestellt, um dieses Verhalten zu korrigieren: linuxthreads-2.1-patch

Bei Verwendung von **glibc-2.2.2** benutzt MySQL-Version 3.23.36 den adaptiven mutex, der sogar viel besser als der gepatchte von **glibc-2.1.3** ist. Seien Sie jedoch davor gewarnt, dass unter bestimmten Umständen der aktuelle mutex-Code in **glibc-2.2.2** überdrehen kann, was die Performance von MySQL beeinträchtigt. Die Gefahr, dass solche Umstände eintreten, kann dadurch verringert werden, dass der `mysqld`-Prozess auf die höchste Priorität gesetzt wird. Zusätzlich konnten wir das Überdrehverhalten mit einem Patch korrigieren, der hier erhältlich ist. Der Patch kombiniert die Korrektur des Überdrehens, die maximale Anzahl von Threads und das Stack-Spacing in einem. Sie wenden es auf das `linuxThreads`-Verzeichnis mit `patch -p0 </tmp/linuxThreads-2.2.2.patch` an. Wir hoffen, dass der Patch in irgend einer Form in zukünftigen Releases von `glibc-2.2` enthalten sein wird. Wie es auch sei, wenn Sie mit `glibc-2.2.2` linken, müssen Sie immer noch `STACK_SIZE` und `PTHREAD_THREADS_MAX` korrigieren. Wir hoffen, dass diese Vorgabewerte zukünftig auf akzeptablere Werte für eine MySQL-Hochlast-Einrichtung gesetzt werden, so dass Ihr eigener Build auf `./configure; make; make install` reduziert werden kann.

Wir empfehlen, dass Die die oben genannten Patches benutzen, um eine spezielle statische Version von `libpThread.a` zu bauen, die Sie nur für statisches Linken mit `MySQL` benutzen. Wir wissen, dass die Patches für `MySQL` sicher sind und seine Performance erheblich verbessern, aber wir können diesbezüglich nichts über andere Applikationen sagen. Wenn Sie andere Applikationen mit der gepatchten Version der Bibliothek linken oder eine gepatchte gemeinsam benutzte (shared) Version bauen und auf Ihrem System installieren, tun Sie das auf eigenes Risiko, was andere Applikationen betrifft, die von `LinuxThreads` abhängen.

Wenn Sie während der Installation von MySQL irgend welche seltsamen Probleme bekommen oder gebräuchliche Utilities hängen bleiben, ist es sehr wahrscheinlich, dass diese entweder Bibliotheks- oder Compiler-bezogen sind. In diesem Fall wird die Benutzung unserer Binärdatei sie beheben.

Ein bekanntes Problem der Binärdistribution ist, dass Sie auf älteren Linux-Systemen, die `libc` benutzen (wie RedHat 4.x oder Slackware) nicht-schwere (non-fatal) Probleme mit der Auflösung von Hostnamen bekommen. See Abschnitt 3.1.1, „MySQL auf Linux installieren“.

Wenn Sie LinuxThreads benutzen, werden Sie feststellen, dass mindestens drei Prozesse laufen. Das sind in Wirklichkeit Threads. Es gibt einen Thread für den LinuxThreads-Manager, einen Thread, um Verbindungen zu handhaben und einen Thread, um Alarme und Signale zu handhaben.

Beachten Sie, dass der Linux-Kernel und die LinuxThread-Bibliothek vorgabemäßig nur 1024 Threads haben können. Das bedeutet, dass Sie auf einem ungepatchten System nur höchstens 1021 Verbindungen zu MySQL haben können. Die Seite http://www.volano.com/linuxnotes.html enthält Informationen, wie man diese Beschränkung umgeht.

Wenn Sie einen toten `mysqld`-Daemon-Prozess mit `ps` sehen, bedeutet das üblicherweise, dass Sie einen Bug in MySQL oder eine zerstörte Tabelle gefunden haben. See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“.

Um auf Linux einen Speicherauszug (Core Dump) zu erhalten, wenn `mysqld` mit einem SIGSEGV-Signal stirbt, können Sie `mysqld` mit der `--core-file`-Option starten. Beachten Sie, dass Sie wahrscheinlich `core file size` hoch setzen müssen, indem Sie `ulimit -c 1000000` zu `safe_mysqld` hinzufügen oder `safe_mysqld` mit `--core-file-sizes=1000000` starten. See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.

Wenn Sie Ihren eigenen MySQL-Client linken und bei der Ausführung diesen Fehler erhalten,

```
ld.so.1: ./my: fatal: libmysqlclient.so.4: open failed: No such file or directory
```

kann das Problem durch eine der folgenden Methoden behoben werden:

- Linken Sie den Client mit dem folgenden Flag (anstelle von `-Lpath`): `-Wl,r/path-libmysqlclient.so`.
- Kopieren Sie `libmysqclient.so` nach `/usr/lib`.
- Fügen Sie der `LD_RUN_PATH`-Umgebungsvariablen den Pfadnamen des Verzeichnisses hinzu, wo `libmysqlclient.so` liegt, bevor Sie Ihren Client laufen lassen.

Wenn Sie den Fujitsu-Compiler `(fcc / FCC)` benutzen, werden Sie beim Kompilieren von MySQL einige Probleme bekommen, weil die Linux-Header-Dateien sehr `gcc`-orientiert sind.

Folgende `configure`-Zeile sollte mit `fcc/FCC` funktionieren:

```
CC=fcc CFLAGS="-O -K fast -K lib -K omitfp -Kpreex -D_GNU_SOURCE -DCONST=konstante -DNO_STRTOLL_PROTO" CXX=FCC CXXFLAGS
```

3.6.1.1. Anmerkungen zur Binärdistribution (Linux)

MySQL benötigt zumindest Linux-Version 2.0.

ACHTUNG: Wir haben Berichte von MySQL-Benutzern erhalten, die schwer wiegende Stabilitätsprobleme mit Linux-Kernel

2.2.14 mitgeteilt haben. Wenn Sie diesen Kernel benutzen, sollten Sie auf 2.2.19 (oder neuer) oder auf einen 2.4-Kernel aktualisieren. Wenn Sie eine Mehrprozessormaschine haben, sollten Sie auf jeden Fall in Betracht ziehen, 2.4 zu benutzen, weil Ihnen das erhebliche Geschwindigkeitssteigerung geben wird.

Die Binärdistribution wird mit `-static` gelinkt, was normalerweise heißt, dass Sie sich nicht um die Version der Systembibliotheken kümmern müssen, die Sie haben. Ausserdem brauchen Sie nicht LinuxThread installieren. Ein Programm, das mit `-static` gelinkt ist, ist etwas größer als ein dynamisch gelinktes Programm, und gleichzeitig etwas schneller (3-5%). Ein Problem liegt jedoch darin, dass Sie bei einem statisch gelinkten Programm keine benutzerdefinierten Funktionen (UDF) benutzen können. Wenn Sie UDF-Funktionen schreiben oder benutzen wollen (das ist nur etwas für C- oder C++-Programmierer), müssen Sie MySQL selbst kompilieren und das dynamische Linken benutzen.

Wenn Sie ein `libc`-basierendes System benutzen (statt eines `glibc2`-Systems), bekommen Sie bei der Binärdistribution wahrscheinlich Probleme mit der Auflösung von Hostnamen und mit `getpwnam()`. (Das liegt daran, dass `glibc` leider von einigen externen Bibliotheken abhängt, um Hostnamen aufzulösen und und `getpwent()`, selbst wenn es mit `-static` kompiliert wird.) In diesem Fall erhalten Sie wahrscheinlich folgende Fehlermeldung, wenn Sie `mysql_install_db` ausführen:

```
Sorry, the host 'xxxx' could not be looked up
```

oder den folgenden Fehler, wenn Sie versuchen, `mysqld` mit der `--user`-Option laufen zu lassen:

```
getpwnam: No such file or directory
```

Sie können dieses Problem auf eine der folgenden Weisen lösen:

- Holen Sie sich eine MySQL-Quelldistribution (eine RPM oder die `tar.gz`-Distribution) und installieren Sie statt dessen diese.
- Führen Sie `mysql_install_db --force` aus. Das führt nicht den `resolveip`-Test in `mysql_install_db` aus. Der Nachteil ist, dass Sie keine Hostnamen in the Berechtigungstabellen benutzen können, sondern nur IP-Nummern (ausser für `localhost`). Wenn Sie ein altes MySQL-Release benutzen, das `--force` nicht unterstützt, müssen Sie den `resolveip`-Test in `mysql_install` mit einem Editor deaktivieren.
- Starten Sie `mysqld` mit `su` anstelle von `--user`.

Die Linux-Intel-Binärdatei und die RPM-Releases von MySQL sind für höchst mögliche Geschwindigkeit konfiguriert. Wir versuchen immer, den schnellsten stabilen Kompiler zu benutzen, der verfügbar ist.

MySQL-Perl-Unterstützung erfordert Perl-Version 5.004_03 oder neuer.

Auf einigen Linux-2.2-Versionen erhalten Sie womöglich den Fehler `Resource temporarily unavailable`, wenn Sie eine Menge neuer Verbindungen zu einem `mysqld`-Server über TCP/IP aufmachen.

Das Problem liegt darin, dass Linux eine Verzögerung zwischen dem Schließen eines TCP/IP-Sockets und dem tatsächlichen Freigeben durch das System hat. Da es nur Platz für eine bestimmte Anzahl von TCP/IP-Slots gibt, bekommen Sie den genannten Fehler, wenn Sie viele neue TCP/IP-Verbindungen innerhalb kurzer Zeit aufbauen, zum Beispiel, wenn Sie den MySQL-`test-connect`-Benchmark über TCP/IP laufen lassen.

Wir haben dieses Problem mehrfach an verschiedene Linux-Mailing-Listen geschrieben, konnten aber bislang keine saubere Lösung erhalten.

Die einzige bekannte 'Behebung' des Problems liegt darin, persistente Verbindungen bei Ihren Clients zu verwenden oder Sockets zu benutzen, wenn Sie den Datenbankserver und die Clients auf derselben Maschine laufen lassen. Wir hoffen, dass zukünftig der `Linux 2.4`-Kernel dieses Problem lösen wird.

3.6.1.2. Anmerkungen zu Linux x86

MySQL erfordert `libc`-Version 5.4.12 oder neuer. Bekannt ist, dass `libc` 5.4.46 funktioniert. `glibc`-Version 2.0.6 und später sollten ebenfalls funktionieren. Es hat einige Probleme mit den `glibc`-RPMs von RedHat gegeben. Wenn Sie Probleme haben, prüfen Sie daher, ob es Updates gibt! Die `glibc`-2.0.7-19- und -2.0.7-29-RPMs funktionieren bekanntermaßen ebenfalls.

Bei einigen älteren Linux-Distributionen kann `configure` einen Fehler wie folgt produzieren:

```
Syntaxfehler in sched.h. Ändern Sie _P zu __P in der
/usr/include/sched.h-Datei. Siehe das Installationskapitel im
Referenzhandbuch.
```

Machen Sie, was die (englischsprachige) Fehlermeldung sagt. Fügen Sie also einen zusätzlichen Unterstrich zum `_P`-Makro hinzu, das nur einen Unterstrich hat, und versuchen Sie es noch einmal.

Möglicherweise erhalten Sie beim Kompilieren Warnungen. Die folgenden davon können ignoriert werden:

```
mysqld.cc -o objs-thread/mysqld.o
mysqld.cc: In function `void init_signals()':
mysqld.cc:315: warning: assignment of negative value `-1' to `long unsigned int'
mysqld.cc: In function `void * signal_hand(void *)':
mysqld.cc:346: warning: assignment of negative value `-1' to `long unsigned int'
```

In Debian-GNU/Linux müssen Sie folgendes tun, damit MySQL beim Hochfahren des Systems automatisch startet:

```
shell> cp support-files/mysql.server /etc/init.d/mysql.server
shell> /usr/sbin/update-rc.d mysql.server defaults 99
```

`mysql.server` befindet sich im `share/mysql`-Verzeichnis unterhalb des MySQL-Installationsverzeichnisses oder im `support-files`-Verzeichnis des MySQL-Source-Trees.

Wenn `mysqld` beim Start immer einen Speicherauszug (Core Dump) erzeugt, kann das Problem darin liegen, dass Sie eine alte `/lib/libc.a` haben. Versuchen Sie sie umzubenennen, entfernen Sie dann `sql/mysqld`, führen Sie ein neues `make install` durch und versuchen Sie es noch einmal. Dieses Problem wurde von einigen Slackware-Installationen berichtet.

Wenn Sie beim Linken von `mysqld` folgenden Fehler erhalten, bedeutet das, dass Ihre `libg++.a` nicht korrekt installiert ist:

```
/usr/lib/libc.a(putc.o): In function `_IO_putc':
putc.o(.text+0x0): multiple definition of `_IO_putc'
```

Sie können vermeiden, dass `libg++.a` benutzt wird, indem Sie `configure` wie folgt ablaufen lassen:

```
shell> CXX=gcc ./configure
```

3.6.1.3. Anmerkungen zu Linux SPARC

Bei einigen Implementationen ist `readdir_r()` fehlerhaft. Das äußert sich darin, dass `SHOW DATABASES` immer einen leeren Satz (Empty Set) zurück gibt. Das kann behoben werden, indem `HAVE_READDIR_R` aus

3.6.1.4. Anmerkungen zu Linux Alpha

MySQL-Version 3.23.12 ist die erste MySQL-Version, die auf Linux-Alpha getestet wurde. Wenn Sie planen, MySQL auf Linux-Alpha einzusetzen, stellen Sie sicher, dass Sie diese oder eine neuere Version haben.

Wir haben MySQL auf Alpha mit unseren Benchmarks und unserer Test-Suite getestet, und es scheint gut zu funktionieren. Hauptsächlich noch nicht getestet haben wird, wie die Dinge mit vielen gleichzeitigen Verbindungen funktionieren.

Wir kompilieren die Standard-MySQL-Binärdatei mit SuSE 6.4, Kernel 2.2.13-SMP, Compaq-C-Kompiler Version 6.2-504 und Compaq-C++-Kompiler Version 6.3-005 auf einer Compaq-DS20-Maschine mit einem Alpha-EV6-Prozessor.

Sie finden die genannten Kompiler auf http://www.support.compaq.com/alpha-tools). Durch die Verwendung dieser Kompiler anstelle von gcc erhalten wir eine 9% bis 14% bessere Performance für MySQL.

Beachten Sie, dass die Konfigurationszeile die Binärversion auf die aktuelle CPU optimiert. Das heißt, dass Sie unsere Binärdatei nur benutzen können, wenn Sie einen Alpha-EV6-Prozessor haben. Ausserdem haben wir statisch kompiliert, um Bibliothek-Probleme zu vermeiden.

```
CC=ccc CFLAGS="-fast" CXX=cxx CXXFLAGS="-fast -noexceptions -nortti" ./configure --prefix=/usr/local/mysql --disable-sh
```

Bei Benutzung von egcs funktionierte bei uns die folgende Konfigurationszeile:

```
CFLAGS="-O3 -fomit-frame-pointer" CXX=gcc CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors -fno-exceptions -fno
```

Einige bekannte Probleme, wenn MySQL auf Linux-Alpha läuft:

- Das Debuggen von threaded Applikationen wie MySQL funktioniert nicht mit `gdb 4.18`. Statt dessen sollten Sie gdb 5.0 herunter laden und benutzen!
- Wenn Sie versuchen, `mysqld` unter Benutzung von `gcc` statisch zu linken, wird das resultierende Image beim Starten einen Speicherauszug (Core Dump) erzeugen. Mit anderen Worten: Benutzen Sie **NICHT** `--with-mysqld-ldflags=-all-static` mit `gcc`.

3.6.1.5. Anmerkungen zu Linux PowerPC

MySQL sollte auf MkLinux mit dem neuesten `glibc`-Paket funktionieren (getestet mit `glibc` 2.0.7).

3.6.1.6. Anmerkungen zu Linux MIPS

Um MySQL auf Qube2 zum Laufen zu bringen (Linux Mips), benötigen Sie die neuesten `glibc`-Bibliotheken (`glibc-2.0.7-29C2` funktioniert bekanntermaßen). Ausserdem müssen Sie den `egcs`-C++-Kompiler (`egcs-1.0.2-9`, `gcc 2.95.2` oder neuer) benutzen.

3.6.1.7. Anmerkungen zu Linux IA64

Um MySQL auf Linux Ia64 zu kompilieren, mussten wir folgendes tun (wir vermuten, dass das leichter wird, wenn die neue gcc-Version für ia64 herausgebracht wird).

Unter Verwendung von `gcc-2.9-final`:

```
CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors -fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mys
```

Nach `make` werden Sie einen Fehler erhalten, dass `sql/opt_range.cc` nicht kompiliert (interner Kompiler-Fehler). Um das zu beheben, gehen Sie ins sql-Verzeichnis und tippen Sie erneut `make` ein. Kopieren Sie die Kompilierzeile, ändern Sie aber -O2 zu -O0. Die Datei sollte nunmehr kompilieren.

Jetzt können Sie folgendes tun:

```
cd ..
make
make_install
```

und `mysqld` sollte lauffähig sein.

Auf Ia64 benutzen die MySQL-Client-Binärdateien gemeinsam genutzte (shared) Bibliotheken. Wenn Sie daher unsere Binärdistribution an anderer Stelle als `/usr/local/mysql` benutzen, müssen Sie entweder `/etc/ld.so.conf` ändern oder den Pfad zum Verzeichnis hinzufügen, wo Sie `libmysqlclient.so` haben, und zwar in der `LD_LIBRARY_PATH`-Umgebungsvariablen.

See Abschnitt A.3.1, „Probleme beim Linken mit der MySQL-Client-Bibliothek“.

3.6.2. Anmerkungen zu Windows

Dieser Abschnitt beschreibt Installation und Benutzung von MySQL auf Windows. Diese Information steht zusätzlich in der `README`-Datei, die mit der MySQL-Windows-Distribution mitgeliefert wird.

3.6.2.1. Wie man MySQL auf Win95 / Win98 startet

MySQL benutzt TCP/IP, um einen Client mit einem Server zu verbinden. (Das erlaubt jeder beliebigen Maschine in Ihrem Netzwerk, sich mit Ihrem MySQL-Server zu verbinden.) Aus diesem Grund müssen Sie MySQL auf Ihrer Maschine installieren, bevor Sie MySQL starten. Sie finden TCP/IP auf Ihrer Windows-CD-ROM.

Beachten Sie, dass Sie bei Verwendung eines alten Win95-Releases (zum Beispiel OSR2) wahrscheinlich ist, dass Sie ein altes Winsock-Paket haben! MySQL erfordert Winsock 2! Sie erhalten das neueste Winsock von http://www.microsoft.com/. Win98 enthält die neue Winsock-2-Bibliothek, deshalb trifft das Gesagte nicht auf Win98 zu.

Um den `mysqld`-Server zu starten, öffnen Sie ein MS-DOS-Fenster (MS-DOS-Eingabeaufforderung) und geben Sie ein:

```
C:\> C:\mysql\bin\mysqld
```

Das startet `mysqld` im Hintergrund ohne Fenster.

Sie können den MySQL-Server killen, indem Sie eingeben:

```
C:\> C:\mysql\bin\mysqladmin -u root shutdown
```

Beachten Sie, dass Win95 und Win98 die Erzeugung von Named Pipes nicht unterstützen. Auf Win95 und Win98 können Sie Named Pipes nur benutzen, um sich zu einem entfernten MySQL-Server zu verbinden, der auf einem NT-Server-Host läuft. (Natürlich muss auch der MySQL-Server Named Pipes unterstützen. Beispielsweise läßt die Verwendung von `mysqld-opt` unter NT keine Named-Pipe-Verbindungen zu. Sie sollten daher entweder `mysqld-nt` oder `mysqld-max-nt` verwenden.)

Wenn `mysqld` nicht startet, überprüfen Sie bitte die `\mysql\data\mysql.err`-Datei um zu sehen, ob der Server eine Meldung ausgegeben hat, die auf die Ursache des Problems hinweist. Sie können auch versuchen, den Server mit `mysqld -`

`-standalone` zu starten. In diesem Fall erscheinen vielleicht nützliche Informationen auf dem Bildschirm, die Ihnen bei der Lösung des Problems helfen.

Die letzte Option besteht darin, `mysqld` mit `--standalone --debug` zu starten. In diesem Fall schreibt `mysqld` eine Log-Datei `C:\mysqld.trace`, die die Ursache enthalten könnte, warum `mysqld` nicht startet. See Abschnitt E.1.2, „Trace-Dateien erzeugen“.

3.6.2.2. MySQL auf Windows NT oder Windows 2000 starten

Der Win95-/Win98-Abschnitt trifft auch auf NT/Win2000 zu, mit folgenden Unterschieden:

Damit MySQL mit TCP/IP auf NT läuft, müssen Sie Service-Pack 3 (oder neuer) installieren!

Beachten Sie, dass alles Folgende, das für NT zutrifft, ebenfalls für Win2000 zutrifft!

Für NT/Win2000 ist der Servername `mysqld-nt`. Normalerweise sollten Sie MySQL auf NT/Win2000 als Systemdienst installieren:

```
C:\> C:\mysql\bin\mysqld-nt --install
```

oder

```
C:\> C:\mysql\bin\mysqld-max-nt --install
```

(Unter Windows NT können Sie in der Tat jede der Server-Binärdateien als Systemdienst installieren, aber nur diejenigen, die Namen haben, die auf `-nt.exe` enden, bieten Unterstützung für Named Pipes.)

Sie können MySQL mit diesen Befehlen starten und anhalten:

```
C:\> NET START mysql
C:\> NET STOP mysql
```

Beachten Sie, dass Sie in diesem Fall keine weiteren Optionen für `mysqld-nt` angeben können!

Sie können `mysqld-nt` auf NT auch als allein ablaufendes Programm (Stand-alone) laufen lassen, wenn Sie `mysqld-nt` mit irgend welchen Optionen starten wollen! Wenn Sie `mysqld-nt` auf NT ohne Optionen laufen lassen, versucht `mysqld-nt`, sich mit den Vorgabeoptionen als Systemdienst zu starten. Wenn Sie `mysqld-nt` angehalten haben, müssen Sie es mit `NET START mysql` neu starten.

Der Systemdienst wird installiert mit dem Namen `MySQL`. Einmal installiert, muss er mit dem Systemdienst-Steuerungs-Manager (SCM) in der Systemsteuerung gestartet werden, oder indem Sie den `NET START MySQL`-Befehl benutzen. Wenn irgend welche Optionen angegeben werden sollen, müssen diese als ``Startparameter'' im SCM-Dienstprogramm angegeben werden, bevor Sie den MySQL-Dienst starten. Wenn `mysqld-nt` läuft, kann er mit `mysqladmin` oder dem SCM-Dienstprogramm angehalten werden, oder indem Sie den Befehl `NET STOP MySQL` benutzen. Wenn Sie SCM benutzen `mysqld-nt`, um den Server anzuhalten, gibt es eine seltsame Meldung von SCM über `mysqld shutdown normally`. Wenn er als Systemdienst läuft, hat `mysqld-nt` keinen Zugriff auf die Konsole. Daher werden auch keine Meldungen angezeigt.

Auf NT erhalten Sie möglicherweise folgende Systemdienst-Fehlermeldungen:

Zugriff verboten	Bedeutung: `mysqld-nt.exe` kann nicht gefunden werden.
Kann nicht registrieren	Bedeutung: Der Pfad ist falsch.
Installation des Systemdienstes fehlgeschlagen.	Bedeutung: Der Systemdienst ist bereits installiert oder der Systemdienst-Steuerungs-Manager ist in einem schlechten Zustand.

Wenn Sie Problem haben, `mysqld-nt` als Systemdienst zu installieren, versuchen Sie, ihn mit dem vollen Pfad zu installieren:

```
C:\> C:\mysql\bin\mysqld-nt --install
```

Wenn das nicht funktioniert, können Sie erreichen, dass `mysqld-nt` korrekt startet, indem Sie den Pfad in der Registrierung korrigieren!

Wenn Sie nicht wollen, dass `mysqld-nt` als Systemdienst startet, können Sie ihn wie folgt starten:

```
C:\> C:\mysql\bin\mysqld-nt --standalone
```

oder

```
C:\> C:\mysql\bin\mysqld --standalone --debug
```

Letztgenanntes gibt Ihnen eine Debug-Spur in `C:\mysqld.trace`. See Abschnitt E.1.2, „Trace-Dateien erzeugen“.

3.6.2.3. MySQL auf Windows laufen lassen

MySQL unterstützt TCP/IP auf allen Windows-Plattformen und Named Pipes auf NT. Vorgabemäßig werden Named Pipes für lokale Verbindungen auf NT und TCP/IP für alle anderen Fälle benutzt, wenn der Client TCP/IP installiert hat. Der Hostname legt fest, welches Protokoll benutzt wird:

Hostname	Protokoll
NULL (keiner)	Auf NT zuerst Named Pipes versuchen. Wenn das nicht funktioniert, TCP/IP benutzen. Auf Win95/Win98 wird TCP/IP benutzt.
.	Named Pipes
localhost	TCP/IP zum aktuellen Host
hostname	TCP/IP

Sie können erzwingen, dass ein MySQL-Client Named Pipes benutzt, indem Sie die `--pipe`-Option oder `.` als Hostnamen angeben. Benutzen Sie die `--socket`-Option, um den Namen der Pipe festzulegen.

Sie können feststellen, ob MySQL funktioniert, indem Sie die folgenden Befehle eingeben:

```
C:\> C:\mysql\bin\mysqlshow
C:\> C:\mysql\bin\mysqlshow -u root mysql
C:\> C:\mysql\bin\mysqladmin version status proc
C:\> C:\mysql\bin\mysql test
```

Wenn `mysqld` nur langsam auf Verbindungen auf Win95/Win98 antwortet, gibt es wahrscheinlich ein Problem mit Ihrem DNS. Starten Sie in diesem Fall `mysqld` mit `--skip-name-resolve` und benutzen Sie nur `localhost` und IP-Nummern in den MySQL Berechtigungstabellen. Sie können DNS bei einer Verbindung zu einem `mysqld-nt`-MySQL-Server, der auf NT läuft, ebenfalls dadurch vermeiden, dass Sie das `--pipe`-Argument verwenden, um die Benutzung von Named Pipes festzulegen. Das funktioniert bei den meisten MySQL-Clients.

Es gibt zwei Versionen des MySQL-Kommadozeilen-Werkzeugs:

`mysql`	Kompiliert auf nativem Windows, was sehr eingeschränkte Texteditiermöglichkeiten bietet.
`mysqlc`	Kompiliert mit dem Cygnus-GNU-Kompiler und -Bibliotheken, was `readline`-Editiermöglichkeit bietet.

Wenn Sie `mysqlc.exe` benutzen wollen, müssen Sie `C:\mysql\lib\cygwinb19.dll` in Ihr Windows-Systemverzeichnis kopieren (`\windows\system` oder ein ähnlicher Ort).

Vorgabemäßig geben die Berechtigungen auf Windows allen lokalen Benutzern volle Zugriffsrechte auf alle Datenbanken, ohne ein Passwort anzugeben. Um MySQL sicherer zu machen, sollten Sie für alle Benutzer ein Passwort setzen und die Zeile in der Tabelle `mysql.user`, die `Host='localhost'` und `User=''` enthält, löschen.

Sie sollten auch für den `root`-Benutzer ein Passwort vergeben. Das folgende Beispiel entfernt den anonymen Benutzer, der von jedem genutzt werden kann, um auf die `test`-Datenbank zuzugreifen und setzt dann für den `root`-Benutzer ein Passwort:

```
C:\> C:\mysql\bin\mysql mysql
mysql> DELETE FROM user WHERE Host='localhost' AND User='';
mysql> QUIT
C:\> C:\mysql\bin\mysqladmin reload
C:\> C:\mysql\bin\mysqladmin -u root password ihr_passwort
```

Nachdem Sie das Passwort gesetzt haben, sollten Sie den `mysqld`-Server herunter fahren, was Sie mit folgendem Befehl bewerkstelligen können:

```
C:\> mysqladmin --user=root --password=ihr_passwort shutdown
```

Wenn Sie die alte Shareware-Version von MySQL-Version 3.21 unter Windows benutzen, schlägt der genannte Befehl mit einem Fehler fehl: `parse error near 'SET OPTION password'`. Die Lösung besteht darin, auf die aktuelle MySQL-Version zu aktualisieren, die frei verfügbar ist.

Mit den neuen MySQL-Versionen können Sie auf einfache Art neue Benutzer hinzufügen und Zugriffsrechte mit den `GRANT`- und `REVOKE`-Befehlen ändern. See Abschnitt 5.3.1, „`GRANT`- und `REVOKE`-Syntax“.

3.6.2.4. Verbinden mit einem entfernten MySQL-Server von Windows mit SSH aus

Hier ist eine Anmerkung dazu, wie man sich über eine sichere Verbindung zu einem entfernten MySQL-Server mit SSH verbindet (von David Carlson <dcarlson@mplcomm.com>):

- Installieren Sie einen SSH-Client auf Ihrer Windows-Maschine. Das beste nicht kostenlose Werkzeug, das ich gefunden habe, ist SecureCRT von http://www.vundyke.com/. Eine andere Option ist f-secure von http://www.f-secure.com/. Sie finden kostenlose Werkzeuge über **Google** auf http://directory.google.com/Top/Computers/Security/Products_and_Tools/Cryptography/SSH/Clients/Windows/.
- Starten Sie Ihren Windows-SSH-Client. Konfigurieren Sie: Host_Name = ihr_mysql_server_URL_oder_IP. Konfigurieren Sie: userid=ihre_userid, um sich an Ihrem Server anzumelden (wahrscheinlich nicht dasselbe wie Ihr MySQL-Benutzername / -Passwort).
- Konfigurieren Sie Port-Forwarding. Machen Sie entweder ein Remote Forward (einstellen: local_port: 3306, remote_host: ihr_mysql_servername_oder_ip, remote_port: 3306) oder ein lokales Forward (einstellen: port: 3306, host: localhost, remote port: 3306).
- Speichern Sie alles, damit Sie es beim nächsten Mal nicht noch einmal eingeben müssen.
- Melden Sie sich an Ihrem Server mit der SSH-Sitzung, die Sie gerade erzeugt haben.
- Starten Sie auf Ihrer Windows-Maschine irgend eine Applikation wie Access.
- Erzeugen Sie unter Windows eine neue Datei und stellen Sie eine Verknüpfung zu MySQL her, indem Sie den ODBC-Treiber so benutzen, wie Sie es normalerweise tun, AUSSER dass Sie localhost als MySQL-Host-Server eingeben - NICHT yourmysqlservername.

Jetzt sollten Sie eine ODBC-Verbindung zu MySQL haben, die mit SSH verschlüsselt ist.

3.6.2.5. Daten auf verschiedenen Platten unter Win32 aufteilen

Ab MySQL-Version 3.23.16 werden die mysqld-max- und mysql-max-nt-Server in der MySQL-Distribution mit der -DUSE_SYMDIR-Option kompiliert. Das gibt Ihnen die Möglichkeit, Datenbanken auf verschiedene Festplatten zu verteilen, indem Sie symbolische Links darauf machen (in ähnlicher Weise, wie symbolische Links unter Unix funktionieren).

Unter Windows legen Sie einen symbolischen Link auf eine Datenbank an, indem Sie eine Datei erzeugen, die den Pfad zum Zielverzeichnis enthält, und diese Datei im mysql_data-Verzeichnis unter dem Dateiname Datenbank.sym speichern. Beachten Sie, dass der symbolische Link nur dann benutzt wird, wenn das Verzeichnis mysql_data_dir\datenbank nicht existiert.

Wenn Ihr MySQL-Daten-Verzeichnis beispielsweise C:\mysql\data ist und Sie die Datenbank foo dort haben wollen, die aber in D:\data\foo liegt, erzeugen Sie die Datei C:\mysql\data\foo.sym, die als Text D:\data\foo\ enthält. Dann werden alle Tabellen, die in der Datenbank foo sind, in D:\data\foo erzeugt.

Beachten Sie, dass wir dieses Feature nicht vorgabemäßig aktiviert haben, weil es mit Geschwindigkeitsnachteilen verbunden ist. Es ist selbst dann nicht aktiviert, wenn Sie MySQL mit Unterstützung dafür kompiliert haben. Um symbolische Links zu aktivieren, müssen Sie in Ihre my.cnf- oder my.ini-Datei folgenden Eintrag machen:

```
[mysqld]
use-symbolic-links
```

In MySQL 4.0 werden symbolische Links vorgabemäßig aktiviert sein. Wenn Sie dies deaktivieren wollen, benutzen Sie die skip-symlink-Option.

3.6.2.6. MySQL-Clients auf Windows kompilieren

In Ihren Quell-Dateien sollten Sie windows.h einschließen, bevor Sie mysql.h einschließen:

```
#if defined(_WIN32) || defined(_WIN64)
#include <windows.h>
#endif
#include <mysql.h>
```

Sie können Ihren Code entweder mit der dynamischen libmysql.lib-Bibliothek linken, die nur ein Wrapper zum Laden der libmysql.dll bei Bedarf ist, oder mit der statischen mysqlclient.lib-Bibliothek.

Beachten Sie, dass MySQL-Client-Bibliotheken als threaded Bibliotheken kompiliert werden, daher sollten Sie auch Ihren Code so

kompilieren, dass er multi-threaded ist!

3.6.2.7. MySQL-Windows im Vergleich zu Unix-MySQL

MySQL-Windows hat sich mittlerweile als sehr stabil erwiesen. Diese Version von MySQL hat dieselben Features wie die entsprechende Unix-Version, allerdings mit folgenden Ausnahmen:

- **Windows 95 und Threads**

  Windows 95 hat ein etwa 200 Bytes großes Hauptspeicher-Leck (Memory Leak) für jede Thread-Erzeugung. Jede Verbindung zu MySQL erzeugt eine neues Thread, daher sollten Sie `mysqld` nicht für längere Zeiträume auf Windows 95 laufen lassen, wenn Ihr Server viele Verbindungen handhabt! Windows NT und Windows 98 haben diesen Bug nicht.

- **Gleichzeitige Lesezugriffe**

  MySQL vertraut auf `pread()`- und `pwrite()`-Aufrufe, um in der Lage zu sein, `INSERT` und `SELECT` zu mischen. Momentan benutzen wir mutexes, um `pread()` / `pwrite()` zu emulieren. Langfristig werden wir die Dateiebenen-Schnittstelle durch eine virtuelle Schnittstelle ersetzen, um die `readfile()`- / `writefile()`-Schnittstelle auf NT mit höherer Geschwindigkeit benutzen zu können. Die aktuelle Implementation begrenzt die Anzahl offener Dateien, die MySQL benutzen kann, auf 1024, was bedeutet, dass Sie nicht so viele gleichzeitige Threads auf NT benutzen können wie auf Unix.

- **Blockierendes Lesen**

  MySQL benutzt blockierendes Lesen (Blocking Read) für jede Verbindung. Das bedeutet in der Anwendung:

  - Eine Verbindung wird nicht automatisch nach 8 Stunden abgebaut, wie es unter der Unix-Version von MySQL der Fall ist.
  - Wenn eine Verbindung hängen bleibt, ist es unmöglich, sie abzubrechen, ohne MySQL zu killen.
  - `mysqladmin kill` funktioniert nicht für schlafende Verbindungen.
  - `mysqladmin shutdown` kann nicht abgebrochen werden, solange es noch schlafende Verbindungen gibt.

  Geplant ist, dieses Problem zu beheben, sobald unsere Windows-Entwickler ein nettes Workaround heraus gefunden haben.

- **UDF-Funktionen**

  Momentan unterstützt MySQL-Windows keine benutzerdefinierten Funktionen (UDF, user defined functions).

- **`DROP DATABASE`**

  Sie können keine Datenbank löschen, die durch irgend einen Thread in Benutzung ist.

- **MySQL vom Task-Manager aus killen**

  Sie können MySQL nicht vom Task-Manager oder mit dem Shutdown-Dienstprogramm unter Windows 95 killen. Sie müssen es mit `mysqladmin shutdown` herunter fahren.

- **Von Groß-/Kleinschreibung unabhängige Namen**

  Unter Windows sind Dateinamen unabhängig von der Groß-/Kleinschreibung. Daher sind Datenbank- und Tabellennamen in MySQL für Windows ebenfalls unabhängig von der Groß-/Kleinschreibung. Die einzige Einschränkung ist die, dass Datenbank- und Tabellennamen innerhalb eines bestimmten Statements dieselbe Groß-/Kleinschreibung haben müssen. See Abschnitt A.5.1, „Groß-/Kleinschreibung beim Suchen“.

- **Das ‘`\`’-Verzeichnis-Zeichen**

  Bestandteile von Pfadnamen werden unter Windows mit dem ‘`\`’-Zeichen getrennt, das in MySQL als Fluchtzeichen (Escape Character) dient. Wenn Sie `LOAD DATA INFILE` oder `SELECT ... INTO OUTFILE` benutzen, müssen Sie ‘`\`’ an solchen Stellen doppelt eingeben:

```
mysql> LOAD DATA INFILE "C:\\tmp\\skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

  Alternativ können Sie auch Dateinamen im Unix-Stil mit ‘`/`’-Zeichen benutzen:

```
mysql> LOAD DATA INFILE "C:/tmp/skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

- **`Can't open named pipe`-Fehler**

  Wenn Sie MySQL-Version 3.22 auf NT mit den neuesten MySQL-Clients benutzen, erhalten Sie folgende Fehlermeldung:

  ```
  error 2017: can't open named pipe to host: . pipe...
  ```

  Das liegt daran, dass die MySQL-Version für NT auf NT vorgabemäßig Named Pipes benutzt. Sie können diesen Fehler vermeiden, indem Sie bei den neuen MySQL-Clients die `--host=localhost`-Option benutzen oder eine Optionsdatei `C:\my.cnf` anlegen, die folgendes enthält:

  ```
  [client]
  host = localhost
  ```

- **`Access denied for user`-Fehler**

  Wenn Sie den Fehler `Access denied for user: 'ein-benutzer@unknown' to database 'mysql'` erhalten, wenn Sie auf einen MySQL-Server auf derselben Maschine zugreifen, heißt das, dass MySQL Ihren Hostnamen nicht richtig auflösen kann.

  Um das zu beheben, legen Sie eine Datei `\windows\hosts` mit folgender Zeile an:

  ```
  127.0.0.1       localhost
  ```

- **`ALTER TABLE`**

  Wenn Sie ein `ALTER TABLE`-Statement ausführen, ist die Tabelle gegen Benutzung durch andere Threads gesperrt. Das hat damit zu tun, dass Sie unter Windows keine Datei löschen können, die durch andere Threads in Benutzung ist. (Zukünftig finden wir möglicherweise einen Weg, dieses Problem zu umgehen.)

- **`DROP TABLE` auf eine Tabelle, die durch eine**

  `MERGE`-Tabelle in Benutzung ist, funktioniert nicht. Der `MERGE`-Handler führt sein Tabellen-Mapping versteckt vor MySQL durch. Weil Windows das Löschen von Dateien verbietet, die offen sind, müssen Sie zuerst alle `MERGE`-Tabellen flushen (mit `FLUSH TABLES`) oder die `MERGE`-Tabelle löschen, bevor Sie die Tabelle löschen. Wir werden das zusammen mit der Einführung von Sichten (`VIEW`s) beheben.

Hier sind einige Themen für diejenigen, die uns beim Windows-Release helfen wollen:

- Einen Ein-Benutzer-Server `MYSQL.DLL` herstellen. Das könnte alles beinhalten, was einen Standard-Server ausmacht, ausser Thread-Erzeugung. Das würde es erheblich erleichtern, MySQL in Applikationen zu benutzen, die keinen echten Client/Server und keinen Zugriff auf den Server von anderen Hosts benötigen.
- Ein paar nette Start- und Stop-Icons zur MySQL-Installation hinzufügen.
- Ein Werkzeug bauen, das Registrierungseinträge für die MySQL-Startoptionen handhabt. Das Lesen der Registrierungseinträge ist bereits in `mysqld.cc` kodiert, sollte aber umgeschrieben werden, damit es mehr Parameter-orientiert ist. Das Werkzeug sollte auch in der Lage sein, die `C:\my.cnf`-Optionsdatei zu aktualisieren, wenn der Benutzer diese lieber als die Registrierungsdatei benutzen will.
- Wenn man `mysqld` als Systemdienst mit `--install` (auf NT) installiert, wäre es nett, wenn man vorgabemäßige Optionen auf der Kommandozeile hinzufügen könnte. Im Moment muss man diese fehlende Möglichkeit durch eine Liste der Parameter in der `C:\my.cnf`-Datei ersetzen.
- Es wäre eine feine Sache, wenn man `mysqld` vom Task-Manager aus killen könnte. Momentan muss man `mysqladmin shutdown` benutzen.
- `readline` auf Windows portieren, damit es im `mysql`-Kommandozeilen-Werkzeug benutzt werden kann.
- GUI-Versionen der Standard-MySQL-Clients (`mysql`, `mysqlshow`, `mysqladmin` und `mysqldump`) wären nett.
- Nett wäre auch, wenn die Socket-Lese- und Schreib-Funktionen in `net.c` unterbrechbar wären. Das würde es ermöglichen, offen Threads mit `mysqladmin kill` auf Windows zu killen.

- following two lines? `mysqld` always starts in the "C" locale und not in the default locale. We would like to have `mysqld` use the current locale für the sort order.
- Benutzerdefinierte Funktionen (UDF) mit `.DLL`s implementieren.
- Makros hinzufügen, um die schnelleren, Thread-sicheren Inkrementierungs-/Dekrementierungsmethoden nutzen zu können, die Windows bietet.

Weitere Windows-spezifische Themen sind in der `README`-Datei beschrieben, die mit der MySQL-Windows-Distribution ausgeliefert wird.

3.6.3. Anmerkungen zu Solaris

Auf Solaris bekommen Sie vielleicht schon Probleme, bevor Sie überhaupt Ihre MySQL-Distribution entpackt haben! Solaris-`tar` kann nicht mit langen Dateinamen umgehen. Daher sehen Sie vielleicht einen Fehler wie den folgenden, wenn Sie MySQL entpacken:

```
x mysql-3.22.12-beta/bench/Results/ATIS-mysql_odbc-NT_4.0-cmp-db2,informix,ms-sql,mysql,oracle,solid,sybase, 0 Bytes, 0
tar: directory checksum error (Verzeichnis-Prüfsummenfehler)
```

In diesem Fall müssen Sie GNU-`tar` (`gtar`) benutzen, um die Distribution zu entpacken. Sie finden eine vorkompilierte Version für Solaris auf http://www.mysql.com/downloads/.

Native Sun-Threads funktinieren nur auf Solaris 2.5 und höher. Auf 2.4 und früher benutzt MySQL automatisch MIT-pThreads. See Abschnitt 3.3.6, „Anmerkungen zu MIT-pThreads“.

Vielleicht erhalten Sie von configure folgenden Fehler:

```
checking for restartable system calls... configure: error can not run test
programs while cross compiling
```

Das bedeutet, dass mit Ihrer Kompiler-Installation etwas nicht stimmt! In diesem Fall sollten Sie Ihren Kompiler auf eine neuere Version aktualisieren. Eventuell sind Sie in der Lage, das Problem zu lösen, indem Sie folgende Zeile in die `config.cache`-Datei einfügen:

```
ac_cv_sys_restartable_syscalls=${ac_cv_sys_restartable_syscalls='no'}
```

Wenn Sie Solaris auf einer SPARC benutzen, ist der empfohlene Kompiler `gcc` 2.95.2. Sie finden ihn auf http://gcc.gnu.org/. Beachten Sie, dass `egcs` 1.1.1 und `gcc` 2.8.1 auf SPARC nicht zuverlässig laufen!

Die empfohlene `configure`-Zeile ist bei der Benutzung von `gcc` 2.95.2:

```
CC=gcc CFLAGS="-O3" \
CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory --enable-assembler
```

Wenn Sie eine Ultra-Sparc haben, erhalten Sie 4 % mehr Performance, wenn Sie "-mcpu=v8 -Wa,-xarch=v8plusa" zu CFLAGS und CXXFLAGS hinzufügen.

Wenn Sie einen Sun Workshop (Fortre) 5.3 (oder neueren) Kompiler haben, können Sie `configure` wie folgt laufen lassen:

```
CC=cc CFLAGS="-Xa -fast -xO4 -native -xstrconst -mt" \
CXX=CC CXXFLAGS="-noex -xO4 -mt" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

In den MySQL-Benchmarks haben wir auf einer Ultra-Sparc 6% Geschwindigkeitssteigerung erreicht, wenn wir Sun Workshop 5.3 benutzen, im Vergleich mit der Benutzung von gcc mit -mcpu-Flags.

Wenn Sie Probleme mit `fdatasync` oder `sched_yield` bekommen, können Sie diese beheben, indem Sie `LIBS=-lrt` zur Konfigurationszeile hinzufügen.

Der folgende Absatz ist nur für ältere Kompiler als WorkShop 5.3 relevant:

Eventuell müssen Sie auch das `configure`-Skript editieren und folgende Zeile ändern:

```
#if !defined(__STDC__) || __STDC__ != 1
```

Ändern zu:

```
#if !defined(__STDC__)
```

Wenn Sie `__STDC__` mit der `-Xc`-Option anschalten, kann der Sun-Kompiler nicht mit der Solaris-`pThread.h`-Header-Datei kompilieren. Das ist ein Bug von Sun (Kompiler-Problem oder beschädigte Include-Datei).

Wenn `mysqld` beim Laufenlassen eine Fehlermeldung wie die unten stehende ausgibt, haben Sie versucht, MySQL mit dem Sun-Kompiler zu kompilieren, ohne die Multi-Thread-Option (`-mt`) anzuschalten:

```
libc internal error: _rmutex_unlock: rmutex not held
```

Fügen Sie `-mt` zu `CFLAGS` und `CXXFLAGS` hinzu und versuchen Sie es noch einmal.

Wenn Sie folgenden Fehler beim Kompilieren von MySQL mit `gcc` erhalten, ist Ihr `gcc` nicht für Ihre Version von Solaris konfiguriert:

```
shell> gcc -O3 -g -O2 -DDBUG_OFF  -o thr_alarm ...
./thr_alarm.c: In function `signal_hand':
./thr_alarm.c:556: too many arguments to function `sigwait'
```

Die einzige richtige Möglichkeit in diesem Fall ist, sich die neueste Version von `gcc` zu besorgen und Sie mit Ihrem aktuellen `gcc`-Kompiler zu kompilieren. Zumindest auf Solaris 2.5 haben fast alle Binärversionen von `gcc` alte, unbrauchbare Include-Dateien, die alle Programme beschädigen, die Threads benutzen (und möglicherweise auch andere Programme)!

Solaris stellt keine statischen Versionen aller Systembibliotheken zur Verfügung (`libpThreads` und `libdl`). Daher können Sie MySQL nicht mit `--static` kompilieren. Wenn Sie es dennoch versuchen, erhalten Sie folgenden Fehler:

```
ld: fatal: library -ldl: not found

oder

undefined reference to `dlopen'

oder

cannot find -lrt
```

Wenn zu viele Prozesse zu schnell hintereinander versuchen, sich mit `mysqld` zu verbinden, werden Sie folgenden Fehler im MySQL-Log sehen:

```
Error in accept: Protocol error
```

Als Workaround können Sie versuchen, den Server mit der `--set-variable back_log=50`-Option zu starten. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

Wenn Sie Ihren eigenen MySQL-Client linken, erhalten Sie möglicherweise folgenden Fehler, wenn Sie versuchen, ihn auszuführen:

```
ld.so.1: ./my: fatal: libmysqlclient.so.#: open failed: No such file or directory
```

Dieses Problem kann mit einer der folgenden Methoden vermieden werden:

- Linken Sie den Client mit folgendem Flag (anstelle von `-Lpath`): `-Wl,r/full-path-to-libmysqlclient.so`.
- Kopieren Sie `libmysqclient.so` nach `/usr/lib`.
- Fügen Sie den Pfadnamen des Verzeichnisses, wo `libmysqlclient.so` liegt, der `LD_RUN_PATH`-Umgebungsvariablen hinzu, bevor Sie Ihren Client laufen lassen.

Wenn Sie die `--with-libwrap`-configure-Option benutzen, müssen Sie auch die Bibliotheken einschließen, die `libwrap.a` benötigt:

```
--with-libwrap="/opt/NUtcpwrapper-7.6/lib/libwrap.a -lnsl -lsocket
```

Wenn Sie Probleme mit configure haben, wenn Sie versuchen, mit `-lz` zu linken und keine `zlib` installiert haben, haben Sie zwei Möglichkeiten:

- Wenn Sie in der Lage sein wollen, dass komprimierte Kommunikationsprotokoll zu benutzen, müssen Sie zlib von ftp.gnu.org laden und installieren.

- Konfigurieren Sie mit `--with-named-z-libs=no`.

Wenn Sie gcc benutzen und Probleme mit dem Laden von `UDF`-Funktionen in MySQL haben, versuchen Sie, `-lgcc` zur Link-Zeile für die `UDF`-Funktion hinzuzufügen.

Wenn Sie wollen, dass MySQL automatisch startet, kopieren Sie `Support-files/mysql.server` nach `/etc/init.d` und erzeugen Sie einen symbolischen Link darauf, den Sie `/etc/rc3.d/S99mysql.server` nennen.

3.6.3.1. Anmerkungen zu Solaris 2.7/2.8

Normalerweise können Sie eine Solaris-2.6-Binärdatei für Solaris 2.7 und 2.8 benutzen. Die meisten Dinge, die Solaris 2.6 betreffen, treffen auch für Solaris 2.7 und 2.8 zu.

Beachten Sie, dass MySQL-Version 3.23.4 und höher in der Lage sein sollte, automatisch neue Versionen von Solaris zu erkennen und Workarounds für die folgenden Probleme zu aktivieren!

Solaris 2.7 / 2.8 hat einige Bugs in den Include-Dateien. Eventuell sehen Sie folgenden Fehler, wenn Sie `gcc` benutzen:

```
/usr/include/widec.h:42: warning: `getwc' redefined
/usr/include/wchar.h:326: warning: this is the location of the previous
definition
```

Wenn das auftritt, können Sie folgendes tun, um das Problem zu lösen:

Kopieren Sie `/usr/include/widec.h` nach `.../lib/gcc-lib/os/gcc-version/include` und ändern Sie Zeile 41 von:

```
#if     !defined(lint) && !defined(__lint)

nach

#if     !defined(lint) && !defined(__lint) && !defined(getwc)
```

Alternativ können Sie `/usr/include/widec.h` direkt editieren. Egal, wie Sie vorgehen: Nachdem Sie die Fehlerbehebung durchgeführt haben, sollten Sie `config.cache` entfernen und `configure` noch einmal laufen lassen!

Wenn Sie beim Laufenlassen von `make` folgende Fehler bekommen, liegt das daran, dass `configure` die `curses.h`-Datei nicht erkannte (vermutlich aufgrund des Fehlers in `/usr/include/widec.h`):

```
In file included by mysql.cc:50:
/usr/include/term.h:1060: syntax error before `,'
/usr/include/term.h:1081: syntax error before `;'
```

Das Problem lösen Sie auf eine der folgenden Weisen:

- Konfigurieren Sie mit `CFLAGS=-DHAVE_CURSES_H CXXFLAGS=-DHAVE_CURSES_H ./configure`.
- Editieren Sie `/usr/include/widec.h`, wie weiter oben gezeigt, und lassen Sie configure noch einmal laufen.
- Entfernen Sie die `#define HAVE_TERM`-Zeile aus der `config.h`-Datei und lassen Sie `make` noch einmal laufen.

Wenn Sie das Problem bekommen, dass Ihr Linker `-lz` nicht finden kann, wenn Sie Ihr Client-Programm linken, liegt das wahrscheinlich daran, dass Ihre `libz.so`-Datei in `/usr/local/lib` installiert ist. Sie können das mit einer der folgenden Methoden beheben:

- Fügen Sie `/usr/local/lib` zu `LD_LIBRARY_PATH` hinzu.
- Fügen Sie einen Link auf `libz.so` von `/lib` hinzu.
- Wenn Sie Solaris 8 benutzen, können Sie die optionale zlib aus Ihrer Solaris-8-CD-Distribution installieren.
- Konfigurieren Sie MySQL mit der `--with-named-z-libs=no`-Option.

3.6.3.2. Anmerkungen zu Solaris x86

Auf Solaris 2.8 auf x86 erzeugt `mysqld` einen Speicherauszug (Core Dump), wenn Sie darin 'strip' laufen lassen.

Wenn Sie `gcc` oder `egcs` auf Solaris x86 benutzen und Probleme mit Speicherauszügen (Core Dumps) unter Last erleben, sollten Sie folgenden `configure`-Befehl benutzen:

```
CC=gcc CFLAGS="-O3 -fomit-frame-pointer -DHAVE_CURSES_H" \
CXX=gcc \
CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors -fno-exceptions -fno-rtti -DHAVE_CURSES_H" \
./configure --prefix=/usr/local/mysql
```

Das vermeidet Probleme mit der `libstdc++`-Bibliothek und mit C++-Ausnahmefehlern.

Wenn das nicht hilft, sollten Sie eine Debug-Version kompilieren und sie mit einer Trace-Datei oder unter `gdb` laufen lassen. See Abschnitt E.1.3, „mysqld unter gdb debuggen“.

3.6.4. Anmerkungen zu BSD

3.6.4.1. Anmerkungen zu FreeBSD

FreeBSD 3.x wird für MySQL empfohlen, weil das Thread-Paket sehr viel integrierter ist.

Die einfachste und daher empfohlene Art der Installation ist die Benutzung der mysql-server- und mysql-client-Ports, die auf http://www.freebsd.org verfügbar sind.

Durch deren Benutzung erhalten Sie:

- Ein funktionierendes MySQL mit allen Optimierungen bereits aktiviert, von denen bekannt ist, dass Sie auf Ihrer Version von FreeBSD funktionieren.
- Automatische Konfiguration, automatisches Build.
- Start-Skripte, die in /usr/local/etc/rc.d installiert werden.
- Die Möglichkeit festzustellen, welche Dateien installiert sind, mit pkg_info -L. Und die Möglichkeit, sie mit pkg_delete zu entfernen, wenn Sie MySQL nicht mehr auf dieser Maschine haben wollen.

Empfohlen wird die Benutzung von MIT-pThreads auf FreeBSD 2.x und von nativen Threads auf Version 3 und höher. Es ist möglich, auf einigen späten 2.2.x-Versionen mit nativen Threads zu arbeiten, aber Sie können beim Herunterfahren von `mysqld` Probleme bekommen.

Die MySQL-`Makefile`-Dateien erfordern GNU-make (`gmake`). Wenn Sie MySQL kompilieren wollen, müssen Sie zuerst GNU-`make` installieren.

Stellen Sie sicher, dass Ihr Namensauflöser (Name Resolver) korrekt eingerichtet ist. Ansonsten erleben Sie vielleicht Resolver-Verzögerungen oder -Fehler, wenn Sie sich mit `mysqld` verbinden.

Stellen Sie sicher, dass der `localhost`-Eintrag in der `/etc/hosts`-Datei stimmt. Ansonsten werden Sie Probleme haben, sich mit der Datenbank zu verbinden. Die `/etc/hosts`-Datei sollte mit folgender Zeile beginnen:

```
127.0.0.1       localhost localhost.ihre.domain
```

Wenn Sie bemerken, dass `configure` MIT-pThreads benutzen wird, lesen Sie die Anmerkungen zu MIT-pThreads. See Abschnitt 3.3.6, „Anmerkungen zu MIT-pThreads“.

Wenn `make install` meldet, dass es `/usr/include/pThreads` nicht finden kann, hat `configure` nicht entdeckt, dass Sie MIT-pThreads benötigen. Das kann durch die Ausführung folgender Befehle behoben werden:

```
shell> rm config.cache
shell> ./configure --with-mit-threads
```

FreeBSD ist dafür bekannt, dass es vorgabemäßig einen sehr niedrigen Wert für Datei-Handles eingestellt hat. See Abschnitt A.2.16, „File Not Found“. Kommentieren Sie den Abschnitt ulimit -n section in safe_mysqld aus oder erhöhen Sie die Werte für den `mysqld`-Benutzer in /etc/login.conf (und bauen Sie es neu mit cap_mkdb /etc/login.conf). Stellen Sie ausserdem sicher, dass Sie die korrekte Klasse für diesen Benutzer in der Passwort-Datei einstellen, wenn Sie nicht den Vorgabewert benutzen (benutzen Sie chpass mysqld-user-name). See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.

Wenn Sie Probleme mit dem aktuellen Datum in MySQL erhalten, wird das Setzen der `TZ`-Variablen das wahrscheinlich beheben. See Anhang F, *Umgebungsvariablen*.

Um ein sicheres, stabiles System zu erhalten, sollten Sie ausschließlich FreeBSD-Kernels benutzen, die als `-STABLE` markiert

sind.

3.6.4.2. Anmerkungen zu NetBSD

Um auf NetBSD zu kompilieren, benötigen Sie GNU `make`. Ansonsten wird das Kompilieren abstürzen, wenn `make` versucht, `lint` auf C++Dateien laufen zu lassen.

3.6.4.3. Anmerkungen zu OpenBSD

3.6.4.4. Anmerkungen zu OpenBSD 2.5

Auf OpenBSD-Version 2.5 können Sie MySQL mit nativen Threads mit folgenden Optionen kompilieren:

```
CFLAGS=-pThread CXXFLAGS=-pThread ./configure --with-mit-threads=no
```

3.6.4.5. Anmerkungen zu OpenBSD 2.8

Unsere Benutzer haben berichtet, dass OpenBSD 2.8 einen Thread-Bug hat, der Probleme mit MySQL verursacht. Die OpenBSD-Entwickler haben das Problem behoben, aber seit dem 25. Januar 2001 ist es nur im ``-current''-Zweig verfügbar. Die Symptome dieses Thread-Bugs sind langsames Antworten, hohe Lase, hohe Prozessorauslastung und Abstürze.

3.6.4.6. Anmerkungen zu BSD/OS

3.6.4.7. Anmerkungen zu BSD/OS Version 2.x

Wenn Sie folgenden Fehler beim Kompilieren von MySQL erhalten, ist Ihr `ulimit`-Wert für virtuellen Speicher zu niedrig:

```
item_func.h: In method `Item_func_ge::Item_func_ge(const Item_func_ge &)':
item_func.h:28: virtual memory exhausted
make[2]: *** [item_func.o] Error 1
```

Versuchen Sie, `ulimit -v 80000` zu benutzen, und lassen Sie `make` erneut laufen. Wenn das nicht funktioniert und Sie `bash` benutzen, versuchen Sie, statt dessen `csh` oder `sh` zu benutzen. Einige BSDI-Benutzer haben Probleme mit `bash` und `ulimit` berichtet.

Wenn Sie `gcc` benutzen, müssen Sie eventuell auch den `--with-low-memory`-Flag für `configure` benutzen, um in der Lage zu sein, `sql_yacc.cc` zu kompilieren.

Wenn Sie Probleme mit dem aktuellen Datum in MySQL erhalten, wird das Setzen der `TZ`-Variablen das wahrscheinlich beheben. See Anhang F, *Umgebungsvariablen*.

3.6.4.8. Anmerkungen zu BSD/OS Version 3.x

Aktualisieren Sie auf BSD/OS Version 3.1. Wenn das nicht möglich ist, installieren Sie BSDI-Patch M300-038.

Benutzen Sie zur Konfiguration von MySQL folgenden Befehl:

```
shell> env CXX=shlicc++ CC=shlicc2 \
       ./configure \
           --prefix=/usr/local/mysql \
           --localstatedir=/var/mysql \
           --without-perl \
           --with-unix-socket-path=/var/mysql/mysql.sock
```

Folgendes funktioniert bekanntermaßen ebenfalls:

```
shell> env CC=gcc CXX=gcc CXXFLAGS=-O3 \
       ./configure \
           --prefix=/usr/local/mysql \
           --with-unix-socket-path=/var/mysql/mysql.sock
```

Wenn Sie wollen, können Sie die Verzeichnisorte ändern oder aber die Vorgabewerte benutzen, indem Sie einfach keine Speicherorte angeben.

Wenn Sie Performance-Probleme unter Hochlast bekommen, versuchen Sie die `--skip-thread-priority`-Option für `mysqld`! Dies führt alle Threads mit derselben Priorität aus. Auf BSDI-Version 3.1 gibt Ihnen das bessere Performance (zumindest solange, bis BSDI ihren Thread-Scheduler in Ordnung bringt).

Wenn Sie beim Kompilieren den Fehler `virtual memory exhausted` erhalten, probieren Sie es mit `ulimit -v 80000` und lassen Sie `make` noch einmal laufen. Wenn das nicht funktioniert und Sie `bash` benutzen, versuchen Sie, statt dessen `csh` oder `sh` zu benutzen. Einige BSDI-Benutzer haben Probleme mit `bash` und `ulimit` berichtet.

3.6.4.9. Anmerkungen zu BSD/OS Version 4.x

BSDI-Version 4.x hat einige auf Threads bezogene Bugs. Wenn Sie auf dieser Plattform MySQL benutzen wollen, sollten Sie alle Patches installieren, die sich auf Threads beziehen. Zumindest M400-023 sollte installiert sein.

Auf einigen Systemen mit BSDI-Version 4.x bekommen Sie vielleicht Probleme mit gemeinsam verwendeten (shared) Bibliotheken. Das äußert sich darin, dass Sie keinerlei Client-Programme wie `mysqladmin` ausführen können. In diesem Fall müssen Sie MySQL so rekonfigurieren, dass keine gemeinsam genutzten Bibliotheken benutzt werden, indem Sie die `--disable-shared`-Option für configure benutzen.

Einige Kunden hatten auf BSDI 4.0.1 Probleme damit, dass die `mysqld`-Binärdatei nach einiger Zeit keine Tabellen mehr öffnen konnte. Das liegt an einigen Bugs, die sich auf Bibliothek / System beziehen, und die `mysqld` veranlassen, das aktuelle Verzeichnis zu wechseln, ohne danach gefragt zu haben!

Die Lösung besteht darin, entweder auf 3.23.34 zu aktualisieren oder nach dem Laufenlassen von `configure` die Zeile `#define HAVE_REALPATH` aus `config.h` zu entfernen, bevor Sie make laufen lassen.

Beachten Sie, dass sich aus dem Gesagten ergibt, dass Sie auf BSDI keine symbolischen Links von Datenbankverzeichnissen zu einem anderen Datenbankverzeichnis oder symbolische Links von einer Tabelle zu einer anderen Datenbank herstellen können! (Ein symbolischer Link auf eine andere Platte ist okay.)

3.6.5. Anmerkungen zu Mac OS X

3.6.5.1. Mac OS X Public Beta

MySQL sollte ohne jedes Problem auf Mac OS X Public Beta (Darwin) laufen. Die pThread-Patches für dieses Betriebssystem benötigen Sie nicht!

3.6.5.2. Mac OS X Server

Bevor Sie versuchen, MySQL auf Mac OS X Server zu konfigurieren, müssen Sie das pThread-Paket von http://www.prnet.de/RegEx/mysql.html installieren.

Unsere Binärdatei für Mac OS X wird kompiliert auf Rhapsody 5.5, mit folgender Konfigurationszeile:

```
CC=gcc CFLAGS="-O2 -fomit-frame-pointer" CXX=gcc CXXFLAGS="-O2 -fomit-frame-pointer" ./configure --prefix=/usr/local/my
```

Wenn Sie der Ressourcen-Datei Ihrer Shell Aliase hinzufügen wollen, um auf `mysql` und `mysqladmin` von der Kommandozeile aus zuzugreifen, geben Sie ein:

```
alias mysql '/usr/local/mysql/bin/mysql'
alias mysqladmin '/usr/local/mysql/bin/mysqladmin'
```

3.6.6. Anmerkungen zu anderen Unixen

3.6.6.1. Anmerkungen zu HP-UX Notes für Binärdistributionen

Einige Binärdistributionen von MySQL für HP-UX werden als HP-Depot-Datei und als Tar-Datei ausgeliefert. Um die Depot-Datei benutzen zu können, müssen Sie mindestens HP-UX 10.x haben, um auf HP's Software-Depot-Werkzeuge zugreifen zu können.

Die HP-Version von MySQL wurde auf einem HP 9000/8xx-Server unter HP-UX 10.20 kompiliert und benutzt MIT-pThreads. Unter dieser Konfiguration arbeitet sie bekanntermaßen gut. MySQL-Version 3.22.26 und neuer kann auch mit HP's nativem Thread-Paket gebaut werden.

Weitere Konfigurationen, die ebenfalls funktionieren können:

- HP 9000/7xx mit HP-UX 10.20+
- HP 9000/8xx mit HP-UX 10.30

Folgende Konfigurationen werden fast mit Sicherheit nicht laufen:

- HP 9000/7xx oder 8xx mit HP-UX 10.x, wobei x < 2

- HP 9000/7xx oder 8xx mit HP-UX 9.x

Um die Distribution zu installieren, benutzen Sie die unten stehenden Befehle, wobei `/pfad/zum/depot` der volle Pfadname der Depot-Datei ist:

- Um alles inklusive Server, Client- und Entwicklungs-Werkzeuge zu installieren:

```
shell> /usr/sbin/swinstall -s /pfad/zum/depot mysql.full
```

- Um nur den Server zu installieren:

```
shell> /usr/sbin/swinstall -s /pfad/zum/depot mysql.server
```

- Um nur das Client-Paket zu installieren:

```
shell> /usr/sbin/swinstall -s /pfad/zum/depot mysql.client
```

- Um nur die Entwicklungs-Werkzeuge zu installieren:

```
shell> /usr/sbin/swinstall -s /pfad/zum/depot mysql.developer
```

Das Depot speichert Binärdateien und Bibliotheken in `/opt/mysql` und Daten in `/var/opt/mysql`. Es legt auch die entsprechenden Einträge in `/etc/init.d` und `/etc/rc2.d` an, um den Server automatisch beim Hochfahren zu starten. Das setzt `root`-Rechte zum Installieren voraus.

Um die HP-UX-tar.gz-Distribution zu installieren, müssen Sie GNU `tar` haben.

3.6.6.2. Anmerkungen zu HP-UX Version 10.20

Es gibt einige kleine Probleme, wenn Sie MySQL auf HP-UX kompilieren. Wir empfehlen, anstelle des nativen HP-UX-Kompilers `gcc` zu benutzen, weil `gcc` besseren Code produziert!

Wir empfehlen die Benutzung von gcc 2.95 auf HP-UX. Benutzen Sie keine hohen Optimierungs-Flags (wie -O6), weil das eventuell für HP-UX nicht sicher ist.

Beachten Sie, dass MIT-pThreads nicht mit dem HP-UX-Kompiler kompiliert werden können, weil dieser keine `.S`-(Assembler)-Dateien kompilieren kann.

Folgende Konfigurationszeile sollte funktionieren:

```
CFLAGS="-DHPUX -I/opt/dce/include" CXXFLAGS="-DHPUX -I/opt/dce/include -felide-constructors -fno-exceptions -fno-rtti"
```

Wenn Sie `gcc` 2.95 selbst kompilieren, sollten Sie ihn NICHT mit den DCE-Bibliotheken (`libdce.a` oder `libcma.a`) linken, wenn Sie MySQL mit MIT-pThreads kompilieren wollen. Wenn Sie DCE- und MIT-pThreads-Pakete mischen, erhalten Sie einen `mysqld`, mit dem Sie sich nicht verbinden können. Entfernen Sie die DCE-Bibliotheken, während Sie `gcc` 2.95 kompilieren!

3.6.6.3. Anmerkungen zu HP-UX Version 11.x

Für HP-UX Version 11.x empfehlen wir MySQL-Version 3.23.15 oder später.

Wegen einiger kritischer Bugs in den Standard-HP-UX-Bibliotheken sollten Sie folgende Patches installieren, bevor Sie MySQL auf HP-UX 11.0 laufen lassen:

```
PHKL_22840 Streams cumulative
PHNE_22397 ARPA cumulative
```

Das löst das Problem, dass man `EWOULDBLOCK` von `recv()` und `EBADF` von `accept()` in threaded Applikationen erhält.

Wenn Sie `gcc` 2.95.1 auf einem nicht-gepatchten HP-UX-11.x-System benutzen, erhalten Sie den Fehler:

```
In file included by /usr/include/unistd.h:11,
                 by ../include/global.h:125,
                 by mysql_priv.h:15,
                 by item.cc:19:
/usr/include/sys/unistd.h:184: declaration of C function ...
/usr/include/sys/pThread.h:440: previous declaration ...
In file included by item.h:306,
                 by mysql_priv.h:158,
```

```
                by item.cc:19:
```

Das Problem liegt darin, dass HP-UX `pThreads_atfork()` nicht konsistent definiert. Es hat konfliktbehaftete Prototypes in `/usr/include/sys/unistd.h`:184 und `/usr/include/sys/pThread.h`:440 (Details weiter unten).

Eine Lösung besteht darin, `/usr/include/sys/unistd.h` nach `mysql/include` zu kopieren und `unistd.h` zu editieren, wobei es so abgeändert wird, dass es der Definition in `pThread.h` entspricht. Hier ist der Diff:

```
183,184c183,184
<      extern int pThread_atfork(void (*prepare)(), void (*parent)(),
<                                void (*child)());
---
>      extern int pThread_atfork(void (*prepare)(void), void (*parent)(void),
>                                void (*child)(void));
```

Danach sollte folgende Konfigurationszeile funktionieren:

```
CFLAGS="-fomit-frame-pointer -O3 -fpic" CXX=gcc CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti -O3" ./configu
```

Hier sind ein paar Informationen über das Kompilieren von MySQL mit dem HP-UX:x-Kompiler, die uns ein Benutzer der HP-UX-Version 11.x geschickt hat:

```
 Environment:
      proper compilers.
         setenv CC cc
         setenv CXX aCC
      flags
         setenv CFLAGS -D_REENTRANT
         setenv CXXFLAGS -D_REENTRANT
         setenv CPPFLAGS -D_REENTRANT
     % aCC -V
     aCC: HP ANSI C++ B3910B X.03.14.06
     % cc -V /tmp/empty.c
     cpp.ansi: HP92453-01 A.11.02.00 HP C Preprocessor (ANSI)
     ccom: HP92453-01 A.11.01.00 HP C Compiler
     cc: "/tmp/empty.c", line 1: warning 501: Empty source file.

  configuration:
     ./configure  --with-pThread        \
     --prefix=/source-control/mysql     \
     --with-named-Thread-libs=-lpThread \
     --with-low-memory

    added '#define _CTYPE_INCLUDED' to include/m_ctype.h. This
    symbol ist the one defined in HP's /usr/include/ctype.h:

     /* Don't include std ctype.h when this is included */
     #define _CTYPE_H
     #define __CTYPE_INCLUDED
     #define _CTYPE_INCLUDED
     #define _CTYPE_USING   /* Don't put names in global namespace. */
```

- Ich muss den Compile-Time-Flag `-D_REENTRANT` benutzen, um den Kompiler dazu zu bringen, den Prototype für `localtime_r` zu erkennen. Alternativ hätte ich auch den Prototype für `localtime_r` bereit stellen können. Aber ich wollte weitere Bugs abfangen, in die ich sonst gerannt wäre. Ich war nicht sicher, wo ich es benötigen würde, daher fügte ich es zu allen Flags hinzu.
- Die Optimierungs-Flags, die MySQL benutzt (-O3), werden von den HP-Kompilern nicht erkannt. Ich habe die Flags nicht geändert.

Wenn Sie folgenden Fehler von `configure` erhalten:

```
checking for cc option to accept ANSI C... no
configure: error: MySQL requires a ANSI C compiler (and a C++ compiler). Try gcc. See the installation chapter in the
```

Überprüfen Sie, dass Sie den Pfad zum K&R-Kompiler nicht vor dem Pfad zum HP-UX-C- und C++-Kompiler haben.

3.6.6.4. Anmerkungen zu IBM-AIX

Automatische Erkennung von `xlC` fehlt bei Autoconf, daher wird ein `configure`-Befehl wie folgender benötigt, wenn Sie MySQL kompilieren (dieses Beispiel benutzt den IBM-Kompiler):

```
export CC="xlc_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192 "
export CXX="xlC_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192"
export CFLAGS="-I /usr/local/include"
export LDLFAGS="-L /usr/local/lib"
export CPPFLAGS=$CFLAGS
```

```
export CXXFLAGS=$CFLAGS

./configure --prefix=/usr/local \
                --localstatedir=/var/mysql \
                --sysconfdir=/etc/mysql \
                --sbindir='/usr/local/bin' \
                --libexecdir='/usr/local/bin' \
                --enable-thread-safe-client \
                --enable-large-files
```

Das sind die Optionen, die benutzt werden, um die MySQL-Distribution zu kompilieren, die sich auf http://www-frec.bull.com/ befindet.

Wenn Sie in obiger Konfigurationszeile `-O3` zu `-O2` ändern, müssen Sie auch die `-qstrict`-Option entfernen (das ist eine Beschränkung im IBM-C-Kompiler).

Wenn Sie `gcc` oder `egcs` benutzen, um MySQL zu kompilieren, **MÜSSEN** Sie den `-fno-exceptions`-Flag benutzen, weil das Exception-Handling in `gcc` / `egcs` nicht Thread-sicher ist! (Das wurde mit `egcs` 1.1. getestet.) Es gibt auch ein paar bekannte Probleme mit dem IBM-Assembler, die dazu führen können, dass schlechter Code erzeugt wird, wenn er zusammen mit gcc benutzt wird.

Wir empfehlen folgende `configure`-Zeile für `egcs` und `gcc 2.95` auf AIX:

```
CC="gcc -pipe -mcpu=power -Wa,-many" \
CXX="gcc -pipe -mcpu=power -Wa,-many" \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory
```

`-Wa,-many` ist notwendig, damit das Kompilieren gelingt. Das Problem ist IBM bekannt, hat es aber nicht eilig, es zu beheben, weil ein Workaround verfügbar ist. Wir wissen nicht, ob `-fno-exceptions` für `gcc 2.95` erforderlich ist, aber weil MySQL keine Exceptions benutzt und die obige Option schnelleren Code erzeugt, empfehlen wir, dass Sie diese Option für `egcs / gcc` immer benutzen.

Wenn Sie ein Problem mit Assembler-Code bekommen, versuchen Sie, -mcpu=xxx so anzupassen, dass es zu Ihrem Prozessor passt. Typischerweise wird man power2, power oder powerpc benutzen, alternativ kann man eventuell 604 oder 604e benutzen. Ich bin nicht ganz sicher, aber ich würde sagen, dass "power" meist sicher sein sollte, selbst auf einer power2-Maschine.

Wenn Sie nicht wissen, welchen Prozessor Sie haben, geben Sie "uname -m" ein. Das gibt eine Zeichenkette zurück, die etwa wie "000514676700" aussieht, mit dem Format xxyyyyyymmss, wobei xx und ss immer die Nullen sind (0). yyyyyy ist eine eindeutige System-ID und mm ist die ID des CPU-Planars. Eine Tabelle dieser Werte liegt auf http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/cmds/aixcmds5/uname.htm. Darin finden Sie Maschinentyp und Maschinenmodell, was Sie benutzen können, um herauszufinden, welchen Prozessortyp Sie haben.

Wenn Sie Probleme mit Signalen haben (MySQL stirbt unerwartet unter hoher Last), haben Sie vielleicht einen Betriebssystem-Bug bei Threads und Signalen gefunden. In diesem Fall können Sie MySQL anweisen, keine Signale zu benutzen, indem Sie es wie folgt konfigurieren:

```
shell> CFLAGS=-DDONT_USE_THR_ALARM CXX=gcc \
       CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti -DDONT_USE_THR_ALARM" \
       ./configure --prefix=/usr/local/mysql --with-debug --with-low-memory
```

Das berührt nicht die Performance von MySQL, hat aber den Nebeneffekt, dass Sie keine Clients auf einer Verbindung mit `mysqladmin kill` oder `mysqladmin shutdown` killen können, die ``schlafen". Statt dessen wird der Client sterben, wenn er den nächsten Befehl sendet.

Bei einigen Versionen von AIX für das Linken mit `libbind.a` bei `getservbyname` zu einem Speicherauszug (Core Dump). Das ist ein AIX-Bug, der IBM berichtet werden sollte.

Bei AIX 4.2.1 und gcc müssen Sie folgende Änderungen durchführen:

Nach dem Konfigurieren müssen Sie `config.h` und `include/my_config.h` editieren und die Zeile ändern, in der steht:

```
#define HAVE_SNPRINTF 1
```

zu

```
#undef HAVE_SNPRINTF
```

Schließlich müssen Sie in `mysqld.cc` einen Prototype für initgoups hinzufügen:

```
#ifdef _AIX41
extern "C" int initgroups(const char *,int);
#endif
```

3.6.6.5. Anmerkungen zu SunOS 4

Auf SunOS 4 werden MIT-pThreads benötigt, um MySQL zu kompilieren, was letztlich bedeutet, dass Sie GNU-`make` benötigen.

Einige SunOS-4-Systeme haben Probleme mit dynamischen Bibliotheken und `libtool`. Sie können folgende `configure`-Zeile benutzen, um das Problem zu vermeiden:

```
shell> ./configure --disable-shared --with-mysqld-ldflags=-all-static
```

Wenn Sie `readline` kompilieren, erhalten Sie vielleicht Warnungen über duplizierte Defines. Diese können ignoriert werden.

Wenn Sie `mysqld` kompilieren, gibt es ein paar `implicit declaration of function`-Warnungen. Diese können ignoriert werden.

3.6.6.6. Anmerkungen zu Alpha-DEC-UNIX (Tru64)

Wenn Sie egcs 1.1.2 auf Digital Unix benutzen, sollten Sie auf gcc 2.95.2 aktualisieren, weil egcs auf DEC einige schwer wiegende Bugs hat!

Wenn Sie threaded Programme unter Digital Unix kompilieren, empfiehlt die Dokumentation, die `-pThread`-Option für `cc` und `cxx` und die Bibliotheken `-lmach -lexc` zu benutzen (zusätzlich zu `-lpThread`). Sie sollten `configure` wie folgt laufen lassen:

```
CC="cc -pThread" CXX="cxx -pThread -O" \
./configure --with-named-thread-libs="-lpThread -lmach -lexc -lc"
```

Wenn Sie `mysqld` kompilieren, sehen Sie eventuell eine Reihe von Warnungen wie die folgende:

```
mysqld.cc: In function void handle_connections()':
mysqld.cc:626: passing long unsigned int *' as argument 3 of
accept(int,sockadddr *, int *)'
```

Sie können diese Warnungen ignorieren. Sie treten auf, weil `configure` nur Fehler entdecken kann, keine Warnungen.

Wenn Sie den Server direkt von the Kommandozeile starten, haben Sie vielleicht Probleme, dass er stirbt, wenn Sie sich ausloggen. (Wenn Sie sich ausloggen, erhalten Ihre offenen Prozesse ein `SIGHUP`-Signal.) Wenn das der Fall ist, starten Sie den Server wie folgt:

```
shell> nohup mysqld [options] &
```

`nohup` bewirkt, dass der folgende Befehl jegliche `SIGHUP`-Signale, die vom Terminal gesendet werden, ignoriert. Alternativ starten Sie den Server mit `safe_mysqld`, was `mysqld` mit `nohup` für Sie aufruft. See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.

Wenn Sie ein Problem beim Kompilieren von mysys/get_opt.c bekommen, entfernen Sie einfach die Zeile #define _NO_PROTO am Anfang dieser Datei!

Wenn Sie den CC-Kompiler von Compaq benutzen, sollte die folgende Konfigurationszeile funktionieren:

```
CC="cc -pThread"
CFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host"
CXX="cxx -pThread"
CXXFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host"
export CC CFLAGS CXX CXXFLAGS
./configure \
--prefix=/usr/local/mysql \
--with-low-memory \
--enable-large-files \
--enable-shared=yes \
--with-named-Thread-libs="-lpThread -lmach -lexc -lc"
gnumake
```

Wenn Sie ein Problem mit libtool beim Kompilieren mit gemeinsam genutzten (shared) Bibliotheken bekommen wie oben, wenn Sie `mysql` linken, sollten Sie dies folgendermaßen umgehen können:

```
cd mysql
/bin/sh ../libtool --mode=link cxx -pThread  -O3 -DDBUG_OFF \
-O4 -ansi_alias -ansi_args -fast -inline speed \
-speculate all \ -arch host  -DUNDEF_HAVE_GETHOSTBYNAME_R \
-o mysql  mysql.o readline.o sql_string.o completion_hash.o \
../readline/libreadline.a -lcurses \
../libmysql/.libs/libmysqlclient.so  -lm
```

```
cd ..
gnumake
gnumake install
Skripts/mysql_install_db
```

3.6.6.7. Anmerkungen zu Alpha-DEC-OSF1

Wenn Sie Probleme beim Kompilieren haben und DEC `CC` und `gcc` installiert sind, versuchen Sie, `configure` wie folgt laufen zu lassen:

```
CC=cc CFLAGS=-O CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Wenn Sie Probleme mit der `c_asm.h`-Datei bekommen, können Sie wie folgt eine 'dummy'-`c_asm.h`-Datei erzeugen und benutzen:

```
touch include/c_asm.h
CC=gcc CFLAGS=-I./include \
CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Beachten Sie, dass die im Folgenden beschriebenen Probleme mit dem `ld`-Programm behoben werden können, indem Sie das neueste DEC-(Compaq)-Patch-Kit herunterladen, und zwar von folgender Seite: http://ftp.Support.compaq.com/public/unix/.

Auf OSF1 V4.0D und Kompiler "DEC C V5.6-071 auf Digital Unix V4.0 (Rev. 878)" zeigt der Kompiler einige seltsame Verhaltensweisen (undefinierte `asm`-Symbole). Ausserdem scheint `/bin/ld` beschädigt zu sein (Probleme mit `_exit undefined`-Fehlern, die auftreten, wenn Sie `mysqld` linken). Auf diesem System konnten wir MySQL mit folgender `configure`-Zeile kompilieren, nachdem wir `/bin/ld` mit der Version von OSF 4.0C ersetzt haben:

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
```

Beim Digital-Kompiler "C++ V6.1-029" sollte folgendes funktionieren:

```
CC=cc -pThread
CFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all -arch host
CXX=cxx -pThread
CXXFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all -arch host -noexceptions -nortti
export CC CFLAGS CXX CXXFLAGS
./configure --prefix=/usr/mysql/mysql --with-mysqld-ldflags=-all-static --disable-shared --with-named-thread-libs="-lma
```

In einigen Versionen von OSF1 ist die `alloca()`-Funktion beschädigt. Beheben Sie dies, indem Sie die Zeile in `config.h` entfernen, die `'HAVE_ALLOCA'` definiert.

Die `alloca()`-Funktion kann ebenfalls einen falschen Prototyp in `/usr/include/alloca.h` haben. Die Warnung, die hieraus resultiert, kann ignoriert werden.

`configure` benutzt automatisch folgenden Thread-Bibliotheken: `--with-named-thread-libs="-lpThread -lmach -lexc -lc"`.

Wenn Sie `gcc` benutzen, können Sie auch versuchen, `configure` wie folgt laufen zu lassen:

```
shell> CFLAGS=-D_PTHREAD_USE_D4 CXX=gcc CXXFLAGS=-O3 ./configure ....
```

Wenn Sie Probleme mit Signalen haben (MySQL stirbt unerwartet unter Hochlast), haben Sie vielleicht einen Betriebssystem-Bug bei Threads und Signalen gefunden. In diesem Fall können Sie MySQL anweisen, keine Signale zu benutzen, indem Sie es wie folgt konfigurieren:

```
shell> CFLAGS=-DDONT_USE_THR_ALARM \
       CXXFLAGS=-DDONT_USE_THR_ALARM \
       ./configure ...
```

Das berührt nicht die Performance von MySQL, hat aber den Nebeneffekt, dass Sie keine Clients auf einer Verbindung mit `mysqladmin kill` oder `mysqladmin shutdown` killen können, die ``schlafen''. Statt dessen wird der Client sterben, wenn er den nächsten Befehl sendet.

Bei `gcc` 2.95.2 erhalten Sie wahrscheinlich folgenden Kompilierfehler:

```
sql_acl.cc:1456: Internal compiler error in `scan_region', at except.c:2566
Please submit a full bug report.
```

Um das zu beheben, wechseln Sie ins `sql`-Verzeichnis und machen ein ``Kopieren und Einfügen'' der letzten `gcc`-Zeile, ändern

aber `-O3` zu `-O0` (oder fügen `-O0` unmittelbar nach `gcc` hinzu, falls Sie keine `-O`-Option auf Ihrer Kompilierzeile haben.) Danach wechseln Sie einfach direkt zurück in oberste Verzeichnis und lassen `make` noch einmal laufen.

3.6.6.8. Anmerkungen zu SGI Irix

Wenn Sie Irix-Version 6.5.3 oder neuer benutzen, kann `mysqld` nur dann Threads erzeugen, wenn Sie ihn als Benutzer mit `CAP_SCHED_MGT`-Zugriffsrechten (wie `root`) laufen lassen oder dem `mysqld`-Server dieses Recht mit dem folgenden Befehl geben:

```
shell> chcap "CAP_SCHED_MGT+epi" /opt/mysql/libexec/mysqld
```

Sie müssen eventuell in `config.h` einige Dinge umdefinieren, nachdem Sie `configure` laufen gelassen haben und vor dem Kompilieren.

In einigen Irix-Implementationen ist die `alloca()`-Funktion beschädigt. Wenn der `mysqld`-Server bei manchen `SELECT`-Statements stirbt, entfernen Sie die Zeilen aus `config.h`, die `HAVE_ALLOC` und `HAVE_ALLOCA_H` definieren. Wenn `mysqladmin create` nicht funktioniert, entfernen Sie die Zeile aus `config.h`, die `HAVE_READDIR_R` definiert. Eventuell müssen Sie auch die `HAVE_TERM_H`-Zeile entfernen.

SGI empfiehlt, dass Sie alle Patches auf dieser Seite auf einmal installieren:
http://Support.sgi.com/surfzone/patches/patchset/6.2_indigo.rps.html

Als absolutes Minimum sollten Sie das letzte Kernel-Rollup installieren, das letzte `rld`-Rollup und das letzte `libc`-Rollup.

In jedem Fall brauchen Sie für die pThread-Unterstützung alle POSIX-Patches auf dieser Seite:

http://Support.sgi.com/surfzone/patches/patchset/6.2_posix.rps.html

Wenn Sie beim Kompilieren von `mysql.cc` etwa folgenden Fehler erhalten:

```
"/usr/include/curses.h", line 82: error(1084): invalid combination of type
```

Geben Sie folgendes im obersten Verzeichnis Ihres MySQL-Source-Trees ein:

```
shell> extra/replace bool curses_bool < /usr/include/curses.h > include/curses.h
shell> make
```

Es wurden ausserdem Scheduling-Probleme berichtet. Wenn nur ein Thread läuft, läuft alles recht langsam. Das können Sie vermeiden, indem Sie einen weiteren Client-Starten. Daraus kann sich eine zwei- bis zehnfache Geschwindigkeitssteigerung für den anderen Thread ergeben. Das liegt an einem Problem mit Irix-Threads, das kaum zu verstehen ist. Eventuell müssen Sie improvisieren, um eine Lösung zu finden, bis dies behoben ist.

Wenn Sie mit `gcc` kompilieren, können Sie folgenden `configure`-Befehl benutzen:

```
CC=gcc CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql --enable-thread-safe-client --with-named-thread-libs=-lpThread
```

Auf Irix 6.5.11 mit nativen Irix-C- und C++-Kompilern der Version 7.3.1.2 soll auch folgendes funktionieren:

```
CC=cc CXX=CC CFLAGS='-O3 -n32 -TARG:platform=IP22 -I/usr/local/include \
-L/usr/local/lib' CXXFLAGS='-O3 -n32 -TARG:platform=IP22 \
-I/usr/local/include -L/usr/local/lib' ./configure --prefix=/usr/local/mysql \
--with-berkeley-db --with-innodb \
--with-libwrap=/usr/local --with-named-curses-libs=/usr/local/lib/libncurses.a
```

3.6.6.9. Anmerkungen zu Caldera

Die aktuelle Portierung wird auf ``sco3.2v5.0.4"- und-``sco3.2v5.0.5"-Systemen getestet. Die Portierung auf ``sco 3.2v4.2" ist ebenfalls weit fortgeschritten.

Momentan ist der empfohlene Kompiler auf OpenServer gcc 2.95.2. Damit sollten Sie in der Lage sein, MySQL einfach durch folgendes zu kompilieren:

```
CC=gcc CXX=gcc ./configure ... (options)
```

1. Bei OpenServer 5.0.X müssen Sie GDS in Skunkware 95 (95q4c) benutzen. Das ist deshalb notwendig, weil GNU-`gcc` 2.7.2 in Skunkware 97 kein GNU-`as` hat. Sie können auch `egcs` 1.1.2 oder neuer benutzen http://www.egcs.com/. Wenn Sie `egcs` 1.1.2 benutzen, müssen Sie folgenden Befehl eingeben:

   ```
   shell> cp -p /usr/include/pThread/stdtypes.h /usr/local/lib/gcc-lib/i386-pc-sco3.2v5.0.5/egcs-2.91.66/include/pThr
   ```

2. Sie brauchen die Portierung von GCC 2.5.x für dieses Produkt und das Entwicklungssystem. Sie werden auf dieser Version von Caldera (SCO) Unix benötigt. Sie können nicht lediglich das GCC-Dev-System benutzen.

3. Sie sollten zuerst das FSU-PThreads-Paket holen und installieren. Dieses finden Sie auf http://www.cs.wustl.edu/~schmidt/ACE_wrappers/FSU-threads.tar.gz. Sie finden ein vorkompiliertes Paket auf http://www.mysql.com/Downloads/SCO/FSU-threads-3.5c.tar.gz.

4. FSU-PThreads kann mit SCO Unix 4.2 mit TCP/IP kompiliert werden. Oder mit OpenServer 3.0 oder Open Desktop 3.0 (OS 3.0 ODT 3.0), mit installiertem Caldera (SCO) Entwicklungssystem unter Benutzung einer guten Portierung von GCC 2.5.x ODT oder OS 3.0. Hierbei brauchen Sie eine gute Portierung von GCC 2.5.x. Ohne gute Portierung gibt es eine Menge Probleme. Die Portierung für dieses Produkt erfordert das Caldera (SCO) Unix-Entwicklungssystem. Ohne dieses fehlen die Bibliotheken und der Linker, die benötigt werden.

5. Um FSU-PThreads auf Ihrem System zu bauen, tun Sie folgendes:

   a. Lassen Sie `./configure` im `Threads/src`-Verzeichnis laufen und wählen Sie die SCO-OpenServer-Option. Dieser Befehl kopiert `Makefile.SCO5` nach `Makefile`.

   b. Lassen Sie `make` laufen.

   c. Um in das vorgabemäßige `/usr/include`-Verzeichnis zu installieren, loggen Sie sich als Root ein und wechseln (`cd`) Sie in das `thread/src`-Verzeichnis. Führen Sie dann `make install` aus.

6. Denken Sie daran, GNU `make` zu benutzen, wenn Sie MySQL machen.

7. Wenn Sie `safe_mysqld` nicht als Root starten, erhalten Sie wahrscheinlich nur die 110 offenen Dateien pro Prozess. `mysqld` macht darüber in der Log-Datei einen Eintrag.

8. Bei SCO 3.2V5.0.5 sollten Sie FSU-PThreads-Version 3.5c oder neuer benutzen. Ausserdem sollten Sie gcc 2.95.2 oder neuer benutzen!

   Folgender `configure`-Befehl sollte funktionieren:

   ```
   shell> ./configure --prefix=/usr/local/mysql --disable-shared
   ```

9. Bei SCO 3.2V4.2 sollten Sie FSU-PThreads-Version 3.5c oder neuer benutzen. Folgender `configure`-Befehl sollte funktionieren:

   ```
   shell> CFLAGS="-D_XOPEN_XPG4" CXX=gcc CXXFLAGS="-D_XOPEN_XPG4" \
          ./configure \
              --prefix=/usr/local/mysql \
              --with-named-thread-libs="-lgThreads -lsocket -lgen -lgThreads" \
              --with-named-curses-libs="-lcurses"
   ```

   Möglicherweise bekommen Sie Probleme mit einigen Include-Dateien. In diesem Fall finden Sie neue, SCO-spezifische Include-Dateien auf http://www.mysql.com/Downloads/SCO/SCO-3.2v4.2-includes.tar.gz. Entpacken Sie diese Datei ins `include`-Verzeichnis Ihres MySQL-Source-Trees.

Anmerkungen zur Caldera (SCO) Entwicklung:

- MySQL kann FSU-PThreads automatisch erkennen und `mysqld` mit `-lgThreads -lsocket -lgThreads` linken.
- Die Caldera (SCO) Entwicklungsbibliotheken sind re-entrant in FSU-PThreads. Caldera behauptet, dass seine Bibliotheken-Funktionen re-entrant sind, daher müssen sie mit FSU-PThreads re-entrant sein. FSU-PThreads auf OpenServer versucht, das SCO-Scheme zu benutzen, um Bibliotheken re-entrant zu machen.
- FSU-PThreads (zumindest die Version auf http://www.mysql.com/) wird mit gelinktem GNU-`malloc` ausgeliefert. Wenn Sie Problemen mit der Speicherbenutzung begegnen, stellen Sie sicher, dass `gmalloc.o` in `libgThreads.a` und `libgThreads.so` beinhaltet ist.
- In FSU-PThreads achten folgende Systemaufrufe auf pThreads: `read()`, `write()`, `getmsg()`, `connect()`, `accept()`, `select()` und `wait()`.

Wenn Sie DBI auf Caldera (SCO) installieren wollen, müssen Sie `Makefile` in DBI-xxx und jedem Unterverzeichnis editieren.

Beachten Sie, dass folgendes gcc 2.95.2 oder neuer voraussetzt:

```
ALT:                                    NEU:
CC = cc                                 CC = gcc
CCCDLFLAGS = -KPIC -W1,-Bexport         CCCDLFLAGS = -fpic
CCDLFLAGS = -wl,-Bexport                CCDLFLAGS =

LD = ld                                 LD = gcc -G -fpic
LDDLFLAGS = -G -L/usr/local/lib         LDDLFLAGS = -L/usr/local/lib
LDFLAGS = -belf -L/usr/local/lib        LDFLAGS = -L/usr/local/lib

LD = ld                                 LD = gcc -G -fpic
OPTIMISE = -Od                          OPTIMISE = -O1

OLD:
CCCFLAGS = -belf -dy -w0 -U M_XENIX -DPERL_SCO5 -I/usr/local/include

NEW:
CCFLAGS = -U M_XENIX -DPERL_SCO5 -I/usr/local/include
```

Das liegt daran, dass der Perl-dynaloader keine `DBI`-Module lädt, die mit `icc` oder `cc` kompiliert wurden.

Perl funktioniert am besten, wenn es mit `cc` kompiliert wird.

3.6.6.10. Anmerkungen zu Caldera Unixware Version 7.0

Sie benötigen mindestens MySQL-Version 3.22.13, weil diese Version einige Portabilitätsprobleme unter Unixware behebt.

Wir waren in der Lage, MySQL mit folgendem `configure`-Befehl auf Unixware Version 7.0.1 zu kompilieren:

```
CC=cc CXX=CC ./configure --prefix=/usr/local/mysql
```

Wenn Sie `gcc` benutzen wollen, müssen Sie `gcc` 2.95.2 oder neuer benutzen.

3.6.7. Anmerkungen zu OS/2

MySQL benutzt eine ganze Menge offener Dateien. Deswegen sollten Sie Ihrer `CONFIG.SYS`-Datei folgendes hinzufügen:

```
SET EMXOPT=-c -n -h1024
```

Wenn Sie das nicht tun, erhalten Sie wahrscheinlich folgenden Fehler:

```
File 'xxxx' not found (Errcode: 24)
```

Wenn Sie MySQL auf OS/2 Warp 3 einsetzen, wird FixPack 29 oder höher benötigt. Bei OS/2 Warp 4 wird FixPack 4 oder höher benötigt. Das erfordert die PThreads-Bibliothek. MySQL muss auf einer Partition installiert werden, die lange Dateinamen unterstützt, also zum Beispiel HPFS, FAT32 usw.

Das `INSTALL.CMD`-Skript muss von OS/2's eigener `CMD.EXE` aus laufen gelassen werden und funktioniert eventuell nicht mit Ersatz-Shells wie `4OS2.EXE`.

Das `scripts/mysql-install-db`-Skript wurde umbenannt. Es heißt jetzt `install.cmd` und ist ein REXX-Skript, welches die vorgabemäßigen MySQL-Sicherheitseinstellungen einstellt und die WorkPlace-Shell-Icons für MySQL erstellt.

Unterstützung für dynamische Module wird einkompiliert, ist aber noch nicht komplett durchgetestet. Dynamische Module sollten unter Benutzung der PThreads-Runtime-Bibliothek kompiliert werden.

```
gcc -Zdll -Zmt -Zcrtdll=pthrdrtl -I../include -I../regex -I.. \
    -o example udf_example.cc -L../lib -lmysqlclient udf_example.def
mv example.dll example.udf
```

Beachten Sie: Aufgrund von Beschränkungen in OS/2 dürfen UDF-module-name-stems nicht länger als 8 Zeichen sein. Module werden im `/mysql2/udf`-Verzeichnis gespeichert; das `safe-mysqld.cmd`-Skript trägt dieses Verzeichnis in die `BEGINLIBPATH`-Umgebungsvariable ein. Wenn Sie UDF-Module benutzen, werden festgelegte Erweiterungen ignoriert - es wird nicht angenommen, dass sie `.udf` sind. Unter Unix zum Beispiel könnte das gemeinsam genutzte (shared) Module `example.so` benannt sein. Sie würden daraus eine Funktion wie folgt laden:

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "example.so";
```

Unter OS/2 würde das Modul `example.udf` heißen, aber Sie würden nicht die Modul-Erweiterung angeben:

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "example";
```

3.6.8. Anmerkungen zu BeOS

Wir sind sehr daran interessiert, MySQL auf BeOS ans Laufen zu bringen, aber leider kennen wir niemanden, der sich mit BeOS auskennt oder Zeit hat, eine Portierung durchzuführen.

Wir sind daran interessiert, jemanden für eine Portierung zu finden, und wir werden ihn / sie bei allen technischen Fragen helfen, die bei einer Portierung auftreten können.

Wir haben vor einiger Zeit mit einigen BeOS-Entwicklern gesprochen, die uns sagten, dass MySQL zu 80% auf BeOS portiert ist, aber wir haben schon eine Weile nichts von ihnen gehört.

3.6.9. Anmerkungen zu Novell NetWare

Wir sind sehr daran interessiert, MySQL auf NetWare ans Laufen zu bringen, aber leider kennen wir niemanden, der sich mit NetWare auskennt oder Zeit hat, eine Portierung durchzuführen.

Wir sind daran interessiert, jemanden für eine Portierung zu finden, und wir werden ihn / sie bei allen technischen Fragen helfen, die bei einer Portierung auftreten können.

3.7. Anmerkungen zur Perl-Installation

`DBI`/`DBD`-Schnittstelle

3.7.1. Installation von Perl unter Unix

Perl-Unterstützung für MySQL wird durch die `DBI`/`DBD`- Client-Schnittstelle zur Verfügung gestellt. See Abschnitt 9.2, „MySQL-Perl-API“. Der Perl- `DBD`/`DBI`-Client-Code erfordert Perl Version 5.004 oder später. Die Schnittstelle **funktioniert nicht**, wenn Sie eine ältere Version von Perl haben.

MySQL-Perl-Unterstützung erfordert ausserdem, dass Sie MySQL-Client- Programmierunterstützung installiert haben. Wenn Sie MySQL von RPM- Dateien installiert haben, sind Client-Programme im Client-RPM enthalten, aber Client-Programmierunterstützung ist im Entwickler-RPM. Stellen Sie sicher, dass Sie auch das letztgenannte RPM installiert haben.

Ab Version 3.22.8 wird Perl-Unterstützung getrennt von der Haupt-MySQL- Unterstützung ausgeliefert. Wenn Sie Perl-Unterstützung installieren wollen, können Sie die benötigten Dateien von http://www.mysql.com/downloads/api-dbi.html herunter laden.

Die Perl-Distributionen werden als komprimierte `tar`-Archive zur Verfügung gestellt und haben Namen wie `MODULE-VERSION.tar.gz`, wobei `MODULE` der Modulname und `VERSION` die Versionsnummer ist. Sie sollten die `Data-Dumper`-, `DBI`- und `Msql- Mysql-modules`-Distributionen laden und sie in dieser Reihenfolge installieren. Die Installationsprozedur ist unten dargestellt. Das Beispiel gilt für das `Data-Dumper`-Modul, ist aber für alle drei Distributionen dieselbe:

1. Entpacken Sie die Distribution ins aktuelle Verzeichnis.

   ```
   shell> gunzip < Data-Dumper-VERSION.tar.gz | tar xvf -
   ```

   Dieser Befehl erzeugt ein Verzeichnis namens `Data-Dumper- VERSION`.

2. Wechseln Sie ins oberste Verzeichnis der entpackten Distribution:

   ```
   shell> cd Data-Dumper-VERSION
   ```

3. Bauen Sie die Distribution und kompilieren Sie alles:

   ```
   shell> perl Makefile.PL
   shell> make
   shell> make test
   shell> make install
   ```

Der `make test`-Befehl ist wichtig, weil er sicherstellt, dass die Module funktionieren. Beachten Sie, dass der MySQL-Server während der Befehlsausführung bei der `Msql-Mysql-modules`-Installation laufen muss, um den Schnittstellen-Code auszuführen, den ansonsten schlägt der Test fehl.

Es ist eine gute Idee, die `Msql-Mysql-modules`-Distribution neu zu kompilieren und zu installieren, wenn Sie ein neues Release von MySQL installieren, insbesondere, wenn Sie Symptome feststellen wie dass alle Ihre `DBI`-Skripte einen Coredump liefern, nachdem Sie auf eine höhere Version von MySQL aktualisiert haben.

Wenn Sie keine Rechte haben, die Perl-Module im Systemverzeichnis zu installieren, oder wenn Sie lokale Perl-Module

installieren wollen, könnte Ihnen der folgende Link helfen:

```
http://www.iserver.com/support/contrib/perl5/modules.html
```

Suchen Sie nach der Überschrift `Installing New Modules that Require Locally Installed Modules`.

3.7.2. Installation von ActiveState-Perl unter Windows

Um das MySQL-`DBD`-Modul mit ActiveState-Perl unter Windows zu installieren, gehen Sie wie folgt vor:

- Laden Sie ActiveState-Perl von http://www.activestate.com/Products/ActivePerl/ und installieren Sie es.
- Öffnen Sie eine MS-DOS-Eingabeaufforderung.
- Setzen Sie - falls erforderlich - die HTTP_proxy-Variable, zum Beispiel wie folgt:

  ```
  set HTTP_proxy=my.proxy.com:3128
  ```

- Starten Sie das PPM-Programm:

  ```
  C:\> c:\perl\bin\ppm.pl
  ```

- Falls noch nicht geschehen, installieren Sie `DBI`:

  ```
  ppm> install DBI
  ```

- Wenn das erfolgreich verlief, führen Sie folgenden Befehl aus:

  ```
  install
  ftp://ftp.de.uu.net/pub/CPAN/authors/id/JWIED/DBD-mysql-1.2212.x86.ppd
  ```

Das sollte zumindest bei ActiveState-Perl Version 5.6 funktionieren.

Wenn Sie es nicht schaffen, dass oben Genanntes funktioniert, sollten Sie statt dessen den **MyODBC**-Treiber installieren und sich mit dem MySQL-Server über ODBC verbinden:

```
use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn","$user","$password") ||
  die "Fehler $DBI::errstr beim Verbinden mit $dsn\n";
```

3.7.3. Installation der MySQL-Perl-Distribution unter Windows

Die MySQL-Perl-Distribution enthält `DBI`, `DBD:MySQL` und `DBD:ODBC`.

- Laden Sie die Perl-Distribution für Windows von http://www.mysql.com/download.html.
- Entpacken Sie die Distribution in `C:`, so dass Sie ein `C:\PERL`-Verzeichnis erhalten.
- Fügen Sie Ihrem Pfad `C:\PERL\BIN` hinzu.
- Fügen Sie Ihrem Pfad das Verzeichnis `C:\PERL\BIN\MSWIN32-x86- thread` oder `C:\PERL\BIN\MSWIN32-x86` hinzu.
- Testen Sie, ob `perl` funktioniert, indem Sie `perl -v` in einer MS-DOS-Eingabeaufforderung ausführen.

3.7.4. Probleme bei der Benutzung von Perl `DBI`/`DBD`-Schnittstelle

Wenn Perl ausgibt, dass es das `../mysql/mysql.so`-Modul nicht finden kann, liegt das Problem wahrscheinlich darin, dass Perl die gemeinsam genutzte `libmysqlclient.so` nicht findet.

Das können Sie mit einer der folgenden Methoden beheben:

- Kompilieren Sie die `Msql-Mysql-modules`-Distribution mit `perl Makefile.PL -static -config` statt mit

`perl Makefile.PL`.

- Kopieren Sie `libmysqlclient.so` in das Verzeichnis, in dem Ihre anderen gemeinsam genutzten Bibliotheken liegen (wahrscheinlich `/usr/lib` oder `/lib`).
- Unter Linux können Sie der `/etc/ld.so.conf`-Datei den Pfadnamen des Verzeichnisses hinzufügen, in dem `libmysqlclient.so` liegt.
- Fügen Sie der `LD_RUN_PATH`-Umgebungsvariablen den Pfadnamen des Verzeichnisses hinzu, in dem `libmysqlclient.so` liegt.

Wenn Sie folgende Fehler von `DBD-mysql` erhalten, benutzen Sie wahrscheinlich `gcc` (oder eine alte Binärdatei, die mit `gcc` kompiliert wurde):

```
/usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

Fügen Sie `-L/usr/lib/gcc-lib/... -lgcc` zum Link-Befehl hinzu, wenn die `mysql.so`-Bibliothek gebaut wird (überprüfen Sie die Ausgabe von `make` nach `mysql.so`, wenn Sie den Perl-Client kompilieren). Die `-L`-Option sollte den Pfadnamen des Verzeichnisses angeben, in dem `libgcc.a` auf Ihrem System liegt.

Ein weiterer Grund für dieses Problem kann sein, dass Perl und MySQL nicht beide mit `gcc` kompiliert wurden. In diesem Fall können Sie die fehlende Übereinstimmung (Mismatch) durch Kompilieren von beiden mit `gcc` aufheben.

Wenn Sie folgende Fehler von `Msql-Mysql-modules` erhalten, wenn Sie die Tests laufen lassen:

```
t/00base............install_driver(mysql) failed: Can't load
'../blib/arch/auto/DBD/mysql/mysql.so' for module DBD::mysql:
../blib/arch/auto/DBD/mysql/mysql.so: undefined symbol: uncompress at
/usr/lib/perl5/5.00503/i586-linux/DynaLoader.pm line 169.
```

Bedeutet das, dass Sie die Kompressionsbibliothek (-lz) in die Link- Zeile einschließen müssen. Das kann man durch folgende Änderung in der Datei `lib/DBD/mysql/Install.pm` tun:

```
$sysliblist .= " -lm";

ändern in

$sysliblist .= " -lm -lz";
```

Danach **müssen** Sie 'make realclean' laufen lassen und danach mit der Installation von Anfang an beginnen.

Wenn Sie das Perl-Modul auf einem System laufen lassen wollen, das dynamisches Linken nicht unterstützt (wie Caldera/SCO), können Sie eine statische Version von Perl erzeugen, die `DBI` und `DBD-mysql` enthält. Das bringt man zum Laufen, indem man eine Version von Perl erzeugt, in der der `DBI`-Code eingelinkt ist, und diese über das aktuelle Perls installiert. Dann benutzen Sie diese, um eine Version von Perl zu bauen, die zusätzlich den `DBD`-Code eingelinkt hat, und installieren diese.

Unter Caldera (SCO) müssen folgende Umgebungsvariablen gesetzt sein:

```
shell> LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/usr/progressive/lib
or
shell>
LD_LIBRARY_PATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:/usr/progressive/lib:/usr/skunk/lib
shell>
LIBPATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:/usr/progressive/lib:/usr/skunk/lib
shell>
MANPATH=scohelp:/usr/man:/usr/local1/man:/usr/local/man:/usr/skunk/man:
```

Erzeugen Sie zuerst ein Perl, das ein statisch gelinktes `DBI` enthält, indem Sie diese Befehle im Verzeichnis ausführen, in dem Ihre `DBI`-Distribution liegt:

```
shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Dann müssen Sie das neue Perl installieren. Die Ausgabe von `make perl` zeigt den genauen `make`-Befehl an, den Sie für die Installation ausführen müssen. Unter Caldera (SCO) ist das `make -f Makefile.aperl inst_perl MAP_TARGET=perl`.

Benutzen Sie als nächstes dieses soeben erzeugte Perl, um ein weiteres Perl zu erzeugen, dass auch ein statisch gelinktes `DBD::mysql` enthält, indem Sie diese Befehle im Verzeichnis ausführen, in dem Ihre `Msql-Mysql-modules`-Distribution liegt:

```
shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Zum Schluss müssen Sie dieses neue Perl installieren. Hierbei zeigt die Ausgabe von `make perl` wiederum, welcher Befehl benutzt werden muss.

Kapitel 4. Einführung in MySQL: Ein MySQL-Tutorial

Dieses Kapitel enthält eine Einführung in MySQL in Form eines Tutorials. Datei wird gezeigt, wie Sie das `mysql`-Client-Programm benutzen, um eine einfache Datenbank zu erzeugen und zu benutzen. `mysql` (auch ``Terminal-Monitor'' oder einfach ``Monitor'' genannt) ist ein interaktives Programm, mit dem Sie sich mit einem MySQL-Server verbinden, Anfragen (Queries) absetzen und die Ergebnisse ansehen können. `mysql` kann auch im Stapelbetrieb (Batch Mode) benutzt werden: Sie schreiben Ihre Anfragen zuerst in eine Datei und veranlassen dann `mysql`, die Inhalte dieser Datei auszuführen. Hier werden beide Möglichkeiten beschrieben, `mysql` zu benutzen.

Sie können `mysql` mit der `--help`-Option aufrufen, um eine Liste der Optionen zu sehen:

```
shell> mysql --help
```

Dieses Kapitel setzt voraus, dass `mysql` auf Ihrer Maschine installiert ist und dass ein MySQL-Server verfügbar ist, mit dem Sie sich verbinden können. Wenn das nicht der Fall sein sollte, setzen Sie sich mit Ihrem MySQL-Administrator in Verbindung. (Wenn *Sie* der Administrator sind, müssen Sie in anderen Abschnitten des Handbuchs nachsehen.)

Dieses Kapitel beschreibt den gesamten Prozess der Einrichtung und Benutzung einer Datenbank. Wenn Sie lediglich wissen wollen, wie man auf eine bereits existierende Datenbank zugreift, können Sie die Abschnitte überspringen, die beschreiben, wie man eine Datenbank und die Tabellen, die sie enthält, erzeugt.

Weil dieses Kapitel ein Tutorial ist, wurden notwendigerweise viele Details ausgelassen. Sehen Sie in den relevanten Abschnitten dieses Handbuchs nach, wenn Sie weitere Informationen zu den Themen suchen, die hier besprochen werden.

4.1. Verbindung zum Server herstellen und trennen

Um sich zum Server zu verbinden, müssen Sie beim Aufruf von `mysql` in der Regel einen MySQL-Benutzernamen und üblicherweise auch ein Passwort angeben. Wenn der Server auf einer anderen Maschine als der läuft, von der Sie sich einloggen, müssen Sie auch einen Hostnamen angeben. Setzen Sie sich mit Ihrem Administrator in Verbindung, um herauszubekommen, welche Verbindungsparameter Sie benutzen sollten (das heißt welchen Host, welchen Benutzername und welches Passwort Sie verwenden sollen). Wenn Sie die richtigen Parameter kennen, sollten Sie sich wie folgt verbinden können:

```
shell> mysql -h host -u user -p
Enter password: ********
```

`********` steht für Ihr Passwort. Geben Sie es ein, wenn `mysql` `Enter password:` anzeigt.

Wenn das funktioniert hat, sehen Sie ein paar einführende Informationen, gefolgt von der `mysql>`-Eingabeaufforderung:

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end mit ; or \g.
Your MySQL connection id ist 459 to server version: 3.22.20a-log

Type 'help' for help.

mysql>
```

Die Eingabeaufforderung sagt Ihnen, dass `mysql` bereit für die Eingabe von Befehlen ist.

Einige Einige MySQL-Installationen erlauben Benutzern, sich als anonyme (unbenannte) Benutzer mit dem Server auf dem lokalen Host zu verbinden. Wenn das auf Ihrer Maschine der Fall ist, können Sie sich mit diesem Server verbinden, indem Sie `mysql` ohne irgend welche Optionen aufrufen:

```
shell> mysql
```

Nachdem Sie sich erfolgreich verbunden haben, können Sie die Verbindung jederzeit trennen, indem Sie `QUIT` an der `mysql>`-Eingabeaufforderung eingeben.

```
mysql> QUIT
Bye
```

Sie können die Verbindung auch trennen, indem Sie STRG+D eingeben.

Die meisten Beispiele der folgenden Abschnitte setzen voraus, dass Sie mit dem Server verbunden sind. Das wird durch `mysql>` angezeigt.

4.2. Anfragen eingeben

Stellen Sie sicher, dass Sie mit dem Server verbunden sind, wie im vorherigen Abschnitt erörtert. Dadurch wird noch keine Datenbank ausgewählt, mit der Sie arbeiten können, aber das ist in Ordnung. Hier ist es erst einmal wichtiger, herauszufinden, wie Sie Anfragen (Queries) absetzen, als direkt mit dem Erzeugen von Tabellen, dem Einladen von Daten in diese und der Abfrage von Daten aus diesen zu beginnen. Dieser Abschnitt beschreibt die grundlegenden Prinzipien der Befehlseingabe, indem etliche Anfragen gezeigt werden, die Sie ausprobieren können, um sich mit der Arbeitsweise von `mysql` vertraut zu machen.

Hier ist ein einfacher Befehl, der den Server bittet, Ihnen seine Versionsnummer und das aktuelle Datum mitzuteilen. Geben Sie folgendes an der `mysql>`-Eingabeaufforderung ein und drücken Sie die Eingabetaste:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| version()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

Diese Anfrage erläutert verschiedene Dinge über `mysql`:

- Ein Befehl besteht normalerweise aus einem SQL-Statement, gefolgt von einem Semikolon. (Es gibt ein paar Ausnahmen, bei denen das Semikolon nicht benötigt wird. `QUIT`, das vorher erwähnt wurde, stellt eine solche Ausnahme dar. Wir kommen später noch zu anderen Ausnahmen.)
- Wenn Sie einen Befehl absetzen, sendet `mysql` ihn zum Server, der ihn ausführt, und zeigt die Ergebnisse an. Danach wird eine neue `mysql>`-Eingabeaufforderung angezeigt, um klar zu machen, dass es für einen weiteren Befehl bereit ist.
- `mysql` zeigt die Ausgabe der Anfrage in Tabellenform an (Zeilen und Spalten). Die erste Zeile enthält Spaltenüberschriften. Die folgenden Zeilen sind die Ergebnisse der Anfrage. Normalerweise sind die Spaltenüberschriften die Spaltennamen der Tabellen, die Sie abfragen. Wenn Sie statt der Spaltennamen den Wert eines Ausdrucks abfragen (wie im gezeigten Beispiel), beschriftet `mysql` die Spaltenüberschrift mit dem Ausdruck selbst.
- `mysql` zeigt, wie viele Zeilen zurück gegeben wurden und wie lang die Ausführung der Anfrage dauerte, was ihnen eine grobe Einschätzung der Server-Performance ermöglicht. Diese Werte sind ungenau, weil sie die Wanduhrzeit repräsentieren (und nicht die Prozessor- oder Maschinenzeit), und weil sie von Faktoren wie der Serverlast und der Netzwerk-Wartezeit beeinflusst werden. (Um uns kurz zu fassen, zeigen wir die ``rows in set''-Zeile in den weiteren Beispielen dieses Kapitels nicht mehr an.)

Schlüsselwörter können in beliebiger Schreibweise (Groß und klein) eingegeben werden. Folgende Anfragen sind gleichwertig:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Hier kommt eine weitere Anfrage. Sie zeigt, wie Sie `mysql` als einfachen Rechner benutzen können:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

Die bislang gezeigten Befehle sind relativ kurze, einzeilige Statements. Sie können allerdings auch mehrfache Statements auf einer einzelnen Zeile eingeben. Beenden Sie einfach jedes davon mit einem Semikolon:

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| version()    |
+--------------+
| 3.22.20a-log |
+--------------+

+---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Ein Befehl muss nicht auf einer einzelnen Zeile eingegeben werden, so dass längere Befehle, die mehrere Zeilen erfordern, kein Problem darstellen. `mysql` stellt anhand des beendenden Semikolons fest, wo Ihr Statement endet, und nicht etwa anhand des Zeilenendes. (Mit anderen Worten akzeptiert `mysql` Freiformat-Eingaben: Es sammelt Eingabezeilen, führt sie aber solange nicht aus, bis es das Semikolon sieht.)

Hier ist ein einfaches Statement, auf mehrere Zeilen verteilt:

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith@localhost | 1999-03-18   |
+--------------------+--------------+
```

Sehen Sie, wie sich die Eingabeaufforderung von `mysql>` zu `->` ändert, nachdem Sie die erste Zeile der Mehrzeilen-Anfrage eingegeben haben. Auf diese Weise zeigt `mysql` an, dass es noch nicht das komplette Statement gesehen hat und auf den Rest wartet. Die Eingabeaufforderung ist Ihr Freund, denn sie stellt wertvolle Rückmeldungen zur Verfügung. Wenn Sie diese Rückmeldungen nutzen, werden Sie immer dessen gewahr sein, worauf `mysql` wartet.

Wenn Sie den Befehl, den Sie gerade eingeben, nicht ausführen wollen, können Sie ihn mit `\c` abbrechen:

```
mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Beachten Sie auch hierbei die Eingabeaufforderung. Sie ändert sich zurück zu `mysql>`, nachdem Sie `\c` eingegeben haben, und diese Rückmeldung bedeutet, dass `mysql` für einen weiteren Befehl bereit ist.

Folgende Tabelle zeigt alle Eingabeaufforderungen, denen Sie begegnen können, und fasst zusammen, was sie über den Zustand von `mysql` aussagen:

Eingabeaufforderung	Bedeutung
`mysql>`	Bereit für den nächsten Befehl.
`->`	Wartet auf die nächste Zeile eines mehrzeiligen Befehls.
`'>`	Wartet auf die nächste Zeile und fasst eine Zeichenkette zusammen, die mit einem Apostroph (‘`'`’) beginnt.
`">`	Wartet auf die nächste Zeile und fasst eine Zeichenkette zusammen, die mit Anführungszeichen (‘`"`’) beginnt.

Mehrzeilige Statements passieren häufig aus Versehen, wenn Sie vorhaben, einen Befehl auf einer einzelnen Zeile abzusetzen, aber das beendende Semikolon vergessen. In diesem Fall wartet `mysql` auf weitere Eingaben:

```
mysql> SELECT USER()
    ->
```

Wenn so etwas passiert (Sie dachten, Sie hätten ein Statement eingegeben, aber die einzige Antwort ist die `->`-Eingabeaufforderung), wartet `mysql` höchstwahrscheinlich auf das Semikolon. Wenn Sie nicht darauf achten, was Ihnen die Eingabeaufforderung mitteilen will, könnten Sie eine ganze Weile herum sitzen, bevor Sie feststellen, was Sie tun müssen. Geben Sie ein Semikolon ein, um das Statement zu vollenden, und `mysql` wird es ausführen:

```
mysql> SELECT USER()
    -> ;
+--------------------+
| USER()             |
+--------------------+
| joesmith@localhost |
+--------------------+
```

Die `'>`- und `">`-Eingabeaufforderungen kommen bei der Sammlung von Zeichenketten vor. In MySQL können Sie Zeichenketten wahlweise in ‘`'`’- oder ‘`"`’-Zeichen eingeschlossen eingeben (zum Beispiel `'hallo'` oder `"tschüß"`), und `mysql` läßt Sie auch Zeichenketten eingeben, die sich über mehrere Zeilen erstrecken. Wenn Sie eine `'>`- oder `">`-Eingabeaufforderung sehen, heißt das, dass Sie eine Zeile eingegeben haben, die eine Zeichenkette enthält, die mit ‘`'`’ oder ‘`"`’ beginnt, dass Sie aber noch nicht das entsprechende beendende Zeichen (ebenfalls ‘`'`’ oder ‘`"`’) eingegeben haben. Das ist in Ordnung, wenn Sie tatsächlich eine mehrzeilige Zeichenkette eingeben, aber wie wahrscheinlich ist das? Nicht sehr wahrscheinlich. Wahrscheinlicher ist, dass die `'>`- und `">`-Eingabeaufforderungen anzeigen, dass Sie versehentlich ein ‘`'`’- oder ‘`"`’-Zeichen ausgelassen haben. Beispiel:

```
mysql> SELECT * FROM meine_tabelle WHERE name = "Schmidt AND age < 30;
    ">
```

Wenn Sie dieses `SELECT`-Statement eingeben, dann EINGABE drücken und auf das Ergebnis warten, wird nichts passieren. Statt sich zu fragen, warum diese Anfrage so lange dauert, beachten Sie des Rätsels Lösung, die die `">`-Eingabeaufforderung anzeigt. Sie sagt Ihnen, dass `mysql` auf den Rest einer nicht beendeten Zeichenkette wartet. (Sehen Sie den Fehler im Statement? Der Zeichenkette `"Schmidt` fehlt das zweite Anführungszeichen.)

Was machen Sie in diesem Fall? Das einfachste ist, den Befehl abzubrechen. Sie können jetzt allerdings nicht einfach `\c` eingeben, weil `mysql` es als Teil der Zeichenkette interpretieren würde, die es gerade sammelt! Geben Sie daher zuerst das schließende Anführungszeichen ein, damit `mysql` weiß, dass die Zeichenkette zuende ist, und erst danach `\c`:

```
mysql> SELECT * FROM meine_tabelle WHERE name = "Schmidt AND age < 30;
    "> "\c
mysql>
```

Die Eingabeaufforderung ändert sich wieder zu `mysql>` und zeigt damit an, dass `mysql` für einen weiteren Befehl bereit ist.

Es ist wichtig, die Bedeutung der `'>`- und `">`-Eingabeaufforderungen zu kennen, denn wenn Sie versehentlich eine nicht beendete Zeichenkette eingeben, werden alle folgenden Zeilen, die Sie eingeben, von `mysql` ignoriert - inklusive einer Zeile, die `QUIT` enthält! Das kann recht verwirrend sein, besonders dann, wenn Sie nicht wissen, dass Sie das schließende Anführungszeichen eingeben müssen, bevor Sie den aktuellen Befehl abbrechen können.

4.3. Eine Datenbank erzeugen und benutzen

Jetzt, wo Sie wissen, wie Sie Befehle eingeben, ist es Zeit, auf eine Datenbank zuzugreifen.

Nehmen wir an, Sie haben zuhause mehrere Haustiere (Ihre Menagerie) und Sie wollen unterschiedliche Arten von Information über sie im Überblick behalten. Das können Sie tun, indem Sie Tabellen erzeugen, die Ihre Daten beinhalten, und die Sie mit den gewünschten Informationen füllen. Dann können Sie verschiedene Arten von Fragen über Ihre Haustiere beantworten, indem Sie Daten aus den Tabellen abrufen. Dieser Abschnitt zeigt Ihnen:

- Wie Sie eine Datenbank erzeugen
- Wie Sie eine Tabelle anlegen
- Wie Sie Daten in die Tabelle laden
- Wie Sie auf unterschiedliche Weise Daten aus der Tabelle abfragen
- Wie Sie mehrere Tabellen benutzen

Die Menagerie-Datenbank wird (bewusst) einfach sein, aber man kann sich leicht Situationen im echten Leben vorstellen, in denen ein ähnlicher Typ von Datenbank benutzt werden könnte. Beispielsweise könnte ein Bauer eine solche Datenbankbenutzung, um den Überblick über sein Vieh zu behalten, oder ein Tierarzt, um seine Patientendaten im Griff zu haben. Sie finden eine Menagerie-Distribution, die einige der benutzten Anfragen und Beispielsdaten enthält, auf der MySQL-Website. Sie finden die Distribution entweder unter komprimiertes `tar`-Format oder unter Zip-Format.

Benutzen Sie das `SHOW`-Statement, um herauszufinden, welche Datenbanken zur Zeit auf dem Server existieren:

```
mysql> SHOW DATABASES;
+----------+
| database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

Die Liste der Datenbanken weicht wahrscheinlich von derjenigen auf Ihrer Maschine ab, aber wahrscheinlich befinden sich die `mysql`- und `test`-Datenbanken darunter. Die `mysql`-Datenbank ist notwendig, weil darin die Zugriffsrechte für Benutzer gespeichert sind. Die `test`-Datenbank ist meist als Arbeitsbereich zum Ausprobieren dabei.

Wenn die `test`-Datenbank existiert, versuchen Sie, darauf zuzugreifen:

```
mysql> USE test
database changed
```

Beachten Sie, dass `USE` - wie `QUIT` - kein Semikolon erfordert. (Sie können solche Statements mit einem Semikolon beenden, wenn Sie wollen, es schadet nicht.) Das `USE`-Statement ist auch auf andere Art besonders: Es muss auf einer einzigen Zeile eingegeben werden.

Sie können die `test`-Datenbank für die folgenden Beispiele benutzen (wenn Sie Zugriff darauf haben), aber alles, was Sie dort anlegen, kann von jedem sonstigen, der Zugriff darauf hat, entfernt werden. Aus diesem Grund sollten Sie besser Ihren MySQL-Administrator um Erlaubnis bitten, eine eigene Datenbankbenutzung zu können. Nehmen wir an, Sie wollen Ihre Datenbank `menagerie` nennen. Dafür muss der Administrator folgenden Befehl eingeben:

```
mysql> GRANT ALL ON menagerie.* TO ihr_mysql_name;
```

Wobei `ihr_mysql_name` der MySQL-Benutzername ist, der Ihnen zugewiesen wurde.

4.3.1. Eine Datenbank erzeugen und auswählen

Wenn der Administrator für Sie eine Datenbank erzeugt, wenn er Ihre Zugriffsrechte einträgt, können Sie sie unmittelbar benutzen. Ansonsten müssen Sie sie selbst anlegen:

```
mysql> CREATE DATABASE menagerie;
```

Unter Unix sind Datenbanknamen abhängig von der Groß-/Kleinschreibung (im Gegensatz zu SQL-Schlüsselwörtern), daher müssen Sie sich auf Ihre Datenbank immer mit `menagerie` beziehen, nicht mit `Menagerie`, `MENAGERIE` oder irgend einer anderen Variante. Dasselbe gilt für Tabellennamen. (Unter Windows trifft diese Beschränkung nicht zu. Dennoch muss man sich bei einer gegebenen Anfrage auf Datenbanken und Tabellen in derselben Schreibweise beziehen.)

Das Erzeugen einer Datenbank wählt diese nicht zur Benutzung aus. Das müssen Sie explizit tun. Um `menagerie` zur aktuell ausgewählten Datenbank zu machen, benutzen Sie folgenden Befehl:

```
mysql> USE menagerie
database changed
```

Ihre Datenbank muss nur einmal erzeugt werden, aber Sie müssen sie jedes Mal zur Benutzung auswählen, wenn Sie eine `mysql`-Sitzung beginnen. Das können Sie durch die Eingabe eines `USE`-Statements wie oben beschrieben tun. Alternativ können Sie die Datenbank auf der Kommandozeile auswählen, wenn Sie `mysql` aufrufen. Geben Sie einfach ihren Namen nach den Verbindungsparametern ein, die Sie ansonsten eingeben müssen. Beispiel:

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

Beachten Sie, dass `menagerie` auf der gezeigten Kommandozeile nicht Ihr Passwort ist! Wenn Sie Ihr Passwort auf der Kommandozeile nach der `-p`-Option eingeben wollen, müssen Sie das ohne Leerzeichen dazwischen machen (beispielsweise als `-pmeinpasswort`, nicht als `-p meinpasswort`). Es wird allerdings nicht empfohlen, das Passwort auf der Kommandozeile einzugeben, denn dann kann es durch andere Benutzer, die auf Ihrer Maschine eingeloggt sind, ausspioniert werden.

4.3.2. Eine Tabelle erzeugen

Die Datenbank zu erzeugen ist leicht, aber bis jetzt ist sie noch leer, wie Ihnen `SHOW TABLES` zeigt:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

Der schwierigere Teil besteht darin, sich zu entscheiden, wie die Struktur Ihrer Datenbank sein sollte: Welche Tabellen Sie benötigen und welche Spalten in jeder Tabelle enthalten sein sollen.

Sie brauchen eine Tabelle, die für jedes Ihrer Haustiere einen Datensatz enthält. Diese kann `pet`-Tabelle genannt werden, und sie sollte zumindest den Namen jedes Tiers enthalten. Weil lediglich der Name nicht besonders interessant ist, sollte die Tabelle weitere Informationen enthalten. Wenn zum Beispiel mehr als eine Person in Ihrer Familie ein Haustier hält, würden Sie den Namen des Besitzers jedes Haustiers auflisten wollen. Ausserdem wollen Sie vielleicht ein paar grundlegende Informationen wie Art und Geschlecht einfügen.

Was ist mit dem Alter? Diese Information könnte interessant sein, aber es ist keine gute Idee, sie in der Datenbank zu speichern. Das Alter ändert sich, wenn die Zeit vergeht, was bedeutet, dass Sie Ihre Datensätze oft aktualisieren müssen. Statt dessen ist es besser, einen festen Wert wie das Geburtsdatum zu speichern. Immer, wenn Sie dann das Alter benötigen, berechnen Sie es als Differenz zwischen dem aktuellen Datum und dem Geburtstag. MySQL stellt Funktionen für Datumsberechnungen zur Verfügung, daher ist so etwas nicht schwer. Ausserdem hat die Speicherung von Geburtsdaten anstelle von Alter weitere Vorteile:

- Sie können die Datenbank für Aufgaben wie die Erzeugung einer Liste bevorstehender Tier-Geburtstage benutzen. (Wenn Sie das etwas albern finden, bedenken Sie, dass sich dieselbe Frage zum Beispiel bei einer Geschäftsdatenbank stellt, um Kunden herauszufinden, denen Sie in Kürze Geburtstagswünsche schicken wollen, also für die Computer-unterstützte persönliche Note.)
- Sie können Altersberechnungen mit anderen Bezugsdaten als dem aktuellen Datum durchführen. Wenn Sie das Sterbedatum speichern, können Sie zum Beispiel leicht errechnen, wie alt ein Haustier war, als es starb.

Wahrscheinlich fallen Ihnen weitere Informationen ein, die sinnvoller Weise in der `pet`-Tabelle gespeichert werden könnten. Für unser Beispiel sollen die bisher identifizierten Informationen fürs Erste ausreichen: Name, Besitzer, Art, Geschlecht, Geburtstag und Sterbetag.

Legen Sie mit einem `CREATE TABLE`-Statement das Layout Ihrer Tabelle fest:

```
mysql> CREATE TABLE pet (name VARCHAR(20), besitzer VARCHAR(20),
    -> art VARCHAR(20), geschlecht CHAR(1), geburtstag DATE, sterbetag DATE);
```

`VARCHAR` ist für die `name`-, `besitzer`- und `art`-Spalten eine gute Wahl, weil die Spaltenwerte in der Länge variieren werden. Diese Spalten müssen auch nicht alle gleich sein, also `20` Zeichen lang. Sie können jede beliebige Länge zwischen `1` und `255` wählen, was immer Ihnen vernünftig erscheint. (Wenn Sie eine schlechte Wahl getroffen haben und sich später herausstellt, dass Sie eine längere Spalte brauchen, stellt MySQL ein `ALTER TABLE`-Statement zur Verfügung.)

Das Geschlecht der Tiere kann vielfältig dargestellt werden, zum Beispiel als `"m"` und `"w"`, oder auch als `"männlich"` und `"weiblich"`. Am einfachsten ist es, hierfür einzelne Zeichen wie `"m"` und `"w"` zu verwenden.

Der `DATE`-Datentyp für `geburtstag` und `sterbetag` liegt auf der Hand.

Nachdem Sie eine Tabelle angelegt haben, sollte `SHOW TABLES` auch etwas zeigen:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| pet                 |
+---------------------+
```

Um sicherzustellen, dass Ihre Tabelle so wie erwartet angelegt wurde, benutzen Sie das `DESCRIBE`-Statement:

```
mysql> DESCRIBE pet;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| name       | varchar(20) | YES  |     | NULL    |       |
| besitzer   | varchar(20) | YES  |     | NULL    |       |
| art        | varchar(20) | YES  |     | NULL    |       |
| geschlecht | char(1)     | YES  |     | NULL    |       |
| geburtstag | date        | YES  |     | NULL    |       |
| sterbetag  | date        | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Sie können `DESCRIBE` jederzeit benutzen, zum Beispiel, wenn Sie die Namen der Spalten Ihrer Tabelle vergessen haben oder von welchem Datentyp sie sind.

4.3.3. Daten in Tabellen einladen

Nachdem Sie Ihre Tabelle erzeugt haben, müssen Sie sie mit Daten füllen. Hierfür sind die `LOAD DATA`- und `INSERT`-Statements nützlich.

Nehmen wir an, Sie haben Haustiere wie unten aufgeführt. (Achten Sie bei den Datumsangaben bitte darauf, dass MySQL Daten im `YYYY-MM-DD`-Format erwartet, was von dem Format abweichen kann, an das Sie gewohnt sind.)

name	besitzer	art	geschlecht	geburtstag	sterbetag
Fluffy	Harold	Katze	w	1993-02-04	
Claws	Gwen	Katze	m	1994-03-17	
Buffy	Harold	Hund	w	1989-05-13	
Fang	Benny	Hund	m	1990-08-27	
Bowser	Diane	Hund	m	1998-08-31	1995-07-29
Chirpy	Gwen	Vogel	w	1998-09-11	
Whistler	Gwen	Vogel		1997-12-09	
Slim	Benny	Schlange	m	1996-04-29	

Weil Sie mit einer leeren Tabelle beginnen, ist eine einfache Möglichkeit, diese mit Daten zu füllen, dass Sie eine Textdatei erzeugen, die eine Zeile für jedes Ihrer Tiere enthält, und die Inhalte dieser Datei dann mit einem einzigen Statement in die Tabelle laden.

Erzeugen Sie also eine Textdatei `pet.txt`, die einen Datensatz pro Zeile enthält, mit Werten, die durch TAB-Zeichen getrennt sind, und zwar in der Reihenfolge, in der die Spalten im `CREATE TABLE`-Statement aufgeführt waren. Fehlende Werte (wie unbekanntes Geschlecht oder Sterbedaten für Tiere, die noch leben) ersetzen Sie mit `NULL`-Werten. Um das in Ihrer Textdatei darzustellen, nehmen Sie `\N`. Der Datensatz für den Vogel Whistler zum Beispiel sieht wie folgt aus (wobei der Leerraum zwischen den Werten ein einzelnes TAB-Zeichen darstellt):

Whistler	Gwen	Vogel	\N	1997-12-09	\N

Um die Textdatei `pet.txt` in die `pet`-Tabelle zu laden, benutzen Sie folgenden Befehl:

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Sie können das Trennzeichen für die Spalten und das Zeichen für Zeilenende im `LOAD DATA`-Statement explizit festlegen, wenn Sie wollen, aber vorgabemäßig sind das das TAB-Zeichen und das Zeilenvorschub-Zeichen. Das reicht für das Statement aus, um die Datei `pet.txt` korrekt einzulesen.

Wenn Sie einzeln neue Datensätze hinzufügen wollen, ist das `INSERT`-Statement nützlich. In seiner einfachsten Form geben Sie für jede Spalte Werte an, in genau der Reihenfolge, in der die Spalten im `CREATE TABLE`-Statement aufgeführt wurden. Nehmen wir an, dass Diane einen neuen Hamster namens Puffball bekommt. Sie fügen einen neuen Datensatz mittels `INSERT`-Statement wie folgt hinzu:

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','Hamster','w','1999-03-30',NULL);
```

Beachten Sie, dass hierbei Zeichenketten- und Datumswerte in Anführungszeichen stehen. Mit `INSERT` können Sie auch direkt `NULL` einfügen, um einen fehlenden Wert darzustellen. Sie können dafür nicht `\N` wie bei `LOAD DATA` verwenden.

Diesem Beispiel können Sie auch entnehmen, dass es einiger Tipparbeit bedurft hätte, die anfänglichen Datensätze mit mehreren `INSERT`-Statements einzufügen, statt hierfür ein einziges `LOAD DATA`-Statement zu verwenden.

4.3.4. Informationen aus einer Tabelle abfragen

Das `SELECT`-Statement wird benutzt, um Informationen aus einer Tabelle herauszuziehen. Die allgemeine Form des Statements ist:

```
SELECT auszuwählende_spalten
FROM tabelle
WHERE gewünschte_bedingungen
```

`auszuwählende_spalten` bezeichnet, was Sie sehen wollen. Das kann entweder eine Liste von Spalten sein oder `*`, um ``alle Spalten" zu bezeichnen. `tabelle` kennzeichnet die Tabelle, aus der Sie Spalten abfragen wollen. Die `WHERE`-Klausel ist optional. Wenn sie vorhanden ist, kennzeichnet `gewünschte_bedingungen` die Bedingungen, mit denen die Zeilen übereinstimmen müssen, damit sie abgefragt werden.

4.3.4.1. Alle Daten auswählen

Die einfachste Form von `SELECT` fragt alles aus einer Tabelle ab:

```
mysql> SELECT * FROM pet;
+----------+----------+----------+------------+------------+------------+
| name     | besitzer | art      | geschlecht | geburtstag | sterbetag  |
+----------+----------+----------+------------+------------+------------+
| Fluffy   | Harold   | Katze    | w          | 1993-02-04 | NULL       |
| Claws    | Gwen     | Katze    | m          | 1994-03-17 | NULL       |
| Buffy    | Harold   | Hund     | w          | 1989-05-13 | NULL       |
| Fang     | Benny    | Hund     | m          | 1990-08-27 | NULL       |
| Bowser   | Diane    | Hund     | m          | 1998-08-31 | 1995-07-29 |
| Chirpy   | Gwen     | Vogel    | w          | 1998-09-11 | NULL       |
| Whistler | Gwen     | Vogel    | NULL       | 1997-12-09 | NULL       |
| Slim     | Benny    | Schlange | m          | 1996-04-29 | NULL       |
| Puffball | Diane    | Hamster  | w          | 1999-03-30 | NULL       |
+----------+----------+----------+------------+------------+------------+
```

Diese Form von `SELECT` ist nützlich, wenn Sie Ihre gesamte Tabelle abfragen wollen, zum Beispiel, wenn Sie sich gerade mit einem anfänglichen Satz Daten geladen haben. Wie das so passiert, zeigt die Ausgabe einen Fehler auf: Bowser scheint gestorben zu sein, bevor er geboren wurde! In den Original-Stammbaum-Papieren finden Sie, dass das korrekte Geburtsjahr 1989 ist, nicht 1998.

Es gibt eine ganze Reihe Möglichkeiten, das zu beheben:

- Editieren Sie die Datei `pet.txt` und beheben Sie den Fehler. Leeren Sie dann die Tabelle und laden Sie erneut Daten hinein, indem Sie zuerst `DELETE` und dann `LOAD DATA` benutzen:

  ```
  mysql> SET AUTOCOMMIT=1;  # Für schnelles Neuerzeugen der Tabelle
  mysql> DELETE FROM pet;
  mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
  ```

Wenn Sie das jedoch tun, müssen Sie die Daten für Puffball erneut eingeben.

- Den fehlerhaften Datensatz mit einem `UPDATE`-Statement in Ordnung bringen:

```
mysql> UPDATE pet SET geburtstag = "1989-08-31" WHERE name = "Bowser";
```

Wie gezeigt ist es einfach, eine ganze Tabelle abzufragen. Aber typischerweise wird das selten gewünscht sein, besonders, wenn die Tabelle Groß wird. Statt dessen werden Sie an der Antwort auf bestimmte Fragen interessiert sein, wobei Sie ein paar Beschränkungen in Bezug auf die Informationen, die Sie wollen, festlegen. Schauen wir uns einige Auswahl-Anfragen an, hinsichtlich der Fragen in Bezug auf Ihre Haustiere, die sie beantworten.

4.3.4.2. Bestimmte Zeilen auswählen

Sie können nur bestimmte Zeilen Ihrer Tabelle auswählen. Wenn Sie zum Beispiel die Geburtstags-Änderung von Bowser überprüfen wollen, wählen Sie Bowsers Datensatz wie folgt aus:

```
mysql> SELECT * FROM pet WHERE name = "Bowser";
+--------+----------+---------+-------------+------------+------------+
| name   | besitzer | art     | geschlecht  | geburtstag | sterbetag  |
+--------+----------+---------+-------------+------------+------------+
| Bowser | Diane    | dog     | m           | 1989-08-31 | 1995-07-29 |
+--------+----------+---------+-------------+------------+------------+
```

Die Ausgabe bestätigt, dass das Jahr inzwischen korrekt als 1989, nicht 1998, eingetragen ist.

Vergleiche von Zeichenketten achten normalerweise nicht auf Groß-/Kleinschreibung, daher können Sie den Namen als `"bowser"`, `"BOWSER"` usw. angeben. Das Anfrageergebnis wird dasselbe bleiben.

Sie können für jede Spalte Bedingungen festlegen, nicht nur für `name`. Wenn Sie zum Beispiel wissen wollen, welche Tiere nach 1998 geboren wurden, formulieren Sie eine Bedingung für die `geburtstag`-Spalte:

```
mysql> SELECT * FROM pet WHERE geburtstag >= "1998-1-1";
+----------+----------+---------+-------------+------------+-----------+
| name     | besitzer | art     | geschlecht  | geburtstag | sterbetag |
+----------+----------+---------+-------------+------------+-----------+
| Chirpy   | Gwen     | Vogel   | w           | 1998-09-11 | NULL      |
| Puffball | Diane    | Hamster | w           | 1999-03-30 | NULL      |
+----------+----------+---------+-------------+------------+-----------+
```

Sie können Bedingungen kombinieren, um zum Beispiel weibliche Hunde festzustellen:

```
mysql> SELECT * FROM pet WHERE art = "Hund" AND geschlecht = "w";
+-------+----------+---------+------------+------------+-----------+
| name  | besitzer | art     | geschlecht | geburtstag | sterbetag |
+-------+----------+---------+------------+------------+-----------+
| Buffy | Harold   | Hund    | w          | 1989-05-13 | NULL      |
+-------+----------+---------+------------+------------+-----------+
```

Die vorherige Anfrage benutzt den logischen Operator `AND`. Es gibt auch einen `OR`-Operator:

```
mysql> SELECT * FROM pet WHERE art = "Schlange" OR art = "Vogel";
+----------+----------+---------+-------------+------------+-----------+
| name     | besitzer | art     | geschlecht  | geburtstag | sterbetag |
+----------+----------+---------+-------------+------------+-----------+
| Chirpy   | Gwen     | Vogel   | w           | 1998-09-11 | NULL      |
| Whistler | Gwen     | Vogel   | NULL        | 1997-12-09 | NULL      |
| Slim     | Benny    | Schlange| m           | 1996-04-29 | NULL      |
+----------+----------+---------+-------------+------------+-----------+
```

`AND` und `OR` können gemischt werden. Wenn Sie das tun, ist es eine gute Idee, Klammern zu verwenden, um anzuzeigen, wie die Bedingungen gruppiert werden sollen:

```
mysql> SELECT * FROM pet WHERE (art = "Katze" AND geschlecht = "m")
    -> OR (art = "Hund" AND geschlecht = "w");
+-------+----------+---------+-------------+------------+-----------+
| name  | besitzer | art     | geschlecht  | geburtstag | sterbetag |
+-------+----------+---------+-------------+------------+-----------+
| Claws | Gwen     | Katze   | m           | 1994-03-17 | NULL      |
| Buffy | Harold   | Hund    | w           | 1989-05-13 | NULL      |
+-------+----------+---------+-------------+------------+-----------+
```

4.3.4.3. Bestimmte Spalten auswählen

Wenn Sie nicht ganze Zeilen Ihrer Tabelle sehen wollen, geben Sie einfach die Spalten an, an denen Sie interessiert sind, getrennt

durch Kommas. Wenn Sie zum Beispiel wissen wollen, wann Ihre Tiere geboren wurden, wählen Sie die `name`- und `geburtstag`-Spalten aus:

```
mysql> SELECT name, geburtstag FROM pet;
+----------+------------+
| name     | geburtstag |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Um herauszufinden, wem welches Haustier gehört, benutzen Sie diese Anfrage:

```
mysql> SELECT besitzer FROM pet;
+----------+
| besitzer |
+----------+
| Harold   |
| Gwen     |
| Harold   |
| Benny    |
| Diane    |
| Gwen     |
| Gwen     |
| Benny    |
| Diane    |
+----------+
```

Beachten Sie jedoch, dass diese Anfrage einfach die `besitzer`-Spalte jedes Datensatzes abfragt, wodurch mehrere von ihnen mehrfach erscheinen. Um die Ausgabe zu minimieren, fragen Sie jeden eindeutigen Datensatz nur einmal ab, indem Sie das Schlüsselwort `DISTINCT` verwenden:

```
mysql> SELECT DISTINCT besitzer FROM pet;
+-----------+
| besitzer  |
+-----------+
| Benny     |
| Diane     |
| Gwen      |
| Harold    |
+-----------+
```

Sie können eine `WHERE`-Klausel verwenden, um die Auswahl von Zeilen mit der Auswahl von Spalten zu kombinieren. Um zum Beispiel nur die Geburtstage von Hunden und Katzen zu erhalten, benutzen Sie diese Anfrage:

```
mysql> SELECT name, art, geburtstag FROM pet
    -> WHERE art = "Hund" OR art = "Katze";
+--------+---------+------------+
| name   | art     | geburtstag |
+--------+---------+------------+
| Fluffy | Katze   | 1993-02-04 |
| Claws  | Katze   | 1994-03-17 |
| Buffy  | Hund    | 1989-05-13 |
| Fang   | Hund    | 1990-08-27 |
| Bowser | Hund    | 1989-08-31 |
+--------+---------+------------+
```

4.3.4.4. Zeilen sortieren

Sie haben bei den vorherigen Beispielen vielleicht bemerkt, dass die Ergebniszeilen in keiner bestimmten Reihenfolge angezeigt werden. Häufig ist es jedoch einfacher, die Ausgabe der Anfrage zu überprüfen, wenn die Zeilen auf sinnvolle Art sortiert werden. Um ein Ergebnis zu sortieren, benutzen Sie die `ORDER BY`-Klausel.

Hier sind die Geburtstage der Haustiere, sortiert nach Geburtstag:

```
mysql> SELECT name, geburtstag FROM pet ORDER BY geburtstag;
+----------+------------+
| name     | geburtstag |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
```

```
| Puffball | 1999-03-30 |
+----------+------------+
```

Um in umgekehrter Reihenfolge zu sortieren, fügen Sie das `DESC`- (descending) Schlüsselwort zum Namen der Spalte, die Sie sortieren wollen, hinzu:

```
mysql> SELECT name, geburtstag FROM pet ORDER BY geburtstag DESC;
+----------+------------+
| name     | geburtstag |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Sie können über mehrere Spalten sortieren. Um beispielsweise zuerst nach der Art des Tieres und dann nach dem Geburtsdatum innerhalb der Tierart zu sortieren (die jüngsten Tiere zuerst), benutzen Sie folgende Anfrage:

```
mysql> SELECT name, art, geburtstag FROM pet ORDER BY art, geburtstag DESC;
+----------+----------+------------+
| name     | art      | geburtstag |
+----------+----------+------------+
| Chirpy   | Vogel    | 1998-09-11 |
| Whistler | Vogel    | 1997-12-09 |
| Claws    | Katze    | 1994-03-17 |
| Fluffy   | Katze    | 1993-02-04 |
| Fang     | Hund     | 1990-08-27 |
| Bowser   | Hund     | 1989-08-31 |
| Buffy    | Hund     | 1989-05-13 |
| Puffball | Hamster  | 1999-03-30 |
| Slim     | Schlange | 1996-04-29 |
+----------+----------+------------+
```

Beachten Sie, dass sich das `DESC`-Schlüsselwort nur auf die Spalte bezieht, die unmittelbar davor steht (`geburtstag`). `art`-Werte werden nach wie vor in aufsteigender Reihenfolge sortiert.

4.3.4.5. Datumsberechnungen

MySQL stellt etliche Funktionen zur Verfügung, mit denen Sie Datumsberechnungen wie Altersberechnungen oder das Extrahieren von Datumsteilen durchführen können.

Um festzustellen, wie alt jedes Ihrer Haustiere ist, berechnen Sie die Differenz im Jahresanteil des aktuellen Datums und des Geburtstags und subtrahieren eins, wenn das aktuelle Datum früher im Kalender erscheint als das Geburtsdatum. Folgende Anfrage zeigt für jedes Haustier das Geburtsdatum, das aktuelle Datum und das Alter in Jahren:

```
mysql> SELECT name, geburtstag, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(geburtstag))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(geburtstag,5))
    -> AS age
    -> FROM pet;
+----------+------------+--------------+------+
| name     | geburtstag | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
+----------+------------+--------------+------+
```

Hier zieht `YEAR()` den Jahresanteil eines Datums heraus. `RIGHT()` zieht die rechts stehenden fünf Zeichen heraus, die für den `MM-DD`-Teil des Datums stehen. Der Teil in dem Ausdruck, der die `MM-DD`-Werte vergleicht, wird zu 1 oder 0 ausgewertet, was die Jahresdifferenz ein Jahr nach unten anpasst, wenn `CURRENT_DATE` früher im Jahr erscheint als `geburtstag`. Der gesamte Ausdruck ist als Überschrift etwas plump, daher wir ein Alias (`age`) benutzt, um die Spaltenüberschrift etwas lesbarer zu machen.

Die Anfrage funktioniert, aber das Ergebnis könnte leichter überblickt werden, wenn die Zeilen in einer bestimmten Reihenfolge angezeigt würden. Das kann man erreichen, indem man eine `ORDER BY name`-Klausel hinzufügt, um die Ausgabe nach Namen zu sortieren:

```
mysql> SELECT name, geburtstag, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(geburtstag))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(geburtstag,5))
```

```
    -> AS age
    -> FROM pet ORDER BY name;
+----------+------------+--------------+------+
| name     | geburtstag | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
+----------+------------+--------------+------+
```

Um die Ausgabe nach Alter (`age`) statt nach `name` zu sortieren, benutzen Sie einfach eine andere `ORDER BY`-Klausel:

```
mysql> SELECT name, geburtstag, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(geburtstag))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(geburtstag,5))
    -> AS age
    -> FROM pet ORDER BY age;
+----------+------------+--------------+------+
| name     | geburtstag | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
+----------+------------+--------------+------+
```

Eine ähnliche Anfrage kann benutzt werden, um das Alter am Sterbetag bei Tieren festzustellen, die gestorben sind. Das können Sie feststellen, indem Sie überprüfen, ob der `sterbetag`-Wert `NULL` ist. Dann berechnen Sie für diejenigen Tiere mit Nicht-`NULL`-Werten den Unterschied zwischen `sterbetag`- und `geburtstag`-Werten:

```
mysql> SELECT name, geburtstag, sterbetag,
    -> (YEAR(sterbetag)-YEAR(geburtstag)) - (RIGHT(sterbetag,5)<RIGHT(geburtstag,5))
    -> AS age
    -> FROM pet WHERE sterbetag IS NOT NULL ORDER BY age;
+--------+------------+------------+------+
| name   | geburtstag | sterbetag  | age  |
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 |    5 |
+--------+------------+------------+------+
```

Die Anfrage benutzt `sterbetag IS NOT NULL` statt `sterbetag != NULL`, weil `NULL` ein spezieller Wert ist. Das wird später erklärt. See Abschnitt 4.3.4.6, „Mit NULL-Werten arbeiten“.

Was ist, wenn Sie wissen wollen, welche Tiere nächsten Monat Geburtstag haben? Für diese Art von Berechnung sind Jahre und Tage irrelevant. Sie wollen lediglich den Monatsanteil der `geburtstag`-Spalte extrahieren. MySQL bietet etliche Funktionen für die Extraktion von Datumsanteilen, wie `YEAR()`, `MONTH()` und `DAYOFMONTH()`. `MONTH()` ist hier die richtige Funktion. Um festzustellen, wie sie funktioniert, geben Sie eine Anfrage ein, die sowohl die Werte von `geburtstag` als auch die von `MONTH(geburtstag)` ausgibt:

```
mysql> SELECT name, geburtstag, MONTH(geburtstag) FROM pet;
+----------+------------+-------------------+
| name     | geburtstag | MONTH(geburtstag) |
+----------+------------+-------------------+
| Fluffy   | 1993-02-04 |                 2 |
| Claws    | 1994-03-17 |                 3 |
| Buffy    | 1989-05-13 |                 5 |
| Fang     | 1990-08-27 |                 8 |
| Bowser   | 1989-08-31 |                 8 |
| Chirpy   | 1998-09-11 |                 9 |
| Whistler | 1997-12-09 |                12 |
| Slim     | 1996-04-29 |                 4 |
| Puffball | 1999-03-30 |                 3 |
+----------+------------+-------------------+
```

Tiere mit Geburtstagen im kommenden Monat zu finden ist ebenfalls leicht. Nehmen wir an, der aktuelle Monat ist April. Dann ist der Monatswert 4 und Sie suchen nach Tieren, die im Mai (Monat 5) geboren sind, wie folgt:

```
mysql> SELECT name, geburtstag FROM pet WHERE MONTH(geburtstag) = 5;
+-------+------------+
| name  | geburtstag |
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+
```

Ein bisschen komplizierter ist es, wenn der aktuelle Monat Dezember ist. Hier können Sie nicht einfach eins zur Monatszahl (`12`) hinzufügen, weil es keinen 13. Monat gibt. Statt dessen suchen Sie nach Tieren, die im Januar (Monat 1) geboren sind.

Sie können die Anfrage sogar so schreiben, dass sie unabhängig davon funktioniert, was der aktuelle Monat ist. Auf diese Art brauchen Sie keine bestimmte Monatszahl in der Anfrage benutzen. `DATE_ADD()` erlaubt Ihnen, einem gegebenen Datum ein Zeitintervall hinzuzufügen. Wenn Sie dem Wert von `NOW()` einen Monat hinzufügen und dann den Monatsanteil mit `MONTH()` extrahieren, ergibt das den Monat, der die kommenden Geburtstage enthält:

```
mysql> SELECT name, geburtstag FROM pet
    -> WHERE MONTH(geburtstag) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));
```

Eine andere Möglichkeit, diese Aufgabe zu erfüllen, ist, `1` zu addieren, um den nächsten Monat nach dem aktuellen zu erhalten (nach Gebrauch der Modulo-Funktion (`MOD`), um den Monatswert auf `0` zu stellen, falls er aktuell `12`) ist:

```
mysql> SELECT name, geburtstag FROM pet
    -> WHERE MONTH(geburtstag) = MOD(MONTH(NOW()), 12) + 1;
```

`MONTH` gibt eine Zahl zwischen 1 und 12 zurück. `MOD(irgendwas,12)` gibt eine Zahl zwischen 0 und 11 zurück. Daher muss die Addition nach `MOD()` erfolgen, weil wir ansonsten von November (11) bis Januar (1) gehen würden.

4.3.4.6. Mit `NULL`-Werten arbeiten

Der `NULL`-Wert birgt Überraschungen, bis Sie mit ihm vertraut sind. Konzeptionell bedeutet `NULL` einen fehlenden oder unbekannten Wert. Er wird in einiger Hinsicht anders als andere Werte behandelt. Um auf `NULL` zu testen, können Sie nicht die arithmetischen Vergleichoperatoren wie `=`, `<` oder `!=` verwenden. Um sich das zu veranschaulichen, probieren Sie folgenden Anfrage:

```
mysql> SELECT 1 = NULL, 1 != NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 != NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

Wie man sieht, erhält man aus diesen Vergleichen keine sinnvollen Ergebnisse. Benutzen Sie statt dessen die `IS NULL`- und `IS NOT NULL`-Operatoren:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
|         0 |             1 |
+-----------+---------------+
```

In MySQL bedeutet 0 oder `NULL` logisch Falsch und alles sonstige bedeutet logisch Wahr. Der vorgabemäßige Wahrheitswert einer Boolschen Operation ist 1.

Diese besondere Behandlung von `NULL` ist der Grund, warum es im vorherigen Abschnitt notwendig war, mit `sterbetag IS NOT NULL` anstelle von `sterbetag != NULL` festzustellen, welche Tiere nicht mehr leben.

4.3.4.7. Übereinstimmende Suchmuster

MySQL stellt Standard-SQL-Suchmuster-Übereinstimmung zur Verfügung, ebenso wie eine Art der Suchmuster-Übereinstimmung, die auf regulären Ausdrücken basiert, die denen ähnlich sind, die von Unix-Hilfsprogrammen wie `vi`, `grep` und `sed` benutzt werden.

SQL-Suchmuster-Übereinstimmung gestattet Ihnen, '`_`' zu benutzen, um ein einzelnes Zeichen und '`%`', um eine beliebige Anzahl von Zeichen (inklusive des 0-Zeichens) zu finden. In den MySQL-SQL-Suchmustern spielt die Groß-/Kleinschreibung vorgabemäßig keine Rolle. Einige Beispiele sind unten dargestellt. Beachten Sie, dass Sie `=` oder `!=` nicht benutzen können, wenn Sie SQL-Suchmuster benutzen. Stattdessen müssen Sie die `LIKE`- oder `NOT LIKE`-Vergleichsoperatoren benutzen.

So finden Sie Namen, die mit '`b`' anfangen:

```
mysql> SELECT * FROM pet WHERE name LIKE "b%";
+--------+----------+------+------------+------------+------------+
| name   | besitzer | art  | geschlecht | geburtstag | sterbetag  |
+--------+----------+------+------------+------------+------------+
| Buffy  | Harold   | Hund | w          | 1989-05-13 | NULL       |
| Bowser | Diane    | Hund | m          | 1989-08-31 | 1995-07-29 |
+--------+----------+------+------------+------------+------------+
```

So finden Sie Namen, die auf '`fy`' enden:

```
mysql> SELECT * FROM pet WHERE name LIKE "%fy";
+--------+----------+-------+------------+------------+-----------+
| name   | besitzer | art   | geschlecht | geburtstag | sterbetag |
+--------+----------+-------+------------+------------+-----------+
| Fluffy | Harold   | Katze | w          | 1993-02-04 | NULL      |
| Buffy  | Harold   | Hund  | w          | 1989-05-13 | NULL      |
+--------+----------+-------+------------+------------+-----------+
```

So finden Sie Namen, die ‘w’ enthalten:

```
mysql> SELECT * FROM pet WHERE name LIKE "%w%";
+----------+----------+---------+-------------+------------+------------+
| name     | besitzer | art     | geschlecht  | geburtstag | sterbetag  |
+----------+----------+---------+-------------+------------+------------+
| Claws    | Gwen     | Katze   | m           | 1994-03-17 | NULL       |
| Bowser   | Diane    | Hund    | m           | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen     | bird    | NULL        | 1997-12-09 | NULL       |
+----------+----------+---------+-------------+------------+------------+
```

Um Namen zu finden, die genau fünf Zeichen enthalten, benutzen Sie das ‘_’-Suchmuster-Zeichen:

```
mysql> SELECT * FROM pet WHERE name LIKE "_____";
+-------+----------+---------+-------------+------------+-----------+
| name  | besitzer | art     | geschlecht  | geburtstag | sterbetag |
+-------+----------+---------+-------------+------------+-----------+
| Claws | Gwen     | Katze   | m           | 1994-03-17 | NULL      |
| Buffy | Harold   | Hund    | w           | 1989-05-13 | NULL      |
+-------+----------+---------+-------------+------------+-----------+
```

Die andere Art von Suchmuster-Übereinstimmung benutzt erweiterte reguläre Ausdrücke. Wenn Sie bei dieser Art von Suchmuster auf Übereinstimmung prüfen, benutzen Sie die `REGEXP`- und `NOT REGEXP`-Operatoren (oder `RLIKE` und `NOT RLIKE`, die synonym sind).

Einige Charakteristika erweiterter regulärer Ausdrücke sind:

- ‘`.`’ findet jedes beliebige Zeichen.
- Eine Zeichenklasse ‘`[...]`’ findet jedes Zeichen innerhalb der eckigen Klammern. So stimmt zum Beispiel ‘`[abc]`’ mit ‘`a`’, ‘`b`’ oder ‘`c`’ überein. Um einen Bereich von Zeichen zu benennen, benutzen Sie einen Bindestrich. ‘`[a-z]`’ stimmt mit jedem Buchstaben in Kleinschreibung überein, wohingegen ‘`[0-9]`’ mit jeder Ziffer übereinstimmt.
- ‘`*`’ stimmt mit null oder mehr Instanzen der Sache überein, die ihm voransteht. ‘`x*`’ zum Beispiel stimmt mit einer beliebigen Anzahl von ‘`x`’-Zeichen überein. ‘`[0-9]*`’ stimmt mit einer beliebigen Anzahl von Ziffern überein, und ‘`.*`’ mit jeder Anzahl von irgendetwas.
- Reguläre Ausdrücke achten auf Groß-/Kleinschreibung, aber Sie können eine Zeichenklasse benutzen, um beide Schreibungen zu finden, wenn Sie wollen. ‘`[aA]`’ zum Beispiel stimmt mit ‘`a`’ in Groß- und Kleinschreibung überein und ‘`[a-zA-Z]`’ mit jedem Buchstaben in Groß- und Kleinschreibung.
- Das Suchmuster stimmt überein, wenn es irgendwo in dem Wert auftaucht, der überprüft wird. (SQL-Suchmuster stimmen nur überein, wenn sie mit dem gesamten Wert übereinstimmen.)
- Um ein Suchmuster so zu verankern, dass er mit dem Anfang oder dem Ende des überprüften Werts übereinstimmen muss, benutzen Sie ‘`^`’ am Anfang oder ‘`$`’ am Ende des Suchmusters.

Um darzustellen, wie erweiterte reguläre Ausdrücke funktionieren, werden die `LIKE`-Anfragen von oben noch einmal mit `REGEXP` gezeigt.

Um Namen zu finden, die mit ‘`b`’ anfangen, benutzen Sie ‘`^`’, um auf Übereinstimmung am Anfang des Namens zu prüfen:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^b";
+--------+----------+---------+-------------+------------+------------+
| name   | besitzer | art     | geschlecht  | geburtstag | sterbetag  |
+--------+----------+---------+-------------+------------+------------+
| Buffy  | Harold   | Hund    | w           | 1989-05-13 | NULL       |
| Bowser | Diane    | Hund    | m           | 1989-08-31 | 1995-07-29 |
+--------+----------+---------+-------------+------------+------------+
```

Vor MySQL-Version 3.23.4 achtet `REGEXP` auf Groß-/Kleinschreibung. Daher gibt diese Anfrage ein Ergebnis ohne Zeilen zurück. Um sowohl Groß- als auch Kleinschreibung von ‘`b`’ zu finden, benutzen Sie statt dessen folgende Anfrage:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^[bB]";
```

Ab MySQL 3.23.4 müssen Sie, um die Beachtung der Groß-/Kleinschreibung in einem `REGEXP`-Vergleich zu erzwingen, das

`BINARY`-Schlüsselwort verwenden, um eine der Zeichenketten zu einer binären Zeichenkette zu machen. Diese Anfrage stimmt nur mit ‘`b`’ in Kleinschreibung am Anfang eines Namens überein:

```
mysql> SELECT * FROM pet WHERE name REGEXP BINARY "^b";
```

Um Namen zu finden, die auf ‘`fy`’ enden, benutzen Sie ‘`$`’, um Übereinstimmung am Ende des Namens zu finden:

```
mysql> SELECT * FROM pet WHERE name REGEXP "fy$";
+--------+----------+---------+------------+------------+-----------+
| name   | besitzer  | art     | geschlecht  | geburtstag | sterbetag |
+--------+----------+---------+------------+------------+-----------+
| Fluffy | Harold    | Katze   | w           | 1993-02-04 | NULL      |
| Buffy  | Harold    | Hund    | w           | 1989-05-13 | NULL      |
+--------+----------+---------+------------+------------+-----------+
```

Um Namen zu finden, die ‘`w`’ in Groß- oder Kleinschreibung enthalten, benutzen Sie diese Anfrage:

```
mysql> SELECT * FROM pet WHERE name REGEXP "w";
+----------+----------+---------+------------+------------+------------+
| name     | besitzer | art     | geschlecht  | geburtstag | sterbetag  |
+----------+----------+---------+------------+------------+------------+
| Claws    | Gwen     | Katze   | m           | 1994-03-17 | NULL       |
| Bowser   | Diane    | Hund    | m           | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen     | bird    | NULL        | 1997-12-09 | NULL       |
+----------+----------+---------+------------+------------+------------+
```

Weil ein Suchmuster mit regulären Ausdrücken an beliebiger Stelle im Wert gefunden wird, ist es bei der vorherigen Anfrage nicht notwendig, ein Jokerzeichen (Wildcard) auf irgendeine Seite des Suchmusters zu setzen, um nach Übereinstimmung im gesamten Wert zu suchen, wie es bei SQL-Suchmustern der Fall sein müsste.

Um Namen zu finden, die genau fünf Zeichen enthalten, benutzen Sie ‘`^`’ und ‘`$`’, um mit Anfang und Ende des Namens Übereinstimmung zu finden, und fünf Instanzen von ‘`.`’ dazwischen:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.....$";
+-------+----------+---------+------------+------------+-----------+
| name  | besitzer  | art     | geschlecht  | geburtstag | sterbetag |
+-------+----------+---------+------------+------------+-----------+
| Claws | Gwen      | Katze   | m           | 1994-03-17 | NULL      |
| Buffy | Harold    | Hund    | w           | 1989-05-13 | NULL      |
+-------+----------+---------+------------+------------+-----------+
```

Sie könnten die vorherige Anfrage auch unter Verwendung des ‘`{n}`’- ``wiederhole-`n`-mal"-Operators schreiben:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.{5}$";
+-------+----------+---------+------------+------------+-----------+
| name  | besitzer  | art     | geschlecht  | geburtstag | sterbetag |
+-------+----------+---------+------------+------------+-----------+
| Claws | Gwen      | Katze   | m           | 1994-03-17 | NULL      |
| Buffy | Harold    | Hund    | w           | 1989-05-13 | NULL      |
+-------+----------+---------+------------+------------+-----------+
```

4.3.4.8. Zeilen zählen

Datenbanken werden oft benutzt, um die Frage zu beantworten, wie oft eine bestimmte Art von Daten in einer Tabelle erscheint. Sie wollen beispielsweise wissen, wie viele Haustiere Sie haben, oder wie viele Haustiere jeder Besitzer hat, oder Sie wollen verschiedene Arten von Zählungen Ihrer Tiere durchführen.

Die Gesamtzahl der Tiere zählen, die Sie haben, ist dieselbe Frage wie ``Wie viele Zeilen sind in der `pet`-Tabelle?", denn es gibt einen Datensatz pro Haustier. Die `COUNT()`-Funktion zählt die Anzahl von Nicht-`NULL`-Ergebnissen, daher lautet die Anfrage, um Ihre Tiere zu zählen, wie folgt:

```
mysql> SELECT COUNT(*) FROM pet;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

Sie haben vorher schon einmal die Namen der Leute abgefragt, die Haustiere besitzen. Sie können `COUNT()` benutzen, wenn Sie herausfinden wollen, wie viele Tiere jeder Besitzer hat:

```
mysql> SELECT besitzer, COUNT(*) FROM pet GROUP BY besitzer;
+-----------+----------+
| besitzer  | COUNT(*) |
+-----------+----------+
| Benny     |        2 |
| Diane     |        2 |
| Gwen      |        3 |
| Harold    |        2 |
```

```
+-----------+----------+
```

Beachten Sie die Benutzung von `GROUP BY`, um alle Datensätze für jeden `besitzer` zu gruppieren. Ohne das erhalten Sie eine Fehlermeldung:

```
mysql> SELECT besitzer, COUNT(besitzer) FROM pet;
ERROR 1140 at line 1: Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

`COUNT()` und `GROUP BY` sind nützlich, um Ihre Daten auf verschiedene Weise zu charakterisieren. Die folgenden Beispiele zeigen verschiedene Möglichkeiten, um Zählungen Ihrer Tiere durchzuführen.

Anzahl der Tiere pro Art:

```
mysql> SELECT art, COUNT(*) FROM pet GROUP BY art;
+---------+----------+
| art     | COUNT(*) |
+---------+----------+
| Vogel   |        2 |
| Katze   |        2 |
| Hund    |        3 |
| Hamster |        1 |
| Schlange|        1 |
+---------+----------+
```

Anzahl der Tiere pro Geschlecht:

```
mysql> SELECT geschlecht, COUNT(*) FROM pet GROUP BY geschlecht;
+-------------+----------+
| geschlecht  | COUNT(*) |
+-------------+----------+
| NULL        |        1 |
| w           |        4 |
| m           |        4 |
+-------------+----------+
```

(In dieser Ausgabe zeigt `NULL` an, dass das Geschlecht unbekannt ist.)

Anzahl der Tiere pro Kombination von Art und Geschlecht:

```
mysql> SELECT art, geschlecht, COUNT(*) FROM pet GROUP BY art, geschlecht;
+---------+-------------+----------+
| art     | geschlecht  | COUNT(*) |
+---------+-------------+----------+
| Vogel   | NULL        |        1 |
| Vogel   | w           |        1 |
| Katze   | w           |        1 |
| Katze   | m           |        1 |
| Hund    | w           |        1 |
| Hund    | m           |        2 |
| Hamster | w           |        1 |
| Schlange| m           |        1 |
+---------+-------------+----------+
```

Sie müssen nicht die gesamte Tabelle abfragen, wenn Sie `COUNT()` benutzen. Die vorherige Anfrage beispielsweise sieht lediglich für Hunde und Katzen wie folgt aus:

```
mysql> SELECT art, geschlecht, COUNT(*) FROM pet
    -> WHERE art = "Hund" OR art = "Katze"
    -> GROUP BY art, geschlecht;
+---------+-------------+----------+
| art     | geschlecht  | COUNT(*) |
+---------+-------------+----------+
| Katze   | w           |        1 |
| Katze   | m           |        1 |
| Hund    | w           |        1 |
| Hund    | m           |        2 |
+---------+-------------+----------+
```

Oder wenn Sie die Anzahl von Tieren pro Geschlecht wissen wollen, beschränkt auf die Tiere, deren Geschlecht bekannt ist:

```
mysql> SELECT art, geschlecht, COUNT(*) FROM pet
    -> WHERE geschlecht IS NOT NULL
    -> GROUP BY art, geschlecht;
+---------+-------------+----------+
| art     | geschlecht  | COUNT(*) |
+---------+-------------+----------+
| Vogel   | w           |        1 |
| Katze   | w           |        1 |
| Katze   | m           |        1 |
| Hund    | w           |        1 |
| Hund    | m           |        2 |
| Hamster | w           |        1 |
```

```
| Schlange| m           |        1 |
+---------+-------------+----------+
```

4.3.4.9. Mehr als eine Tabelle benutzen

In der `pet`-Tabelle behalten Sie die Übersicht über Ihre Haustiere. Wenn Sie weitere Informationen über sie aufzeichnen wollen, beispielsweise Ereignisse in ihrem Leben wie Besuche beim Tierarzt oder wenn Nachwuchs zur Welt kommt, brauchen Sie eine weitere Tabelle. Wie sollte diese aussehen? Sie benötigt:

- Den Namen des Haustiers, damit Sie wissen, auf welches Tier sich jedes Ereignis bezieht.
- Ein Datum, damit Sie wissen, wann sich das Ereignis zugetragen hat.
- Ein Feld, um das Ereignis zu beschreiben.
- Ein Feld für den Typ des Ereignisses, wenn Sie in der Lage sein wollen, Ereignisse zu kategorisieren.

Nach diesen Vorüberlegungen könnte das `CREATE TABLE`-Statement für die `ereignis`-Tabelle wie folgt aussehen:

```
mysql> CREATE TABLE ereignis (name VARCHAR(20), datum DATE,
    -> typ VARCHAR(15), bemerkung VARCHAR(255));
```

Wie bei der `pet`-Tabelle ist es am einfachsten, die anfänglichen Datensätze mit Hilfe einer TAB-getrennten Textdatei einzuladen, die folgende Informationen enthält:

Fluffy	1995-05-15	Nachwuchs	4 Kätzchen, 3 weiblich, 1 männlich
Buffy	1993-06-23	Nachwuchs	5 Hündchen, 2 weiblich, 3 männlich
Buffy	1994-06-19	Nachwuchs	3 Hündchen, 3 weiblich
Chirpy	1999-03-21	Tierarzt	Schnabel gerade gebogen
Slim	1997-08-03	Tierarzt	Gebrochene Rippe
Bowser	1991-10-12	Zwinger	
Fang	1991-10-12	Zwinger	
Fang	1998-08-28	Geburtstag	Geschenk: neues Kauspielzeug
Claws	1998-03-17	Geburtstag	Geschenk: neues Flohhalsband
Whistler	1998-12-09	Geburtstag	Erster Geburtstag

Laden Sie die Datensätze wie folgt ein:

```
mysql> LOAD DATA LOCAL INFILE "ereignis.txt" INTO TABLE ereignis;
```

Auf der Grundlage dessen, was Sie durch die Abfragen der `pet`-Tabelle gelernt haben, sollten sie in der Lage sein, Abfragen der Datensätze der `ereignis`-Tabelle durchzuführen, was prinzipiell dasselbe ist. Aber wann ist die `ereignis`-Tabelle allein nicht ausreichend, um Fragen zu beantworten, die Sie stellen könnten?

Nehmen wir an, Sie wollen herausfinden, wie alt jedes Haustier war, als es Nachwuchs bekam. In der `ereignis`-Tabelle steht, wann das geschah, aber um das Alter der Mutter auszurechnen, wird ihr Geburtstag benötigt. Weil dieser in der `pet`-Tabelle steht, brauchen Sie für diese Anfrage beide Tabellen:

```
mysql> SELECT pet.name, (TO_DAYS(datum) - TO_DAYS(geburtstag))/365 AS age,anmerkung
    -> FROM pet, ereignis
    -> WHERE pet.name = ereignis.name AND typ = "Nachwuchs";
+--------+------+-----------------------------------+
| name   | age  | anmerkung                         |
+--------+------+-----------------------------------+
| Fluffy | 2.27 | 4 kätzchen, 3 weiblich, 1 männlich |
| Buffy  | 4.12 | 5 hündchen, 2 weiblich, 3 männlich |
| Buffy  | 5.10 | 3 hündchen, 3 weiblich            |
+--------+------+-----------------------------------+
```

Zu dieser Anfrage gibt es einiges anzumerken:

- In der `FROM`-Klausel stehen zwei Tabellen, weil die Anfrage aus beiden Tabellen Informationen herausziehen muss.
- Wenn Sie Informationen aus mehreren Tabellen verbinden (englisch: join), müssen Sie angeben, wie Datensätze in der einen

Tabelle mit solchen in der anderen Tabelle in Übereinstimmung gebracht werden können. Das ist einfach, weil beide eine `name`-Spalte haben. Die Anfrage benutzt die `WHERE`-Klausel, um Datensätze beider Tabellen basierend auf den `name`-Werten in Übereinstimmung zu bringen.

- Weil die `name`-Spalte in beiden Tabellen vorkommt, müssen Sie angeben, welche Tabelle Sie meinen, wenn Sie auf die Spalte verweisen. Das wird gemacht, indem dem Spaltennamen der Tabellenname voran gestellt wird.

Sie müssen nicht unbedingt zwei verschiedene Tabellen haben, um eine Verknüpfung (Join) durchzuführen. Manchmal ist es nützlich, eine Tabelle mit sich selbst zu verknüpfen, wenn Sie nämlich Datensätze in einer Tabelle mit Datensätze in derselben Tabelle vergleichen wollen. Um zum Beispiel Zuchtpaare unter Ihren Haustieren zu finden, können Sie die `pet`-Tabelle mit sich selbst verknüpfen, um Paare von männlichen und weiblichen Tieren derselben Art zusammen zu bringen:

```
mysql> SELECT p1.name, p1.geschlecht, p2.name, p2.geschlecht, p1.art
    -> FROM pet AS p1, pet AS p2
    -> WHERE p1.art = p2.art AND p1.geschlecht = "w" AND p2.geschlecht = "m";
+--------+-------------+--------+-------------+---------+
| name   | geschlecht  | name   | geschlecht  | art     |
+--------+-------------+--------+-------------+---------+
| Fluffy | w           | Claws  | m           | Katze   |
| Buffy  | w           | Fang   | m           | Hund    |
| Buffy  | w           | Bowser | m           | Hund    |
+--------+-------------+--------+-------------+---------+
```

In dieser Anfrage legen wir Aliase für den Tabellennamen fest, um auf die Spalten verweisen zu können und um auseinander zu halten, auf welche Instanz der Tabelle sich jede Spaltenreferenz bezieht.

4.4. Informationen über Datenbanken und Tabellen

Was ist, wenn Sie den Namen einer Datenbank oder Tabelle vergessen haben oder für eine gegebene Tabelle die Struktur nicht mehr kennen (wie zum Beispiel die Spalten heißen)? MySQL löst solcherlei Probleme mit diversen Statements, die Informationen über die Datenbanken und Tabellen bereitstellen, die es unterstützt.

`SHOW DATABASES` kennen Sie schon. Dieses listet die Datenbanken auf, die vom Server verwaltet werden. Um herauszufinden, welche Datenbank aktuell ausgewählt ist, benutzen Sie die `DATABASE()`-Funktion:

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

Wenn Sie noch keine Datenbank ausgewählt haben, ist das Ergebnis leer.

Um herauszufinden, welche Tabellen die aktuelle Datenbank enthält (wenn Sie sich zum Beispiel über den Namen einer Tabelle nicht sicher sind), benutzen Sie folgenden Befehl:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| ereignis            |
| pet                 |
+---------------------+
```

Wenn Sie die Struktur einer Tabelle sehen wollen, ist der `DESCRIBE`-Befehl nützlich. Er zeigt Informationen über jede Tabellenspalte an:

```
mysql> DESCRIBE pet;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| name       | varchar(20) | YES  |     | NULL    |       |
| besitzer   | varchar(20) | YES  |     | NULL    |       |
| art        | varchar(20) | YES  |     | NULL    |       |
| geschlecht | char(1)     | YES  |     | NULL    |       |
| geburtstag | date        | YES  |     | NULL    |       |
| sterbetag  | date        | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

`Field` zeigt den Spaltennamen, `Type` ist der Datentyp der Spalte, `Null` zeigt an, ob die Spalte `NULL`-Werte enthalten darf oder nicht, `Key` zeigt an, ob die Spalte indiziert ist oder nicht und `Default` legt den Vorgabewert der Spalte fest.

Wenn Sie Indexe auf eine Tabelle haben, zeigt Ihnen `SHOW INDEX FROM tabelle` Informationen über diese an.

4.5. Beispiele gebräuchlicher Anfragen (Queries)

Hier finden sich Beispiele, wie geläufige Probleme mit MySQL gelöst werden können.

Einige der Beispiele benutzen die Tabelle `shop`, die den Stückpreis für jeden Artikel für bestimmte Händler enthält. Unter der Annahme, dass jeder Händler einen einzelnen fest Preis pro Artikel hat, ist (`artikel`, `haendler`) der Primärschlüssel für diese Datensätze.

Starten Sie das Kommandozeilen-Werkzeug `mysql` und wählen Sie eine Datenbank aus:

```
mysql ihr-datenbank-name
```

(Bei den meisten MySQL-Installationen können Sie die Datenbank 'test' auswählen.)

Erzeugen Sie die Beispiel-Tabelle wie folgt:

```
CREATE TABLE shop (
 artikel  INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
 haendler CHAR(20)                 DEFAULT ''     NOT NULL,
 preis    DOUBLE(16,2)             DEFAULT '0.00' NOT NULL,
 PRIMARY KEY(artikel, haendler));

INSERT INTO shop VALUES
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
(3,'D',1.25),(4,'D',19.95);
```

Die Beispieldaten sehen jetzt so aus:

```
mysql> SELECT * FROM shop;

+---------+---------+-------+
| artikel | haendler| preis |
+---------+---------+-------+
|    0001 | A       |  3.45 |
|    0001 | B       |  3.99 |
|    0002 | A       | 10.99 |
|    0003 | B       |  1.45 |
|    0003 | C       |  1.69 |
|    0003 | D       |  1.25 |
|    0004 | D       | 19.95 |
+---------+---------+-------+
```

4.5.1. Der höchste Wert einer Spalte

``Was ist die höchste Artikelnummer?''

```
SELECT MAX(artikel) AS artikel FROM shop

+---------+
| artikel |
+---------+
|       4 |
+---------+
```

4.5.2. Die Zeile, die den höchsten Wert einer bestimmten Spalte enthält

``Suche Artikelnummer, Händler und Preis des teuersten Artikels.''

In ANSI-SQL wird das mit einer Unterabfrage (Sub-Query) durchgeführt:

```
SELECT artikel, haendler, preis
FROM   shop
WHERE  preis=(SELECT MAX(preis) FROM shop)
```

In MySQL (was noch keine Unterabfragen hat) führen Sie das in zwei Schritten durch:

1. Mit einem `SELECT`-Statement ermitteln Sie den höchsten Preis in der Tabelle.
2. Mit diesem Wert stellen Sie die aktuelle Anfrage zusammen:

   ```
   SELECT artikel, haendler, preis
   FROM   shop
   WHERE  preis=19.95
   ```

Eine andere Lösung besteht darin, alle Zeilen absteigend nach Preis zu sortieren und nur die erste Zeile zu nehmen, indem Sie die MySQL-spezifische `LIMIT`-Klausel benutzen:

```
SELECT artikel, haendler, preis
FROM   shop
ORDER BY preis DESC
LIMIT 1
```

ACHTUNG: Wenn es mehrere teuerste Artikel gibt (die zum Beispiel alle 19.95 kosten), zeigt die `LIMIT`-Lösung nur einen davon!

4.5.3. Höchster Wert einer Spalte pro Gruppe

``Was ist der höchste Preis pro Artikel?"

```
SELECT artikel, MAX(preis) AS preis
FROM   shop
GROUP BY artikel

+---------+-------+
| artikel | preis |
+---------+-------+
|    0001 |  3.99 |
|    0002 | 10.99 |
|    0003 |  1.69 |
|    0004 | 19.95 |
+---------+-------+
```

4.5.4. Die Zeilen, die das gruppenweise Maximum eines bestimmten Felds enthalten

``Suche für jeden Artikel den oder die Händler mit den teuersten Preisen."

In ANSI-SQL würden Sie das wie folgt mit einer Unterabfrage erledigen:

```
SELECT artikel, haendler, preis
FROM   shop s1
WHERE  preis=(SELECT MAX(s2.preis)
              FROM shop s2
              WHERE s1.artikel = s2.artikel);
```

In MySQL macht man das am besten in mehreren Schritten:

1. Die Liste (artikel,maxpreis) holen.
2. Für jeden Artikel die korrespondierenden Zeilen holen, die den höchsten Preis gespeichert haben.

Das kann auf einfache Weise mit einer temporären Tabelle geschehen:

```
CREATE TEMPORARY TABLE tmp (
        artikel INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
        preis   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL);

LOCK TABLES shop read;

INSERT INTO tmp SELECT artikel, MAX(preis) FROM shop GROUP BY artikel;

SELECT shop.artikel, haendler, shop.preis FROM shop, tmp
WHERE shop.artikel=tmp.artikel AND shop.preis=tmp.preis;

UNLOCK TABLES;

DROP TABLE tmp;
```

Wenn Sie keine `TEMPORARY`-Tabelle benutzen, müssen Sie zusätzlich die 'tmp'-Tabelle sperren.

``Kann das mit einer einzigen Anfrage durchgeführt werden?"

Ja, aber nur unter Verwendung eines recht ineffizienten Tricks, den wir den ``MAX-CONCAT-Trick" nennen:

```
SELECT artikel,
       SUBSTRING( MAX( CONCAT(LPAD(preis,6,'0'),haendler) ), 7) AS haendler,
  0.00+LEFT(      MAX( CONCAT(LPAD(preis,6,'0'),haendler) ), 6) AS preis
FROM   shop
GROUP BY artikel;
```

```
+---------+---------+-------+
| artikel | haendler| preis |
+---------+---------+-------+
|    0001 | B       |  3.99 |
|    0002 | A       | 10.99 |
|    0003 | C       |  1.69 |
|    0004 | D       | 19.95 |
+---------+---------+-------+
```

Das letzte Beispiel kann etwas effizienter gemacht werden, wenn man das Aufteilen der verketteten Spalte im Client durchführt.

4.5.5. Wie Benutzer-Variablen verwendet werden

Sie können MySQL-Benutzer-Variablen verwenden, um Ergebnisse zwischenzuspeichern, ohne sie in temporäre Variablen im Client speichern zu müssen. See Abschnitt 7.1.4, „Benutzer-Variablen“.

Um zum Beispiel die Artikel mit dem höchsten und dem niedrigsten Preis herauszufinden, können Sie folgendes machen:

```
select @min_preis:=min(preis),@max_preis:=max(preis) from shop;
select * from shop where preis=@min_preis or preis=@max_preis;

+---------+---------+-------+
| artikel | haendler| preis |
+---------+---------+-------+
|    0003 | D       |  1.25 |
|    0004 | D       | 19.95 |
+---------+---------+-------+
```

4.5.6. Wie Fremdschlüssel (Foreign Keys) verwendet werden

Sie brauchen keine Fremdschlüssel, um zwei Tabellen zu verknüpfen.

Das einzige, was MySQL nicht durchführt, ist der `CHECK`, um sicherzustellen, dass die Schlüssel, die Sie benutzen, in der oder den Tabelle(n) existieren, auf die Sie verweisen, und es löscht auch nicht automatisch Zeilen aus einer Tabelle mit einer Fremdschlüssel-Definition. Wenn Sie Ihre Schlüssel wie gewöhnlich benutzen, funktioniert das gut:

```
CREATE TABLE personen (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE hemden (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    stil ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    farbe ENUM('rot', 'blau', 'orange', 'weiß', 'schwarz') NOT NULL,
    besitzer SMALLINT UNSIGNED NOT NULL references personen,
    PRIMARY KEY (id)
);


INSERT INTO personen VALUES (NULL, 'Antonio Paz');

INSERT INTO hemden VALUES
(NULL, 'polo', 'blau', LAST_INSERT_ID()),
(NULL, 'dress', 'weiß', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blau', LAST_INSERT_ID());


INSERT INTO personen VALUES (NULL, 'Lilliana Angelovska');

INSERT INTO hemden VALUES
(NULL, 'dress', 'orange', LAST_INSERT_ID()),
(NULL, 'polo', 'rot', LAST_INSERT_ID()),
(NULL, 'dress', 'blau', LAST_INSERT_ID()),
(NULL, 't-shirt', 'weiß', LAST_INSERT_ID());


SELECT * FROM personen;
+----+---------------------+
| id | name                |
+----+---------------------+
|  1 | Antonio Paz         |
|  2 | Lilliana Angelovska |
+----+---------------------+

SELECT * FROM hemden;
+----+---------+--------+----------+
| id | stil    | farbe  | besitzer |
+----+---------+--------+----------+
|  1 | polo    | blau   |     1    |
|  2 | dress   | weiß   |     1    |
|  3 | t-shirt | blau   |     1    |
|  4 | dress   | orange |     2    |
|  5 | polo    | rot    |     2    |
|  6 | dress   | blau   |     2    |
```

```
|  7 | t-shirt | weiß  |     2    |
+----+---------+--------+----------+


SELECT h.* FROM personen p, hemden h
 WHERE p.name LIKE 'Lilliana%'
   AND h.besitzer = p.id
   AND h.farbe <> 'weiß';

+----+-------+--------+----------+
| id | stil  | farbe  | besitzer |
+----+-------+--------+----------+
|  4 | dress | orange |     2    |
|  5 | polo  | rot    |     2    |
|  6 | dress | blau   |     2    |
+----+-------+--------+----------+
```

4.5.7. Über zwei Schlüssel suchen

MySQL optimiert derzeit noch nicht, wenn Sie über zwei unterschiedliche Schlüssel suchen, die mit `OR` kombiniert werden (eine Suche mit einem Schlüssel mit verschiedenen `OR`-Teilen wird recht gut optimiert):

```
SELECT feld1_index, feld2_index FROM test_tabelle WHERE feld1_index = '1'
OR feld2_index = '1'
```

Der Grund liegt darin, dass wir bislang noch keine Zeit hatten, hierfür eine effiziente Möglichkeit zu implementieren, die das für allgemeine Fälle abhandelt. (Die `AND`-Handhabung ist im Vergleich jetzt komplett allgemein und funktioniert sehr gut.)

In der Zwischenzeit können Sie dieses Problem sehr effizient lösen, indem Sie eine `TEMPORARY`-Tabelle verwenden. Diese Art der Optimierung ist ebenfalls sehr gut, wenn Sie sehr komplizierte Anfragen verwenden, bei denen der SQL-Server die Optimierungen in falscher Reihenfolge durchführt.

```
CREATE TEMPORARY TABLE tmp
SELECT feld1_index, feld2_index FROM test_tabelle WHERE feld1_index = '1';
INSERT INTO tmp
SELECT feld1_index, feld2_index FROM test_tabelle WHERE feld2_index = '1';
SELECT * from tmp;
DROP TABLE tmp;
```

Diese Möglichkeit der Anfrage ist im Endeffekt ein `UNION` von zwei Anfragen.

4.5.8. Besuche pro Tag berechnen

Folgendes zeigt, wie Sie die Bit-Gruppen-Funktionen benutzen können, um die Anzahl der Tage pro Monat zu zählen, in denen ein Benutzer eine Web-Seite besucht hat.

```
CREATE TABLE t1 (jahr YEAR(4), monat INT(2) UNSIGNED ZEROFILL, tag INT(2) UNSIGNED ZEROFILL);
INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),(2000,2,23),(2000,2,23);

SELECT jahr,monat,BIT_COUNT(BIT_OR(1<<tag)) AS tage FROM t1 GROUP BY jahr,monat;

Das gibt folgendes Ergebnis zurück:

+------+-------+------+
| jahr | monat | tage |
+------+-------+------+
| 2000 |    01 |    3 |
| 2000 |    02 |    2 |
+------+-------+------+
```

Dies berechnet, wie viele verschiedene Tage für eine gegebene Jahr-Monats-Kombination benutzt wurden, bei automatischer Entfernung doppelter Einträge (Duplikate).

4.6. `mysql` im Stapelbetrieb (Batch Mode)

In den vorherigen Abschnitten haben Sie `mysql` interaktiv benutzt, um Anfragen einzugeben und die Ergebnisse zu betrachten. Sie können `mysql` auch im Stapelbetrieb benutzen. Dafür schreiben Sie dei Befehle, die Sie ausführen wollen, in eine Datei, und teilen `mysql` dann mit, seine Eingaben aus dieser Datei zu lesen:

```
shell> mysql < stapel-datei
```

Wenn Sie auf der Kommandozeile Verbindungsparameter angeben müssen, könnte der Befehl wie folgt aussehen:

```
shell> mysql -h host -u user -p < stapel-datei
Enter password: ********
```

Wenn Sie `mysql` auf diese Weise benutzen, erzeugen Sie eine Skript-Datei und führen dann das Skript aus.

Warum sollten Sie ein Skript benutzen? Hier sind ein paar Gründe:

- Wenn Sie eine Anfrage wiederholt ausführen (sagen wir jeden Tag oder jede Woche), vermeiden Sie mit einem Skript, dass Sie sie jedes Mal zur Ausführung erneut eintippen müssen.
- Sie können aus existierenden Anfragen neue Anfragen erzeugen, die ähnlich sind, indem Sie die Skript-Dateien kopieren und editieren.
- Der Stapelbetrieb kann auch für die Entwicklung einer Anfrage nützlich sein, insbesondere, wenn Sie mehrzeilige Befehle oder Befehlssequenzen aus mehreren Statements entwickeln. Wenn Sie einen Fehler machen, müssen Sie nicht alles noch einmal tippen, sondern editieren einfach Ihr Skript, um den Fehler zu beheben, und weisen `mysql` an, es erneut auszuführen.
- Wenn Sie eine Anfrage haben, die eine größere Ausgabe erzeugt, können Sie die Ausgabe durch einen Pager laufen lassen, statt zuzusehen, wie Sie über den Bildschirm flimmert:

  ```
  shell> mysql < stapel-datei | more
  ```

- Für weitere Verarbeitung können Sie die Ausgabe auch in eine Datei lenken:

  ```
  shell> mysql < stapel-datei > mysql.ausgabe
  ```

- Sie können Ihr Skript an andere Leute verteilen, so dass auch sie die Befehle laufen lassen können.
- In einigen Situationen ist interaktive Benutzung nicht angebracht, zum Beispiel dann, wenn Sie eine Anfrage durch einen `cron`-Job ausführen lassen. In diesem Fall brauchen Sie Stapelbetrieb.

Das Standard-Ausgabeformat ist anders (präziser), wenn Sie `mysql` im Stapelbetrieb laufen lassen, als wenn Sie es interaktiv nutzen. Die Ausgabe von `SELECT DISTINCT art FROM pet` zum Beispiel sieht so aus, wenn Sie sie interaktiv laufen lassen:

```
+---------+
| art     |
+---------+
| Vogel   |
| Katze   |
| Hund    |
| Hamster |
| Schlange|
+---------+
```

Aber wie folgt, wenn sie im Stapelbetrieb läuft:

```
art
Vogel
Katze
Hund
Hamster
Schlange
```

Wenn Sie im Stapelbetrieb das interaktive Ausgabeformat haben wollen, benutzen Sie `mysql -t`. Um die Befehle auszugeben, die ausgeführt werden, benutzen Sie `mysql -vvv`.

4.7. Anfragen aus dem Zwillings-Projekt

Bei Analytikerna und Lentus haben wir die Systeme und die Feldarbeit für ein großes Forschungsprojekt gemacht. Dieses Projekt ist eine Zusammenarbeit zwischen dem Institut für Umweltmedizin des Karolinska Institutes, Stockholm, und der Abteilung für klinische Forschung bei Altersprozessen und Psychologie der University of Southern California.

Das Projekt beinhaltet einen Screening-Teil, bei dem alle Zwillinge in Schweden, die älter als 65 Jahre sind, per Telefon interviewt wurden. Zwillinge, die bestimmte Kriterien erfüllen, werden im nächsten Schritt weiter untersucht. In diesem späteren Stadium werden Zwillinge, die teilnehmen wollen, von einem Arzt-Schwester-Team besucht. Einige Untersuchungen beinhalten physische und neuropsychologische Untersuchungen, Labortests, Neuroimaging, Bewertungen des psychischen Zustands und eine Sammlung der Familiengeschichten. Zusätzlich werden Daten über medizinische und umweltbedingte Risikofaktoren gesammelt.

Weitere Informationen zu den Zwillingsstudien finden Sie hier:

```
http://www.imm.ki.se/TWIN/TWINGREATBRITAINW.HTM
```

Der spätere Teil des Projekts wird mit einer Web-Schnittstelle verwaltet, die Perl und MySQL benutzt.

Jeden Abend werden alle Daten der Interviews in eine MySQL-Datenbank verschoben.

4.7.1. Alle nicht verteilten Zwillinge finden

Mit folgender Anfrage wird festgelegt, wer in den zweiten Teil des Projekts geht:

```
select
        concat(p1.id, p1.tvab) + 0 as tvid,
        concat(p1.christian_name, " ", p1.surname) as Name,
        p1.postal_code as Code,
        p1.city as City,
        pg.abrev as Area,
        if(td.participation = "Aborted", "A", " ") as A,
        p1.dead as dead1,
        l.event as event1,
        td.suspect as tsuspect1,
        id.suspect as isuspect1,
        td.severe as tsevere1,
        id.severe as isevere1,
        p2.dead as dead2,
        l2.event as event2,
        h2.nurse as nurse2,
        h2.doctor as doctor2,
        td2.suspect as tsuspect2,
        id2.suspect as isuspect2,
        td2.severe as tsevere2,
        id2.severe as isevere2,
        l.finish_date
from
        twin_project as tp
        /* For Twin 1 */
        left join twin_data as td on tp.id = td.id and tp.tvab = td.tvab
        left join informant_data as id on tp.id = id.id and tp.tvab = id.tvab
        left join harmony as h on tp.id = h.id and tp.tvab = h.tvab
        left join lentus as l on tp.id = l.id and tp.tvab = l.tvab
        /* For Twin 2 */
        left join twin_data as td2 on p2.id = td2.id and p2.tvab = td2.tvab
        left join informant_data as id2 on p2.id = id2.id and p2.tvab = id2.tvab
        left join harmony as h2 on p2.id = h2.id and p2.tvab = h2.tvab
        left join lentus as l2 on p2.id = l2.id and p2.tvab = l2.tvab,
        person_data as p1,
        person_data as p2,
        postal_groups as pg
where
        /* p1 gets main twin and p2 gets his/her twin. */
        /* ptvab is a field inverted by tvab */
        p1.id = tp.id and p1.tvab = tp.tvab and
        p2.id = p1.id and p2.ptvab = p1.tvab and
        /* Just the sceening survey */
        tp.survey_no = 5 and
        /* Skip if partner died before 65 but allow emigration (dead=9) */
        (p2.dead = 0 or p2.dead = 9 or
         (p2.dead = 1 and
          (p2.sterbetag_date = 0 or
           (((to_days(p2.sterbetag_date) - to_days(p2.geburtstagday)) / 365)
            >= 65))))
        and
        (
        /* Twin is suspect */
        (td.future_contact = 'Yes' and td.suspect = 2) or
        /* Twin is suspect - Informant is Blessed */
        (td.future_contact = 'Yes' and td.suspect = 1 and id.suspect = 1) or
        /* No twin - Informant is Blessed */
        (ISNULL(td.suspect) and id.suspect = 1 and id.future_contact = 'Yes') or
        /* Twin broken off - Informant is Blessed */
        (td.participation = 'Aborted'
         and id.suspect = 1 and id.future_contact = 'Yes') or
        /* Twin broken off - No inform - Have partner */
        (td.participation = 'Aborted' and ISNULL(id.suspect) and p2.dead = 0))
        and
        l.event = 'Finished'
        /* Get at area code */
        and substring(p1.postal_code, 1, 2) = pg.code
        /* Not already distributed */
        and (h.nurse is NULL or h.nurse=00 or h.doctor=00)
        /* Has not refused or been aborted */
        and not (h.status = 'Refused' or h.status = 'Aborted'
        or h.status = 'Died' or h.status = 'Other')
order by
        tvid;
```

Einige Erläuterungen:

- `concat(p1.id, p1.tvab) + 0 as tvid`

  Wir wollen nach den verketteten `id` und `tvab` in numerischer Reihenfolge sortieren. Indem wir `0` hinzufügen, bringen wir

MySQL dazu, das Ergebnis als Zahl zu behandeln.

- Spalte `id`

  Diese identifiziert ein Zwillingspaar. Sie ist in allen Tabellen Schlüssel.

- Spalte `tvab`

  Diese identifiziert ein Zwillingspaar. Sie hat einen Wert von `1` oder `2`.

- Spalte `ptvab`

  Sie ist die Umkehrung von `tvab`. Wenn `tvab 1` ist, ist sie `2`, und umgekehrt. Sie ist dafür da, MySQL die Optimierung der Anfrage zu erleichtern.

Diese Anfrage demonstriert unter anderem, wie man ein Nachschlagen (Lookup) in einer Tabelle von derselben Tabelle aus mit einem Join durchführt (`p1` und `p2`). In dem Beispiel wird das dazu benutzt, um festzustellen, ob der Partner eines Zwillings vor Erreichen des 65. Lebensjahrs starb. Wenn das der Fall ist, wird die Zeile nicht zurückgegeben.

Das Geschilderte existiert in allen Tabellen mit zwillingsbezogenen Informationen. Wir haben einen Schlüssel auf beide `id,tvab` (alle Tabellen), und auf `id,ptvab` (`person_data`), um Anfragen schneller zu machen.

Auf unserer Produktionsmaschine (einer 200MHz-UltraSPARC) gibt diese Anfrage etwa 150 bis 200 Zeilen zurück und benötigt weniger als eine Sekunde.

Die aktuelle Anzahl von Datensätzen in den oben benutzten Tabellen:

Tabelle	Zeilen
`person_data`	71074
`lentus`	5291
`twin_project`	5286
`twin_data`	2012
`informant_data`	663
`harmony`	381
`postal_groups`	100

4.7.2. Eine Tabelle über den Zustand von Zwillingspaaren zeigen

Jedes Interview endet mit einem Statuscode, genannt `ereignis`. Die unten stehende Anfrage wird benutzt, um eine Tabelle über alle Zwillingspaare anzuzeigen, kombiniert mit dem Ereignis. Das zeigt an, wie viele Paare beider Zwillingen im Zustand beendet sind, bei wie vielen Paaren ein Zwilling im Zustand beendet ist, welche ein Interview abgelehnt haben usw.

```
select
        t1.event,
        t2.event,
        count(*)
from
        lentus as t1,
        lentus as t2,
        twin_project as tp
where
        /* We are looking at one pair at a time */
        t1.id = tp.id
        and t1.tvab=tp.tvab
        and t1.id = t2.id
        /* Just the sceening survey */
        and tp.survey_no = 5
        /* This makes each pair only appear once */
        and t1.tvab='1' and t2.tvab='2'
group by
        t1.event, t2.event;
```

4.8. MySQL mit Apache benutzen

Der Contrib-Abschnitt beinhaltet Programme, mit denen Sie Ihre Benutzer durch eine MySQL-Datenbank authentifizieren können, und mit denen Sie Ihre Logdateien in eine MySQL-Tabelle schreiben können.

Sie können das Log-Format von Apache so ändern, dass es durch MySQL leicht gelesen werden kann, indem Sie folgendes in die

Apache-Konfigurationsdatei schreiben:

```
LogFormat \
        "\"%h\",%{%Y%m%d%H%M%S}t,%>s,\"%b\",\"%{Content-Type}o\",  \
        \"%U\",\"%{Referer}i\",\"%{User-Agent}i\""
```

In MySQL können Sie dann etwas wie das hier tun:

```
LOAD DATA INFILE '/local/access_log' INTO TABLE tabelle
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```

Kapitel 5. MySQL-Datenbankadministration

5.1. MySQL konfigurieren

5.1.1. mysqld-Kommandozeilenoptionen

`mysqld` akzeptiert folgende Kommandozeilenoptionen:

- `--ansi`

  ANSI-SQL-Syntax anstelle von MySQL-Syntax benutzen. See Abschnitt 2.7.2, „MySQL im ANSI-Modus laufen lassen“.

- `-b, --basedir=path`

  Pfad zum Installationsverzeichnis. Gewöhnlich werden alle Pfade relativ zu diesem aufgelöst.

- `--big-tables`

  große Ergebnismengen zulassen, indem alle temporären Mengen in eine Datei gesichert werden. Das löst die meisten 'table full'-Fehler, verlangsamt aber in den Fällen Anfragen, in denen Tabellen im Speicher ausreichen würden. Ab Version 3.23.2 ist MySQL in der Lage, das automatisch zu lösen, indem für kleine temporäre Tabellen der Arbeitsspeicher benutzt wird und auf Festplatten-Tabellen umgeschaltet wird, wenn das nötig ist.

- `--bind-address=IP`

  IP-Adresse zum Anbinden (bind).

- `--character-sets-dir=path`

  Verzeichnis, wo Zeichensätze sind. See Abschnitt 5.6.1, „Der für Daten und Sortieren benutzte Zeichensatz“.

- `--chroot=path`

  Chroot den `mysqld`-Daemon beim Start. Empfohlene Sicherheitsmaßnahme. Wird allerdings `LOAD DATA INFILE` und `SELECT ... INTO OUTFILE` etwas einschränken.

- `--core-file`

  Schreibt eine Core-Datei, wenn `mysqld` stirbt. Auf manchen Systemen müssen Sie zusätzliche `--core-file-size` für `safe_mysqld` angeben. See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.

- `-h, --datadir=path`

  Pfad zum Datenbank-Wurzelverzeichnis.

- `--default-character-set=charset`

  Setzt den vorgabemäßigen Zeichensatz. See Abschnitt 5.6.1, „Der für Daten und Sortieren benutzte Zeichensatz“.

- `--default-table-type=type`

  Setzt den vorgabemäßigen Tabellentyp für Tabellen. See Kapitel 8, *MySQL-Tabellentypen*.

- `--debug[...]=`

  Wenn MySQL mit `--with-debug` konfiguriert ist, können Sie diese Option benutzen, um eine Trace-Datei darüber zu erhalten, was `mysqld` tut. See Abschnitt E.1.2, „Trace-Dateien erzeugen“.

- `--delay-key-write-for-all-tables`

  Schlüsselpuffer (Key Buffer) für jegliche `MyISAM`-Tabellen nicht leeren (flush). See Abschnitt 6.5.2, „Serverparameter tunen“.

- `--enable-locking`

  System-Sperren einschalten. Beachten Sie, dass Sie bei der Benutzung dieser Option auf Systemen, die kein voll funktionsfähiges lockd() besitzen (wie Linux), mysqld leicht zum Deadlock bringen können.

- `-T, --exit-info`

Eine Bit-Maske verschiedener Flags, mit denen man den mysqld-Server debuggen kann. Man sollte diese Option nicht benutzen, wenn man nicht ganz genau weiß, was sie tut!

- `--flush`

  Alle Änderungen nach jedem SQL-Befehl auf Platte zurückschreiben (flush). Normalerweise schreibt MySQL alle Änderungen nach jedem SQL-Befehl auf Platte und läßt das Betriebssystem sich um das Synchronisieren auf Platte kümmern. See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“.

- `-?, --help`

  Kurze Hilfe ausgeben und beenden.

- `--init-file=file`

  Beim Start SQL-Befehle aus dieser Datei lesen.

- `-L, --language=...`

  Client-Fehlermeldungen in der angegebenen Sprache. Kann als voller Pfad angegeben werden. See Abschnitt 5.6.2, „Nicht englische Fehlermeldungen“.

- `-l, --log[=datei]`

  Loggt Verbindungen und Anfragen in datei. See Abschnitt 5.9.2, „Die allgemeine Anfragen-Log-Datei“.

- `--log-isam[=datei]`

  Loggt alle ISAM- / MyISAM-Änderungen in datei (wird nur benutzt, um ISAM / MyISAM zu debuggen).

- `--log-slow-queries[=datei]`

  Loggt alle Anfragen, die länger als `long_query_time` Sekunden für die Ausführung benötigt haben, in datei. See Abschnitt 5.9.5, „Die Anfragen-Log-Datei für langsame Anfragen“.

- `--log-update[=datei]`

  Loggt Updates in `datei.#`, wobei # eine eindeutige Zahl ist, falls nicht vorgegeben. See Abschnitt 5.9.3, „Die Update-Log-Datei“.

- `--log-long-format`

  Loggt einige zusätzliche Informationen ins Update-Log. Wenn Sie `--log-slow-queries` benutzen, werden Anfragen, die keine Indexe benutzen, in die Langsame-Anfragen-Log-Datei geloggt.

- `--low-priority-updates`

  Operationen, die Tabellen ändern (`INSERT`/`DELETE`/`UPDATE`), haben geringere Priorität als Selects. Das kann auch mit `{INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ...` durchgeführt werden, um lediglich die Priorität einer einzelnen Anfrage zu verringern, oder mit `SET OPTION SQL_LOW_PRIORITY_UPDATES=1`, um die Priorität in einem Thread zu ändern. See Abschnitt 6.3.2, „Themen, die Tabellensperren betreffen“.

- `--memlock`

  Sperrt den `mysqld`-Prozess in den Arbeitsspeicher. Das funktioniert nur, wenn Ihr System den `mlockall()`-Systemaufruf versteht (wie Solaris). Das kann helfen, wenn Sie Probleme damit haben, dass Ihr Betriebssystem `mysqld` veranlasst, auf Platte zu swappen.

- `--myisam-recover [=option[,option...]]], wobei option eine`

  Kombination von `DEFAULT`, `BACKUP`, `FORCE` oder `QUICK` ist. Sie können sie auch explizit auf `""` setzen, wenn Sie diese Option ausschalten wollen. Wenn die Option benutzt wird, überprüft `mysqld` beim Öffnen, ob die Tabelle als zerstört markiert ist oder ob die Tabelle nicht ordnungsgemäß geschlossen wurde. (Die letzte Option funktioniert nur, wenn Sie mysqld mit `--skip-locking` laufen lassen). Wenn das der Fall ist, läßt `mysqld` eine Überprüfung der Tabelle laufen. Wenn die Tabelle beschädigt war, versucht `mysqld`, sie zu reparieren.

  Folgende Optionen beeinflussen, wie repair funktioniert.

DEFAULT	Dasselbe, als würde man für `--myisam-recover` keine Option angeben.
BACKUP	Wenn die Tabelle während der Wiederherstellung geändert wurde, eine Datensicherung

	der `tabelle.MYD`-Datendatei als `tabelle-datetime.BAK` speichern.
FORCE	Eine Wiederherstellung selbst dann laufen lassen, wenn man mehr als eine Zeile aus der .MYD-Datei verlieren wird.
QUICK	Die Zeilen der Tabelle nicht überprüfen, wenn es keine gelöschten Blocks gibt.

Bevor eine Tabelle automatisch repariert wird, fügt MySQL darüber eine Bemerkung in das Fehler-Log. Wenn Sie in der Lage sein wollen, die meisten Sachen ohne Benutzer-Intervention zu beheben, sollten Sie die Optionen `BACKUP,FORCE` benutzen. Das erzwingt ein Reparieren einer Tabelle, selbst wenn dabei einige Zeilen gelöscht würden, erhält aber die alte Datendatei als Datensicherung, so dass Sie später herausfinden können, was passiert ist.

- `--pid-file=pfad`

  Pfad zur pid-Datei, die von `safe_mysqld` benutzt wird.

- `-P, --port=...`

  Port-Nummer, um auf TCP/IP-Verbindungen zu warten (listen).

- `-o, --old-protocol`

  Das 3.20-Protokoll für Kompatibilität mit einigen sehr alten Clients benutzen.

- `--one-thread`

  Nur einen Thread benutzen (zum Debuggen unter Linux). See Abschnitt E.1, „Einen MySQL-Server debuggen“.

- `-O, --set-variable var=option`

  Weist einer Variablen einen Wert zu. `--help` listet Variablen auf. Sie finden eine komplette Beschreibung aller Variablen im `SHOW VARIABLES`-Abschnitt dieses Handbuchs. See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“. Der Abschnitt über das Tunen der Serverparameter enthält Informationen darüber, wie man diese optimiert. See Abschnitt 6.5.2, „Serverparameter tunen“.

- `--safe-mode`

  Einige Optimierungsschritte überspringen. Setzt `--skip-delay-key-write` voraus.

- `--safe-show-database`

  Keine Datenbanken anzeigen, für die der Benutzer keine Zugriffsrechte hat.

- `--safe-user-create`

  Wenn das angeschaltet ist, kann ein Benutzer keine neuen Benutzer mit dem GRANT-Befehl anlegen, wenn der Benutzer kein `INSERT`-Zugriffsrecht auf die `mysql.user`-Tabelle oder irgend welche Spalten dieser Tabelle hat.

- `--skip-concurrent-insert`

  Die Fähigkeit abschalten, gleichzeitig auf `MyISAM`-Tabellen auszuwählen (select) und einzufügen (insert). (Sollte nur benutzt werden, wenn Sie der Meinung sind, ein Bug in diesem Feature gefunden zu haben.)

- `--skip-delay-key-write`

  Die `delay_key_write`-Option für alle Tabellen ignorieren. See Abschnitt 6.5.2, „Serverparameter tunen“.

- `--skip-grant-tables`

  Diese Option veranlasst den Server, das Zugriffsrechte-System überhaupt nicht zu benutzen. Das gibt jedem *vollen Zugriff* auf alle Datenbanken! (Einen laufenden Server können Sie anweisen, die Berechtigungstabellen erneut zu verwenden, indem Sie `mysqladmin flush-privileges` oder `mysqladmin reload` ausführen.)

- `--skip-host-cache`

  Nie den Host-Name-Cache für schnellere Name-IP-Auflösung benutzen, sondern statt dessen bei jeder Verbindung beim DNS-Server anfragen. See Abschnitt 6.5.5, „Wie MySQL DNS benutzt“.

- `--skip-locking`

System-Sperren nicht benutzen. Um `isamchk` oder `myisamchk` auszuführen, müssen Sie den Server herunter fahren. See Abschnitt 2.2.2, „Wie stabil ist MySQL?“. Beachten Sie, dass Sie in MySQL-Version 3.23 `REPAIR` und `CHECK` benutzen können, um `MyISAM`-Tabellen zu reparieren / zu prüfen.

- `--skip-name-resolve`

  Hostnamen werden nicht aufgelöst. Alle `Host`-Spaltenwerte in den Berechtigungstabellen müssen IP-Nummern oder `localhost` sein. See Abschnitt 6.5.5, „Wie MySQL DNS benutzt“.

- `--skip-networking`

  Auf überhaupt keine TCP/IP-Verbindungen warten (listen). Jede Interaktion mit `mysqld` muss über Unix-Sockets erfolgen. Diese Option wird ausdrücklich empfohlen für Systeme, auf denen nur lokale Anfragen (Requests) erlaubt sind. See Abschnitt 6.5.5, „Wie MySQL DNS benutzt“.

- `--skip-new`

  Keine neuen, möglicherweise falschen Routinen benutzen. Setzt `--skip-delay-key-write` voraus. Setzt ausserdem den vorgabemäßigen Tabellentyp auf `ISAM`. See Abschnitt 8.3, „ISAM-Tabellen“.

- `--skip-symlink`

  Keine Dateien löschen oder umbenennen, auf die eine mit Symlink verknüpfte Datei im Daten-Verzeichnis zeigt.

- `--skip-safemalloc`

  Wenn MySQL mit `--with-debug=full` konfiguriert wird, überprüfen alle Programme den Arbeitsspeicher auf Überlauf, bei jeder Speicher-Allokation und -Freigabe. Da dieses Prüfen sehr langsam ist, können Sie es vermeiden, wenn Sie keine Arbeitsspeicherprüfung benötigten, indem Sie diese Option benutzen.

- `--skip-show-database`

  Keine 'SHOW DATABASE'-Befehle zulassen, wenn der Benutzer keine **process**-Berechtigung hat.

- `--skip-stack-trace`

  Keine Stack-Traces schreiben. Diese Option ist nützlich, wenn Sie `mysqld` unter einem Debugger laufen lassen. See Abschnitt E.1, „Einen MySQL-Server debuggen“.

- `--skip-thread-priority`

  Benutzung von Thread-Prioritäten abschalten, um schnellere Antwortzeiten zu erzielen.

- `--socket=pfad`

  Socket-Datei, die anstelle des vorgabemäßigen `/tmp/mysql.sock` für lokale Verbindungen benutzt wird.

- `--sql-mode=option[,option[,option...]]`

  Option kann jede beliebige Kombination von `REAL_AS_FLOAT`, `PIPES_AS_CONCAT`, `ANSI_QUOTES`, `IGNORE_SPACE`, `SERIALIZE` und `ONLY_FULL_GROUP_BY` sein. Sie kann auch leer sein (`""`), wenn Sie dies zurücksetzen wollen.

  Alle oben angegebenen Optionen festlegen ist dasselbe wie --ansi benutzen. Mit dieser Option kann man nur benötigte SQL-Modi anschalten. See Abschnitt 2.7.2, „MySQL im ANSI-Modus laufen lassen“.

- `transaction-isolation= { READ-UNCOMMITTED | READ-COMMITTED | REPEATABLE-READ | SERIALIZABLE }`

  Setzt das vorgabemäßige Transaktions-Isolations-Level.

  See Abschnitt 7.7.3, „`SET TRANSACTION`-Syntax“.

- `-t, --tmpdir=pfad`

  Pfad für temporäre Dateien. Es kann nützlich sein, wenn Ihr vorgabemäßiges `/tmp`-Verzeichnis auf einer Partition liegt, die zu klein ist, um temporäre Tabellen zu speichern.

- `-u, --user=benutzername`

  Den `mysqld`-Daemon unter dem Benutzer `benutzername` laufen lassen. Diese Option ist *zwingend notwendig*, wenn `mysqld` als Root gestartet wird.

- `-V, --version`

  Versionsinformationen ausgeben und beenden.

- `-W, --warnings`

  Warnmeldungen wie `Aborted connection...` in die `.err`-Datei ausgeben. See Abschnitt A.2.9, „Kommunikationsfehler / Abgebrochene Verbindung“.

5.1.2. my.cnf-Optionsdateien

Seit Version 3.22 kann MySQL vorgabemäßige Startoptionen für den Server und für Clients aus Optionsdateien lesen.

MySQL liest Vorgabeoptionen aus folgenden Dateien unter Unix:

Dateiname	Zweck
`/etc/my.cnf`	Globale Optionen
`DATADIR/my.cnf`	Server-spezifische Optionen
`defaults-extra-file`	Die Datei, die mit --defaults-extra-file=# festgelegt wird
`~/.my.cnf`	Benutzerspezifische Optionen

`DATADIR` ist das MySQL-Daten-Verzeichnis (typischerweise `/usr/local/mysql/data` bei einer Binärinstallation oder `/usr/local/var` bei einer Quellinstallation). Beachten Sie, dass das das Verzeichnis ist, das zur Konfigurationszeit festgelegt wurde, nicht das, das mit `--datadir` festgelegt wird, wenn `mysqld` startet! (`--datadir` hat keinen Einfluss darauf, wo der Server nach Optionsdateien sucht, denn er sucht nach ihnen, bevor er irgend welche Kommandozeilenargumente verarbeitet.)

MySQL liest Vorgabeoptionen aus folgenden Dateien unter Windows:

Dateiname	Zweck
`Windows-System-Verzeichnis\my.ini`	Globale Optionen
`C:\my.cnf`	Globale Optionen
`C:\mysql\data\my.cnf`	Server-spezifische Optionen

Beachten Sie, dass Sie unter Windows alle Pfade mit `/` statt mit `\` angeben sollten. Wenn Sie `\` benutzen, müssen Sie das doppelt (`\\`) tun, weil `\` in MySQL das Fluchtzeichen (Escape-Character) ist.

MySQL versucht, Optionsdateien in der oben angegebenen Reihenfolge zu lesen. Wenn es mehrere Optionsdateien gibt, erlangt eine Option, die in einer Datei festgelegt wird, die später gelesen wird, Vorrang über dieselbe Option, die in einer sonstigen Optionsdatei festgelegt wurde. Optionen, die auf der Kommandozeile festgelegt werden, erlangen Vorrang vor Optionen in jeglichen Optionsdateien. Einige Optionen können durch Umgebungsvariablen festgelegt werden. Optionen, die auf der Kommandozeile oder in Optionsdateien festgelegt werden, haben Vorrang vor Werten in Umgebungsvariablen. See Anhang F, *Umgebungsvariablen*. Folgende Programme unterstützen Optionsdateien: `mysql`, `mysqladmin`, `mysqld`, `mysqldump`, `mysqlimport`, `mysql.server`, `myisamchk` und `myisampack`.

Sie können Optionsdateien benutzen, um jede beliebig lange Option festzulegen, die ein Programm unterstützt! Starten Sie das Programm mit `--help`, um eine Liste der verfügbaren Optionen zu erhalten.

Eine Optionsdatei kann Zeilen der folgenden Formate enthalten:

- `#Kommentar`

  Kommentarzeilen fangen mit ‘`#`’ oder ‘`;`’ an. Leere Zeilen werden ignoriert.

- `[group]`

  `group` ist der Name des Programms oder der Gruppe, für das oder die Sie Optionen setzen wollen. Nach einer Gruppen-Zeile beziehen sich alle `option`- oder `set-variable`-Zeilen auf die benannte Gruppe, bis zum Ende der Optionsdatei oder bis eine andere Gruppe angegeben wird.

- `option`

  Das ist äquivalent zu `--option` auf der Kommandozeile.

- `option=value`

  Das ist äquivalent zu `--option=value` auf der Kommandozeile.

- `set-variable = variable=value`

  Das ist äquivalent zu `--set-variable variable=value` auf der Kommandozeile. Diese Syntax muss verwendet werden, um eine `mysqld`-Variable zu setzen.

Die `client`-Gruppe gestattet Ihnen, Optionen anzugeben, die sich auf alle MySQL-Clients (nicht auf `mysqld`) beziehen. Diese Gruppe eignet sich bestens dafür, das Passwort festzulegen, das Sie benutzen, um sich mit dem Server zu verbinden. (Stellen Sie jedoch sicher, dass die Optionsdatei nur für Sie les- und schreibbar ist.)

Beachten Sie, dass bei Optionen und Werten alle führenden Leerzeichen und solche am Zeilenende automatisch entfernt werden. Sie können in der Zeichenkette für den Wert die Escape-Sequenzen '`\b`', '`\t`', '`\n`', '`\r`', '`\\`' und '`\s`' benutzen ('`\s`' ist das Leerzeichen).

Hier ist eine typische globale Optionsdatei:

```
[client]
port=3306
socket=/tmp/mysql.sock

[mysqld]
port=3306
socket=/tmp/mysql.sock
set-variable = key_buffer_size=16M
set-variable = max_allowed_packet=1M

[mysqldump]
quick
```

Hier ist eine typische Benutzer-Optionsdatei:

```
[client]
# Folgendes Passwort wird an alle Standard-MySQL-Clients geschickt:
password=mein_password

[mysql]
no-auto-rehash
set-variable = connect_timeout=2

[mysqlhotcopy]
interactive-timeout
```

Wenn Sie eine Quelldistribution haben, finden Sie Beispielkonfigurationen in den Dateien mit Namen `my-xxxx.cnf` im `Support-files`-Verzeichnis. Wenn Sie eine Binärdistribution haben, suchen Sie im `DIR/support-files`-Verzeichnis, wobei `DIR` der Pfadname zum MySQL-Installationsverzeichnis ist (typischerweise `/usr/local/mysql`). Aktuell finden Sie dort beispielhafte Konfigurationsdateien für kleine, mittlere, große und sehr große Systeme. Sie können `my-xxxx.cnf` in Ihr Heimatverzeichnis kopieren, um damit zu experimentieren (benennen Sie die Kopie in `.my.cnf` um).

Alle MySQL-Clients, die Optionsdateien unterstützen, unterstützen folgende Optionen:

--no-defaults	Keine Optionsdateien einlesen.
--print-defaults	Den Programmnamen und alle Optionen, die das Programm erhalten wird, ausgeben.
--defaults-file=voller-pfad-zur-vorgabe-datei	Nur die angegebene Konfigurationsdatei benutzen.
--defaults-extra-file=voller-pfad-zur-vorgabe-datei	Diese Konfigurationsdatei nach der globalen Konfigurationsdatei einlesen, aber vor der Benutzer-Konfigurationsdatei.

Beachten Sie, dass die oben aufgeführten Optionen auf der Kommandozeile zuerst angegeben werden müssen, damit sie funktionieren! `--print-defaults` kann jedoch direkt nach den `--defaults-xxx-file`-Befehlen angegeben werden.

Hinweis für Entwickler: Optionsdatei-Handhabung ist schlicht dadurch implementiert, dass alle übereinstimmenden Optionen verarbeitet werden (das heißt, Optionen in der entsprechenden Gruppe), vor jeglichen Kommandozeilen-Argumenten. Das funktioniert sehr gut bei Programmen, die die letzte Instanz einer Option benutzen, die mehrfach festgelegt wurde. Wenn Sie ein altes Programm benutzen, das mehrfach festgelegte Optionen auf diese Art handhabt, aber keine Optionsdateien liest, müssen Sie nur zwei Zeilen hinzufügen, um diese Fähigkeit hinzuzufügen. Sehen Sie im Quellcode irgend eines Standard-MySQL-Clients nach, wie das gemacht wird.

In Shellskripts können Sie den `my_print_defaults`-Befehl benutzen, um die Konfigurationsdateien zu parsen:

```
shell> my_print_defaults client mysql
--port=3306
--socket=/tmp/mysql.sock
--no-auto-rehash
```

Die Ausgabe enthält alle Optionen für die Gruppen 'client' und 'mysql'.

5.1.3. Viele Server auf derselben Maschine installieren

In einigen Fällen brauchen Sie vielleicht viele verschiedene `mysqld`-Daemons (Server), die auf derselben Maschine laufen. Beispielsweise wollen Sie eine neue MySQL-Version zum Testen benutzen, während gleichzeitig eine alte Version für die Produktion läuft, oder Sie wollen verschiedenen Benutzern Zugriff auf verschiedene `mysqld`-Server geben, die sie selbst verwalten.

Eine Möglichkeit, einen neuen Server laufen zu lassen, besteht darin, ihn mit einem anderen Socket und einem anderen Port wie folgt zu starten:

```
shell> MYSQL_UNIX_PORT=/tmp/mysqld-neu.sock
shell> MYSQL_TCP_PORT=3307
shell> export MYSQL_UNIX_PORT MYSQL_TCP_PORT
shell> scripts/mysql_install_db
shell> bin/safe_mysqld &
```

Der Umgebungsvariablen-Appendix beinhaltet eine Liste anderer Umgebungsvariablen, die Sie benutzen können, um `mysqld` zu steuern. See Anhang F, *Umgebungsvariablen*.

Der oben gezeigte Weg ist die 'schnelle und schmutzige' Lösung, die man üblicherweise zum Testen benutzt. Das nette daran ist, dass alle Verbindungen, die Sie in obiger Shell aufbauen, automatisch an den neuen laufenden Server weiter geleitet werden!

Wenn Sie dasselbe dauerhafter durchführen wollen, sollten Sie für jeden Server eine Optionsdatei erzeugen. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“. In Ihrem Startskript, das beim Hochfahren ausgeführt wird (mysql.server?) sollten Sie für beide Server folgendes festlegen:

```
safe_mysqld --default-file=pfad-zur-optionsdatei
```

Zumindest folgende Optionen sollten für jeden Server unterschiedlich sein:

- `port=#`, `socket=pfad`, `pid-file=pfad`

Folgende Optionen sollten unterschiedlich sein, wenn sie benutzt werden:

- `log=pfad`, `log-bin=pfad`, `log-update=pfad`, `log-isam=pfad`, `bdb-logdir=pfad`

Wenn Sie mehr Performance erreichen wollen, können Sie auch folgendes unterschiedlich festlegen:

- `tmpdir=pfad`, `bdb-tmpdir=pfad`

See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

Wenn Sie binäre MySQL-Versionen installieren (.tar-Dateien) und sie mit `./bin/safe_mysqld` starten, müssen Sie in den meisten Fällen lediglich die `socket`- und `port`-Argumente in `safe_mysqld` hinzufügen / ändern.

5.1.4. Viele MySQL-Server auf derselben Maschine laufen lassen

Unter bestimmten Umständen wollen Sie vielleicht mehrere Server auf derselben Maschine laufen lassen. Beispielsweise wollen Sie ein neues MySQL-Release testen, Ihre bestehende Produktionseinrichtung aber unangetastet lassen. Oder Sie sind ein Internet-Service-Provider, der unabhängige MySQL-Installationen für verschiedene Kunden hat.

Wenn Sie mehrere Server laufen lassen wollen, ist es am einfachsten, die Server mit unterschiedlichen TCP/IP-Ports und Socket-Dateien laufen zu lassen, damit sie nicht beide auf demselben TCP/IP-Port oder derselben Socket-Datei auf Verbindungen warten. See Abschnitt 5.7.3, „mysqld_multi, Programm zur Verwaltung mehrerer MySQL-Server“.

Nehmen wir einen existierenden Server an, der auf die existierende Port-Nummer und Socket-Datei konfiguriert ist. Sie konfigurieren einen neuen Server mit einem `configure`-Befehl, etwa wie folgt:

```
shell> ./configure  --with-tcp-port=port_nummer \
            --with-unix-socket-path=datei \
            --prefix=/usr/local/mysql-3.22.9
```

Hier müssen `port_nummer` und `datei` anders als die vorgabemäßigen Werte sein. Der `--prefix`-Wert sollte ein Installationsverzeichnis festlegen, das anders ist als dasjenige, unter dem die existierende MySQL-Installation liegt.

Sie können den Socket, der vom aktuell laufenden MySQL-Server benutzt wird, mit folgendem Befehl feststellen:

```
shell> mysqladmin -h hostname --port=port_nummer variables
```

Wenn Sie ``localhost'' als Hostnamen festlegen, benutzt `mysqladmin` Unix-Sockets anstelle von TCP/IP.

Wenn Sie einen MySQL-Server auf dem Port laufen haben, den Sie benutzt haben, bekommen Sie eine Liste der wichtigsten konfigurierbaren Variablen in MySQL, inklusive des Socketnamens.

Sie müssen keinen neuen MySQL-Server kompilieren, nur um ihn mit einem anderen Port und Socket zu starten. Sie können Port und Socket zur Laufzeit als Optionen von `safe_mysqld` festlegen:

```
shell> /pfad/zu/safe_mysqld --socket=datei --port=port_nummer
```

`mysqld_multi` kann ebenfalls `safe_mysqld` (oder `mysqld`) als Argument nehmen und die Optionen von einer Konfigurationsdatei an `safe_mysqld` und weiter an `mysqld` durchreichen.

Wenn Sie den neuen Server mit demselben Datenbankverzeichnis laufen lassen und Loggen angeschaltet haben, sollten Sie auch den Namen der Logdateien für `safe_mysqld` mit `--log`, `--log-update` oder `--log-slow-queries` festlegen. Ansonsten versuchen beide Server, in dieselbe Logdatei zu schreiben.

ACHTUNG: Normalerweise sollten Sie nie zulassen, dass zwei Server Daten in derselben Datenbank aktualisieren! Wenn Ihr Betriebssystem kein fehlerfreies System-Sperren (System Locking) unterstützt, führt das zu unliebsamen Überraschungen!

Wenn Sie für den zweiten Server ein anderes Datenbankverzeichnis benutzen wollen, können Sie das mit der `--datadir=path`-Option für `safe_mysqld` angeben.

HINWEIS: Mehrere MySQL-Server (`mysqld`) auf verschiedenen Maschinen laufen lassen, die auf ein gemeinsames Datenverzeichnis über `NFS` zugreifen, ist generell eine **SCHLECHTE IDEE**! Das Problem liegt darin, dass `NFS` zum Flaschenhals in Punkto Geschwindigkeit wird, denn es ist nicht für solche Zwecke gedacht. Und letztlich müssten Sie immer noch eine Lösung dafür finden, dass sich zwei oder mehr `mysqlds` nicht in die Quere kommen. Momentan gibt es keine Plattform, die mit 100%-iger Zuverlässigkeit Datei-Sperren (File Locking, gewöhnlich mit dem `lockd`-Daemon) in jeder Situation durchführt. Dennoch stellt `NFS` ein weiteres mögliches Risiko dar, denn es macht es dem `lockd`-Daemon noch schwieriger, Datei-Sperren zu handhaben. Machen Sie es sich also leicht und vergessen Sie diese Idee! Die funktionierende Lösung ist, einen Computer mit einem Betriebssystem einzusetzen, dass Threads effizient handhabt und mehrere Prozessoren hat.

Wenn Sie sich mit einem MySQL-Server verbinden wollen, der mit einem anderen Port läuft als mit dem, der in Ihren Client kompiliert ist, können Sie folgende Methoden benutzen:

- Starten Sie den Client mit `--host 'hostname' --port=port_nummer`, um sich über TCP/IP zu verbinden, oder mit `[--host localhost] --socket=datei`, um sich über ein Unix-Socket zu verbinden.
- In Ihren C- oder Perl-Programmen können Sie die Port- oder Socket-Argumente angeben, wenn Sie sich mit dem MySQL-Server verbinden.
- Wenn Sie das Perl-`DBD::mysql`-Modul benutzen, können Sie die Optionen aus den MySQL-Optionsdateien lesen. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

  ```
  $dsn = "DBI:mysql:test;mysql_read_default_group=client;mysql_read_default_file=/usr/local/mysql/data/my.cnf"
  $dbh = DBI->connect($dsn, $user, $password);
  ```

- Setzen Sie die `MYSQL_UNIX_PORT`- und `MYSQL_TCP_PORT`-Umgebungsvariablen, so dass sie auf den Unix-Socket und TCP/IP-Port zeigen, bevor Sie Ihre Clients starten. Wenn Sie normalerweise eine speziellen Socket oder Port benutzen, sollten Sie die Befehle zum Setzen dieser Umgebungsvariablen in Ihrer `.login`-Datei unterbringen. See Anhang F, *Umgebungsvariablen*.
- Legen Sie den vorgabemäßigen Socket und TCP/IP-Port in der `.my.cnf`-Datei in Ihrem Heimatverzeichnis fest. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

5.2. Allgemeine Sicherheitsthemen und das MySQL-

Zugriffsberechtigungssystem

MySQL hat ein fortgeschrittenes, aber nicht standardisiertes Sicherheits- bzw. Berechtigungssystem. Dieser Abschnitt beschreibt, wie es funktioniert.

5.2.1. Allgemeine Sicherheitsrichtlinien

Jeder, der MySQL auf einem Computer benutzt, der mit dem Internet verbunden ist, sollte diesen Abschnitt lesen, um die gebräuchlichsten Sicherheitsfehler zu vermeiden.

Wenn wir über Sicherheit sprechen, unterstreichen wir die Notwendigkeit, den gesamten Server-Host (und nicht nur den MySQL-Server) gegen alle Arten möglicher Angriffe abzusichern: Lauschangriffe, Änderungen (Altering), Playback und Dienstverweigerung (Denial of Service). Dieser Abschnitt deckt nicht alle Aspekte von Verfügbarkeit und Fehlertoleranz ab.

MySQL benutzt ein Sicherheitssystem, das auf Zugriffssteuerungslisten (Access Control Lists, ACLs) für alle Verbindungen, Anfragen und sonstige Operationen basiert, die ein Benutzer durchführen kann. Zusätzlich gibt es einige Unterstützung für SSL-verschlüsselte Verbindungen zwischen MySQL-Clients und -Servern. Viele der hier geschilderten Konzepte sind überhaupt nicht spezifisch für MySQL, sondern beziehen sich auf fast alle Applikationen.

Wenn Sie MySQL laufen lassen, sollten Sie möglichst immer folgende Richtlinien beachten:

- GEBEN SIE NIEMALS JEMANDEM AUSSER DEM MySQL-ROOT-BENUTZER ZUGRIFF AUF DIE `user`-TABELLE IN DER `mysql`-DATENBANK! Das verschlüsselte Passwort ist das echte Passwort in MySQL. Wenn Sie das in der the `user`-Tabelle aufgeführte Passwort für einen gegebenen Benutzer kennen, können Sie sich leicht als dieser Benutzer einloggen, wenn Sie Zugriff auf den Host haben, der für dieses Benutzerkonto aufgeführt ist.
- Lernen Sie das MySQL-Zugriffsberechtigungssystem. Die `GRANT`- und `REVOKE`-Befehle werden benutzt, um den Zugriff auf MySQL zu steuern. Gewähren Sie nicht mehr Zugriffsrechte als notwendig. Gewähren Sie niemals Zugriffsrechte für alle Hosts.

  Checkliste:

  - Probieren Sie `mysql -u root`. Wenn es Ihnen gelingt, sich erfolgreich mit dem Server zu verbinden, ohne nach einem Passwort gefragt zu werden, haben Sie ein Problem, denn jeder kann sich als MySQL-`root`-Benutzer mit dem Server verbinden und hat volle Berechtigungen! Lesen Sie in diesem Fall noch einmal die MySQL-Installationsanweisungen durch und achten Sie insbesondere auf den Teil, der sich mit dem Setzen des `root`-Passworts beschäftigt.
  - Benutzen Sie den Befehl `SHOW GRANTS` und prüfen Sie nach, wer Zugriff auf was hat. Entfernen Sie die Berechtigungen, die nicht notwendig sind, indem Sie den `REVOKE`-Befehl benutzen.
- Halten Sie keine Klartext-Passwörter in Ihrer Datenbank. Wenn Ihr Computer kompromittiert wird, kann der Einbrecher die gesamte Liste von Passwörtern nehmen und benutzen. Benutzen Sie statt dessen `MD5()` oder eine andere Einweg-Hash-Funktion.
- Benutzen Sie keine Passwörter aus Lexika. Es gibt spezielle Programme, um diese zu knacken. Selbst Passwörter wie ``xfish98'' sind sehr schlecht. Viel besser ist ``duag98'', was dasselbe Wort ``fish'' enthält, aber um eine Taste nach links auf einer QUERTZ-Tastatur verschoben. Eine weitere Methode ist, etwas wie ``Mhall'' zu benutzen, was die ersten Buchstaben des Satzes ``Mary had a little lamb'' enthält. Das läßt sich leicht merken und eintippen, aber schwierig durch jemanden erraten, der es nicht kennt.
- Investieren Sie in eine Firewall. Diese schützt sie vor mindestens 50% aller Exploits in jeglicher Software. Installieren Sie MySQL hinter einer Firewall oder in einer entmilitarisierten Zone (Demilitarized Zone, DMZ).

  Checkliste:

  - Versuchen Sie, Ihre Ports vom Internet aus zu scannen, indem Sie ein Werkzeug wie `nmap` benutzen. MySQL benutzt vorgabemäßig Port 3306. Dieser Port sollte von nicht vertrauenswürdigen Hosts aus unerreichbar sein. Eine weitere einfache Methode, um zu überprüfen, ob Ihr MySQL-Port offen ist oder nicht, ist, den folgenden Befehl von einer entfernten Maschine aus zu benutzen, wobei `server_host` der Hostname Ihres MySQL-Servers ist:

    ```
    shell> telnet server_host 3306
    ```

    Wenn Sie eine Verbindung und einige sinnlose Zeichen erhalten, ist der Port offen und sollte auf Ihrer Firewall oder Ihrem Router geschlossen werden, sofern Sie nicht einen wirklich guten Grund haben, ihn offen zu halten. Wenn `telnet` einfach hängt oder die Verbindung abgelehnt wird, ist alles in Ordnung, der Port ist blockiert.
- Trauen Sie keinen Daten, die von Benutzern eingegeben werden. Sie können versuchen, Ihren Code auszutricksen, indem Sie spezielle oder escapete Zeichenfolgen in Web-Formulare, URLs oder sonstige Applikationen, die Sie hergestellt haben, eingeben. Stellen Sie sicher, dass Ihre Applikation sicher bleibt, wenn ein Benutzer etwas wie ``; DROP DATABASE

`mysql;`" eingibt. Das ist ein extremes Beispiel, aber große Sicherheitslücken und Datenverlust können eintreten, wenn ein Hacker ähnliche Techniken benutzt und Sie nicht darauf vorbereitet sind.

Denken Sie auch daran, numerische Daten zu überprüfen. Ein häufiger Fehler besteht darin, nur Zeichenketten zu schützen. Manchmal denken Leute, dass eine Datenbank, die nur öffentlich zugängliche Daten enthält, nicht geschützt werden muss. Das stimmt nicht. Auf solche Datenbanken können zumindest Dienstverweigerungsangriffe (Denial-of-Service-Attacken) durchgeführt werden. Die einfachste Art, sich vor dieser Art von Angriffen zu schützen, ist, Apostrophe um numerische Konstanten herum zu benutzen: `SELECT * FROM tabelle WHERE ID='234'` statt `SELECT * FROM tabelle WHERE ID=234`. MySQL wandelt diese Zeichenkette automatisch in eine Zahl um und entfernt alle nicht-numerischen Zeichen aus ihr.

Checkliste:

- Alle Web-Applikationen:
  - Versuchen Sie, '`'`' und '`"`' in alle Ihr Web-Formulare einzugeben. Wenn Sie irgend welche MySQL-Fehler erhalten, untersuchen Sie das Problem unverzüglich!
  - Versuchen Sie, jedwede dynamischen URLs zu ändern, indem Sie `%22` ('`"`'), `%23` ('`#`') und `%27` ('`'`') zu den URLs hinzufügen.
  - Versuchen Sie, Datentypen in dynamischen URLs von numerischen zu Zeichentypen zu ändern, die Zeichen aus den vorherigen Beispielen enthalten. Ihre Applikation sollte gegen solche und ähnliche Angriffe sicher sein.
  - Versuchen Sie Buchstaben, Leerzeichen und Sonderzeichen anstelle von Zahlen in numerische Felder einzugeben. Ihre Applikation sollte diese entfernen, bevor sie sie MySQL übergibt, und Ihre Applikation sollte einen Fehler erzeugen. Es ist sehr gefährlich, nicht geprüfte Werte an MySQL zu übergeben!
  - Überprüfen Sie Datengrößen, bevor Sie sie an MySQL übergeben.
  - Überlegen Sie, ob es sinnvoll ist, dass sich Ihre Applikation mit einem anderen Benutzernamen mit der Datenbank verbindet als mit dem, den Sie für Verwaltungszwecke benutzen. Geben Sie Applikationen nicht mehr Zugriffsberechtigungen als sie brauchen.
- Benutzer von PHP:
  - Sehen Sie sich die `addslashes()`-Funktion an. Ab PHP 4.0.3 ist eine `mysql_escape_string()`-Funktion verfügbar, die auf der Funktion mit demselben Namen in der MySQL-C-API basiert.
- Benutzer der MySQL-C-API:
  - Sehen Sie sich den `mysql_escape_string()`-API-Aufruf an.
- Benutzer von MySQL++:
  - Sehen Sie sich die `escape`- und `quote`-Modifier für Query-Streams an.
- Benutzer der Perl-DBI:
  - Sehen Sie sich die `quote()`-Methode an oder benutzen Sie Platzhalter.
- Benutzer von Java-JDBC:
  - Benutzen Sie ein `PreparedStatement`-Objekt und Platzhalter.

- Übermitteln Sie keine Klartextdaten (unverschlüsselte Daten) über das Internet. Diese Daten sind für jeden zugänglich, der Zeit und Möglichkeit hat, sie abzuhören und sie für die eigenen Zwecke zu benutzen. Benutzen Sie statt dessen ein verschlüsseltes Protokoll wie SSL oder SSH. MySQL unterstützt ab Version 4.0.0 interne SSL-Verbindungen. SSH-Port-Forwarding kann benutzt werden, um einen verschlüsselten (und komprimierten) Kommunikationstunnel zu erzeugen.
- Lernen Sie die Benutzung der `tcpdump`- und `strings`-Utilities. In den meisten Fällen können Sie mit einem Befehl wie dem folgenden feststellen, ob MySQL-Datenströme verschlüsselt sind oder nicht:

```
shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings
```

(Das funktioniert unter Linux und sollte mit kleineren Änderungen auf anderen Systemen funktionieren.) Achtung: Wenn Sie keine Daten sehen, heißt das nicht immer, dass sie verschlüsselt sind. Wenn Sie hohe Sicherheit benötigen, sollten Sie sich mit einem Sicherheitsexperten in Verbindung setzen.

5.2.2. Wie Sie MySQL gegen Cracker sicher machen

Wenn Sie sich mit einem MySQL-Server verbinden, sollten Sie normalerweise ein Passwort benutzen. Das Passwort wird nicht als Klartext über die Verbindung übermittelt. Allerdings ist der Verschlüsselungsalgorithmus nicht sehr stark, so dass ein cleverer Angreifer mit einiger Mühe das Passwort knacken kann, wenn er in der Lage ist, den Verkehr zwischen Client und Server abzuhören. Wenn die Verbindung zwischen Client und Server über ein nicht vertrauenswürdiges Netzwerk geht, sollten Sie einen SSH-Tunnel benutzen, um die Kommunikation zu verschlüsseln.

Jede sonstige Information wird als Klartext übermittelt, die von jedem gelesen werden kann, der in der Lage ist, die Verbindung abzuhören. Wenn Sie das beunruhigt, können Sie das komprimierte Protokoll benutzen (ab MySQL-Version 3.22), um so etwas zu erschweren. Um die Dinge noch sicherer zu machen, sollten Sie `ssh` benutzen. Sie finden einen Open-Source- `ssh`-Client auf http://www.openssh.org und einen kommerziellen `ssh`-Client auf http://www.ssh.com. Mit diesen erhalten Sie eine verschlüsselte TCP/IP-Verbindung zwischen einem MySQL-Server und einem MySQL-Client.

Um ein MySQL-System sicher zu machen, sollten Sie auf jeden Fall folgende Vorschläge in Betracht ziehen:

- Benutzen Sie Passwörter für alle MySQL-Benutzer. Bedenken Sie, dass sich jeder beliebige als andere Person einloggen kann, und zwar so einfach wie `mysql -u anderer_benutzer db_name`, wenn `anderer_benutzer` kein Passwort hat. Es ist ein normales Verhalten bei Client-Server-Applikationen, dass der Client einen beliebigen Benutzernamen angeben kann. Sie können das Passwort für alle Benutzer ändern, indem Sie das `mysql_install_db`-Skript editieren, bevor Sie es laufen lassen, oder nur das Passwort für den MySQL-`root`-Benutzer, wie folgt:

  ```
  shell> mysql -u root mysql
  mysql> UPDATE user SET Password=PASSWORD('neues_passwort')
              WHERE user='root';
  mysql> FLUSH PRIVILEGES;
  ```

- Lassen Sie den MySQL-Daemon nicht als Unix-`root`-Benutzer laufen. Das ist sehr gefährlich, denn jeder Benutzer mit `FILE`-Berechtigung ist dann in der Lage, Dateien als `root` zu erzeugen (zum Beispiel `~root/.bashrc`). Um das zu verhindern, weigert sich `mysqld`, als `root` zu laufen, es sei denn, das wird direkt durch die `--user=root`-Option angegeben.

  `mysqld` kann unter einem gewöhnlichen Benutzer ohne besondere Rechte laufen. Sie können auch einen neuen Unix-Benutzer `mysql` anlegen, um alles noch sicherer zu machen. Wenn Sie `mysqld` als ein anderer Unix-Benutzer laufen lassen, müssen Sie nicht den `root`-Benutzernamen in der `user`-Tabelle ändern, denn MySQL-Benutzernamen haben nichts mit den Unix-Benutzernamen zu tun. Um `mysqld` als anderer Unix-Benutzer zu starten, fügen Sie eine `user`-Zeile hinzu, die den Benutzernamen zur `[mysqld]`-Gruppe der `/etc/my.cnf`-Optionsdatei oder der `my.cnf`-Optionsdatei im Daten-Verzeichnis des Servers hinzufügt. Beispiel:

  ```
  [mysqld]
  user=mysql
  ```

  Das bewirkt, dass der Server als der festgelegte Benutzer gestartet wird, egal ob Sie ihn manuell oder mit `safe_mysqld` oder `mysql.server` starten. Weitere Details finden Sie unter Abschnitt 5.3.3, „Wann Berechtigungsänderungen wirksam werden“.

- Unterstützen Sie keine Symlinks auf Tabellen (das kann mit der `--skip-symlink`-Option abgeschaltet werden). Das ist insbesondere wichtig, wenn Sie `mysqld` als Root laufen lassen, weil jeder, der Schreibzugriff auf das mysqld-Daten-Verzeichnis hat, dann jede Datei im System zerstören könnte! See Abschnitt 6.6.1.2, „Benutzung symbolischer Links für Tabellen“.

- Überprüfen Sie, dass der Unix-Benutzer, der `mysqld` laufen läßt, der einzige Benutzer mit Lese-/Schreibzugriffen auf die Datenbankverzeichnisse ist.

- Geben Sie nicht allen Benutzern das **process**-Zugriffsrecht. Die Ausgabe von `mysqladmin processlist` zeigt den Text der aktuell ausgeführten Anfragen, so dass jeder, der diesen Befehl ausführen darf, in der Lage wäre, eine Anfrage eines anderen Benutzers wie `UPDATE user SET password=PASSWORD('not_secure')` einzusehen.

  `mysqld` reserviert eine zusätzliche Verbindung für Benutzer, die das **process**-Zugriffsrecht haben, so dass sich ein MySQL-`root`-Benutzer einloggen und Dinge überprüfen kann, selbst wenn alle normalen Verbindungen in Benutzung sind.

- Geben Sie nicht allen Benutzern das **file**-Zugriffsrecht. Jeder Benutzer, der dieses Zugriffsrecht hat, kann irgendwo im Dateisystem Dateien mit den Rechten des the `mysqld`-Daemons schreiben! Um das etwas sicherer zu machen, sind alle Dateien, die mit `SELECT ... INTO OUTFILE` erzeugt werden, für jeden lesbar und können keine existierenden Dateien überschreiben.

  Das **file**-Zugriffsrecht kann auch benutzt werden, um jede Datei zu lesen, auf die der Unix-Benutzer Zugriff hat, als der der Server läuft. Das könnte zum Beispiel durch Benutzung von `LOAD DATA` missbraucht werden, um `/etc/passwd` in eine Tabelle zu laden, die anschließend mit `SELECT` gelesen wird.

- Wenn Sie Ihrem DNS nicht trauen, sollten Sie IP-Nummern anstelle von Hostnamen in den Berechtigungstabellen verwenden.

In jedem Fall sollten Sie sehr vorsichtig damit sein, Einträge in Berechtigungstabellen vorzunehmen, die Hostnamen mit Platzhaltern (Wildcards) verwenden!

- Wenn Sie die Anzahl der Verbindungen für einen einzelnen Benutzer beschränken wollen, können Sie das tun, indem Sie die `max_user_Verbindungen`-Variable in `mysqld` setzen.

5.2.3. Startoptionen für `mysqld` in Bezug auf Sicherheit

Folgende `mysqld`-Optionen berühren Sicherheitsaspekte:

- `--safe-show-database`

  Mit dieser Option gibt `SHOW DATABASES` nur die Datenbanken zurück, für die der Benutzer irgend welche Rechte hat.

- `--safe-user-create`

  Wenn das angeschaltet ist, kann ein Benutzer keine neuen Benutzer mit dem `GRANT`-Befehl anlegen, wenn der kein `INSERT`-Zugriffsrecht auf die `mysql.user`-Tabelle hat. Wenn Sie dem Benutzer nur das Recht geben wollen, neue Benutzer mit den Berechtigungen anzulegen, die er vergeben darf, sollten Sie ihm folgende Berechtigung geben:

  ```
  GRANT INSERT(benutzer) on mysql.user to 'benutzer'@'hostname';
  ```

  Das stellt sicher, dass der Benutzer keine Berechtigungsspalten direkt ändern kann, sondern dafür den `GRANT`-Befehl benutzen muss.

- `--skip-grant-tables`

  Diese Option veranlasst den Server, das Berechtigungssystem überhaupt nicht zu benutzen. Das gibt jedem *vollen Zugriff* auf alle Datenbanken! (Einen laufenden Server können Sie veranlassen, die Berechtigungstabellen erneut zu verwenden, indem Sie `mysqladmin flush-privileges` oder `mysqladmin reload` ausführen.)

- `--skip-name-resolve`

  Hostnamen werden nicht aufgelöst. Alle `Host`-Spaltenwerte in den Berechtigungstabellen müssen IP-Nummern oder `localhost` sein.

- `--skip-networking`

  Keine TCP/IP-Verbindungen über das Netzwerk zulassen. Alle Verbindungen zu `mysqld` müssen über Unix-Sockets gemacht werden. Diese Option ist ungeeignet für Systeme, die MIT-pThreads benutzen, weil das MIT-pThreads-Paket keine Unix-Sockets unterstützt.

- `--skip-show-database`

  Mit dieser Option gibt das `SHOW DATABASES`-Statement nichts zurück.

5.2.4. Was das Berechtigungssystem macht

Die primäre Funktion des MySQL-Berechtigungssystem ist, einen Benutzer zu authentifizieren, der sich von einem gegebenen Host aus verbindet, und diesen Benutzer Berechtigungen auf eine Datenbank zuzuordnen, wie **select**, **insert**, **update** und **delete**.

Zusätzliche Funktionalität beinhaltet die Möglichkeit, einen anonymen Benutzer anzulegen und Berechtigungen für MySQL-spezifische Funktionen wie `LOAD DATA INFILE` und für administrative Operationen zu gewähren.

5.2.5. Wie das Berechtigungssystem funktioniert

Das MySQL-Berechtigungssystem stellt sicher, dass alle Benutzer nur genau die Dinge tun dürfen, zu denen sie berechtigt sind. Wenn Sie sich mit einem MySQL-Server verbinden, wird Ihre Identität **durch den Host, von dem Sie sich aus verbinden,** festgelegt und **durch den Benutzernamen, den Sie angeben**. Das System gewährt Berechtigungen gemäß Ihrer Identität und gemäß dem, **was Sie tun wollen**.

MySQL zieht sowohl Hostnamen als auch Benutzernamen heran, um Sie zu identifizieren, weil es kaum Grund gibt anzunehmen, dass ein gegebener Benutzername derselben Person woanders auf dem Internet gehört. So muss zum Beispiel der Benutzer `bill`, der sich von `whitehouse.gov` aus verbindet, nicht notwendigerweise dieselbe Person sein, die sich als Benutzer `bill` von `microsoft.com` aus verbindet. MySQL erlaubt Ihnen deshalb, Benutzer auf unterschiedlichen Hosts auseinander zu halten, die

zufällig denselben Namen haben: Sie können `bill` einen Satz von Berechtigungen für Verbindungen von `whitehouse.gov` und einen anderen Satz von Berechtigungen für Verbindungen von `microsoft.com` aus gewähren.

Die MySQL-Zugriffskontrolle läuft in zwei Phasen ab:

- Phase 1: Der Server überprüft, ob Sie das Recht haben, sich verbinden zu können.
- Phase 2: Angenommen, Sie haben das Recht, sich zu verbinden, dann überprüft der Server jede Anfrage, die Sie absetzen, um festzustellen, ob Sie ausreichende Rechte haben, um diese auszuführen. Wenn Sie zum Beispiel Zeilen aus einer Tabellen in einer Datenbank auswählen oder eine Tabelle in einer Datenbank löschen, stellt der Server sicher, dass Sie die **select**-Berechtigung für die Tabelle bzw. die **drop**-Berechtigung für die Datenbank haben.

Der Server benutzt die `user`-, `db`- und `host`-Tabellen in der `mysql`-Datenbank in beiden Phasen der Zugriffskontrolle. Die Felder in diesen Berechtigungstabellen sind unten dargestellt:

Tabellenname	`user`	`db`	`host`
Geltungsbereichs-Felder	`Host`	`Host`	`Host`
	`User`	`Db`	`Db`
	`Password`	`User`	
Berechtigungs-Felder	`Select_priv`	`Select_priv`	`Select_priv`
	`Insert_priv`	`Insert_priv`	`Insert_priv`
	`Update_priv`	`Update_priv`	`Update_priv`
	`Delete_priv`	`Delete_priv`	`Delete_priv`
	`Index_priv`	`Index_priv`	`Index_priv`
	`Alter_priv`	`Alter_priv`	`Alter_priv`
	`Create_priv`	`Create_priv`	`Create_priv`
	`Drop_priv`	`Drop_priv`	`Drop_priv`
	`Grant_priv`	`Grant_priv`	`Grant_priv`
	`References_priv`		
	`Reload_priv`		
	`Shutdown_priv`		
	`Process_priv`		
	`File_priv`		

In der zweiten Phase der Zugriffskontrolle (Anfrage-Verifikation), zieht der Server gegebenenfalls zusätzlich die `tables_priv`- und `columns_priv`-Tabellen heran, falls Ihre Anfrage Tabellen betrifft. Die Felder in diesen Tabellen sind unten dargestellt:

Tabellenname	`tables_priv`	`columns_priv`
Geltungsbereichs-Felder	`Host`	`Host`
	`Db`	`Db`
	`User`	`User`
	`Table_name`	`Table_name`
		`Column_name`
Berechtigungs-Felder	`Table_priv`	`Column_priv`
	`Column_priv`	
Sonstige Felder	`Timestamp`	`Timestamp`
	`Grantor`	

Jede Berechtigungstabelle enthält Geltungsbereichsfelder und Berechtigungsfelder.

Geltungsbereichsfelder legen den Geltungsbereich jedes Eintrags in den Tabellen fest, das heißt, der Kontext, für den der Eintrag gilt. So würde zum Beispiel ein `user`-Tabelleneintrag mit `Host`- und `User`-Werten von `'thomas.loc.gov'` und `'bob'` benutzt werden, um Verbindungen zum Server zu authentifizieren, die von `bob` vom Host `thomas.loc.gov` gemacht werden. In ähnlicher Weise bewirkt ein `db`-Tabelleneintrag in die Felder `Host`, `User` und `Db` mit `'thomas.loc.gov'`, `'bob'` und `'reports'`, dass diese benutzt werden, wenn sich `bob` vom Host `thomas.loc.gov` verbindet und auf die `reports`-

Datenbank zugreift. Die `tables_priv`- und `columns_priv`-Tabellen enthalten Geltungsbereichsfelder, die Tabellen oder Tabellen-Spalten-Kombinationen angeben, auf die sich der jeweilige Eintrag bezieht.

Für Zwecke der Zugriffsprüfung sind Vergleiche von `Host`-Werten unabhängig von der verwendeten Groß-/Kleinschreibung. `User`, `Password`, `Db` und `Table_name`-Werte sind abhängig von der verwendeten Groß-/Kleinschreibung. `Column_name`-Werte sind ab MySQL-Version 3.22.12 unabhängig von der verwendeten Groß-/Kleinschreibung.

Berechtigungsfelder zeigen die Berechtigungen an, die durch den Tabelleneintrag gewährt werden, das heißt, welche Operationen durchgeführt werden können. Der Server kombiniert die Informationen in den verschiedenen Berechtigungstabellen, um daraus eine komplette Beschreibung der Berechtigungen des Benutzers zu formulieren. Die Regeln, nach denen hierbei vorgegangen wird, sind in Abschnitt 5.2.9, „Zugriffskontrolle, Phase 2: Anfrageüberprüfung“ beschrieben.

Geltungsbereichsfelder sind Zeichenketten, die wie unten dargestellt deklariert werden. Der Vorgabewert für jedes Feld ist die leere Zeichenkette:

Feldname	Typ	
`Host`	`CHAR(60)`	
`User`	`CHAR(16)`	
`Password`	`CHAR(16)`	
`Db`	`CHAR(64)`	(`CHAR(60)` für die `tables_priv`- und `columns_priv`-Tabellen)
`Table_name`	`CHAR(60)`	
`Column_name`	`CHAR(60)`	

In den `user`-, `db`- und `host`-Tabellen werden alle Felder als `ENUM('N','Y')` deklariert. Jedes Feld kann einen Wert von `'N'` oder `'Y'` haben. Der Vorgabewert ist `'N'`.

In den `tables_priv`- und `columns_priv`-Tabellen werden Felder als `SET`-Felder deklariert:

Tabellenname	Feldname	Mögliche Set-Elemente
`tables_priv`	`Table_priv`	`'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'Referenzs', 'Index', 'Alter'`
`tables_priv`	`Column_priv`	`'Select', 'Insert', 'Update', 'References'`
`columns_priv`	`Column_priv`	`'Select', 'Insert', 'Update', 'References'`

Kurz gesagt benutzt der Server die Berechtigungstabellen wie folgt:

- Das `user`-Tabellenbereichsfeld legt fest, ob eingehende Verbindungen zugelassen oder abgewiesen werden. Bei zugelassenen Verbindungen zeigen Berechtigungen, die in der `user`-Tabelle vergeben sind, die globalen (Superuser-) Rechte des Benutzers an. Diese Berechtigungen treffen auf **alle** Datenbanken auf dem Server zu.
- Die `db`- und `host`-Tabellen werden zusammen benutzt:
  - Die Geltungsbereichsfelder der `db`-Tabelle legen fest, welche Benutzer auf welche Datenbanken von welchen Hosts aus zugreifen können. Die Berechtigungsfelder legen fest, welche Operationen zugelassen sind.
  - Die `host`-Tabelle wird als Erweiterung der `db`-Tabelle benutzt, wenn Sie wollen, dass ein gegebener `db`-Tabelleneintrag auf verschiedene Hosts zutrifft. Wenn Sie zum Beispiel wollen, dass ein Benutzer eine Datenbank von mehreren Hosts in Ihrem Netzwerk aus benutzen kann, lassen Sie den `Host`-Wert in der `db`-Tabelle des Benutzers leer, und füllen dann die `host`-Tabelle mit einem Eintrag für jeden dieser Hosts. Dieser Mechanismus ist ausführlicher in Abschnitt 5.2.9, „Zugriffskontrolle, Phase 2: Anfrageüberprüfung“ beschrieben.
- Die `tables_priv`- und `columns_priv`-Tabellen sind der `db`-Tabelle ähnlich, aber feinkörniger: Sie beziehen sich auf Tabellen- und Spaltenebenen und nicht auf Datenbankebene.

Beachten Sie, dass die Verwaltungsberechtigungen (**reload**, **shutdown** usw.) nur in der `user`-Tabelle festgelegt werden. Das liegt daran, dass Verwaltungsoperationen Operationen auf dem Server selbst sind und nicht Datenbank-spezifisch, so dass es keinen Grund gibt, solche Berechtigungen in den anderen Berechtigungstabellen aufzuführen. So muss nur die `user`-Tabelle untersucht werden um festzustellen, ob man Verwaltungsoperationen durchführen kann oder nicht.

Das **file**-Zugriffsrecht wird auch nur in der `user`-Tabelle festgelegt. Es ist als solches keine Verwaltungsberechtigung, aber Ihre Möglichkeit, Dateien auf dem Server zu lesen oder zu schreiben, ist unabhängig von der Datenbank, auf die Sie zugreifen.

Der `mysqld`-Server liest die Inhalte der Berechtigungstabellen einmal, und zwar beim Start. Änderungen in den

Berechtigungstabellen werden wirksam wie in Abschnitt 5.3.3, „Wann Berechtigungsänderungen wirksam werden“ geschildert.

Wenn Sie die Inhalte der Berechtigungstabellen ändern, sollten Sie sicherstellen, dass Ihre Änderungen Berechtigungen einführen, die Sie so haben wollen. Hilfe bei der Diagnose von Problemen finden Sie unter Abschnitt 5.2.10, „Gründe für `Access denied`-Fehler“. Hinweise zu Sicherheitsthemen finden Sie unter see Abschnitt 5.2.2, „Wie Sie MySQL gegen Cracker sicher machen“.

Ein nützliches Diagnosetool ist das `mysqlaccess`-Skript, das Yves Carlier für die MySQL-Distribution bereit gestellt hat. Rufen Sie `mysqlaccess` mit der `--help`-Option auf, um herauszufinden, wie es funktioniert. Beachten Sie, dass `mysqlaccess` den Zugriff nur anhand der `user`-, `db`- und `host`-Tabellen überprüft. Es überprüft keine Tabellen- oder Spaltenebenen-Berechtigungen.

5.2.6. Von MySQL zur Verfügung gestellte Berechtigungen

Informationen über Benutzerberechtigungen sind in den `user`-, `db`-, `host`-, `tables_priv`- und `columns_priv`-Tabellen in der `mysql`-Datenbank gespeichert (das heißt in der Datenbank, die `mysql` heißt). Der MySQL-Server liest die Inhalte dieser Tabellen, wenn er startet, und in den Fällen, die unter Abschnitt 5.3.3, „Wann Berechtigungsänderungen wirksam werden“ geschildert sind.

Die Namen, die in diesem Handbuch benutzt werden, um auf die Berechtigungen zu verweisen, die MySQL zur Verfügung stellt, sind unten dargestellt, zusammen mit den Tabellenspaltennamen, die jeder Berechtigung in the Berechtigungstabellen zugeordnet sind, und dem Kontext, auf den die Berechtigung zutrifft.

Berechtigung	Spalte	Kontext
select	`Select_priv`	Tabellen
insert	`Insert_priv`	Tabellen
update	`Update_priv`	Tabellen
delete	`Delete_priv`	Tabellen
index	`Index_priv`	Tabellen
alter	`Alter_priv`	Tabellen
create	`Create_priv`	Datenbanken, Tabellen oder Indexe
drop	`Drop_priv`	Datenbanken oder Tabellen
grant	`Grant_priv`	Datenbanken oder Tabellen
References	`References_priv`	Datenbanken oder Tabellen
reload	`Reload_priv`	Serververwaltung
shutdown	`Shutdown_priv`	Serververwaltung
process	`Process_priv`	Serververwaltung
file	`File_priv`	Dateizugriff auf den Server

Die **select**-, **insert**-, **update**- und **delete**-Berechtigungen erlauben Ihnen, Operationen auf Zeilen in existierenden Tabellen in einer Datenbank durchzuführen.

`SELECT`-Statements erfordern die **select**-Berechtigung nur dann, wenn tatsächlich Zeilen aus einer Tabelle abgerufen werden. Sie können bestimmte `SELECT`-Statements selbst ohne Berechtigung durchführen, um auf jede der Datenbanken auf dem Server zuzugreifen. Beispielsweise könnten Sie den `mysql`-Client als einfachen Taschenrechner benutzen:

```
mysql> SELECT 1+1;
mysql> SELECT PI()*2;
```

Die **index**-Berechtigung erlaubt Ihnen, Indexe zu erzeugen oder zu entfernen.

Die **alter**-Berechtigung erlaubt Ihnen, `ALTER TABLE` zu benutzen.

Die **create**- und **drop**-Berechtigungen erlauben Ihnen, neue Datenbanken und Tabellen zu erzeugen oder bestehende Datenbanken und Tabellen zu entfernen.

Denken Sie daran, dass ein Benutzer, dem Sie die **drop**-Berechtigung für die `mysql`-Datenbank gewähren, in der Lage ist, die Datenbank zu löschen, in der die MySQL-Zugriffsberechtigungen gespeichert sind!

Die **grant**-Berechtigung erlaubt Ihnen, die Berechtigungen, die Sie selbst besitzen, an andere Benutzer zu vergeben.

Die **file**-Berechtigung erlaubt Ihnen, Dateien auf dem Server zu lesen und zu schreiben, wenn Sie die `LOAD DATA INFILE`- und `SELECT ... INTO OUTFILE`-Statements benutzen. Jeder Benutzer, dem diese Berechtigung gewährt wurde, kann jedwede Datei lesen oder schreiben, die der MySQL-Server lesen oder schreiben darf.

Die restlichen Berechtigungen werden für Verwaltungsoperationen benutzt, die mit dem `mysqladmin`-Programm durchgeführt werden. Die unten stehende Tabelle zeigt, welche `mysqladmin`-Befehle mit jeder Verwaltungsberechtigung ausgeführt werden können:

Berechtigung	Befehle, die dem Berechtigten erlaubt sind
reload	`reload`, `refresh`, `flush-privileges`, `flush-hosts`, `flush-logs` und `flush-tables`
shutdown	`shutdown`
process	`processlist`, `kill`

Der `reload`-Befehl weist den Server an, die Berechtigungstabellen neu einzulesen. Der `refresh`-Befehl schreibt alle Tabellen auf Platte (flush) und öffnet und schließt die Log-Dateien. `flush-privileges` ist ein Synonym für `reload`. Die anderen `flush-*`-Befehle führen Funktionen aus, die `refresh` ähnlich sind, aber im Umfang beschränkter und daher in einigen Fällen zu bevorzugen. Wenn Sie zum Beispiel nur die Log-Dateien flushen wollen, ist `flush-logs` `refresh` vorzuziehen.

Der `shutdown`-Befehl fährt den Server herunter.

Der `processlist`-Befehl zeigt Informationen über die Threads an, die im Server ausgeführt werden. Der `kill`-Befehl killt Server-Threads. Ihre eigenen Threads können Sie jederzeit anzeigen oder killen, aber Sie brauchen die **process**-Berechtigung, um Threads anzuzeigen oder zu killen, die von anderen Benutzern initiiert wurden. See Abschnitt 5.5.4, „`KILL`-Syntax“.

Es ist generell eine gute Idee, Berechtigungen nur den Nutzern zu gewähren, die diese tatsächlich brauchen, aber speziell bei folgenden Berechtigungen sollten Sie besondere Vorsicht walten lassen:

- Die **grant**-Berechtigung erlaubt Benutzern, Ihre Berechtigungen an andere Benutzer zu übertragen. Zwei Benutzer mit unterschiedlichen Berechtigungen und mit der **grant**-Berechtigung sind in der Lage, Ihre Berechtigungen zu kombinieren.
- Die **alter**-Berechtigung kann benutzt werden, um das Berechtigungssystem zu unterlaufen, indem Tabellen umbenannt werden.
- Die **file**-Berechtigung kann missbraucht werden, um jede öffentlich lesbare Datei auf dem Server in eine Datenbanktabelle einzulesen, auf deren Inhalte dann mit `SELECT` zugegriffen werden kann. Das beinhaltet die Inhalte aller Datenbanken, die vom Server gehostet werden!
- Die **shutdown**-Berechtigung kann missbraucht werden, um andere Benutzer komplett vom Server auszuschließen, indem der Server beendet wird.
- Die **process**-Berechtigung kann benutzt werden, um den Klartext von momentan ablaufenden Anfragen einzusehen, inklusive Anfragen, die Passwörter setzen oder ändern.
- Zugriffsrechte auf die `mysql`-Datenbank können benutzt werden, um Passwörter zu ändern und auf sonstige Berechtigungsinformationen zuzugreifen. (Passwörter werden verschlüsselt gespeichert, daher kann ein böswilliger Benutzer sie nicht einfach lesen und anschließend die Klartext-Passwörter kennen.) Wenn man auf die `mysql.user`-Passwort-Spalte zugreifen kann, kann man das nutzen, um sich als beliebiger Benutzer am MySQL-Server anzumelden. (Mit ausreichenden Rechten kann derselbe Benutzer dann Passwörter durch eigene ersetzen.)

Es gibt einige Dinge, die Sie mit dem MySQL-Berechtigungssystem nicht tun können:

- Sie können nicht ausdrücklich festlegen, dass ein bestimmter Benutzer keinen Zugriff haben soll. Das heißt, Sie können nicht explizit mit einem bestimmten Benutzer vergleichen und dann die Verbindung ablehnen.
- Sie können nicht festlegen, dass ein Benutzer das Recht hat, Tabellen in einer Datenbank zu erzeugen oder zu löschen, aber nicht die Datenbank selbst zu erzeugen oder zu löschen.

5.2.7. Verbinden mit dem MySQL-Server

MySQL-Client-Programme erfordern im Allgemeinen, dass Sie Verbindungsparameter festlegen, wenn Sie sich mit einem MySQL-Server verbinden wollen: Der Host, mit dem Sie sich verbinden wollen, Ihr Benutzername und Ihr Passwort. Beispielsweise kann der `mysql`-Client wie folgt gestartet werden (optionale Argumente sind in ‘`[`’ und ‘`]`’ eingeschlossen):

```
shell> mysql [-h hostname] [-u benutzername] [-pihr_passwort]
```

Alternative Formen der `-h`-, `-u`- und `-p`-Optionen sind `--host=hostname`, `--user=benutzername` und `--password=ihr_passwort`. Beachten Sie, dass zwischen `-p` oder `--password=` und dem folgenden Passwort *kein Leerzeichen* steht!

ACHTUNG: Ein Passwort auf der Kommandozeile anzugeben ist nicht sicher! Jeder Benutzer auf Ihrem System kann dann Ihr Passwort herausfinden, indem er einen Befehl wie `ps auxww` eingibt. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

`mysql` benutzt Vorgabewerte für Verbindungsparameter, die auf der Kommandozeile nicht angegeben sind:

- Der vorgabemäßige Hostname ist `localhost`.
- Der vorgabemäßige Benutzername ist Ihr Unix-Loginname.
- Es wird kein Passwort übergeben, wenn `-p` fehlt.

Für einen Unix-Benutzer `joe` sind daher folgende Befehle gleichbedeutend:

```
shell> mysql -h localhost -u joe
shell> mysql -h localhost
shell> mysql -u joe
shell> mysql
```

Andere MySQL-Clients verhalten sich ähnlich.

Auf Unix-Systemen können Sie andere Vorgabewerte festlegen, die benutzt werden, wenn Sie eine Verbindung aufmachen, so dass Sie diese nicht jedes Mal auf der Kommandozeile eingeben müssen, wenn Sie ein Client-Programm aufrufen. Das kann auf verschiedene Weise gemacht werden:

- Sie können Verbindungsparameter im `[client]`-Abschnitt der `.my.cnf`-Konfigurationsdatei in Ihrem Heimatverzeichnis festlegen. Der relevante Abschnitt der Datei sieht etwa wie folgt aus:

  ```
  [client]
  host=hostname
  user=benutzername
  password=ihr_passwort
  ```

  See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

- Sie können Verbindungsparameter festlegen, indem Sie Umgebungsvariablen benutzen. Der Host kann für `mysql` festgelegt werden, indem `MYSQL_HOST` benutzt wird. Der MySQL-Benutzername kann mit `USER` festgelegt werden (nur für Windows). Das Passwort kann mit `MYSQL_PWD` festgelegt werden (aber das ist unsicher, siehe nächster Abschnitt). See Anhang F, *Umgebungsvariablen*.

5.2.8. Zugriffskontrolle, Phase 1: Verbindungsüberprüfung

Wenn Sie versuchen, sich mit einem MySQL-Server zu verbinden, akzeptiert der Server die Verbindung oder weist sie zurück, abhängig von Ihrer Identität und davon, ob Sie diese mit dem korrekten Passwort verifizieren können. Falls nicht, lehnt der Server den Zugriff vollständig ab. Ansonsten akzeptiert der Server die Verbindung, geht dann in Phase 2 und wartet auf Anfragen.

Ihre Identität basiert auf zwei Informationsbestandteilen:

- Dem Host, von dem Sie sich verbinden
- Ihrem MySQL-Benutzernamen

Die Identitätsüberprüfung wird anhand der drei Geltungsbereichs-Felder der `user`-Tabelle, nämlich (`Host`, `User` und `Password`) durchgeführt. Der Server akzeptiert die Verbindung nur, wenn ein `user`-Tabelleneintrag mit Ihrem Hostnamen und Benutzernamen übereinstimmt und Sie das korrekte Passwort angeben können.

Werte in den Geltungsbereichs-Feldern der `user`-Tabelle können wie folgt festgelegt werden:

- Ein `Host`-Wert kann ein Hostname oder eine IP-Nummer sein, oder `'localhost'`, was die lokale Maschine angibt.
- Sie können die Platzhalterzeichen ‘`%`’ und ‘`_`’ im `Host`-Feld benutzen.
- Ein `Host`-Wert `'%'` stimmt mit jedem Hostnamen überein.
- Ein leerer `Host`-Wert bedeutet, dass die Berechtigung zusammen mit dem Eintrag in der `host`-Tabelle gilt, der mit dem angegebenen Hostnamen übereinstimmt. Weitere Informationen hierzu finden Sie im nächsten Kapitel.

- Ab MySQL-Version 3.23 können `Host`-Werte als IP-Nummern festgelegt werden, und Sie können eine Netmask festlegen, die angibt, wie viele Adress-Bits für die Netzwerknummer benutzt werden. Beispiel:

  ```
  GRANT ALL PRIVILEGES on db.* to david@'192.58.197.0/255.255.255.0';
  ```

  Das erlaubt jedem, sich von einer IP zu verbinden, bei der folgendes gilt:

  ```
  benutzer_ip & netmask = host_ip.
  ```

  Im obigen Beispiel können sich alle IP's im Intervall zwischen 192.58.197.0 bis 192.58.197.255 mit dem MySQL-Server verbinden.

- Platzhalterzeichen sind im `User`-Feld nicht erlaubt. Sie können aber einen leeren Wert angeben, der mit jedem Namen übereinstimmt. Wenn der Eintrag in der `user`-Tabelle, der mit einer hereinkommenden Verbindung übereinstimmt, einen leeren Benutzernamen hat, wird angenommen, dass der Benutzer der anonyme Benutzer ist (der Benutzer ohne Namen), und nicht der Name, den der Client tatsächlich angegeben hat. Das bedeutet, dass ein leerer Benutzername für alle weiteren Zugriffsüberprüfungen während der laufenden Verbindung benutzt wird (also während Phase 2).

- Das `Password`-Feld kann leer sein. Das bedeutet nicht, dass jedes Passwort übereinstimmt, sondern dass der Benutzer sich ohne Angabe eines Passworts verbinden muss.

Nicht-leere `Password`-Werte repräsentieren verschlüsselte Passwörter. MySQL speichert Passwörter nicht im Klartext, so dass jeder sie sehen könnte. Statt dessen wird das Passwort eines Benutzers, der sich zu verbinden versucht, verschlüsselt (unter Benutzung der `PASSWORD()`-Funktion). Das verschlüsselte Passwort wird dann benutzt, wenn Client / Server prüfen, ob das Passwort korrekt ist (das geschieht, ohne dass das verschlüsselte Passwort jemals über die Verbindung übertragen wird). Beachten Sie, dass aus der Sicht von MySQL das verschlüsselte Passwort das ECHTE Passwort ist, daher sollten Sie niemandem Zugriff darauf geben! Insbesondere sollten Sie keinem normalen Benutzer Lesezugriff auf die Tabellen der `mysql`-Datenbank geben!

Die unten stehenden Beispiele zeigen, wie unterschiedliche Kombinationen von `Host`- und-`User`-Werten in den `user`-Tabelleneinträgen auf hereinkommende Verbindungen zutreffen:

`Host` **Wert**	`User` **Wert**	**Verbindungen, die mit dem Eintrag übereinstimmen**
`'thomas.loc.gov'`	`'fred'`	`fred`, der sich von `thomas.loc.gov` aus verbindet
`'thomas.loc.gov'`	`''`	Jeder Benutzer, der sich von `thomas.loc.gov` aus verbindet
`'%'`	`'fred'`	`fred`, der sich von jedem Host aus verbindet
`'%'`	`''`	Jeder Benutzer, der sich von jedem Host aus verbindet
`'%.loc.gov'`	`'fred'`	`fred`, der sich von jedem beliebigen Host in der `loc.gov`-Domäne aus verbindet
`'x.y.%'`	`'fred'`	`fred`, der sich von `x.y.net`, `x.y.com`, `x.y.edu` usw. aus verbindet (wahrscheinlich eher unsinnig)
`'144.155.166.177'`	`'fred'`	`fred`, der sich vom Host mit der IP-Adresse `144.155.166.177` aus verbindet
`'144.155.166.%'`	`'fred'`	`fred`, der sich von jedem beliebigen Host im Class-C-Subnet `144.155.166` aus verbindet
`'144.155.166.0/255.255.255.0'`	`'fred'`	Dasselbe wie im vorherigen Beispiel

Weil Sie im `Host`-Feld IP-Platzhalterwerte verwenden können (beispielsweise `'144.155.166.%'`, was mit jedem Host in einem Subnet übereinstimmt), besteht die Möglichkeit, dass jemand diese Fähigkeit ausbeutet, indem er einen Host zum Beispiel `144.155.166.somewhere.com` nennt. Um solche Versuche zu vereiteln, verbietet MySQL den Vergleich mit Hostnamen, die mit Ziffern und einem Punkt übereinstimmen. Wenn Sie daher einen Host haben, der so wie `1.2.foo.com` benannt ist, wird sein Name nie mit der `Host`-Spalte der Berechtigungstabellen übereinstimmen. Nur eine IP-Nummer kann mit dem IP-Platzhalterwert übereinstimmen.

Eine hereinkommende Verbindung kann mit mehr als einem Eintrag in der `user`-Tabelle übereinstimmen. Beispielsweise würde eine Verbindung von `thomas.loc.gov` aus durch `fred` mit mehreren der oben genannten Einträge übereinstimmen. Wie entscheidet der Server, welcher der Einträge benutzt werden soll, wenn mehrere zutreffen? Der Server löst dieses Problem, indem er die `user`-Tabelle nach dem Einlesen beim Start sortiert, und danach die Einträge in sortierter Form durchsieht, wenn ein Benutzer versucht, sich zu verbinden. Der erste übereinstimmende Eintrag ist der, der benutzt wird.

Das Sortieren der `user`-Tabelle funktioniert wie folgt. Nehmen Sie an, dass die `user`-Tabelle so aussieht:

```
+-----------+----------+-
| Host      | User     | ...
```

```
+-----------+----------+-
| %         | root     | ...
| %         | jeffrey  | ...
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

Wenn der Server die Tabelle liest, ordnet er die Einträge mit den spezifischsten Einträgen für die `Host`-Werte zuerst ein (`'%'` in der `Host`-Spalte bedeutet ``jeder Host'' und ist am unspezifischsten). Einträge mit denselben `Host`-Werten werden mit den spezifischsten `User`-Werten zuerst geordnet (ein leerer `User`-Wert bedeutet ``jeder Benutzer'' und ist am unspezifischsten). Die daraus resultierende sortierte `user`-Tabelle sieht wie folgt aus:

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
| %         | jeffrey  | ...
| %         | root     | ...
+-----------+----------+-
```

Beim Versuch einer Verbindung durchsucht der Server die sortierten Einträge und benutzt die ersten übereinstimmenden. Bei einer Verbindung von `localhost` aus durch `jeffrey` stimmen die Werte zuerst mit den Einträgen von `'localhost'` in der `Host`-Spalte überein. Hiervon stimmt der Eintrag mit dem leeren Benutzernamen sowohl mit dem verbindenden Host als auch mit dem Benutzernamen überein. (`'%'/'jeffrey'` hätte auch übereingestimmt, aber er ist nicht der erste Tabelleneintrag, der gefunden wird.)

Hier ist ein weiteres Beispiel. Nehmen Sie an, die `user`-Tabelle sieht wie folgt aus:

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| %              | jeffrey  | ...
| thomas.loc.gov |          | ...
+----------------+----------+-
```

Die sortierte Tabelle sieht wie folgt aus:

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| thomas.loc.gov |          | ...
| %              | jeffrey  | ...
+----------------+----------+-
```

Eine Verbindung von `thomas.loc.gov` aus durch `jeffrey` stimmt mit dem ersten Eintrag überein, wohingegen eine Verbindung von `whitehouse.gov` aus durch `jeffrey` mit dem zweiten Eintrag übereinstimmt.

Ein häufiges Missverständnis besteht darin zu denken, dass bei einem angegebenen Benutzernamen alle Einträge, die explizit den Benutzer nennen, zuerst benutzt werden, wenn der Server versucht, eine Übereinstimmung für die Verbindung zu finden. Das stimmt schlicht nicht. Das vorherige Beispiel stellt das dar, wobei eine Verbindung von `thomas.loc.gov` aus durch `jeffrey` zuerst gerade nicht mit dem Eintrag übereinstimmt, der `'jeffrey'` als `User`-Feldwert enthält, sondern mit dem Eintrag, der keinen Benutzernamen enthält!

Wenn Sie Probleme haben, sich mit dem Server zu verbinden, geben Sie die `user`-Tabelle aus und sortieren Sie sich von Hand, um zu sehen, wo die erste Übereinstimmung stattfindet.

5.2.9. Zugriffskontrolle, Phase 2: Anfrageüberprüfung

Wenn Sie erst einmal eine Verbindung hergestellt haben, geht der Server in Phase 2. Bei jeder Anfrage, die über diese Verbindung hereinkommt, prüft der Server, ob Sie ausreichende Berechtigungen haben, sie auszuführen, wobei es auf die Operation ankommt, die Sie ausführen wollen. Hier kommen die Berechtigungsfelder der Berechtigungstabellen ins Spiel. Diese Berechtigungen können aus jeder der `user`-, `db`-, `host`-, `tables_priv`- oder `columns_priv`-Tabellen stammen. Die Berechtigungstabellen werden mit `GRANT`- und `REVOKE`-Befehlen verändert. See Abschnitt 5.3.1, „GRANT- und REVOKE-Syntax“. (Hilfreich sind die Ausführungen unter Abschnitt 5.2.5, „Wie das Berechtigungssystem funktioniert“, wo die Felder aufgelistet sind, die sich in jeder der Berechtigungstabellen finden.)

Die `user`-Tabelle gewährt Berechtigungen, die Ihnen auf globaler Ebene zugeordnet sind und die unabhängig von der gerade aktuellen Datenbank zutreffen. Wenn beispielsweise die `user`-Tabelle Ihnen die **delete**-Berechtigung gewährt, können Sie Zeilen aus jeder Datenbank auf dem Server-Host löschen! Mit anderen Worten: Berechtigungen in der `user`-Tabelle sind Superuser-Berechtigungen. Es ist klug, Berechtigungen in der `user`-Tabelle nur Superusern wie Server- oder Datenbankverwaltern zu gewähren. Bei anderen Benutzern sollten Sie Berechtigungen in der `user`-Tabelle auf `'N'` gesetzt lassen und Berechtigungen nur auf Datenbank-Ebene gewähren, indem Sie die `db`- und `host`-Tabellen benutzen.

Die `db`- und `host`-Tabellen gewähren Datenbank-spezifische Berechtigungen. Werte in den Geltungsbereichs-Feldern können wie folgt festgelegt werden:

- Die Platzhalterzeichen '`%`' und '`_`' können in den `Host`- und `Db`-Feldern jeder Tabelle benutzt werden.
- Ein `'%'`-`Host`-Wert in der `db`-Tabelle bedeutet ``jeder Host." Ein leerer `Host`-Wert in der `db`-Tabelle bedeutet ``sieh in der `host`-Tabelle wegen weiterer Informationen nach".
- Ein `'%'`- oder leerer `Host`-Wert in der `host`-Tabelle bedeutet ``jeder Host".
- Ein `'%'`- oder leerer `Db`-Wert in einer der Tabellen bedeutet ``jede Datenbank".
- Ein leerer `User`-Wert in einer der Tabellen entspricht dem anonymen Benutzer.

Die `db`- und `host`-Tabellen werden eingelesen und sortiert, wenn der Server hoch fährt (zur gleichen Zeit, wenn er die `user`-Tabelle einliest). Die `db`-Tabelle wird nach den Geltungsbereichs-Feldern `Host`, `Db` und `User` sortiert. Die `host`-Tabelle wird nach den Geltungsbereichs-Feldern `Host` und `Db` sortiert. Bei der `user`-Tabelle werden die spezifischsten Werte zuerst und die unspezifischsten Werte zuletzt einsortiert, und wenn der Server nach übereinstimmenden Einträgen sucht, benutzt er die erste Übereinstimmung, die er findet.

Die `tables_priv`- und `columns_priv`-Tabellen gewähren Tabellen- und Spalten-spezifische Berechtigungen. Werte in der Geltungsbereichs-Feldern können wie folgt festgelegt werden:

- Die Platzhalterzeichen '`%`' und '`_`' können im `Host`-Feld beider Tabellen benutzt werden.
- Ein `'%'`- oder leerer `Host`-Wert in jeder der beiden Tabellen bedeutet ``jeder Host."
- Die `Db`-, `Table_name`- und `Column_name`-Felder dürfen in beiden Tabellen keine Platzhalter enthalten oder leer sein.

Die `tables_priv`- und `columns_priv`-Tabellen werden nach den `Host`-, `Db`- und `User`-Feldern sortiert. Das geschieht ähnlich wie das Sortieren der `db`-Tabelle, wenngleich das Sortieren einfacher ist, weil nur das `Host`-Feld Platzhalter enthalten darf.

Der Prozess der Anfragenüberprüfung ist weiter unten beschrieben. (Wenn Sie mit dem Quelltext für die Zugangsüberprüfung vertraut sind, werden Sie feststellen, dass die Beschreibung hier leicht vom im Code verwendeten Algorithmus abweicht. Die Beschreibung stellt dar, was der Code tatsächlich tut; sie weicht nur deshalb ab, um die Erklärung zu erleichtern.)

Bei Verwaltungsanfragen (**shutdown**, **reload** usw.) prüft der Server nur den `user`-Tabelleneintrag, weil das die einzige Tabelle ist, die Verwaltungsberechtigungen festlegt. Zugriff wird gewährt, wenn der Eintrag die verlangte Operation erlaubt, ansonsten wird er verweigert. Wenn Sie zum Beispiel `mysqladmin shutdown` ausführen wollen, aber Ihr `user`-Tabelleneintrag Ihnen nicht die **shutdown**-Berechtigung gewährt, wird der Zugriff verweigert, ohne dass die `db`- oder `host`-Tabellen geprüft werden. (Sie enthalten keine `Shutdown_priv`-Spalte, daher gibt es keinen Grund, sie zur Prüfung heranzuziehen.)

Bei Datenbank-bezogenen Anfragen (**insert**, **update** usw.) prüft der Server zuerst die globalen (superuser-) Berechtigungen, indem er im `user`-Tabelleneintrag nachsieht. Wenn der Eintrag die verlangte Operation erlaubt, wird der Zugriff gewährt. Wenn die globalen Berechtigungen in der `user`-Tabelle unzureichend sind, stellt der Server die Datenbank-spezifischen Berechtigungen des Benutzers fest, indem er die `db`- und `host`-Tabellen prüft:

1. Der Server sieht in der `db`-Tabelle nach einer Übereinstimmung in den `Host`-, `Db`- und `User`-Feldern nach. In den `Host`- und `User`-Feldern wird nach Übereinstimmung mit dem Hostnamen gesucht, von dem aus sich der Benutzer verbindet, und nach Übereinstimmung mit dem MySQL-Benutzernamen. Im `Db`-Feld wird nach Übereinstimmung mit der Datenbank gesucht, mit der sich der Benutzer verbinden will. Wenn es keinen Eintrag für `Host` und `User` gibt, wird der Zugriff verweigert.
2. Wenn es keinen übereinstimmenden `db`-Tabelleneintrag gibt und das `Host`-Feld nicht leer ist, bestimmt dieser Eintrag die Datenbank-spezifischen Berechtigungen des Benutzers.
3. Wenn das `Host`-Feld des übereinstimmenden `db`-Tabelleneintrags leer ist, bedeutet das, dass die `host`-Tabelle festlegt, welchen Hosts Zugriff auf die Datenbank erlaubt werden soll. In diesem Fall schlägt der Server weiter in der `host`-Tabelle nach, um eine Übereinstimmung in den `Host`- und `Db`-Feldern zu finden. Wenn kein `host`-Tabelleneintrag passt, wird der Zugriff verweigert. Bei einer Übereinstimmung werden die Datenbank-spezifischen Berechtigungen des Benutzers als Schnittmenge (*nicht* Vereinigungsmenge!) der Berechtigungen in den `db`- und `host`-Tabelleneinträgen berechnet, was die Berechtigungen ergibt, die in beiden Einträgen `'Y'` sind. (Auf diese Weise können Sie allgemeine Berechtigungen in den `db`-Tabelleneinträgen vergeben und diese dann fallweise von Host zu Host beschränken, indem Sie die `host`-Tabelleneinträge benutzen.)

Nachdem die Datenbank-spezifischen Berechtigungen festgestellt wurden, die durch die `db`- und `host`-Tabelleneinträge gewährt werden, fügt der Server diese zu den globalen Berechtigungen in der `user`-Tabelle hinzu. Wenn das Ergebnis die verlangte Operation erlaubt, wird der Zugriff gewährt. Ansonsten prüft der Server die Tabellen- und Spalten-Berechtigungen des Benutzers in den `tables_priv`- und `columns_priv`-Tabellen und fügt diese zu den Benutzerberechtigungen hinzu. Aus dem Ergebnis ergibt sich, ob der Zugriff erlaubt oder verweigert wird.

Als Boole'scher Term ausgedrückt kann die vorstehende Beschreibung der Berechnung der Benutzerrechte wie folgt zusammengefasst werden:

```
globale Berechtigungen
ODER (Datenbankberechtigungen UND Hostberechtigungen)
ODER Tabellenberechtigungen
ODER Spaltenberechtigungen
```

Vielleicht ist es nicht offensichtlich, warum der Server bei anfänglich als unzureichend herausgefundenen globalen `user`-Eintragsberechtigungen für die verlangte Operation diese Berechtigungen anschließend zu den Datenbank-, Tabellen- und Spalten-spezifischen Berechtigungen hinzuzählt. Der Grund liegt darin, dass eine Anfrage möglicherweise mehr als eine Sorte von Berechtigungen erfordert. Wenn Sie beispielsweise ein `INSERT ... SELECT`-Statement ausführen, brauchen Sie eventuell sowohl die **insert**- als auch die **select**-Berechtigung. Ihre Berechtigungen mögen so sein, dass der `user`-Tabelleneintrag eine Berechtigung enthält und der `db`-Tabelleneintrag die andere. In diesem Fall haben Sie die notwendigen Berechtigungen, die Anfrage auszuführen, aber das Server kann das nicht aus nur einer der beiden Tabellen heraus erkennen, sondern muss dafür die Einträge beider Tabellen kombinieren.

Die `host`-Tabelle kann benutzt werden, um eine Liste sicherer Server zu pflegen.

Bei TcX enthält die `host`-Tabelle eine Liste aller Maschine des lokalen Netzwerks. Diesen werden alle Berechtigungen gewährt.

Sie können die `host`-Tabelle auch dazu benutzen, die Host aufzuführen, die *nicht* sicher sind. Nehmen Sie an, Sie haben eine Maschine `oeffentlich.ihre.domaene`, die an einem öffentlichen Ort ist, den Sie als nicht sicher erachten. Sie können allen Hosts in Ihrem Netzwerk Zugriff gewähren ausser dieser Maschine, indem Sie die `host`-Tabelleneinträge wie folgt benutzen:

```
+--------------------------+----+-
| Host                     | Db | ...
+--------------------------+----+-
| oeffentlich.ihre.domane  | %  | ... (alle Berechtigungen auf 'N' gesetzt)
| %.ihre.domaene           | %  | ... (alle Berechtigungen auf 'Y' gesetzt)
+--------------------------+----+-
```

Natürlich sollten Sie Ihre Einträge in die Berechtigungstabellen immer testen (indem Sie zum Beispiel `mysqlaccess` benutzen), um sicherzustellen, dass Ihre Zugriffsberechtigungen tatsächlich so gesetzt sind, wie Sie denken.

5.2.10. Gründe für `Access denied`-Fehler

Wenn Sie beim Verbindungsversuch zu einem MySQL-Server `Access denied`-Fehler bekommen, gibt Ihnen die folgende Liste ein paar Hinweise, das Problem zu beheben:

- Haben Sie nach der Installation von MySQL das `mysql_install_db`-Skript laufen lassen, um die anfänglichen Berechtigungstabelleninhalte zu konfigurieren? Wenn nicht, tun Sie das! See Abschnitt 5.3.4, „Einrichtung der anfänglichen MySQL-Berechtigungen“. Testen Sie die anfänglichen Berechtigungen, indem Sie folgenden Befehl ausführen:

  ```
  shell> mysql -u root test
  ```

  Der Server sollte die Verbindung ohne Fehlermeldung zulassen. Stellen Sie auch sicher, dass Sie eine Datei `user.MYD` im MySQL-Datenbankverzeichnis haben. Üblicherweise ist das `PFAD/var/mysql/user.MYD`, wobei `PFAD` der Pfadname zum MySQL-Installationsverzeichnis ist.

- Nach einer gerade durchgeführten Installation sollten Sie sich mit dem Server verbinden und Ihre Benutzer und deren Zugriffsberechtigungen einrichten:

  ```
  shell> mysql -u root mysql
  ```

  Der Server sollte die Verbindung zulassen, weil der MySQL-`root`-Benutzer anfänglich kein Passwort hat. Das ist ein Sicherheitsrisiko, daher sollten Sie das `root`-Passwort einrichten, während Sie Ihre anderen MySQL-Benutzer einrichten.

  Wenn Sie versuchen, sich als `root` zu verbinden, und folgenden Fehler erhalten:

  ```
  Access denied for user: '@unknown' to database mysql
  ```

  heißt das, dass Sie in der `user`-Tabelle keinen Eintrag `'root'` im `User`-Spaltenwert haben und dass `mysqld` den

Hostnamen für Ihren Client nicht auflösen kann. In diesem Fall müssen Sie den Server mit der `--skip-grant-tables`-Option neu starten und Ihrer `/etc/hosts`- oder `\windows\hosts`-Datei einen Eintrag für Ihren Host hinzufügen.

- Wenn Sie einen Fehler wie folgt erhalten:

```
shell> mysqladmin -u root -pxxxx ver
Access denied for user: 'root@localhost' (Using password: YES)
```

bedeutet das, dass Sie ein falsches Passwort benutzen. See Abschnitt 5.3.7, „Passwörter einrichten“.

Wenn Sie das Root-Passwort vergessen haben, können Sie `mysqld` mit `--skip-grant-tables` neu starten, um das Passwort zu ändern. Diese Option wird weiter hinten im Handbuch ausführlicher beschrieben.

Wenn Sie den obigen Fehler erhalten, obwohl Sie kein Passwort angegeben haben, bedeutet das, dass in einer der `my.ini`-Dateien ein falsches Passwort steht. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“. Sie können die Benutzung der Optionsdateien mit der `--no-defaults`-Option wie folgt verhindern:

```
shell> mysqladmin --no-defaults -u root ver
```

- Wenn Sie eine bestehende MySQL-Installation von einer Version vor 3.22.11 auf Version 3.22.11 oder später aktualisiert haben, haben Sie das `mysql_fix_privilege_tables`-Skript ausgeführt? Falls nicht, tun Sie das! Die Struktur der Berechtigungstabellen hat sich ab MySQL-Version 3.22.11 geändert, als das `GRANT`-Statement mit Funktion erfüllt wurde.
- Falls es aussieht, als hätten sich Ihre Berechtigungen mitten in einer Sitzung geändert, kann es sein, dass ein Superuser sie geändert hat. Das Neuladen der Berechtigungstabellen betrifft neue Client-Verbindungen, aber auch bestehende Verbindungen, wie in Abschnitt 5.3.3, „Wann Berechtigungsänderungen wirksam werden“ beschrieben.
- Wenn Sie es nicht schaffen, dass Ihr Passwort funktioniert, denken Sie daran, dass Sie die `PASSWORD()`-Funktion benutzen müssen, wenn Sie das Passwort mit den `INSERT`-, `UPDATE`- oder `SET PASSWORD`-Statements setzen. Die `PASSWORD()`-Funktion wird nicht benötigt, wenn Sie das Passwort mit dem `GRANT ... INDENTIFIED BY`-Statement oder dem `mysqladmin password`-Befehl setzen. See Abschnitt 5.3.7, „Passwörter einrichten“.
- `localhost` ist ein Synonym für Ihren lokalen Hostnamen und gleichzeitig der vorgabemäßige Host, mit dem sich Clients versuchen zu verbinden, wenn Sie nicht explizit einen Hostnamen angeben. Verbindungen zu `localhost` funktionieren jedoch nicht, wenn Sie auf einem System arbeiten, das MIT-pThreads benutzt (`localhost`-Verbindungen werden über Unix-Sockets hergestellt, die von MIT-pThreads nicht unterstützt werden). Um auf solchen Systemen Probleme zu vermeiden, sollten Sie die `--host`-Option zu benutzen, um den Serverhost explizit anzugeben. Das stellt eine TCP/IP-Verbindung zum `mysqld`-Server her. In diesem Fall muss Ihr echter Hostname in den `user`-Tabelleneinträgen auf dem Server-Host stehen. (Das gilt sogar dann, wenn Sie ein Client-Programm auf demselben Host fahren, wo der Server läuft.)
- Wenn Sie beim Versuch, sich mit `mysql -u user_name db_name` mit einer Datenbank zu verbinden, einen `Access denied`-Fehler erhalten, gibt es eventuell ein Problem mit der `user`-Tabelle. Das können Sie überprüfen, indem Sie `mysql -u root mysql` und folgendes SQL-Statement absetzen:

```
mysql> SELECT * FROM user;
```

Das Ergebnis sollte einen Eintrag enthalten, in dem die `Host`- und `User`-Spalten mit dem Hostnamen Ihres Computers und Ihrem MySQL-Benutzernamen übereinstimmen.

- Die `Access denied`-Fehlermeldung sagt Ihnen, als wer Sie sich versuchen einzuloggen, den Host, von dem aus Sie versuchen, sich zu verbinden, und ob Sie ein Passwort benutzen oder nicht. Normalerweise sollten Sie in der `user`-Tabelle einen Eintrag haben, der exakt mit Ihrem Hostnamen und Ihrem Benutzernamen übereinstimmt, die in der Fehlermeldung ausgegeben wurden. Wenn Sie zum Beispiel eine Fehlermeldung erhalten, die `Using password: NO` enthält, bedeutet das, dass Sie versuchen sich einzuloggen, ohne ein Passwort anzugeben.
- Wenn Sie folgenden Fehler erhalten, wenn Sie sich von einem anderen Host als dem, auf dem der MySQL-Server läuft, zu verbinden, gibt es keine Zeile in der `user`-Tabelle, die mit Ihrem Host übereinstimmt:

```
Host ... is not allowed to connect to this MySQL server
```

Das können Sie mit dem Kommandozeilentool `mysql` beheben (auf dem Serverhost!) und eine Zeile zur `user`-, `db`- oder `host`-Tabelle hinzufügen, die eine Benutzername-/Hostname-Kombination enthält, von wo aus Sie sich verbinden wollen; danach führen Sie `mysqladmin flush-privileges` aus. Wenn Sie nicht MySQL-Version 3.22 laufen lassen und die IP-Nummer oder den Hostnamen der Maschine nicht kennen, von der aus Sie sich verbinden, sollten Sie einen Eintrag mit `'%'` als `Host`-Spaltenwert in die `user`-Tabelle einfügen und `mysqld` mit der `--log`-Option auf der Servermaschine neu starten. Nach dem Verbinden von der Client-Maschine aus zeigt die Information im MySQL-Log an, wie Sie sich wirklich verbunden haben. (Ersetzen Sie danach `'%'` im `user`-Tabelleneintrag durch den tatsächlichen Hostnamen, der im Log steht. Ansonsten erhalten Sie ein System, das unsicher ist.)

Ein weiterer Grund für diesen Fehler unter Linux kann sein, dass Sie eine Binärversion von MySQL benutzen, die mit einer anderen glibc-Version kompiliert wurde als die, die Sie benutzen. In diesem Fall sollten Sie entweder die glibc Ihres Betriebssystems aktualisieren oder die Quellversion von MySQL herunter laden und sie selbst kompilieren. Ein Quell-RPM läßt sich normalerweise sehr einfach kompilieren und installieren, daher stellt dies kein großes Problem dar.

- Wenn Sie eine Fehlermeldung erhalten, in der der Hostname nicht angezeigt wird oder eine IP-Nummer ist, obwohl Sie sich mit einem Hostnamen versuchen zu verbinden:

```
shell> mysqladmin -u root -pxxxx -h ein-hostname ver
Access denied für user: 'root@' (Using password: YES)
```

  bedeutet das, dass MySQL einen Fehler beim Auflösen der IP zu einem Hostnamen erhielt. In diesem Fall können Sie `mysqladmin flush-hosts` ausführen, um den internen DNS-Cache zu flushen. See Abschnitt 6.5.5, „Wie MySQL DNS benutzt“.

  Einige dauerhafte Lösungen sind:

  - Versuchen Sie herauszufinden, was mit Ihrem DNS-Server nicht funktioniert, und beheben Sie das Problem.
  - Geben Sie in den MySQL-Berechtigungstabellen IP-Nummern statt Hostnamen an.
  - Starten Sie `mysqld` mit `--skip-name-resolve`.
  - Starten Sie `mysqld` mit `--skip-host-cache`.
  - Verbinden Sie sich zu `localhost` wenn Sie Server und Client auf derselben Maschine laufen lassen.
  - Tragen Sie die Client-Maschinennamen in `/etc/hosts` ein.

- Wenn `mysql -u root test` funktioniert, aber `mysql -h your_hostname -u root test` zu `Access denied` führt, haben Sie eventuell nicht den korrekten Namen Ihres Hosts in der `user`-Tabelle. Ein häufiges Problem hierbei ist, dass der `Host`-Wert im `user`-Tabelleneintrag einen unqualifizierten Hostnamen festlegt, die Namensauflösungsroutinen Ihres Systems aber einen voll qualifizierten Domänennamen zurückgeben (oder umgekehrt). Wenn Sie zum Beispiel einen Eintrag mit dem Host `'tcx'` in der `user`-Tabelle haben, Ihr DNS MySQL aber mitteilt, dass Ihr Hostname `'tcx.subnet.se'` ist, funktioniert der Eintrag nicht. Fügen Sie der `user`-Tabelle einen Eintrag hinzu, der die IP-Nummer Ihres Hosts als `Host`-Spaltenwert enthält. (Alternativ könnten Sie der `user`-Tabelle einen Eintrag mit einem `Host`-Wert hinzufügen, der einen Platzhalter enthält, zum Beispiel `'tcx.%'`. Allerdings ist die Benutzung von Hostnamensendungen mit ‘`%`’ *unsicher* und wird daher *nicht* empfohlen!)

- Wenn `mysql -u benutzername test` funktioniert, aber `mysql -u benutzername andere_datenbank` nicht, haben Sie wahrscheinlich keinen Eintrag für `andere_datenbank` in der `db`-Tabelle.

- Wenn `mysql -u benutzername datenbankname` funktioniert, wenn es auf der Servermaschine ausgeführt wird, aber `mysql -u hostname -u benutzername datenbankname` nicht, wenn es auf einer anderen Clientmaschine ausgeführt wird, ist die Clientmaschine wahrscheinlich nicht in der `user`-Tabelle oder der `db`-Tabelle aufgeführt.

- Wenn Sie gar nicht herausfinden können, warum Sie `Access denied` erhalten, entfernen Sie aus der `user`-Tabelle alle Einträge, die `Host`-Werte haben, die Platzhalter enthalten (Einträge, die ‘`%`’ oder ‘`_`’ enthalten). Ein sehr häufiger Fehler besteht darin, einen neuen Eintrag mit `Host='%'` und `User='irgendein_benutzer'` in der Annahme hinzuzufügen, dass einem das erlaubt, `localhost` anzugeben, um sich von derselben Maschine aus zu verbinden. Der Grund, warum das nicht funktioniert, ist, dass die vorgabemäßigen Berechtigungen einen Eintrag mit `Host='localhost'` und `User=''` enthalten. Weil dieser Eintrag einen `Host`-Wert `'localhost'` hat, der spezifischer ist als `'%'`, wird er vorrangig vor dem neuen Eintrag benutzt, wenn man sich von `localhost` verbindet! Das korrekte Vorgehen ist, einen zweiten Eintrag mit `Host='localhost'` und `User='irgendein_benutzer'` hinzuzufügen, oder den Eintrag mit `Host='localhost'` und `User=''` zu entfernen.

- Wenn Sie den folgenden Fehler erhalten, gibt es eventuell Probleme mit der `db`- oder der `host`-Tabelle:

```
Access to database denied
```

  Wenn der aus der `db`-Tabelle ausgewählte Eintrag einen leeren Wert in der `Host`-Spalte hat, stellen Sie sicher, dass es einen oder mehrere korrespondierende Einträge in der `host`-Tabelle gibt, die festlegen, auf welche Hosts der `db`-Tabelleneintrag zutrifft.

  Wenn Sie bei der Benutzung der SQL-Befehle `SELECT ... INTO OUTFILE` oder `LOAD DATA INFILE` einen Fehler erhalten, enthält Ihr Eintrag in der `user`-Tabelle wahrscheinlich keine angeschaltete **file**-Berechtigung.

- Denken Sie daran, dass Client-Programme Verbindungsparameter benutzen, die in Konfigurationsdateien oder Umgebungsvariablen festgelegt sind. See Anhang F, *Umgebungsvariablen*. Wenn ein Client anscheinend falsche

vorgabemäßige Verbindungsparameter sendet, wenn Sie diese nicht auf der Kommandozeile angeben, überprüfen Sie Ihre Umgebung und die `.my.cnf`-Datei in Ihrem Heimatverzeichnis. Überprüfen Sie gegebenenfalls auch systemweite MySQL-Konfigurationsdateien, obwohl es sehr viel unwahrscheinlicher ist, dass Client-Verbindungsparameter in diesen festgelegt werden. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“. Wenn Sie beim Laufenlassen eines Clients ohne irgend welche Optionen `Access denied` erhalten, stellen Sie sicher, dass Sie kein altes Passwort in irgendeiner Optionsdatei angegeben haben! See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

- Wenn Sie in den Berechtigungstabellen direkte Änderungen vornehmen (indem Sie ein `INSERT`- oder `UPDATE`-Statement benutzen) und Ihre Änderungen anscheinend ignoriert werden, denken Sie daran, dass sie ein `FLUSH PRIVILEGES`-Statement absetzen müssen oder einen `mysqladmin flush-privileges`-Befehl ausführen, um den Server zu veranlassen, die Berechtigungstabellen neu einzulesen. Ansonsten haben Ihre Änderungen keine Auswirkung, bis der Server das nächste Mal gestartet wird. Denken Sie auch daran, wenn Sie ein `root`-Passwort mit einem `UPDATE`-Befehl festgelegt haben, dass Sie dieses solange nicht angeben müssen, bis Sie die Berechtigungen flushen, weil der Server vorher nicht weiß, dass Sie Ihr Passwort geändert haben!

- Wenn Sie Zugriffsprobleme mit einem Perl-, PHP-, Python- oder ODBC-Programm haben, versuchen Sie, sich mit `mysql -u benutzername datenbankname` oder `mysql -u benutzername -pihr_passwort datenbankname` zu verbinden. Wenn es Ihnen gelingt, sich mittels des `mysql`-Clients zu verbinden, gibt es ein Problem mit Ihrem Programm und nicht mit den Zugriffsberechtigungen. (Beachten Sie, dass zwischen `-p` und dem Passwort kein Leerzeichen steht; alternativ können Sie auch die `--password=ihr_passwort`-Syntax benutzen, um Ihr Passwort anzugeben. Wenn Sie die `-p`-Option allein benutzen, wird MySQL eine Eingabeaufforderung für das Passwort anzeigen.)

- Zum Testen starten Sie den `mysqld`-Daemon mit der `--skip-grant-tables`-Option. Anschließend können Sie die MySQL-Berechtigungstabellen ändern und das `mysqlaccess`-Skript benutzen, um zu sehen, ob Ihre Änderungen den gewünschten Effekt haben oder nicht. Wenn Sie mit Ihren Änderungen zufrieden sind, führen Sie `mysqladmin flush-privileges` aus, um `mysqld` mitzuteilen, die neuen Berechtigungstabellen zu benutzen. **Beachten Sie:** Das Neuladen der Berechtigungstabellen überschreibt die `--skip-grant-tables`-Option. Das erlaubt Ihnen, den Server zu veranlassen, die Berechtigungstabellen wieder zu benutzen, ohne ihn herunter und dann wieder herauf fahren zu müssen.

- Wenn alles andere fehlschlägt, starten Sie den `mysqld`-Daemon mit einer Debugging-Option (zum Beispiel `--debug=d,general,query`). Das gibt Host- und Benutzerinformationen über Verbindungsversuche aus sowie Informationen über jeden abgesetzten Befehl. See Abschnitt E.1.2, „Trace-Dateien erzeugen“.

- Wenn Sie irgend welche anderen Probleme mit den MySQL-Berechtigungstabellen haben und meinen, das Problem der Mailing-Liste mitteilen zu müssen, stellen Sie immer einen Auszug Ihrer MySQL-Berechtigungstabellen zur Verfügung. Sie können einen Auszug der Tabellen mit dem `mysqldump mysql`-Befehl erzeugen. Berichten Sie Ihr Problem - wie immer - unter Benutzung des `mysqlbug`-Skripts. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“. In einigen Fällen müssen Sie vielleicht `mysqld` mit `--skip-grant-tables` neu starten, um `mysqldump` benutzen zu können.

5.3. MySQL-Benutzerkonten-Verwaltung

5.3.1. `GRANT`- und `REVOKE`-Syntax

```
GRANT berechtigung_art [(spalten_liste)] [, berechtigung_art [(spalten_liste)] ...]
    ON {tabelle | * | *.* | datenbank.*}
    TO benutzername [IDENTIFIED BY 'passwort']
        [, benutzername [IDENTIFIED BY 'passwort'] ...]
    [REQUIRE
            [{SSL| X509}]
        [CIPHER cipher [AND]]
        [ISSUER issuer [AND]]
        [SUBJECT subject]]
    [WITH GRANT OPTION]

REVOKE berechtigung_art [(spalten_liste)] [, berechtigung_art [(spalten_liste)] ...]
    ON {tabelle | * | *.* | datenbank.*}
    FROM benutzername [, benutzername ...]
```

`GRANT` ist implementiert ab MySQL Version 3.22.11. Bei früheren MySQL-Versionen bewirkt das `GRANT`-Statement nichts.

Die `GRANT`- und `REVOKE`-Befehle erlauben Systemverwaltern, Benutzer anzulegen und MySQL-Benutzern Rechte auf vier Berechtigungsebenen zu gewähren und zu entziehen:

- **Globale Ebene**

  Globale Berechtigungen betreffen alle Datenbanken auf einem gegebenen Server. Diese Berechtigungen werden in der `mysql.user`-Tabelle gespeichert.

- **Datenbank-Ebene**

Datenbank-Berechtigungen betreffen alle Tabellen in einer gegebenen Datenbank. Diese Berechtigungen werden in den `mysql.db`- und `mysql.host`-Tabellen gespeichert.

- **Tabellen-Ebene**

  Tabellen-Berechtigungen betreffen alle Spalten in einer gegebenen Tabelle. Diese Berechtigungen werden in der `mysql.tables_priv`-Tabelle gespeichert.

- **Spalten-Ebene**

  Spalten-Berechtigungen betreffen einzelne Spalten in einer gegebenen Tabelle. Diese Berechtigungen werden in der `mysql.columns_priv`-Tabelle gespeichert.

Wenn Sie ein `GRANT` für einen Benutzer angeben, den es nicht gibt, wird dieser Benutzer erzeugt. Beispiele, wie `GRANT` funktioniert, finden Sie unter Abschnitt 5.3.5, „Neue MySQL-Benutzer hinzufügen“.

Bei `GRANT` und `REVOKE`-Statements kann `berechtigung_art` wie folgt angegeben werden:

```
ALL PRIVILEGES      FILE                RELOAD
ALTER               INDEX               SELECT
CREATE              INSERT              SHUTDOWN
DELETE              PROCESS             UPDATE
DROP                REFERENCES          USAGE
```

`ALL` ist ein Synonym für `ALL PRIVILEGES`. `REFERENCES` ist noch nicht implementiert. `USAGE` ist momentan ein Synonym für ``keine Berechtigungen". Es kann benutzt werden, um einen Benutzer zu erzeugen, der keine Berechtigungen hat.

Um einem Benutzer die **grant**-Berechtigung zu entziehen, benutzen Sie einen `berechtigung_art`-Wert `GRANT OPTION`:

```
REVOKE GRANT OPTION ON ... FROM ...;
```

Die einzigen `berechtigung_art`-Werte, die Sie für eine Tabelle festlegen können, sind `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `DROP`, `GRANT`, `INDEX` und `ALTER`.

Die einzigen `berechtigung_art`-Werte, die Sie für eine Spalte festlegen können (im Falle, dass Sie eine `spalten_liste`-Klausel benutzen), sind `SELECT`, `INSERT` und `UPDATE`.

Sie können globale Berechtigungen setzen, indem Sie die `ON *.*`-Syntax benutzen. Datenbank-Berechtigungen setzen Sie mit der `ON datenbank.*`-Syntax. Wenn Sie `ON *` setzen und eine aktuelle Datenbank ausgewählt haben, setzen Sie die Berechtigungen für diese Datenbank. (**ACHTUNG:** Wenn Sie `ON *` festlegen und *keine* aktuelle Datenbank ausgewählt haben, betrifft das die globalen Berechtigungen!)

Um die Rechtegewährung für Benutzer von uneindeutigen Hosts aus zu ermöglichen, unterstützt MySQL den `benutzername`-Wert in der Form `benutzer@host`. Wenn Sie eine `user`-Zeichenkette festlegen wollen, die Sonderzeichen enthält (wie ‘`-`’), oder eine `host`-Zeichenkette, die Sonderzeichen oder Platzhalterzeichen enthält (wie ‘`%`’), können Sie Benutzernamen oder Hostnamen in Anführungszeichen setzen (beispielsweise `'test-benutzer'@'test-hostname'`).

Sie können im Hostnamen Platzhalter angeben. `benutzer@"%.loc.gov"` zum Beispiel trifft auf `benutzer` für jeden Host in der Domäne `loc.gov` zu. `benutzer@"144.155.166.%"` trifft auf `benutzer` für jeden Host im `144.155.166`-Class-C-Subnetz zu.

Die einfache Form `benutzer` ist ein Synonym für `benutzer@"%"`. **ACHTUNG:** Wenn Sie anonymen Benutzern erlauben, sich mit dem MySQL-Server zu verbinden (was vorgabemäßig der Fall ist), sollten Sie auch alle lokalen Benutzer als `benutzer@localhost` hinzufügen, weil ansonsten der Eintrag für den anonymen Benutzer für den lokalen Host in der `mysql.user`-Tabelle benutzt wird, wenn der Benutzer versucht, sich von der lokalen Maschine in den MySQL-Server einzuloggen! Anonyme Benutzer werden definiert, indem Einträge mit `User=''` in die `mysql.user`-Tabelle eingefügt werden. Das können Sie mit folgender Anfrage überprüfen:

```
mysql> SELECT Host,User FROM mysql.user WHERE User='';
```

Momentan unterstützt `GRANT` nur Host-, Datenbank-, Tabellen- und Spaltennamen mit maximal 60 Zeichen. Ein Benutzername kann bis zu 16 Zeichen lang sein.

Die Berechtigungen für eine Tabelle oder Spalte werden durch ein logisches ODER der Berechtigungen auf jeder der vier Berechtigungsebenen zusammen gesetzt. Wenn die `mysql.user`-Tabelle beispielsweise festlegt, dass ein Benutzer eine globalen **select**-Berechtigung hat, kann diese nicht durch Einträge auf Datenbank-, Tabellen- oder Spaltenebene widerrufen werden.

Die Berechtigungen für eine Spalte können wie folgt berechnet werden:

```
Globale Berechtigungen
ODER (Datenbank-Berechtigungen UND Host-Berechtigungen)
ODER Tabellen-Berechtigungen
ODER Spalten-Berechtigungen
```

In den meisten Fällen können Sie einem Benutzer Rechte auf lediglich einer der Berechtigungsebenen gewähren, wodurch das Leben nicht so kompliziert ist wie oben dargestellt. Die Details der Prozedur zur Überprüfung der Berechtigungen sind in Abschnitt 5.2, „Allgemeine Sicherheitsthemen und das MySQL-Zugriffsberechtigungssystem“ dargestellt.

Wenn Sie Berechtigungen für eine Benutzer-/Hostname-Kombination gewähren, die in der `mysql.user`-Tabelle nicht existiert, wird ein Eintrag hinzugefügt und verbleibt dort, bis der mit einem `DELETE`-Befehl gelöscht wird. Mit anderen Worten: `GRANT` kann eventuell `user`-Tabelleneinträge erzeugen, aber `REVOKE` entfernt diese nicht, sondern Sie müssen das explizit mit `DELETE` machen.

Ab MySQL-Version 3.22.12 wird, wenn ein neuer Benutzer erzeugt wird oder wenn Sie globale Grant-Berechtigungen haben, das Passwort des Benutzers durch die `IDENTIFIED BY`-Klausel festgelegt, wenn eine angegeben wird. Wenn der Benutzer bereits ein Passwort hat, wird es durch das neue ersetzt.

ACHTUNG: Wenn Sie einen neuen Benutzer anlegen, aber keine `IDENTIFIED BY`-Klausel angeben, hat der neue Benutzer kein Passwort. Das ist unsicher.

Passwörter können auch mit dem `SET PASSWORD`-Befehl gesetzt werden. See Abschnitt 6.5.6, „`SET`-Syntax“.

Wenn Sie Berechtigungen für eine Datenbank gewähren, wird ein Eintrag in der `mysql.db`-Tabellen erzeugt, falls notwendig. Wenn alle Berechtigungen für die Datenbank mit `REVOKE` widerrufen wurden, wird dieser Eintrag gelöscht.

Wenn ein Benutzer überhaupt keine Berechtigungen auf eine Tabelle hat, wird die Tabelle nicht angezeigt, wenn der Benutzer nach einer Liste von Tabellen anfragt (zum Beispiel mit einem `SHOW TABLES`-Statement).

Die `mit GRANT OPTION`-Klausel gibt dem Benutzer die Möglichkeit, anderen Benutzern jegliche der Berechtigungen zu vergeben, die der Benutzer auf der angegebenen Berechtigungsebene hat. Sie sollten vorsichtig damit sein, wem Sie die **grant**-Berechtigung geben, denn zwei Benutzer mit unterschiedlichen Berechtigungen können in der Lage sein, Ihre Berechtigungen zu addieren!

Sie können einem Benutzer keine Berechtigung gewähren, die Sie selbst nicht haben; die **grant**-Berechtigung erlaubt Ihnen nur, die Berechtigungen zu vergeben, die Sie selbst besitzen.

Wenn Sie einem Benutzer die **grant**-Berechtigung auf einer bestimmten Berechtigungsebene geben, denken Sie daran, dass der Benutzer jegliche Berechtigungen, die der Benutzer schon besitzt (oder die ihm in Zukunft gewährt werden!), auf dieser Ebene auch an andere Benutzer gewährt werden können. Nehmen Sie an, Sie gewähren einem Benutzer die **insert**-Berechtigung auf eine Datenbank. Wenn Sie danach die **select**-Berechtigung auf die Datenbank mit `WITH GRANT OPTION` gewähren, kann der Benutzer nicht nur die **select**-Berechtigung weiter geben, sondern auch **insert**. Wenn Sie dem Benutzer danach die **update**-Berechtigung auf die Datenbank gewähren, kann der Benutzer insgesamt **insert**, **select** und **update** weiter geben.

Sie sollten einem normalen Benutzer keine **alter**-Berechtigung gewähren. Wenn Sie das tun, kann der Benutzer versuchen, das Berechtigungssystem zu unterlaufen, indem er Tabellen umbenennt!

Beachten Sie: Wenn Sie Tabellen- oder Spalten-Berechtigungen auch nur für einen Benutzer gewähren, untersucht der Server Tabellen- und Spalten-Berechtigungen für alle Benutzer. Dadurch wird MySQL etwas langsamer.

Wenn `mysqld` startet, werden alle Berechtigungen in den Speicher eingelesen. Datenbank-, Tabellen- und Spalten-Berechtigungen werden sofort wirksam. Berechtigungen auf Benutzerebene werden wirksam, wenn sich der Benutzer das nächste Mal verbindet. Änderungen in den Berechtigungstabellen, die Sie mit `GRANT` oder `REVOKE` durchführen, werden vom Server sofort bemerkt. Wenn Sie Berechtigungstabellen manuell ändern (mit `INSERT`, `UPDATE` usw.), müssen Sie ein `FLUSH PRIVILEGES`-Statement ausführen oder `mysqladmin flush-privileges` laufen lassen, um den Server zu veranlassen, die Berechtigungstabellen neu zu laden. See Abschnitt 5.3.3, „Wann Berechtigungsänderungen wirksam werden“.

Die größten Unterschiede zwischen ANSI SQL und MySQL-Versionen von `GRANT` sind:

- In MySQL werden Berechtigungen für eine Benutzername-/Hostname-Kombination vergeben und nicht nur für einen Benutzernamen.
- ANSI SQL hat keine globalen oder Datenbankebene-Berechtigungen und unterstützt nicht alle Berechtigungsarten, die MySQL unterstützt. MySQL unterstützt nicht die ANSI-SQL-`TRIGGER`-, `EXECUTE`- oder `UNDER`-Berechtigungen.
- ANSI-SQL-Berechtigungen werden auf hierarchische Art strukturiert. Wenn Sie einen Benutzer entfernen, werden alle Berechtigungen, die dieser Benutzer gewährt hat, widerrufen. In MySQL werden die gewährten Berechtigungen nicht automatisch widerrufen, sondern Sie müssen das selbst tun.
- Wenn Sie in MySQL das `INSERT`-Recht nur für Teile der Spalten einer Tabelle haben, können Sie dennoch `INSERT`-

Statements auf der Tabelle ausführen. Die Spalten, für die Sie keine `INSERT`-Berechtigung haben, werden auf ihre Vorgabewerte gesetzt. ANSI SQL erfordert, dass Sie die `INSERT`-Berechtigung auf alle Spalten haben.

- Wenn Sie eine Tabelle in ANSI SQL löschen, werden alle Berechtigungen für die Tabelle widerrufen. Wenn Sie eine Berechtigung in ANSI SQL widerrufen, werden alle Berechtigungen, die auf dieser Berechtigung basierend gewährt wurden, widerrufen. In MySQL können Berechtigungen nur explizit mit `REVOKE`-Befehlen oder durch die Manipulation der MySQL-Berechtigungstabellen widerrufen werden.

---------- MySQL unterstützt SSL-verschlüsselte Verbindungen. Um zu verstehen, wie MySQL SSL benutzt, müssen wir einige Grundlagen von SSL und X509 erläutern. Leute, die damit schon vertraut sind, können dieses Kapitel überspringen.

Vorgabemäßig benutzt MySQL unverschlüsselte Verbindungen zwischen Client und Server. Das heißt, dass jeder auf dem Weg dazwischen lauschen und Ihre Daten, die übertragen werden, mitlesen kann. Darüber hinaus könnten einige Leute auch den Inhalt von Daten ändern, die zwischen Client und Server ausgetauscht werden. Möglicherweise haben Sie auch wirklich geheime Daten über öffentliche Netzwerke zu übertragen, und eine Öffentlichkeit solcher Art ist unakzeptabel.

SSL ist ein Protokoll, das unterschiedliche Verschlüsselungsalgorithmen benutzt, um sicherzustellen, dass Daten aus einem öffentlichen Netzwerk vertraut werden kann. Es besitzt Mechanismen, um Veränderungen, Verlust oder wiederholtes Abspielen (Replay) von Daten zu entdecken. SSL enthält auch Algorithmen, um die Identität zu erkennen und zu überprüfen, indem der X509-Standard benutzt wird.

Mittels Verschlüsselung werden jegliche Arten von Daten unlesbar gemacht. Darüber hinaus werden in der heutigen Praxis Verschlüsselungsalgorithmen viele weitere Elemente hinzugefügt. Sie sollten vielen Arten bekannter Angriffe widerstehen, wie dem Herumspielen mit der Reihenfolge verschlüsselter Nachrichten oder dem doppelten Abspielen (Replay) von Daten.

X509 ist der Standard, der es ermöglicht, jemanden im Internet zu identifizieren. Er wird meistens beim E-Commerce über das Internet benutzt. Kurz gesagt sollte es ein Unternehmen namens "Zertifizierungsautorität" geben, die jedem elektronische Zertifikate zuordnet, der diese braucht. Zertifikate beruhen auf asymmetrischen Verschlüsselungsalgorithmen, die zwei Verschlüsselungsschlüssel haben - öffentlichen und geheimen. Zertifikatsbesitzer können ihre Identität jeder anderen Seite beweisen. Zertifikate beinhalten den öffentlichen Schlüssel des Besitzers. Alle Daten, die damit verschlüsselt werden, können nur vom Besitzer des geheimen Schlüssels entschlüsselt werden.

Frage: Warum benutzt MySQL nicht standardmäßig verschlüsselte Verbindungen? Antwort: Weil es MySQL langsamer macht. Jede zusätzliche Funktionalität erfordert, dass ein Computer zusätzliche Arbeit verrichtet, und das Verschlüsseln von Daten ist eine CPU-intensive Operation, die leicht die Zeit und Leistung übertreffen kann, die MySQL selbst verbraucht und benötigt. MySQL ist vorgabemäßig auf Geschwindigkeit optimiert. Frage: Ich brauche mehr Informationen über SSL / X509 / Verschlüsselung usw. Antwort: Benutzen Sie Ihre bevorzugte Internet-Suchmaschine und suchen Sie nach den Schlüsselwörtern, die Sie interessieren.

MySQL kann x509-Zertifikat-Attribute prüfen, zusätzlich zum meist benutzten Benutzername-/Passwort-Schema. Alle gewöhnlich Optionen werden immer noch benötigt (Benutzername, Passwörter, IP-Adressmaske, Datenbank-/Tabellenname).

Es gibt verschiedene Möglichkeiten, Verbindungen zu begrenzen:

- Ohne jegliche SSL-/X509-Optionen werden alle Arten verschlüsselter und unverschlüsselter Verbindungen zugelassen, wenn Benutzername und Passwort gültig sind.
- Die `REQUIRE SSL`-Option erzwingt SSL-verschlüsselte Verbindungen. Beachten Sie, dass dieses Erfordernis übergangen werden kann, wenn es irgend welche weiteren ACL-Datensätze gibt, die Verbindungen ohne SSL zulassen.

  Beispiel:

  ```
  GRANT ALL PRIVILEGES ON test.* TO root@localhost IDENTIFIED BY "goodsecret" REQUIRE SSL
  ```

- * `REQUIRE X509` Wenn ein X509-Zertifikat erforderlich ist, bedeutet das, dass der Client ein gültiges Zertifikat haben muss, aber wir kümmern uns nicht um das genaue Zertifikat, den Herausgeber (Issuer) oder den Betreff (Subject). Die einzige Einschränkung ist, dass es möglich sein sollte, seine Unterschrift (Signature) mit einigen unserer CA-Zertifikate zu überprüfen.

  Beispiel:

  ```
  GRANT ALL PRIVILEGES ON test.* TO root@localhost IDENTIFIED BY "goodsecret" REQUIRE X509
  ```

- `REQUIRE ISSUER issuer` macht Verbindungen restriktiver: Jetzt muss der Client ein gültiges X509-Zertifikat vorlegen, das von einem CA-Issuer herausgegeben wurde. Die Benutzung von X509-Zertifikaten impliziert immer Verschlüsselung, daher wird die Option "SSL" nicht mehr benötigt.

  Beispiel:

  ```
  GRANT ALL PRIVILEGES ON test.* TO root@localhost IDENTIFIED BY "goodsecret" REQUIRE ISSUER "C=FI, ST=Some-State, L=
  ```

- `REQUIRE SUBJECT betreff` erfordert, dass der Client ein gültiges X509-Zertifikat mit dem Betreff "betreff" darauf hat. Wenn der Client ein gültiges Zertifikat hat, was aber einen anderen Betreff besitzt, wird die Verbindung nicht zugelassen.

  Beispiel:

  ```
  GRANT ALL PRIVILEGES ON test.* TO root@localhost IDENTIFIED BY "goodsecret" REQUIRE SUBJECT "C=EE, ST=Some-State, L
  ```

- `REQUIRE CIPHER cipher` wird benötigt um sicherzustellen, dass Chiffrierungen und Schlüssellängen benutzt werden, die stark genug sind. SSL selbst kann schwach sein, wenn alte Algorithmen mit kurzen Verschlüsselungsschlüsseln benutzt werden. Wenn diese Option benutzt wird, können wir exakte Chiffrierungen anfordern, bevor die Verbindung erlaubt wird.

  Beispiel:

  ```
  GRANT ALL PRIVILEGES ON test.* TO root@localhost IDENTIFIED BY "goodsecret" REQUIRE CIPHER "EDH-RSA-DES-CBC3-SHA"
  ```

  Es ist erlaubt, die Optionen in Kombination wie folgt zu benutzen:

  ```
  GRANT ALL PRIVILEGES ON test.* TO root@localhost IDENTIFIED BY "goodsecret"
          REQUIRE SUBJECT "C=EE, ST=Some-State, L=Tallinn, O=MySQL demo client certificate, CN=Tonu Samuel/Email=tonu
          AND ISSUER "C=FI, ST=Some-State, L=Helsinki, O=MySQL Finnland AB, CN=Tonu Samuel/Email=tonu@mysql.com"
          AND CIPHER "EDH-RSA-DES-CBC3-SHA"
  ```

  Es ist aber nicht erlaubt, irgend eine der Optionen doppelt zu benutzen. Nur unterschiedliche Optionen dürfen gemischt werden.

5.3.2. MySQL-Benutzernamen und -Passwörter

Es gibt mehrere Unterschiede in der Art, wie Benutzernamen und Passwörter von MySQL benutzt werden, und der Art, wie sie von Unix oder Windows benutzt werden:

- Benutzernamen, wie sie von MySQL für Authentifizierungszwecke benutzt werden, haben nicht zu tun mit Unix-Benutzernamen (Login-Namen) oder Windows-Benutzernamen. Die meisten MySQL-Clients versuchen sich zwar vorgabemäßig einzuloggen, indem sie den aktuellen Unix-Benutzernamen als den MySQL-Benutzernamen verwenden, aber das geschieht nur aus Gründen der Bequemlichkeit. Client-Programme lassen zu, dass ein anderer Name mit den `-u`- oder `--user`-Optionen angegeben wird. Das bedeutet, dass Sie eine Datenbank nicht auf irgend eine Weise sicher machen können, wenn nicht alle MySQL-Benutzernamen Passwörter haben. Jeder kann versuchen, sich mit dem Server zu verbinden, indem er irgend einen Namen angibt, und wird damit Erfolg haben, wenn er einen Namen angibt, der kein Passwort hat.
- MySQL-Benutzernamen können bis zu 16 Zeichen lang sein; Unix-Benutzernamen sind typischerweise auf 8 Zeichen begrenzt.
- MySQL-Passwörter haben nichts mit Unix-Passwörtern zu tun. Es gibt keine notwendige Verbindungen zwischen dem Passwort, das Sie benutzen, um sich an einer Unix-Maschine anzumelden, und dem Passwort, das Sie benutzen, um auf eine Datenbank auf dieser Maschine zuzugreifen.
- MySQL verschlüsselt Passwörter mit einem anderen Algorithmus als dem, der während des Unix-Login-Prozesses benutzt wird, siehe die Beschreibungen der `PASSWORD()`- und `ENCRYPT()`-Funktionen in Abschnitt 7.3.5.2, „Verschiedene Funktionen“. Beachten Sie, dass trotz der Tatsache, dass das Passwort 'zerhackt' gespeichert wird, es ausreicht, Ihr 'zerhacktes' Passwort zu kennen, um sich am MySQL-Server anmelden zu können!

MySQL-Benutzer und ihre Berechtigungen werden normalerweise mit dem `GRANT`-Befehl erzeugt. See Abschnitt 5.3.1, „`GRANT`- und `REVOKE`-Syntax“.

Wenn Sie sich an einem MySQL-Server mit einem Kommandozeilen-Client anmelden, sollten Sie das Passwort mit `--password=ihr-passwort` eingeben. See Abschnitt 5.2.7, „Verbinden mit dem MySQL-Server“.

```
mysql --user=monty --password=rate_mal datenbankname
```

Wenn Sie möchten, dass der Client eine Eingabeaufforderung für das Passwort präsentiert, sollten Sie `--password` ohne Argument benutzen.

```
mysql --user=monty --password datenbankname
```

Oder in der kurzen Form:

```
mysql -u monty -p datenbankname
```

Beachten Sie, dass in den letzten Beispielen 'datenbankname' **NICHT** das Passwort ist.

Wenn Sie die `-p`-Option zur Eingabe des Passworts benutzen wollen, tun Sie das wie folgt:

```
mysql -u monty -prate_mal datenbankname
```

Auf einigen Systemen kürzt die Bibliothek, die MySQL benutzt, um die Eingabeaufforderung für das Passwort auszugeben, das Passwort auf 8 Zeichen. Intern hat MySQL keine Beschränkung hinsichtlich der Länge des Passworts.

5.3.3. Wann Berechtigungsänderungen wirksam werden

Wenn `mysqld` startet, werden alle Berechtigungstabelleninhalte in den Arbeitsspeicher eingelesen und werden zu diesem Zeitpunkt wirksam.

Änderungen in den Berechtigungstabellen, die mit `GRANT`, `REVOKE` oder `SET PASSWORD` durchgeführt werden, werden unmittelbar vom Server bemerkt.

Wenn Sie die Berechtigungstabellen manuell ändern (mit `INSERT`, `UPDATE` usw.), müssen Sie ein `FLUSH PRIVILEGES`-Statement ausführen oder `mysqladmin flush-privileges` oder `mysqladmin reload` laufen lassen, um den Server anzuweisen, die Berechtigungstabellen neu einzulesen. Ansonsten haben Ihre Änderungen *keine Auswirkung*, bis Sie den Server neu starten. Wenn Sie die Berechtigungstabellen manuell ändern, aber vergessen, die Berechtigungen neu zu laden, werden Sie sich wundern, warum trotz Ihrer Änderungen kein Unterschied zu bemerken ist!

Wenn der Server bemerkt, dass sich die Berechtigungstabellen geändert haben, werden bestehende Client-Verbindungen wie folgt davon betroffen:

- Tabellen- und Spalten-Berechtigungsänderungen werden bei der nächsten Anfrage des Clients wirksam.
- Datenbank-Berechtigungsänderungen werden beim nächsten `USE datenbank`-Befehl wirksam.

Globale Berechtigungsänderungen und Passwortänderungen werden beim nächsten Mal wirksam, wenn sich der Client verbindet.

5.3.4. Einrichtung der anfänglichen MySQL-Berechtigungen

Nach der Installation von MySQL konfigurieren Sie die anfänglichen Zugriffsberechtigungen, indem Sie `scripts/mysql_install_db` laufen lassen. See Abschnitt 3.3.1, „Schnellinstallation, Überblick“. Das `mysql_install_db`-Skript startet den `mysqld`-Server und initialisiert dann die Berechtigungstabellen, so dass diese folgenden Satz an Berechtigungen enthalten:

- Der MySQL-`root`-Benutzer wird als Superuser angelegt, der alles tun darf. Verbindungen müssen vom lokalen Host aus gemacht werden.

  HINWEIS: Das anfängliche `root`-Passwort ist leer, daher kann sich jeder als `root` *ohne Passwort* verbinden und hat alle Berechtigungen.

- Ein anonymer Benutzer wird erzeugt, der mit Datenbanken, die den Namen `'test'` haben oder mit `'test_'` anfangen, alles tun darf. Verbindungen müssen vom lokalen Host aus gemacht werden. Das heißt, dass sich jeder lokale Benutzer ohne Passwort verbinden kann und als anonymer Benutzer behandelt wird.
- Andere Berechtigungen werden verweigert. Beispielsweise können normale Benutzer nicht `mysqladmin shutdown` oder `mysqladmin processlist` benutzen.

HINWEIS: Die vorgabemäßigen Berechtigungen sind unter Windows anders. See Abschnitt 3.6.2.3, „MySQL auf Windows laufen lassen“.

Weil Ihre Installation anfangs weit offen ist, sollten Sie als eins der ersten Dinge ein Passwort für den MySQL-`root`-Benutzer anlegen. Das können Sie wie folgt tun (beachten Sie, dass das Passwort mit der `PASSWORD()`-Funktion angegeben wird):

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('neues_passwort')
           WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

Ab MySQL-Version 3.22 können Sie das `SET PASSWORD`-Statement benutzen:

```
shell> mysql -u root mysql
mysql> SET PASSWORD FOR root=PASSWORD('neues_passwort');
```

Eine weitere Möglichkeit, das Passwort zu setzen, besteht in der Benutzung des `mysqladmin`-Befehls:

```
shell> mysqladmin -u root password neues_passwort
```

Nur Benutzer mit Schreib-/Aktualisierungszugriff auf die `mysql`-Datenbank können das Passwort für andere Benutzer ändern. Alle normalen Benutzer (nicht anonyme Benutzer) können nur ihr eigenes Passwort ändern, entweder mit einem der obigen Befehle oder mit `SET PASSWORD=PASSWORD('neues_passwort')`.

Denken Sie daran, wenn Sie das Passwort in der `user`-Tabelle direkt mit der ersten Methode ändern, dass Sie den Server anweisen müssen, die Berechtigungstabellen neu einzulesen (mit `FLUSH PRIVILEGES`), weil die Änderungen ansonsten nicht wahrgenommen werden.

Sobald das `root`-Passwort gesetzt wurde, müssen Sie in der Folge immer das Passwort angeben, wenn Sie sich als `root` mit dem Server verbinden.

Eventuell wollen Sie das `root`-Passwort leer lassen, damit Sie es für die weitere Konfiguration oder für Tests nicht angeben müssen. Stellen Sie jedoch sicher, dass Sie es setzen, bevor Sie Ihre Installation für irgend welche Produktionsaufgaben benutzen.

Sehen Sie im `scripts/mysql_install_db`-Skript nach, wie es die vorgabemäßigen Berechtigungen installiert. Sie können das als Grundlage für das Hinzufügen weiterer Benutzer nehmen.

Wenn Sie wollen, dass die anfänglichen Berechtigungen anders sind als die gerade beschriebenen, können Sie `mysql_install_db` abändern, bevor Sie es benutzen.

Um die Berechtigungstabellen komplett neu zu erzeugen, entfernen Sie alle `.frm`-, `.MYI`- und `.MYD`-Dateien im Verzeichnis, das die `mysql`-Datenbank enthält. (Das ist das Verzeichnis namens `mysql` unter dem Datenbank-Verzeichnis, was aufgelistet wird, wenn Sie `mysqld --help` laufen lassen.) Lassen Sie dann das `mysql_install_db`-Skript laufen, eventuell nachdem Sie es editiert haben, um die Berechtigungen zu enthalten, die Sie haben wollen.

HINWEIS: Bei MySQL-Versionen vor Version 3.22.10 sollten Sie die `.frm`-Dateien NICHT löschen. Wenn Sie das versehentlich doch tun, müssen Sie sie aus Ihrer MySQL-Distribution zurück kopieren, bevor Sie `mysql_install_db` laufen lassen.

5.3.5. Neue MySQL-Benutzer hinzufügen

Sie können Benutzer auf zwei Arten hinzufügen: Indem Sie `GRANT`-Statements verwenden oder indem Sie die MySQL-Berechtigungstabellen direkt verändern. Die bevorzugte Methode ist, `GRANT`-Statements zu benutzen, denn sie sind präziser und weniger fehleranfällig. See Abschnitt 5.3.1, „GRANT- und REVOKE-Syntax“.

Ausserdem gibt es eine Menge von Dritten beigesteuerte Programme wie `phpmyadmin`, die benutzt werden können, um Benutzer zu erzeugen und zu verwalten.

Die unten stehenden Beispiele zeigen, wie man den `mysql`-Client benutzt, um neue Benutzer zu erzeugen. Die Beispiele setzen voraus, dass Berechtigungen mit den Vorgabewerten eingerichtet wurden, die im vorherigen Abschnitt beschrieben wurden. Um also Änderungen machen zu können, müssen Sie sich von derselben Maschine aus verbinden, wo `mysqld` läuft, und Sie müssen sich als MySQL-`root`-Benutzer verbinden, und der `root`-Benutzer muss die **insert**-Berechtigung für die `mysql`-Datenbank und die **reload**-Verwaltungsberechtigung haben. Wenn Sie bereits das `root`-Benutzerpasswort geändert haben, müssen Sie es für die unten stehenden `mysql`-Befehle eingeben.

Sie fügen neue Benutzer mit `GRANT`-Statements hinzu:

```
shell> mysql --user=root mysql
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
           IDENTIFIED BY 'ein_passwort' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@"%"
           IDENTIFIED BY 'ein_passwort' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO admin@localhost;
mysql> GRANT USAGE ON *.* TO dummy@localhost;
```

Diese `GRANT`-Statements richten drei neue Benutzer ein:

- `monty`

  Einen echten Superuser, der sich von irgendwo her mit dem Server verbinden kann, aber das Passwort `'ein_passwort'` dafür verwenden muss. Beachten Sie, dass man `GRANT`-Statements sowohl für `monty@localhost` als auch für `monty@"%"` verwenden muss. Wenn man keinen Eintrag mit `localhost` hinzufügt, hat der Eintrag für den anonymen Benutzer für `localhost` Vorrang, der durch `mysql_install_db` angelegt wird, wenn man sich vom lokalen Host aus

verbindet, weil dieser einen spezifischeren `Host`-Feldwert hat und daher früher in der `user`-Tabellen-Sortierreihenfolge auftaucht.

- `admin`

  Ein Benutzer, der sich ohne Passwort von `localhost` aus verbinden kann und der die **reload**- und **process**-Verwaltungsberechtigungen hat. Das erlaubt dem Benutzt, die `mysqladmin reload`-, `mysqladmin refresh`- und `mysqladmin flush-*`-Befehle sowie `mysqladmin processlist` auszuführen. Es werden keine Datenbank-bezogenen Berechtigungen gewährt. (Diese können später gewährt werden, indem zusätzliche `GRANT`-Statements ausgeführt werden.)

- `dummy`

  Ein Benutzer, der sich ohne Passwort verbinden kann, aber nur vom lokalen Host aus. Die globalen Berechtigungen sind alle auf `'N'` gesetzt - diese `USAGE`-Berechtigung erlaubt Ihnen, einen Benutzer ohne Berechtigungen anzulegen. Es wird angenommen, dass Sie später Datenbank-spezifische Berechtigungen gewähren.

Sie können dieselben Benutzerzugriffsinformationen direkt mittels `INSERT`-Statements eingeben und dann den Server anweisen, die Berechtigungstabellen neu zu laden:

```
shell> mysql --user=root mysql
mysql> INSERT INTO user VALUES('localhost','monty',PASSWORD('ein_passwort'),
                'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user VALUES('%','monty',PASSWORD('ein_passwort'),
                 'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user SET Host='localhost',User='admin',
                  Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
                       VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

Abhängig von Ihrer MySQL-Version müssen Sie oben eventuell eine andere Anzahl von `'Y'`-Werten eingeben (Versionen vor Version 3.22.11 hatten weniger Berechtigungsspalten). Beim `admin`-Benutzer wird die besser lesbare `INSERT`-Syntax benutzt, die ab Version 3.22.11 verfügbar ist.

Beachten Sie, dass Sie für die Einrichtung eines Superusers lediglich einen `user`-Tabelleneintrag mit Berechtigungsfeldern einrichten müssen, die auf `'Y'` gesetzt sind. Es sind keine `db`- oder `host`-Tabelleneinträge nötig.

The Berechtigungsspalten in der `user`-Tabelle wurden im letzten `INSERT`-Statement nicht explizit gesetzt (für den Benutzer `dummy`), daher erhalten diese Spalten ihren Vorgabewert von `'N'`. Das ist dasselbe, was `GRANT USAGE` macht.

Das folgende Beispiel fügt einen Benutzer `custom` hinzu, der sich von `localhost`, `server.domain` und `whitehouse.gov` aus verbinden kann. Er will auf die `bankkonto`-Datenbank nur von `localhost` aus zugreifen, auf die `spesen`-Datenbank nur von `whitehouse.gov` aus und auf die `kunde`-Datenbank von allen drei Hosts aus. Er will von allen drei Hosts aus das Passwort `dumm` benutzen.

Um die Berechtigungen dieses Benutzers mit `GRANT`-Statements einzurichten, geben Sie folgende Befehle ein:

```
shell> mysql --user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
           ON bankkonto.*
           TO custom@localhost
           IDENTIFIED BY 'dumm';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
           ON spesen.*
           TO custom@whitehouse.gov
           IDENTIFIED BY 'dumm';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
           ON kunde.*
           TO custom@'%'
           IDENTIFIED BY 'dumm';
```

Der Grund, warum wir Grant-Statements für den Benutzer 'custom' eingeben, ist, dass wir dem Benutzer Zugriff auf MySQL sowohl von der lokalen Maschine mit Unix-Sockets als auch von der entfernten Maschine 'whitehouse.gov' über TCP/IP geben wollen.

Um die Benutzerberechtigungen durch direkte Änderungen an den Berechtigungstabellen einzugeben, geben Sie folgende Befehle ein (beachten Sie das `FLUSH PRIVILEGES` am Ende):

```
shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
       VALUES('localhost','custom',PASSWORD('dumm'));
mysql> INSERT INTO user (Host,User,Password)
       VALUES('server.domain','custom',PASSWORD('dumm'));
mysql> INSERT INTO user (Host,User,Password)
       VALUES('whitehouse.gov','custom',PASSWORD('dumm'));
mysql> INSERT INTO db
```

```
       (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       VALUES
       ('localhost','bankkonto','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
       (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       VALUES
       ('whitehouse.gov','spesen','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
       (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       VALUES('%','kunde','custom','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
```

Die ersten drei `INSERT`-Statements fügen `user`-Tabelleneinträge hinzu, die dem Benutzer `custom` erlauben, sich von den verschiedenen Hosts aus mit dem gegebenen Passwort zu verbinden, gewähren ihm aber keine Berechtigungen (alle Berechtigungen werden auf den Vorgabewert `'N'` gesetzt). Die nächsten drei `INSERT`-Statements fügen `db`-Tabelleneinträge hinzu, die `custom` Berechtigungen für die `bankkonto`-, `spesen`- und `kunde`-Datenbanken gewähren, aber nur, wenn auf sie von den korrekten Hosts aus zugegriffen wird. Wie immer, wenn die Berechtigungstabellen direkt verändert werden, muss dem Server gesagt werden, dass er sie neu laden muss (mit `FLUSH PRIVILEGES`), damit die Berechtigungsänderungen wirksam werden.

Wenn Sie einem bestimmten Benutzer Zugriff von irgendeiner Maschine in einer gegebenen Domäne geben wollen, können Sie ein `GRANT`-Statement wie das folgende absetzen:

```
mysql> GRANT ...
           ON *.*
           TO benutzername@"%.domaene.de"
           IDENTIFIED BY 'passwort';
```

Um dasselbe durch direkte Änderung der Berechtigungstabellen einzugeben, machen Sie folgendes:

```
mysql> INSERT INTO user VALUES ('%.domaene.de', 'benutzername',
           PASSWORD('passwort'),...);
mysql> FLUSH PRIVILEGES;
```

5.3.6. Limiting user resources

Starting from MySQL 4.0.2 one can limit certain resources per user.

So far, the only available method of limiting user usage of MySQL server resources has been setting the `max_user_connections` startup variable to a non-zero value. But this method is strictly global and does not allow for management of individual users, which could be of paricular interest to Internet Service Providers.

Therefore, management of three resources is introduced on the individual user level:

- Number of all queries per hour: All commands that could be run by a user.
- Number of all updates per hour: Any command that changes any table or database.
- Number of connections made per hour: New connections opened per hour.

A user in the aforementioned context is a single entry in the `user` table, which is uniquely identified by its `user` and `host` columns.

All users are by default not limited in using the above resources, unless the limits are granted to them. These limits can be granted **only** via global `GRANT (*.*)`, using this syntax:

```
GRANT ... WITH MAX_QUERIES_PER_HOUR = N1
               MAX_UPDATES_PER_HOUR = N2
               MAX_CONNECTIONS_PER_HOUR = N3;
```

One can specify any combination of the above resources. N1, N2 and N3 are integers and stands for count / hour.

If user reaches any of the above limits withing one hour, his connection will be terminated or refused and the appropriate error message shall be issued.

Current usage values for a particular user can be flushed (set to zero) by issuing a `GRANT` statement with any of the above clauses, including a `GRANT` statement with the current values.

Also, current values for all users will be flushed if privileges are reloaded (in the server or using `mysqladmin reload`) or if the `FLUSH USER_RESOURCES` command is issued.

The feature is enabled as soon as a single user is granted with any of the limiting `GRANT` clauses.

As a prerequisite for enabling this feature, the `user` table in the `mysql` database must contain the additional columns, as defined in the table creation scripts `mysql_install_db` and `mysql_install_db.sh` in `scripts` subdirectory.

5.3.7. Passwörter einrichten

In den meisten Fällen sollten Sie `GRANT` benutzen, um Ihre Benutzer / Passwörter einzurichten, daher trifft das folgende nur für fortgeschrittene Benutzer zu. See Abschnitt 5.3.1, „GRANT- und REVOKE-Syntax“.

Die Beispiele in den vorherigen Abschnitten erläutern ein wichtiges Prinzip: Wenn Sie ein nicht leeres Passwort mit `INSERT`- oder `UPDATE`-Statements setzen, müssen Sie die `PASSWORD()`-Funktion benutzen, um es zu verschlüsseln. Das liegt daran, dass die `user`-Tabelle Passwörter in verschlüsselter Form speichert, nicht als Klartext. Wenn Sie diese Tatsache vergessen, ist es möglich, dass sie Passwörter wie folgt setzen:

```
shell> mysql -u root mysql
mysql> INSERT INTO user (Host,User,Password)
       VALUES('%','heinzholger','keks');
mysql> FLUSH PRIVILEGES;
```

Das Ergebnis ist, dass der Klartextwert `'keks'` als Passwort in der `user`-Tabelle gespeichert ist. Wenn der Benutzer `heinzholger` versucht, sich mittels dieses Passworts mit dem Server zu verbinden, verschlüsselt der `mysql`-Client es mit `PASSWORD()`, erzeugt damit einen Authentifikationsvektor, der auf dem **verschlüsselten** Passwort und einer Zufallszahl basiert, die er vom Server erhält, und schickt das Ergebnis zum Server. Der Server benutzt den `password`-Wert in der `user`-Tabelle (den **nicht verschlüsselten** Wert `'keks'`), um dieselben Berechnungen durchzuführen, und vergleicht die Ergebnisse. Der Vergleich schlägt fehl und der Server verweigert die Verbindung:

```
shell> mysql -u heinzholger -pkeks test
Access denied
```

Passwörter müssen verschlüsselt sein, wenn sie in die `user`-Tabelle eingefügt werden, daher hätte das `INSERT`-Statement also wie folgt formuliert sein müssen:

```
mysql> INSERT INTO user (Host,User,Password)
       VALUES('%','heinzholger',PASSWORD('keks'));
```

Sie müssen die `PASSWORD()`-Funktion auch benutzen, wenn Sie `SET PASSWORD`-Statements gebrauchen:

```
mysql> SET PASSWORD FOR heinzholger@"%" = PASSWORD('keks');
```

Wenn Sie Passwörter mit dem `GRANT ... IDENTIFIED BY`-Statement oder dem `mysqladmin password`-Befehl setzen, wird die `PASSWORD()`-Funktion nicht benötigt. Beide sorgen dafür, dass das Passwort verschlüsselt wird, daher würden Sie ein Passwort `'keks'` wie folgt setzen:

```
mysql> GRANT USAGE ON *.* TO heinzholger@"%" IDENTIFIED BY 'keks';
```

oder

```
shell> mysqladmin -u heinzholger password keks
```

NOTE: `PASSWORD()` verschlüsselt Passwörter nicht auf dieselbe Art, wie das bei Unix-Passwörtern der Fall ist. Wenn daher Ihr Unix-Passwort und Ihr MySQL-Passwort identisch sind, sollten Sie daraus nicht schließen, dass `PASSWORD()` denselben Verschlüsselungswert ergibt wie der, der in der Unix-Passwortdatei gespeichert ist. See Abschnitt 5.3.2, „MySQL-Benutzernamen und -Passwörter“.

5.3.8. Wie Sie Ihre Passwörter sicher halten

Es ist nicht ratsam, Ihr Passwort so einzugeben, dass es von anderen Benutzern entdeckt werden kann. Die verschiedenen Methoden, Passwörter bei der Benutzung von Client-Programmen einzugeben, sind unten aufgeführt, jeweils mit einer Einschätzung des Risikos der Methode:

- Geben Sie einem normalen Benutzer nie Zugriff auf die `mysql.user`-Tabelle. Wenn jemand das verschlüsselte Passwort für einen Benutzer kennt, ermöglicht ihm das, sich als dieser Benutzer einzuloggen. Die Passwörter sind nur 'zerhackt', so dass niemand das echte Passwort sehen können sollte, das Sie benutzen (falls Sie ein ähnliches Passwort für Ihre anderen Applikationen benutzen sollten).
- Sie können auf der Kommandozeile die `-pyour_pass`- oder `--password=your_pass`-Option benutzen. Das ist bequem,

aber unsicher, weil Ihr Passwort für Systemzustandsprogramme (wie `ps`) sichtbar wird, die möglicherweise von anderen Benutzer aufgerufen werden, um Kommandozeilen anzuzeigen. (MySQL-Clients überschreiben typischerweise die Kommandozeilenargumente während der Initialisierungssequenz mit Nullen, dennoch gibt es einen kurzen Zeitraum, während dessen der Wert sichtbar ist.)

- Sie können eine `-p`- oder `--password`-Option (ohne `ihr_passwort`-Wert) benutzen. In diesem Fall erbittet das Client-Programm das Passwort vom Terminal:

```
shell> mysql -u benutzername -p
Enter password: ********
```

Die '`*`'-Zeichen stehen für Ihr Passwort.

Es ist sicherer, Ihr Passwort auf diese Art einzugeben statt auf der Kommandozeile, weil es für andere Benutzer nicht sichtbar wird. Diese Methode ist jedoch nur für Programme geeignet, die interaktiv laufen. Wenn Sie einen Client von einem Skript aus aufrufen wollen, das nicht interaktiv läuft, gibt es keine Möglichkeit, das Passwort vom Terminal aus einzugeben. Auf solchen Systemen kann es sogar vorkommen, dass die erste Zeile Ihres Skripts gelesen und (fälschlicherweise) als Ihr Passwort interpretiert wird!

- Sie können Ihr Passwort in einer Konfigurationsdatei speichern. Beispielsweise können Sie Ihr Passwort im `[client]`-Abschnitt der `.my.cnf`-Datei in Ihrem Heimatverzeichnis aufführen:

```
[client]
password=ihr_passwort
```

Wenn Sie Ihr Passwort in `.my.cnf` speichern, sollte die Datei nicht für die Gruppe (group) lesbar oder schreibbar sein. Stellen Sie sicher, dass der Zugriffsmodus der Datei `400` oder `600` ist.

See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

- Sie können Ihr Passwort in der `MYSQL_PWD`-Umgebungsvariablen speichern, aber diese Methode wird als extrem unsicher erachtet und sollte nicht gewählt werden. Einige Versionen von `ps` beinhalten eine Option, die Umgebung laufender Prozesse anzeigen zu lassen; Ihr Passwort würde dann für alle im Klartext lesbar sein, wenn Sie `MYSQL_PWD` setzen. Selbst auf Systemen ohne eine solche Version von `ps` ist es nicht ratsam, anzunehmen, dass es keine andere Methode gibt, Prozessumgebungen einzusehen. See Anhang F, *Umgebungsvariablen*.

Alles in allem sind die sichersten Methoden, das Passwort entweder durch Client-Programm entgegen nehmen zu lassen oder es in einer sauber abgesicherten `.my.cnf`-Datei einzugeben.

5.4. Katastrophenschutz und Wiederherstellung

5.4.1. Datenbank-Datensicherungen

Weil MySQL-Tabellen als Dateien gespeichert werden, ist es leicht, eine Datensicherung durchzuführen. Um eine konsistente Datensicherung zu erhalten, machen Sie ein `LOCK TABLES` auf die relevanten Tabellen, gefolgt von `FLUSH TABLES` für die Tabellen. See Abschnitt 7.7.2, „`LOCK TABLES/UNLOCK TABLES`-Syntax“. See Abschnitt 5.5.3, „`FLUSH`-Syntax“. Sie brauchen lediglich eine Lesesperre (Read Lock); das erlaubt anderen Threads, die Tabellen weiterhin abzufragen, während Sie eine Kopie der Dateien im Datenbank-Verzeichnis machen. `FLUSH TABLE` wird benötigt, um sicherzustellen, dass alle aktiven Indexseiten auf Platte zurück geschrieben werden, bevor Sie die Datensicherung beginnen.

Wenn Sie eine Tabellensicherung auf SQL-Ebene machen wollen, können Sie `SELECT INTO OUTFILE` oder `BACKUP TABLE` benutzen. See Abschnitt 7.4.1, „`SELECT`-Syntax“. See Abschnitt 5.4.2, „`BACKUP TABLE`-Syntax“.

Eine weitere Möglichkeit, eine Datenbank zu sichern, stellt die Benutzung des `mysqldump`-Programms oder des `mysqlhotcopy-Skript`s dar. See Abschnitt 5.8.5, „mysqldump, Tabellenstrukturen und -daten dumpen“. See Abschnitt 5.8.6, „mysqlhotcopy, MySQL-Datenbanken und Tabellen kopieren“.

1. Machen Sie eine komplette Sicherung Ihrer Datenbanken:

```
shell> mysqldump --tab=/pfad/zum/verzeichnis/ --opt --full

oder

shell> mysqlhotcopy datenbank /pfad/zum/verzeichnis/
```

Sie können auch einfach alle Tabellendateien (`*.frm`-, `*.MYD`- und `*.MYI`-Dateien) kopieren, solange der Server nicht gerade etwas aktualisiert. Das Skript `mysqlhotcopy` benutzt diese Methode.

2. Halten Sie `mysqld` an, wenn er läuft, und starten Sie ihn mit der `--log-update[=datei]`-Option. See Abschnitt 5.9.3, „Die Update-Log-Datei“. Die Update-Log-Datei(en) gibt Ihnen die Information, die Sie dafür benötigen, um Änderungen an der Datenbank zu replizieren, die ab dem Zeitpunkt durchgeführt wurden, als Sie `mysqldump` ausführten.

Wenn Sie etwas wiederherstellen müssen, versuchen Sie zunächst, Ihre Tabellen mit `REPAIR TABLE` oder `myisamchk -r` wieder herzustellen. Das sollte in 99,9% aller Fälle funktionieren. Wenn `myisamchk` fehlschlägt, probieren Sie folgende Prozedur (das funktioniert nur, wenn Sie MySQL mit `--log-update` gestartet haben. See Abschnitt 5.9.3, „Die Update-Log-Datei“.):

1. Stellen Sie die originale `mysqldump`-Datensicherung wieder her.
2. Führen Sie folgenden Befehl aus, um die Aktualisierungen (Updates) im Binär-Log noch einmal laufen zu lassen:

```
shell> mysqlbinlog hostname-bin.[0-9]* | mysql
```

   Wenn Sie das Update-Log benutzen, können Sie folgendes machen:

```
shell> ls -1 -t -r hostname.[0-9]* | xargs cat | mysql
```

`ls` wird benutzt, um alle Update-Log-Dateien in der richtigen Reihenfolge zu erhalten.

Mit `SELECT * INTO OUTFILE 'datei' FROM tabelle` können Sie auch selektive Datensicherungen herstellen und diese wieder herstellen mit `LOAD DATA INFILE 'datei' REPLACE ...`. Um Duplikate zu vermeiden, benötigen Sie einen Primärschlüssel (`PRIMARY KEY`) oder einen eindeutigen Schlüssel (`UNIQUE`) in der Tabelle. Das Schlüsselwort `REPLACE` führt dazu, dass alte Datensätze durch neue ersetzt werden, wenn ein neuer Datensatz einen alten auf einem eindeutigen Schlüsselwert duplizieren würde.

Wenn Sie bei der Datensicherung auf Ihrem System Performance-Probleme bekommen, können Sie diese lösen, indem Sie Replikation einrichten und die Datensicherungen auf dem Slave statt auf dem Master durchführen. See Abschnitt 5.10.1, „Einführung in die Replikation“.

Wenn Sie ein Veritas-Dateisystem benutzen, können Sie folgendes tun:

1. Führen Sie einen Client- (Perl ?) `FLUSH TABLES mit READ LOCK` aus.
2. Forken Sie eine Shell oder führen Sie einen anderen Client aus `mount vxfs snapshot`.
3. Führen Sie im ersten Client `UNLOCK TABLES` aus.
4. Kopieren Sie die Dateien von snapshot
5. Unmounten Sie snapshot

5.4.2. `BACKUP TABLE`-Syntax

```
BACKUP TABLE tabelle[,tabelle...] TO '/pfad/zum/backup/verzeichnis'
```

Machen Sie eine Kopie aller Tabellendateien ins Datensicherungsverzeichnis, was die Mindestanforderung für die Wiederherstellung darstellt. Momentan funktioniert das nur bei `MyISAM`-Tabellen. Bei `MyISAM`-Tabellen kopiert man `.frm`- (Definition) und `.MYD`- (Daten) Dateien. Die Indexdatei kann aus diesen beiden aufgebaut werden.

Bevor Sie diesen Befehl ausführen, sehen Sie bitte unter Abschnitt 5.4.1, „Datenbank-Datensicherungen“ nach.

Während der Datensicherung gilt eine Lesesperre (Read Lock) für jede Tabelle, eine nach der anderen, während sie gesichert werden. Wenn Sie mehrere Tabellen als Schnappschuss sichern wollen, müssen Sie zuerst ein `LOCK TABLES` ausführen, das eine Lesesperre für jede Tabelle in der zu sichernden Gruppe enthält.

Der Befehl gibt eine Tabelle mit folgenden Spalten zurück:

Spalte	**Wert**
Table	Tabellenname
Op	Immer ``backup''
Msg_type	`status`, `error`, `info` oder `warning`.
Msg_text	Die Meldung.

Beachten Sie, dass `BACKUP TABLE` erst ab MySQL 3.23.25 verfügbar ist.

5.4.3. `RESTORE TABLE`-Syntax

```
RESTORE TABLE tabelle[,tabelle...] FROM '/pfad/zum/backup/verzeichnis'
```

Stellt die Tabelle(n) aus der Datensicherung her, die mit `BACKUP TABLE` gesichert wurde(n). Bestehende Tabellen werden nicht überschrieben; wenn Sie über bestehende Tabellen wiederherstellen wollen, erhalten Sie eine Fehlermeldung. RESTORE benötigt länger als Datensicherung, weil der Index neu aufgebaut werden muss. Je mehr Schlüssel Sie haben, desto länger dauert es. Genau wie `BACKUP TABLE` funktioniert RESTORE momentan nur mit `MyISAM`-Tabellen.

Der Befehl gibt eine Tabelle mit folgenden Spalten zurück:

Spalte	**Wert**
Table	Tabellenname
Op	Immer ``restore''
Msg_type	`status`, `error`, `info` oder `warning`.
Msg_text	Die Meldung.

5.4.4. `CHECK TABLE`-Syntax

```
CHECK TABLE tabelle[,tabelle...] [option [option...]]

option = QUICK | FAST | MEDIUM | EXTENDED | CHANGED
```

`CHECK TABLE` funktioniert nur bei `MyISAM`-Tabellen. Bei `MyISAM`-Tabellen ist es dasselbe, wie `myisamchk -m tabelle` über die Tabelle laufen zu lassen.

Wenn Sie keine Option angeben, wird `MEDIUM` benutzt.

Prüft die Tabelle(n) auf Fehler. Bei `MyISAM`-Tabellen werden die Schlüssel-Statistiken aktualisiert. Der Befehl gibt eine Tabelle mit folgenden Spalten zurück:

Spalte	**Wert**
Table	Tabellenname
Op	Immer ``check''.
Msg_type	`status`, `error`, `info` oder `warning`.
Msg_text	Die Meldung.

Beachten Sie, dass Sie viele Zeilen an Information für jede geprüfte Tabelle erhalten. Die letzte Zeile enthält den `Msg_type status` und sollte normalerweise `OK` sein. Wenn Sie nicht `OK` erhalten, oder `Not checked`, sollten Sie im Normalfall eine Reparatur der Tabelle durchführen. See Abschnitt 5.4.6, „Benutzung von `myisamchk` für Tabellenwartung und Absturzreparatur“. `Not checked` bedeutet, dass bei der Tabelle der angegebene `TYPE` MySQL mitgeteilt hat, dass es keinerlei Notwendigkeit gab, die Tabelle zu prüfen.

Die unterschiedlichen Prüfoptionen stehen für folgendes:

Option	**Bedeutung**
`QUICK`	Keine Zeilen nach falschen Verknüpfungen (Links) durchsehen (scannen).
`FAST`	Nur Tabellen prüfen, die nicht ordnungsgemäß geschlossen wurden.
`CHANGED`	Nur Tabellen prüfen, die seit der letzten Prüfung geändert wurden oder die nicht ordnungsgemäß geschlossen wurden.
`MEDIUM`	Zeilen durchsehen (scannen), um zu bestätigen, dass gelöschte Verknüpfungen (Links) in Ordnung sind. Diese Option berechnet auch eine Schlüssel-Prüfsumme für die Zeilen und bestätigt diese mit einer berechneten Prüfsumme für die Schlüssel.
`EXTENDED`	Schlägt komplett alle Schlüssel für jede Zeile nach (Lookup). Hierdurch wird sichergestellt, dass die Tabelle 100% konsistent ist, aber das benötigt lange Zeit!

Bei `MyISAM`-Tabellen dynamischer Größe führt eine Prüfung immer eine `MEDIUM`-Prüfung durch. Bei Zeilen statischer Länge

wird das Durchsehen (Scan) der Zeilen durch `QUICK` und `FAST` übersprungen, weil solche Zeilen sehr selten beschädigt sind.

Sie können Prüfoptionen wie folgt kombinieren:

```
CHECK TABLE test_tabelle FAST QUICK;
```

Das würde nur eine QUICK-Prüfung der Tabelle durchführen, wenn diese nicht ordnungsgemäß geschlossen worden wäre.

HINWEIS: In einigen Fällen kann `CHECK TABLE` zu einer Änderung der Tabelle führen! Das geschieht, wenn die Tabelle als 'beschädigt' oder 'nicht ordnungsgemäß geschlossen' gekennzeichnet ist, aber `CHECK TABLE` keine Probleme in der Tabelle gefunden hat. In diesem Fall kennzeichnet `CHECK TABLE` die Tabelle als in Ordnung.

Wenn eine Tabelle beschädigt ist, liegt das Problem höchst wahrscheinlich in den Indexen und nicht im Daten-Teil. Alle oben genannten Prüfoptionen prüfen die Indexe gründlich und sollten daher die meisten Fehler finden.

Wenn Sie lediglich eine Tabelle prüfen wollen, von der Sie annehmen, dass sie in Ordnung ist, sollten Sie keine Prüfoptionen oder die `QUICK`-Option angeben. Letztere sollte benutzt werden, wenn Sie es eilig haben und das sehr geringe Risiko auf sich nehmen können, dass `QUICK` keinen Fehler in der Daten-Datei findet. (In den meisten Fällen sollte MySQL bei normalem Gebrauch jeden Fehler in der Daten-Datei finden. Wenn das geschieht, wird die Tabelle als 'beschädigt' gekennzeichnet, was bedeutet, dass die Tabelle solange nicht benutzt werden kann, bis sie repariert ist.)

`FAST` und `CHANGED` sind in erster Linie für die Benutzung durch ein Skript vorgesehen (zum Beispiel für die Ausführung durch cron), wenn Sie Ihre Tabellen von Zeit zu Zeit prüfen wollen. Für die meisten Anwendungsfälle sollte man `FAST` vor `CHANGED` bevorzugen. (Der einzige Fall, wo das nicht so ist, ist, wenn Sie vermuten, einen Bug im `MyISAM`-Code gefunden zu haben.)

`EXTENDED` ist nur für den Fall vorgesehen, dass Sie eine normale Prüfung haben durchlaufen lassen, aber immer noch seltsame Fehler von einer Tabelle erhalten, wenn MySQL versucht, eine Zeile zu aktualisieren oder eine Zeile über einen Schlüssel zu finden (das ist sehr unwahrscheinlich, wenn eine normale Prüfung durchgelaufen ist!).

Es wurde berichtet, dass bei der Tabellenprüfung einige Dinge nicht automatisch korrigiert werden können:

- `Found row where the auto_increment column has the value 0`.

  Das bedeutet, dass es in der Tabelle eine Zeile gibt, in der die `auto_increment`-Index-Spalte den Wert 0 enthält. (Es ist möglich, eine Zeile zu erzeugen, in der die auto_increment-Spalte 0 ist, indem man die Spalte explizit mit einem `UPDATE`-Statement auf 0 setzt.)

  Das ist für sich genommen kein Fehler, kann aber Probleme verursachen, wenn Sie die Tabelle dumpen und dann wiederherstellen, oder ein `ALTER TABLE` auf die Tabelle machen. In diesen Fällen ändert sich der Wert der auto_increment-Spalte gemäß den Regeln für auto_increment-Spalten, was Probleme wie doppelte Schlüsseleintragsfehler bringen könnte.

  Um diese Warnmeldung loszuwerden, führen Sie einfach ein `UPDATE`-Statement durch und setzen die Spalte auf irgend einen anderen Wert als 0.

5.4.5. `REPAIR TABLE`-Syntax

```
REPAIR TABLE tabelle[,tabelle...] [QUICK] [EXTENDED]
```

`REPAIR TABLE` funktioniert nur bei `MyISAM`-Tabellen und ist dasselbe, wie `myisamchk -r tabelle` auf die Tabelle auszuführen.

Normalerweise sollten sie diesen Befehl nie ausführen müssen, aber wenn ein Unglück passiert, ist es sehr wahrscheinlich, dass Sie alle Daten einer MyISAM-Tabelle mit `REPAIR TABLE` retten können. Wenn Ihre Tabellen häufig beschädigt werden, sollten Sie versuchen, den Grund hierfür herauszufinden! See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“. See Abschnitt 8.1.3, „MyISAM-Tabellenprobleme“.

`REPAIR TABLE` repariert eine möglicherweise beschädigte Tabelle. Der Befehl gibt eine Tabelle mit folgenden Spalten zurück:

Spalte	**Wert**
Table	Tabellenname
Op	Immer ``repair''
Msg_type	`status`, `error`, `info` oder `warning`.
Msg_text	Die Meldung.

Beachten Sie, dass Sie viele Zeilen an Informationen für jede reparierte Tabelle erhalten. Die letzte Zeile enthält den `Msg_type status` und sollte normalerweise `OK` sein. Wenn Sie nicht `OK` erhalten, sollten Sie versuchen, die Tabelle mit `myisamchk -o` zu reparieren, weil `REPAIR TABLE` noch nicht alle Optionen von `myisamchk` enthält. In naher Zukunft werden wir das flexibler gestalten.

Wenn `QUICK` angegeben wird, versucht MySQL lediglich ein `REPAIR` des Indexbaums.

Wenn Sie `EXTENDED` benutzen, erzeugt MySQL den Index Zeile für Zeile, anstatt einen Index auf einmal durch Sortieren zu erzeugen. Das kann bei Schlüsseln fester Länge besser sein, wenn Sie lange `char()`-Schlüssel haben, die sich gut komprimieren lassen.

5.4.6. Benutzung von `myisamchk` für Tabellenwartung und Absturzreparatur

Ab MySQL-Version 3.23.13 können Sie MyISAM-Tabellen mit dem `CHECK TABLE`-Befehl überprüfen. See Abschnitt 5.4.4, „`CHECK TABLE`-Syntax“. Mit dem `REPAIR TABLE`-Befehl können Sie Tabellen reparieren. See Abschnitt 5.4.5, „`REPAIR TABLE`-Syntax“.

Um MyISAM-Tabellen (`.MYI` und `.MYD`) zu überprüfen und / oder zu reparieren, sollten sie das `myisamchk`-Dienstprogramm benutzen. Um ISAM-Tabellen (`.ISM` und `.ISD`) zu überprüfen und / oder zu reparieren, sollten Sie das `isamchk`-Dienstprogramm benutzen. See Kapitel 8, *MySQL-Tabellentypen*.

Der folgende Text behandelt `myisamchk`, trifft aber voll umfänglich auch auf das alte `isamchk` zu.

Sie können das `myisamchk`-Dienstprogramm benutzen, um Informationen über Ihre Datenbanktabellen zu erhalten, sie zu prüfen und zu reparieren, oder um sie zu optimieren. Die folgenden Abschnitte beschreiben, wie man `myisamchk` aufruft (inklusive einer Beschreibung seiner Optionen), wie man einen Wartungsplan für Tabellen erstellt und wie die unterschiedlichen Funktionen von `myisamchk` benutzt werden.

In den meisten Fällen können Sie auch den Befehl `OPTIMIZE TABLES` benutzen, um Tabellen zu optimieren und zu reparieren, aber dieser ist nicht so schnell und (in Fall wirklich schwerer Fehler) nicht so zuverlässig wie `myisamchk`. Auf der anderen Seite ist `OPTIMIZE TABLE` leichter zu benutzen, und Sie brauchen sich nicht um das Flushen von Tabellen zu kümmern. See Abschnitt 5.5.1, „`OPTIMIZE TABLE`-Syntax“.

Obwohl das Reparieren bei `myisamchk` recht sicher ist, ist es immer eine gute Idee, eine Datensicherung zu machen, bevor eine Reparatur durchgeführt wird (oder etwas Sonstiges, das viele Änderungen an einer Tabelle durchführt).

5.4.6.1. Aufrufsyntax von `myisamchk`

`myisamchk` wird wie folgt aufgerufen:

```
shell> myisamchk [optionen] tabelle
```

`optionen` legt fest, was `myisamchk` tun soll. Die Optionen sind unten beschrieben. (Sie erhalten eine Liste der Optionen, wenn Sie `myisamchk --help` eingeben.) Ohne Optionen aufgerufen prüft `myisamchk` einfach nur Ihre Tabelle. Um mehr Informationen zu erhalten oder `myisamchk` anzuweisen, korrigierende Aktionen durchzuführen, geben Sie Optionen wie unten und in den folgenden Abschnitten beschrieben an.

`tabelle` ist die Datenbanktabelle, die Sie prüfen oder reparieren wollen. Wenn Sie `myisamchk` anderswo als im Datenbank-Verzeichnis ausführen, müssen Sie den Pfad zur Datei angeben, denn `myisamchk` weiß nicht, wo Ihre Datenbank liegt. In der Tat kümmert sich `myisamchk` nicht darum, ob die Dateien, die es bearbeiten soll, in einem Datenbank-Verzeichnis liegen oder nicht; sie können diese Dateien daher an eine andere Stelle kopieren und die Wiederherstellungsoperationen dort durchführen.

Sie können in der `myisamchk`-Befehlszeile mehrere Tabellen angeben, wenn Sie wollen. Sie können auch einen Namen als Indexdateinamen angeben (mit dem Suffix `.MYI`), was Ihnen gestattet, alle Tabellen in einem Verzeichnis anzugeben, indem Sie das Muster `*.MYI` benutzen. Wenn Sie zum Beispiel in einem Datenbank-Verzeichnis sind, können Sie alle Tabellen im Verzeichnis wie folgt prüfen:

```
shell> myisamchk *.MYI
```

Wenn Sie nicht im Datenbank-Verzeichnis sind, können Sie alle dortigen Tabellen prüfen, indem Sie den Pfad zum Verzeichnis angeben:

```
shell> myisamchk /pfad/zum/datenbank_verzeichnis/*.MYI
```

Sie können sogar alle Tabellen in allen Datenbanken prüfen, indem Sie einen Platzhalter im Pfad zum MySQL-Daten-Verzeichnis angeben:

```
shell> myisamchk /pfad/zum/datadir/*/*.MYI
```

Um schnell alle Tabellen zu prüfen, wird folgender Befehl empfohlen:

```
myisamchk --silent --fast /pfad/zum/datadir/*/*.MYI
isamchk --silent /pfad/zum/datadir/*/*.ISM
```

Wenn Sie alle Tabellen prüfen und alle Tabellen reparieren wollen, die beschädigt sind, können Sie folgende Kommandozeile eingeben:

```
myisamchk --silent --force --fast --update-state -O key_buffer=64M -O sort_buffer=64M -O read_buffer=1M -O write_buffe
isamchk --silent --force -O key_buffer=64M -O sort_buffer=64M -O read_buffer=1M -O write_buffer=1M /pfad/zum/datadir/*
```

Hierbei wird angenommen, dass Sie mehr als 64 MB Arbeitsspeicher frei haben.

Wenn Sie einen Fehler wie den folgenden erhalten:

```
myisamchk: warning: 1 clients is using oder hasn't closed the table properly
```

Bedeutet das, dass Sie versuchen, eine Tabelle zu überprüfen, die durch ein anderes Programm aktualisiert wurde (wie dem `mysqld`-Server), das die Datei noch nicht geschlossen hat oder das abgestürzt ist, ohne die Datei ordnungsgemäß zu schließen.

Wenn `mysqld` läuft, müssen Sie ein Sync/Schließen aller Tabellen mit `FLUSH TABLES` erzwingen und sicherstellen, dass niemand die Tabellen benutzt, während Sie `myisamchk` laufen lassen. In MySQL-Version 3.23 ist die einfachste Möglichkeit, dieses Problem zu vermeiden, die Benutzung von `CHECK TABLE` anstelle von `myisamchk`.

5.4.6.2. Allgemeine Optionen für `myisamchk`

`myisamchk` unterstützt folgende Optionen:

- `-# oder --debug=debug_optionen`

  Ausgabe eines Debug-Logs. Die Zeichenkette `debug_optionen` ist häufig `'d:t:o,dateiname'`.

- `-? oder --help`

  Hilfetext ausgeben und beenden.

- `-O var=option, --set-variable var=option`

  Setzt den Wert einer Variablen. Mögliche Variablen und ihre Vorgabewerte für myisamchk können mit `myisamchk --help` herausgefunden werden:

key_buffer_size	523264
read_buffer_size	262136
write_buffer_size	262136
sort_buffer_size	2097144
sort_key_blocks	16
decode_bits	9

  `sort_buffer_size` wird benutzt, wenn Schlüssel repariert werden, indem Schlüssel sortiert werden, was der Normalfall ist, wenn Sie `--recover` benutzen.

  `key_buffer_size` wird benutzt, wenn Sie die Tabelle mit `--extended-check` prüfen oder wenn die Schlüssel repariert werden, indem Schlüssel Zeile für Zeile in die Tabelle eingefügt werden (als wenn normale Einfügeoperationen (Insert) durchgeführt werden). Eine Reparatur mittels Key-Buffer (Schlüsselpuffer) wird in folgenden Fällen benutzt:

  - Wenn Sie `--safe-recover` benutzen.
  - Wenn die temporären Dateien, die benötigt werden, um die Schlüssel zu sortieren, mehr als zweimal so Groß werden würden, als wenn die Schlüsseldatei direkt erzeugt würde. Das ist oft dann der Fall, wenn Sie große `CHAR`-, `VARCHAR`- oder `TEXT`-Schlüssel haben, weil das Sortieren die gesamten Schlüssel während des Sortierens speichern muss. Wenn Sie viel temporären Platz haben und `myisamchk` zwingen können, mittels Sortieren zu reparieren, können Sie die `--sort-recover`-Option benutzen.

  Die Reparatur durch den Key-Buffer (Schlüsselpuffer) nimmt weit weniger Plattenplatz in Anspruch als wenn Sortieren benutzt

wird, ist aber auch viel langsamer.

Wenn Sie eine schnellere Reparatur wollen, setzen Sie die obigen Variablen auf ungefähr 1/4 Ihres verfügbaren Arbeitsspeichers. Sie können beide Variablen auf große Werte setzen, weil nur einer der oben aufgeführten Puffer zur gleichen Zeit benutzt wird.

- `-s oder --silent`

  Schweigsamer Modus. Ausgaben erfolgen nur im Fehlerfall. Sie können `-s` doppelt benutzen (`-ss`), um `myisamchk` sehr schweigsam zu machen.

- `-v oder --verbose`

  Geschwätziger Modus. Es werden mehr Informationen ausgegeben. Dies kann auch bei `-d` und `-e` benutzt werden. Benutzen Sie `-v` mehrfach (`-vv`, `-vvv`), um noch ausführlichere Meldungen auszugeben!

- `-V oder --version`

  Die aktuelle Version von `myisamchk` ausgeben und beenden.

- `-w or, --wait`

  Statt einen Fehler auszugeben, wenn die Tabelle gesperrt ist, warten, bis die Tabelle entsperrt ist, bevor fortgefahren wird. Beachten Sie: Wenn Sie `mysqld` auf der Tabelle mit `--skip-locking` laufen lassen, kann die Tabelle nur mit einem weiteren `myisamchk`-Befehl gesperrt werden.

5.4.6.3. Prüfoptionen für `myisamchk`

- `-c oder --check`

  Tabelle auf Fehler überprüfen. Das ist die vorgabemäßige Operation, wenn Sie `myisamchk` keine sonstigen Optionen angeben, die dies überschreiben.

- `-e oder --extend-check`

  Tabelle SEHR gründlich prüfen (was recht langsam ist, wenn Sie viele Indexe haben). Diese Option sollte nur in Extremfällen benutzt werden. Normalerweise sollten `myisamchk` oder `myisamchk --medium-check` in fast allen Fällen in der Lage sein, herauszufinden, ob es in der Tabelle irgend welche Fehler gibt.

  Wenn Sie `--extended-check` benutzen und viel Arbeitsspeicher haben, setzen Sie den Wert von `key_buffer_size` um etliches herauf!

- `-F oder --fast`

  Nur Tabellen prüfen, die nicht ordnungsgemäß geschlossen wurden.

- `-C oder --check-only-changed`

  Nur Tabellen prüfen, die seit der letzten Prüfung geändert wurden.

- `-f oder --force`

  `myisamchk` mit `-r` (repair) auf die Tabelle neu starten, wenn `myisamchk` in der Tabelle irgend welche Fehler findet.

- `-i oder --information`

  Statistische Informationen über die Tabelle, die geprüft wird, ausgeben.

- `-m oder --medium-check`

  Schneller als extended-check, findet aber nur 99,99% aller Fehler. Das sollte allerdings in den meisten Fällen ausreichen.

- `-U oder --update-state`

  In der `.MYI`-Datei speichern, wann die Tabelle geprüft wurde und ob die Tabelle beschädigt wurde. Das sollte benutzt werden, um vollen Nutzen aus der `--check-only-changed`-Option ziehen zu können. Sie sollten diese Option nicht benutzen, wenn der `mysqld`-Server die Tabelle benutzt und Sie ihn mit `--skip-locking` laufen lassen.

- `-T oder --read-only`

  Die Tabelle nicht als geprüft kennzeichnen. Das ist hilfreich, wenn Sie `myisamchk` benutzen, um eine Tabelle zu prüfen, die von irgend einer anderen Applikation benutzt wird, die kein Sperren durchführt (wie `mysqld --skip-locking`).

5.4.6.4. Reparaturoptionen für `myisamchk`

Folgende Optionen werden benutzt, wenn Sie `myisamchk` mit `-r` oder `-o` starten:

- `-D # oder --data-file-length=#`

  Maximale Länge der Daten-Datei (wenn die Daten-Datei neu erzeugt wird, wenn sie 'voll' ist).

- `-e oder --extend-check`

  Es wird versucht, jede mögliche Zeile der Daten-Datei wiederherzustellen. Normalerweise wird dies auch eine Menge Zeilen-'Müll' finden. Benutzen Sie diese Option nur dann, wenn Sie völlig verzweifelt sind.

- `-f oder --force`

  Alte temporäre Dateien (`tabelle.TMD`) werden überschrieben, anstatt abzubrechen.

- `-k # oder keys-used=#`

  Wenn Sie ISAM benutzen, weist das den ISAM-Tabellen-Handler an, nur die ersten #-Indexe zu benutzen. Wenn Sie `MyISAM` benutzen, sagt es dem Handler, welche Schlüssel benutzt werden sollen, wobei jedes Binärbit für einen Schlüssel steht (der erste Schlüssel ist Bit 0). Das kann benutzt werden, um schnelleres Einfügen (Insert) zu erreichen! Deaktivierte Indexe können reaktiviert werden, indem man `myisamchk -r` benutzt.

- `-l oder --no-symlinks`

  Symbolischen Links wird nicht gefolgt. Normalerweise repariert `myisamchk` die Tabelle, auf die ein Symlink verweist. Diese Option gibt es in MySQL 4.0 nicht, weil MySQL 4.0 während der Reparatur keine Symlinks entfernt.

- `-r oder --recover`

  Kann fast alles reparieren, ausser eindeutige Schlüssel, die nicht eindeutig sind (was ein extrem unwahrscheinlicher Fehler bei ISAM- / MyISAM-Tabellen ist). Wenn Sie eine Tabelle wiederherstellen wollen, sollten Sie zuerst diese Option ausprobieren. Nur wenn myisamchk berichtet, dass die Tabelle mit `-r` nicht wiederhergestellt werden kann, sollten Sie `-o` probieren. (Hinweis: Im unwahrscheinlichen Fall, dass `-r` fehlschlägt, ist die Daten-Datei immer noch intakt.) Wenn Sie viel Arbeitsspeicher haben, sollten Sie die Größe von `sort_buffer_size` herauf setzen!

- `-o oder --safe-recover`

  Benutzt eine alte Wiederherstellungsmethode (liest alle Zeilen der Reihe nach und aktualisiert alle Indexbäume, basierend auf den gefundenen Zeilen); das ist sehr viel langsamer als `-r`, kann aber eine Reihe sehr unwahrscheinlicher Fälle behandeln, die `-r` nicht behandeln kann. Diese Wiederherstellungsmethode benutzt viel weniger Plattenspeicher als `-r`. Normalerweise sollte man immer zuerst versuchen, mit `-r` zu reparieren und nur im Falle des Fehlschlagens `-o` benutzen.

  Wenn Sie viel Arbeitsspeicher haben, sollten Sie die Größe von `key_buffer_size` herauf setzen!

- `-n oder --sort-recover`

  Zwingt `myisamchk` zu sortieren, um Schlüssel aufzulösen, selbst wenn die temporären Dateien sehr Groß sein sollten. Diese Option hat keine Auswirkung, wenn Sie Volltextschlüssel in der Tabelle haben.

- `--character-sets-dir=...`

  Verzeichnis, wo Zeichensätze gespeichert sind.

- `--set-character-set=name`

  Ändert den Zeichensatz, der vom Index benutzt wird.

- `-t oder --tmpdir=path`

  Pfad zum Speichern temporärer Dateien. Wenn dieser nicht gesetzt ist, benutzt `myisamchk` hierfür die Umgebungsvariable `TMPDIR`.

- `-q oder --quick`

  Repariert schneller, indem die Daten-Datei nicht verändert wird. Man kann ein zweites `-q` angeben, um `myisamchk` zu zwingen, die Original-Daten-Datei zu ändern, falls doppelte Schlüssel auftreten.

- `-u oder --unpack`

  Datei entpacken, die mit myisampack gepackt wurde.

5.4.6.5. Weitere Optionen für `myisamchk`

Weitere Aktionen, die `myisamchk` ausführen kann, neben der Prüfung und Reparatur von Tabellen:

- `-a oder --analyze`

  Analysiert die Verteilung von Schlüsseln. Das verbessert die Performance bei Tabellenverknüpfungen (Joins), indem der Join-Optimierer in die Lage versetzt wird, besser auszuwählen, in welcher Reihenfolge die Tabellen verknüpft werden sollten und welche Schlüssel er dabei verwenden sollte: `myisamchk --describe --verbose tabelle'` oder Benutzung von `SHOW KEYS` in MySQL.

- `-d oder --description`

  Gibt ein paar Informationen über die Tabelle aus.

- `-A oder --set-auto-increment[=value]`

  Zwingt auto_increment, mit diesem oder einem höheren Wert anzufangen. Wenn kein Wert angegeben wird, wird der nächste auto_increment-Wert auf den höchsten benutzten Wert für den auto-Schlüssel + 1 gesetzt.

- `-S oder --sort-index`

  Sortiert die Blöcke des Indexbaums in Hoch-Niedrig-Reihenfolge. Das optimiert Suchoperationen und macht das Durchsehen (Scanning) von Tabellen nach Schlüsseln schneller.

- `-R oder --sort-records=#`

  Sortiert Datensätze in Übereinstimmung mit einem Index. Das macht Ihre Daten viel konzentrierter und kann `SELECT` mit Bereichen und `ORDER BY`-Operationen auf diesem Index erheblich beschleunigen. (Beim ersten Sortieren kann das SEHR langsam sein!) Um die Anzahl von Indexen einer Tabelle herauszufinden, benutzen Sie `SHOW INDEX`, was die Indexe einer Tabelle in genau der Reihenfolge zeigt, in der `myisamchk` sie sieht. Indexe werden mit 1 beginnend nummeriert.

5.4.6.6. Speicherbenutzung von `myisamchk`

Die Speicherzuordnung ist wichtig, wenn Sie `myisamchk` laufen lassen. `myisamchk` benutzt nicht mehr Speicher, als Sie mir der `-O`-Option festlegen. Wenn Sie `myisamchk` für sehr große Dateien benutzen wollen, sollten Sie zuerst entscheiden, wieviel Speicher Sie benutzen wollen. Die Vorgabe liegt bei nur etwa 3 MB, um Dinge zu reparieren. Indem größere Werte benutzt werden, können Sie `myisamchk` dazu bringen, schneller zu arbeiten. Wenn Sie beispielsweise 32 MB Arbeitsspeicher haben, könnten Sie Optionen wie die folgende benutzen (zusätzlich zu weiteren Optionen, die Sie eventuell angeben):

```
shell> myisamchk -O sort=16M -O key=16M -O read=1M -O write=1M ...
```

`-O sort=16M` sollte für die meisten Fälle ausreichen.

Denken Sie daran, dass `myisamchk` temporäre Dateien in `TMPDIR` benutzt. Wenn `TMPDIR` auf ein Speicher-Dateisystem zeigt, können Kein-Speicher-Fehler schnell auftreten. Wenn das passiert, setzen Sie `TMPDIR` so, dass es auf ein Verzeichnis mit mehr Speicherplatz zeigt und starten Sie `myisamchk` erneut.

Beim Reparieren benötigt `myisamchk` große Mengen von Festplattenspeicher:

- Die doppelte Größe der Daten-Datei (die Originaldatei und eine Kopie). Dieser Platz wird nicht benötigt, wenn die Reparatur mit `--quick` durchgeführt wird, weil in diesem Fall nur die Index-Datei neu erzeugt wird. Der Platz wird auf derselben Festplatte benötigt, wo die Original-Daten-Datei liegt!

- Platz für die neue Index-Datei, die die alte ersetzt. Die alte Index-Datei wird beim Start beschnitten, daher kann man diesen Platz üblicherweise ignorieren. Der Platz wird auf derselben Platte benötigt, auf der die Original-Index-Datei liegt!

- Wenn Sie `--recover` oder `--sort-recover` benutzen (aber nicht, wenn Sie `--safe-recover` benutzen), brauchen Sie Platz für einen Sortierpuffer (Sort Buffer) für: `(größter_schlüssel + zeilen_zeiger_länge) * anzahl_der_zeilen * 2`. Sie können die Länge der Schlüssel und die Zeilen-Zeiger-Länger mit `myisamchk -dv tabelle` prüfen. Dieser Platz wird auf der temporären Platte zugeordnet (festgelegt durch `TMPDIR` oder `--tmpdir=#`).

Wenn Sie während der Reparatur ein Problem mit dem Plattenplatz bekommen, können Sie `--safe-recover` anstelle von `--recover` ausprobieren.

5.4.6.7. Benutzung von `myisamchk` für die Fehlerbeseitigung nach Abstürzen

Wenn Sie `mysqld` mit `--skip-locking` laufen lassen (was auf einigen Systemen wie Linux die Vorgabe ist), können Sie `myisamchk` nicht zuverlässig dafür benutzen, eine Tabelle zu prüfen, wenn `mysqld` diese Tabelle benutzt. Wenn Sie sicher sein können, dass niemand auf die Tabellen mit `mysqld` zugreift, während Sie `myisamchk` laufen lassen, müssen Sie nur ein `mysqladmin flush-tables` durchführen, bevor Sie anfangen, die Tabellen zu prüfen. Wenn Sie das nicht garantieren können, müssen Sie `mysqld` herunter fahren, während Sie die Tabellen prüfen. Wenn Sie `myisamchk` laufen lassen, währen `mysqld` die Tabellen aktualisiert, erhalten Sie möglicherweise die Meldung, dass eine Tabelle beschädigt ist, selbst wenn sie es nicht ist.

Wenn Sie `--skip-locking` nicht benutzen, können Sie jederzeit `myisamchk` benutzen, um Tabellen zu prüfen. Während Sie das tun, warten alle Clients, die versuchen, die Tabelle zu aktualisieren, bis `myisamchk` fertig ist, bevor sie weiter machen.

Wenn Sie `myisamchk` benutzen, um Tabellen zu reparieren oder zu optimieren, **MÜSSEN** Sie stets sicherstellen, dass der `mysqld`-Server die Tabelle nicht benutzt (das trifft auch zu, wenn Sie `--skip-locking` benutzen). Wenn Sie `mysqld` nicht herunter fahren, sollten Sie zumindest `mysqladmin flush-tables` ausführen, bevor Sie `myisamchk` benutzen.

Dieses Kapitel beschreibt, wie man MySQL-Datenbanken auf Datenbeschädigung prüft und damit umgeht. Wenn Ihre Tabellen häufig beschädigt sind, wollten Sie versuchen, den Grund hierfür herauszufinden! See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“.

Der Abschnitt über `MyISAM`-Tabellen enthält Gründe, warum eine Tabelle beschädigt sein könnte. See Abschnitt 8.1.3, „MyISAM-Tabellenprobleme“.

Wenn Sie eine Wiederherstellung nach einem Absturz durchführen, ist es wichtig zu wissen, das jede Tabelle `tabelle` in einer Datenbank mit drei Dateien im Datenbank-Verzeichnis korrespondiert:

Datei	Zweck
`tabelle.frm`	Tabellendefinitionsdatei (form)
`tabelle.MYD`	Daten-Datei (data)
`tabelle.MYI`	Index-Datei (index)

Jeder der drei Dateitypen kann auf verschiedene Weisen beschädigt werden. Probleme treten aber zumeist bei Daten-Dateien und Index-Dateien auf.

`myisamchk` funktioniert so, dass Zeile für Zeile eine Kopie der `.MYD`-(data)-Datei gemacht wird. Es beendet die Reparaturphase damit, dass die alte `.MYD`-Datei entfernt wird und die neue Datei mit dem Orginal-Dateinamen benannt wird. Wenn Sie `--quick` benutzen, erzeugt `myisamchk` keine temporäre `.MYD`-Datei, sondern nimmt statt dessen an, dass die `.MYD`-Datei korrekt ist, und erzeugt nur eine neue Index-Datei, ohne die `.MYD`-Datei zu berühren. Das ist sicher, weil `myisamchk` automatisch feststellt, wenn die `.MYD`-Datei beschädigt ist, und die Reparatur in diesem Fall abbricht. Sie können `myisamchk` auch mit zwei `--quick`-Optionen aufrufen. In diesem Fall bricht `myisamchk` bei einigen Fehlern (wie doppelten Schlüsseleinträgen) nicht ab, sondern versucht statt dessen, diese aufzulösen, indem die `.MYD`-Datei verändert wird. Normalerweise ist die Benutzung von zwei `--quick`-Optionen nur sinnvoll, wenn Sie zu wenig frei Plattenplatz haben, um eine normale Reparatur durchzuführen. In diesem Fall sollten Sie zumindest eine Datensicherung machen, bevor Sie `myisamchk` laufen lassen.

5.4.6.8. Wie Tabellen auf Fehler überprüft werden

Um eine MyISAM-Tabelle zu prüfen, benutzen Sie folgende Befehle:

- `myisamchk tabelle`

  Das findet 99.99% aller Fehler. Nicht gefunden werden Beschädigungen, die **NUR** die Daten-Datei betreffen (was sehr ungewöhnlich ist). Wenn Sie eine Tabelle prüfen wollen, sollten Sie `myisamchk` normalerweise ohne Optionen oder entweder mit der `-s`- oder `--silent`-Option laufen lassen.

- `myisamchk -m tabelle`

Das findet 99.999% aller Fehler. Zuerst prüft es alle Indexeinträge auf Fehler und liest dann alle Zeilen durch. Es berechnet eine Prüfsumme für alle Schlüssel in den Zeilen und bestätigt dann, dass die Prüfsumme mit der Prüfsumme für die Schlüssel im Indexbaum übereinstimmt.

- `myisamchk -e tabelle`

  Das führt eine vollständige, gründlich Prüfung aller Daten durch (`-e` bedeutet ``extended check'' - erweiterte Prüfung). Es führt ein Prüf-Lesen jedes Schlüssels für jede Zeile durch, um zu bestätigen, dass sie tatsächlich auf die richtige Zeile verweisen. Das kann bei einer großen Tabelle mit vielen Schlüsseln SEHR LANG dauern. `myisamchk` hält normalerweise an, wenn es den ersten Fehler gefunden hat. Wenn Sie mehr Informationen haben wollen, können Sie die `--verbose`-(`-v`)-Option benutzen. Das veranlasst `myisamchk`, weiterzumachen, bis maximal 20 Fehler gefunden wurden. Bei normalem Gebrauch ist ein einfaches `myisamchk` (ohne weitere Argumente ausser dem Tabellennamen) ausreichend.

- `myisamchk -e -i tabelle`

  Wie der vorherige Befehl, jedoch weist die `-i`-Option `myisamchk` an, zusätzlich einige statistische Informationen auszugeben.

5.4.6.9. Wie Tabellen repariert werden

Der folgende Abschnitt behandelt nur die Benutzung von `myisamchk` mit `MyISAM`-Tabellen (Erweiterungen `.MYI` und `.MYD`). Wenn Sie `ISAM`-Tabellen benutzen (Erweiterungen `.ISM` und `.ISD`), sollten Sie statt dessen `isamchk` benutzen.

Ab MySQL-Version 3.23.14 können Sie MyISAM-Tabellen mit dem `REPAIR TABLE`-Befehl reparieren. See Abschnitt 5.4.5, „REPAIR TABLE-Syntax“.

Zu den Symptomen einer beschädigten Tabelle gehören Anfragen, die unerwartet abbrechen, und beobachtbare Fehler wie folgender:

- `tabelle.frm` is locked against change
- Can't find file `tabelle.MYI` (Errcode: ###)
- Unexpected end of file
- Record file is crashed
- Got error ### from table handler

  Um mehr Information über den Fehler zu erhalten, lassen Sie `perror ###` laufen. Hier sind die häufigsten Fehler, die auf ein Problem mit der Tabelle hinweisen:

```
shell> perror 126 127 132 134 135 136 141 144 145
126 = Index-Datei ist beschädigt / falsches Dateiformat
127 = Daten-Datei ist beschädigt
132 = Alte Datenbank-Datei
134 = Datensatz wurde bereits gelöscht (oder Daten-Datei beschädigt)
135 = Kein Platz mehr in der Daten-Datei
136 = Kein Platz mehr in der Index-Datei
141 = Doppelter Eintrag für eindeutigen Schlüssel oder Beschränkung beim Schreiben oder Aktualisiern
144 = Tabelle ist beschädigt und die letzte Reparatur ist fehlgeschlagen
145 = Tabelle ist als beschädigt gekennzeichnet und sollte repariert werden
```

  Beachten Sie, dass Fehler 135 (kein Platz mehr in der Daten-Datei) kein Fehler ist, der durch eine einfache Reparatur behoben werden kann. In diesem Fall müssen Sie folgendes durchführen:

```
ALTER TABLE tabelle MAX_ROWS=xxx AVG_ROW_LENGTH=yyy;
```

In den anderen Fällen müssen Sie Ihre Tabellen reparieren. `myisamchk` kann üblicherweise die meisten Dinge, die schief gehen können, finden und beheben.

Der Reparaturprozess läuft in vier Phasen ab, die unten beschrieben sind. Bevor Sie anfangen, sollten Sie in das Datenbank-Verzeichnis wechseln und die Berechtigungen der Tabellen-Dateien prüfen. Stellen Sie sicher, dass diese durch den Unix-Benutzer lesbar sind, unter dem `mysqld` läuft (und für Sie, weil Sie auf die Dateien zugreifen müssen, wenn Sie sie prüfen). Wenn Sie in der Folge Dateien verändern müssen, müssen diese für Sie auch schreibbar sein.

Wenn Sie MySQL-Version 3.23.16 und höher benutzen, können (und sollten) Sie die `CHECK`- und `REPAIR`-Befehle benutzen, um `MyISAM`-Tabellen zu prüfen und zu reparieren. See Abschnitt 5.4.5, „REPAIR TABLE-Syntax“.

Der Handbuchabschnitt über Tabellenwartung beinhaltet die Optionen für `isamchk` / `myisamchk`. See Abschnitt 5.4.6, „Benutzung von `myisamchk` für Tabellenwartung und Absturzreparatur“.

Der folgende Abschnitt ist für Fälle, in denen die obigen Befehle fehlschlagen, oder wenn Sie die erweiterten Features benutzen wollen, die `isamchk` / `myisamchk` zur Verfügung stellt.

Wenn Sie eine Tabelle von der Kommandozeile aus reparieren wollen, müssen Sie zuerst den `mysqld`-Server herunter fahren. Beachten Sie bei `mysqladmin shutdown` auf einen entfernten Server, dass der `mysqld`-Server noch für eine Weile aktiv bleibt, nachdem `mysqladmin` beendet ist, bis alle Anfragen beendet und alle Schlüssel auf Platte zurück geschrieben (flush) wurden.

Phase 1: Prüfen Ihrer Tabellen

Lassen Sie `myisamchk *.MYI` laufen, oder `myisamchk -e *.MYI`, wenn Sie mehr Zeit haben. Benutzen Sie die `-s`-(silent)-Option, um unnötige Informationen zu unterdrücken.

Wenn der `mysqld`-Server herunter gefahren ist, sollten Sie die `--update`-Option benutzen, um `myisamchk` zu veranlassen, die Tabelle als 'geprüft' zu kennzeichnen.

Sie müssen nur die Tabellen reparieren, bei denen `myisamchk` bekannt gibt, dass sie Fehler enthalten. Gehen Sie bei solchen Tabellen zu Phase 2 über.

Wenn Sie beim Prüfen merkwürdige Fehler erhalten (wie `out of memory`-Fehler), oder wenn `myisamchk` abstürzt, gehen Sie zu Phase 3.

Phase 2: Einfache, sichere Reparatur

HINWEIS: Wenn Sie wollen, dass die Reparatur sehr viel schneller abläuft, sollten Sie allen `isamchk/myisamchk`-Befehlen folgendes hinzufügen: `-O sort_buffer=# -O key_buffer=#` (wobei # etwa 1/4 des verfügbaren Arbeitsspeichers ist).

Probieren Sie zuerst `myisamchk -r -q tabelle` (`-r -q` bedeutet ``quick recovery mode" - schnelles Wiederherstellen). Dies versucht, die Index-Datei zu reparieren, ohne die Daten-Datei zu berühren. Wenn die Daten-Datei alles enthält, was sie sollte, und die Löschverknüpfungen auf die korrekten Stellen in der Daten-Datei zeigen, sollte das funktionieren und die Tabelle ist repariert. Machen Sie dann mit der Reparatur der nächsten Tabelle weiter. Ansonsten führen Sie folgende Prozedur durch:

1. Machen Sie eine Datensicherung der Daten-Datei, bevor Sie fortfahren.
2. Geben Sie `myisamchk -r tabelle` (`-r` bedeutet ``recovery mode" - Wiederherstellung) ein. Das entfernt falsche und gelöschte Datensätze aus der Daten-Datei und stellt die Index-Datei wieder her.
3. Wenn die vorherigen Schritte fehlschlagen, geben Sie `myisamchk --safe-recover tabelle` ein. Der Modus für sicheres Wiederherstellen benutzt eine alte Wiederherstellungsmethode, die ein paar Fälle behandelt, die der normale Wiederherstellungsmodus nicht behandelt (ist aber langsamer).

Wenn Sie bei der Reparatur merkwürdige Fehler erhalten (wie `out of memory`-Fehler), oder wenn `myisamchk` abstürzt, gehen Sie zu Phase 3.

Phase 3: Schwierige Reparatur

Diese Phase sollten Sie nur dann erreichen, wenn der erste 16-KB-Block der Index-Datei zerstört ist oder falsche Informationen enthält, oder wenn die Index-Datei fehlt. In diesem Fall ist es notwendig, eine neue Index-Datei zu erzeugen. Das machen Sie wie folgt:

1. Verschieben Sie die Daten-Datei an einen sicheren Ort.
2. Benutzen Sie die Tabellen-Beschreibungsdatei, um eine neue (leere) Daten-Datei und Index-Dateien zu erzeugen:

   ```
   shell> mysql datenbank
   mysql> SET AUTOCOMMIT=1;
   mysql> TRUNCATE TABLE tabelle;
   mysql> quit
   ```

   Wenn Ihre SQL-Version kein `TRUNCATE TABLE` hat, benutzen Sie statt dessen `DELETE FROM tabelle`.
3. Kopieren Sie Ihre alte Daten-Datei zurück, über die neu erzeugte Daten-Datei. (Verschieben Sie Ihre alte Daten-Datei nicht einfach, damit Sie eine Kopie erhalten, falls etwas schief geht.)

Gehen Sie zurück zu Phase 2. `myisamchk -r -q` sollte jetzt funktionieren. (Das sollte keine Endlosschleife sein.)

Phase 4: Sehr schwierige Reparatur

Diese Phase sollten Sie nur dann erreichen, wenn auch die Beschreibungsdatei beschädigt ist. Das sollte nie passieren, weil die Beschreibungsdatei nicht verändert wird, nachdem die Tabelle erzeugt wurde:

1. Stellen Sie die Beschreibungsdatei von einer Datensicherung wieder her und gehen Sie zurück zu Phase 3. Sie können auch die Index-Datei wiederherstellen und zu Phase 2 zurück gehen. Im letzteren Fall sollten Sie mit `myisamchk -r` anfangen.

2. Wenn Sie keine Datensicherung haben, aber genau wissen, wie die Tabelle erzeugt wurde, erzeugen Sie eine Kopie der Tabelle in einer anderen Datenbank. Entfernen Sie die neue Daten-Datei und verschieben Sie die Beschreibungs- und Index-Dateien von der anderen Datenbank in Ihre beschädigte Datenbank. Das ergibt neue Beschreibungs- und Index-Dateien, läßt aber die Daten-Datei in Ruhe. Gehen Sie zurück zu Phase 2 und versuchen Sie, die Index-Datei wiederherzustellen.

5.4.6.10. Tabellenoptimierung

Um fragmentierte Datensätze zu vereinigen und verschwendeten Speicherplatz zu beseitigen, der sich durch Löschen und Aktualisieren von Datensätzen ergibt, lassen Sie `myisamchk` im Wiederherstellungsmodus laufen:

```
shell> myisamchk -r tabelle
```

Auf dieselbe Weise können Sie eine Tabelle optimieren, indem Sie das SQL-Statement `OPTIMIZE TABLE` benutzen. `OPTIMIZE TABLE` führt eine Reparatur der Tabelle und eine Analyse der Schlüssel durch und sortiert den Indexbaum, um schnelleres Nachschlagen der Schlüssel (Key Lookup) zu ermöglichen. Ausserdem schaltet es die Möglichkeit ungewollter Interaktionen zwischen einem Dienstprogramm und dem Server aus, weil der Server bei der Benutzung von `OPTIMIZE TABLE` die ganze Arbeit verrichtet. See Abschnitt 5.5.1, „`OPTIMIZE TABLE`-Syntax“.

`myisamchk` hat eine Anzahl weiterer Optionen, die Sie für die Verbesserung der Performance einer Tabelle benutzen können:

- `-S, --sort-index`, `-R index_nummer, --sort-records=index_nummer`, `-a, --analyze`

Eine detaillierte Beschreibung der Optionen steht unter See Abschnitt 5.4.6.1, „Aufrufsyntax von `myisamchk`“.

5.4.7. Wartungsplan für Tabellen erstellen

Ab MySQL-Version 3.23.13 können Sie MyISAM-Tabellen mit dem `CHECK TABLE`-Befehl prüfen. See Abschnitt 5.4.4, „`CHECK TABLE`-Syntax“. Sie können Tabellen mit dem `REPAIR TABLE`-Befehl reparieren. See Abschnitt 5.4.5, „`REPAIR TABLE`-Syntax“.

Es ist eine gute Idee, Tabellenüberprüfungen auf regelmäßiger Basis durchzuführen statt darauf zu warten, dass Probleme auftreten. Für Wartungszwecke benutzen Sie `myisamchk -s`, um Tabellen zu überprüfen. Die `-s`-Option (Kurzform für `--silent`) veranlasst `myisamchk`, im schweigsamen Modus zu laufen, wobei Meldungen nur ausgegeben werden, wenn Fehler auftreten.

Ebenfalls eine gute Idee ist es, Tabellen zu überprüfen, wenn der Server hoch fährt. Wenn beispielsweise die Maschine mitten während einer Aktualisierung (Update) neu gebootet hat, müssen Sie üblicherweise alle Tabellen prüfen, die betroffen sein könnten. (Das ist ein Fall von ``erwarteter Tabellenbeschädigung".) Sie können `safe_mysqld` einen Test hinzufügen, der `myisamchk` laufen läßt, um alle Tabellen zu überprüfen, die innerhalb der letzten 24 Stunden geändert wurden, wenn nach einem Reboot eine alte `.pid`-(process ID)-Datei übrig blieb. (Die `.pid`-Datei wird von `mysqld` erzeugt, wenn er hoch fährt, und entfernt, wenn er normal beendet wird. Die Anwesenheit einer `.pid`-Datei beim Systemstart zeigt an, dass `mysqld` regelwidrig abgebrochen wurde.)

Eine noch bessere Testmethode besteht darin, jede Tabelle zu prüfen, deren Zeit der letzten Änderung neuer ist als die der `.pid`-Datei.

Ausserdem sollten Sie Ihre Tabellen regelmäßig während der normalen Systemtätigkeit prüfen. Bei MySQL AB lassen wir einen `cron`-Job laufen, um alle wichtigen Tabellen einmal pro Woche zu prüfen, indem wir folgende Zeile in der `crontab`-Datei benutzen:

```
35 0 * * 0 /pfad/zu/myisamchk --fast --silent /pfad/zu/datadir/*/*.MYI
```

Das gibt Informationen über beschädigte Tabellen aus, so dass wir diese prüfen und reparieren können, falls notwendig.

Da wir mittlerweile seit einigen Jahren keinerlei unerwartet beschädigte Tabellen hatten (Tabellen, die aus anderen Gründen als Hardware-Schäden beschädigt wurden), reicht uns einmal pro Woche völlig aus.

Wir empfehlen, dass Sie jede Nacht `myisamchk -s` auf alle Tabellen ausführen, die während der letzten 24 Stunden aktualisiert

wurden, bis Sie MySQL so sehr vertrauen, wie wir selbst das mittlerweile tun.

Normalerweise brauchen Sie MySQL-Tabellen nicht so sehr warten. Wenn Sie Tabellen mit Zeilen dynamischer Länge ändern (Tabellen mit `VARCHAR`-, `BLOB`- oder `TEXT`-Spalten) oder Tabellen mit vielen gelöschten Zeilen haben, werden Sie diese von Zeit zu Zeit (einmal im Monat?) defragmentieren wollen bzw. freien Speicherplatz schaffen.

Das können Sie mit `OPTIMIZE TABLE` auf die in Frage kommenden Tabellen tun, oder, wenn Sie den `mysqld`-Server für eine Weile herunter fahren können:

```
isamchk -r --silent --sort-index -O sort_buffer_size=16M */*.ISM
myisamchk -r --silent --sort-index  -O sort_buffer_size=16M */*.MYI
```

5.4.8. Informationen über eine Tabelle erhalten

Um eine Beschreibung einer Tabelle oder Statistiken über sie zu erhalten, benutzen Sie die unten stehenden Befehle. Einige davon werden später detaillierter erläutert:

- `myisamchk -d tabelle`

  Läßt `myisamchk` im ``Beschreibungsmodus" laufen, um eine Beschreibung Ihrer Tabelle zu erzeugen. Wenn Sie den MySQL-Server mit der `--skip-locking`-Option starten, kann `myisamchk` eventuell Fehler über eine Tabelle berichten, die aktualisiert wird, während es läuft. Weil `myisamchk` jedoch im Beschreibungsmodus keine Tabelle ändert, gibt es kein Risiko, dass Daten zerstört werden.

- `myisamchk -d -v tabelle`

  Um mehr Informationen über das, was `myisamchk` tut, zu erzeugen, fügen Sie `-v` als Option hinzu, damit es im geschwätzigen Modus läuft.

- `myisamchk -eis tabelle`

  Zeigt nur die wichtigsten Informationen über die Tabelle. Das ist langsam, weil es die ganze Tabelle lesen muss.

- `myisamchk -eiv tabelle`

  Wie `-eis`, sagt aber zusätzlich, was getan werden muss.

Beispiel einer `myisamchk -d`-Ausgabe:

```
MyISAM file:     firma.MYI
Record format:   Fixed length
Data records:    1403698 Deleted blocks:         0
Recordlength:    226

table description:
Key Start Len Index   Type
1   2     8   unique  double
2   15    10  multip. text packed stripped
3   219   8   multip. double
4   63    10  multip. text packed stripped
5   167   2   multip. unsigned short
6   177   4   multip. unsigned long
7   155   4   multip. text
8   138   4   multip. unsigned long
9   177   4   multip. unsigned long
    193   1           text
```

Beispiel einer `myisamchk -d -v`-Ausgabe:

```
MyISAM file:         firma
Record format:       Fixed length
File-version:        1
Creation time:       1999-10-30 12:12:51
Recover time:        1999-10-31 19:13:01
Status:              checked
Data records:          1403698  Deleted blocks:              0
Datafile parts:        1403698  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     3
Max datafile length: 3791650815  Max keyfile length: 4294967294
Recordlength:              226

table description:
Key Start Len Index   Type                 Rec/key     Root Blocksize
1   2     8   unique  double                     1 15845376      1024
2   15    10  multip. text packed stripped       2 25062400      1024
3   219   8   multip. double                    73 40907776      1024
4   63    10  multip. text packed stripped       5 48097280      1024
5   167   2   multip. unsigned short          4840 55200768      1024
6   177   4   multip. unsigned long           1346 65145856      1024
```

```
7   155   4   multip. text                    4995 75090944      1024
8   138   4   multip. unsigned long             87 85036032      1024
9   177   4   multip. unsigned long            178 96481280      1024
    193   1           text
```

Beispiel einer `myisamchk -eis`-Ausgabe:

```
Checking MyISAMDatei: firma
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:  98%  Packed:   17%

Records:          1403698    M.recordlength:     226   Packed:            0%
Recordspace used:     100%   Empty space:          0%  Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

User time 1626.51, System time 232.36
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 627, Swaps 0
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 639, Involuntary context switches 28966
```

Beispiel einer `myisamchk -eiv`-Ausgabe:

```
Checking MyISAM file: firma
Data records: 1403698   Deleted blocks:       0
- check file-size
- check delete-chain
block_size 1024:
index  1:
index  2:
index  3:
index  4:
index  5:
index  6:
index  7:
index  8:
index  9:
No recordlinks
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 2
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
- check data record references index: 3
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
- check data record references index: 5
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 6
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 7
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 8
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 9
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:   9%  Packed:   17%

- check records und index references
[LOTS OF ROW NUMBERS DELETED]

Records:          1403698    M.recordlength:     226   Packed:            0%
Recordspace used:     100%   Empty space:          0%  Blocks/Record:   1.00
Record blocks:    1403698    Delete blocks:        0
Recorddata:     317235748    Deleted data:         0
Lost space:             0    Linkdata:             0

User time 1639.63, System time 251.61
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 10580, Swaps 0
Blocks in 4 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 10604, Involuntary context switches 122798
```

Hier stehen die Größen der Daten- und Index-Dateien der Tabelle, die in den vorstehenden Beispielen benutzt wurde:

```
-rw-rw-r--   1 monty    tcx     317235748 Jan 12 17:30 firma.MYD
-rw-rw-r--   1 davida   tcx      96482304 Jan 12 18:35 firma.MYM
```

Erläuterungen der Informationen, die `myisamchk` erzeugt, werden unten gegeben. ``keyfile" ist die Index-Datei. ``Record" und ``row" sind Synonyme:

- `ISAM file`

  Name der ISAM-(Index)-Datei.

- `Isam-version`

  Version des ISAM-Formats. Momentan immer 2.

- `Creation time`

  Wann die Daten-Datei erzeugt wurde.

- `Recover time`

  Wann die Index-/Daten-Datei das letzte Mal rekonstruiert wurden.

- `Data records`

  Wie viele Datensätze in der Tabelle sind.

- `Deleted blocks`

  Wie viele gelöschte Blöcke noch Platz belegen. Sie können Ihre Tabelle optimieren, um diesen Platz zu minimieren. See Abschnitt 5.4.6.10, „Tabellenoptimierung“.

- `Datafile: Parts`

  Bei dynamischem Datensatzformat zeigt dies an, wie viele Datenblöcke es gibt. Bei einer optimierten Tabelle ohne fragmentierte Datensätze ist das dasselbe wie `Data records`.

- `Deleted data`

  Wie viele Bytes nicht zurückgewonnener gelöschter Daten es gibt. Sie können Ihre Tabelle optimieren, um diesen Platz zu minimieren. See Abschnitt 5.4.6.10, „Tabellenoptimierung“.

- `Datafile pointer`

  Die Größe des Daten-Datei-Zeigers in Bytes. Das sind normalerweise 2, 3, 4 oder 5 Bytes. Die meisten Tabellen schaffen 2 Bytes, aber das kann bislang von MySQL noch nicht gesteuert werden. Bei festen Tabellen ist das eine Datensatzadresse. Bei dynamischen Tabellen ist es eine Byte-Adresse.

- `Keyfile pointer`

  Die Größe des Index-Datei-Zeigers in Bytes. Sie beträgt normalerweise 1, 2 oder 3 Bytes. Die meisten Tabellen schaffen 2 Bytes, aber das wird von MySQL automatisch berechnet. Es ist immer die Block-Adresse.

- `Max datafile length`

  Wie lang die Daten-Datei (`.MYD`-Datei) der Tabelle werden kann, in Bytes.

- `Max keyfile length`

  Wie lang die Index-Datei (`.MYI`-Datei) der Tabelle werden kann, in Bytes.

- `Recordlength`

  Wie viel Platz jeder Datensatz benötigt, in Bytes.

- `Record format`

  Das Format, das benutzt wird, um Tabellenzeilen zu speichern. Die oben stehenden Beispiele benutzen `Fixed length`. Andere mögliche Werte sind `Compressed` und `Packed`.

- `Table description`

  Eine Liste aller Schlüssel in der Tabelle. Für jeden Schlüssel werden einige Low-Level-Informationen angezeigt:

  - `Key`

Die Nummer des Schlüssels.

- `Start`

  Wo im Datensatz dieser Index-Teil anfängt.

- `Len`

  Wie lang dieser Index-Teil ist. Bei gepackten Zahlen sollte das immer die gesamte Länge der Spalte sein. Bei Zeichenketten kann es kürzer als die gesamte Länge der indizierten Spalte sein, weil Sie ein Prefix einer Zeichenkettenspalte indexieren können.

- `Index`

  `unique` oder `multip.` (multiple). Zeigt an, ob ein Wert einmal oder mehrfach in diesem Index vorkommen darf.

- `Type`

  Welchen Datentyp dieser Index-Teil hat. Das ist ein ISAM-Datentyp mit den Optionen `packed`, `stripped` oder `empty`.

- `Root`

  Adresse des Root-Index-Blocks.

- `Blocksize`

  Die Größe jedes Index-Blocks. Vorgabemäßig ist das 1024, doch dieser Wert kann beim Kompilieren geändert werden.

- `Rec/key`

  Das ist ein statistischer Wert, der vom Optimierer benutzt wird. Es sagt aus, wie viele Datensätze es pro Wert für diesen Schlüssel gibt. Ein eindeutiger Schlüssel hat immer einen Wert von 1. Das kann aktualisiert werden, nachdem eine Tabelle geladen wurde (oder in größerem Umfang geändert) mit `myisamchk -a`. Wenn dies überhaupt nicht aktualisiert wurde, wird ein Wert von 30 angenommen.

- Im ersten Beispiel oben ist der neunte Schlüssel ein mehrteiliger Schlüssel mit zwei Teilen.
- `Keyblocks used`

  Welcher Prozentsatz von Schlüsselblöcken benutzt wird. Weil die Tabellen, die in den Beispielen benutzt wurden, direkt vorher mit `myisamchk` reorganisiert wurden, sind diese Werte sehr hoch (sehr nahe am theoretischen Maximum).

- `Packed`

  MySQL versucht, Schlüssel mit einem gemeinsamen Suffix zu packen. Das geht nur bei `CHAR`-, `VARCHAR` und `DECIMAL`-Schlüsseln. Bei langen Zeichenketten wie Namen kann das den benutzten Platz signifikant verringern. Im dritten Beispiel oben ist der vierte Schlüssel zehn Zeichen lang, wodurch ein 60%-ige Verringerung des Platzbedarfs erreicht wird.

- `Max levels`

  Wie tief der B-Baum für diesen Schlüssel ist. große Tabellen mit langen Schlüsseln haben hohe Werte.

- `Records`

  Wie viele Zeilen in der Tabelle enthalten sind.

- `M.recordlength`

  Die durchschnittliche Datensatzlänge. Bei Tabellen mit Datensätzen fester Länge ist das die exakte Datensatzlänge.

- `Packed`

  MySQL schneidet Leerzeichen am Ende von Zeichenketten ab. Der `Packed`-Wert zeigt an, welcher Prozentsatz dadurch gespart wurde.

- `Recordspace used`

  Welcher Prozentsatz der Daten-Datei benutzt wird.

- `Empty space`

Welcher Prozentsatz der Daten-Datei unbenutzt ist.

- `Blocks/Record`

  Durchschnittliche Anzahl der Blöcke pro Datensatz (das heißt, aus wie vielen Verknüpfungen (Links) ein fragmentierter Datensatz zusammengesetzt ist). Bei Tabellen mit festem Format ist das immer 1. Dieser Wert sollte so nah wie möglich an 1,0 bleiben. Wenn er zu Groß wird, können Sie die Tabelle `myisamchk` reorganisieren. See Abschnitt 5.4.6.10, „Tabellenoptimierung“.

- `Recordblocks`

  Wie viele Blöcke (Verknüpfungen, Links) benutzt werden. Bei festem Format ist das die Anzahl der Datensätze.

- `Deleteblocks`

  Wie viele Blöcke (Verknüpfungen, Links) gelöscht sind.

- `Recorddata`

  Wie viele Bytes in der Daten-Datei benutzt sind.

- `Deleted data`

  Wie viele Bytes in der Daten-Datei gelöscht sind (unbenutzt).

- `Lost space`

  Wenn ein Datensatz auf eine kürzere Länge aktualisiert wird, geht etwas Platz verloren. Das ist die Summe aller solcher Verluste in Bytes.

- `Linkdata`

  When das dynamische Tabellenformat benutzt wird, werden Datensatzfragmente mit Zeigern (Pointer) verknüpft (jeder mit 4 bis 7 Bytes). `Linkdata` ist die Summe des Speicherplatzes, der von diesen Zeigern benutzt wird.

Wenn eine Tabelle mit `myisampack` komprimiert wurde, gibt `myisamchk -d` zusätzliche Informationen über jede Tabellenspalte aus, siehe Abschnitt 5.7.4, „myisampack, MySQL-Programm zum Erzeugen komprimierter Nur-Lese-Tabellen“, wo sich ein Beispiel solcher Informationen und was sie bedeuten befindet.

5.5. Datenbankverwaltung Sprachreferenz

5.5.1. `OPTIMIZE TABLE`-Syntax

```
OPTIMIZE TABLE tabelle[,tabelle]...
```

`OPTIMIZE TABLE` sollte benutzt werden, wenn Sie große Teile der Tabelle gelöscht haben oder bei Tabellen mit Zeilen variabler Länge viele Änderungen durchgeführt haben (Tabellen, die `VARCHAR`-, `BLOB`- oder `TEXT`-Spalten enthalten). Gelöschte Datensätze werden in einer verknüpften Liste vorgehalten, und nachfolgenden `INSERT`-Operationen benutzen die Positionen alter Datensätze. Sie können `OPTIMIZE TABLE` benutzen, um unbenutzten Platz freizugeben und die Daten-Datei zu defragmentieren.

Momentan funktioniert `OPTIMIZE TABLE` nur auf **MyISAM**- und `BDB`-Tabellen. Bei `BDB`-Tabellen ist `OPTIMIZE TABLE` momentan auf `ANALYZE TABLE` gemappt. See Abschnitt 5.5.2, „`ANALYZE TABLE`-Syntax“.

Sie können `OPTIMIZE TABLE` auf andere Tabellentypen zum Laufen bringen, indem Sie `mysqld` mit `--skip-new` oder `--safe-mode` starten, aber in diesem Fall wird `OPTIMIZE TABLE` lediglich auf `ALTER TABLE` gemappt.

`OPTIMIZE TABLE` funktioniert wie folgt:

- Wenn die Tabelle gelöschte oder aufgeteilte Zeilen hat, wird sie repariert.
- Wenn die Index-Seiten nicht sortiert sind, werden sie sortiert.
- Wenn die Statistiken nicht aktuell sind (und eine Reparatur nicht durch das Sortieren des Indexes durchgeführt werden könnte), werden sie aktualisiert.

`OPTIMIZE TABLE` für `MyISAM`-Tabellen ist äquivalent zum Laufenlassen von `myisamchk --quick -`

`-check-changed-tables --sort-index --analyze` auf die Tabelle.

Beachten Sie, dass die Tabelle während der Zeit, in der `OPTIMIZE TABLE` läuft, gesperrt ist!

5.5.2. `ANALYZE TABLE`-Syntax

```
ANALYZE TABLE tabelle[,tabelle...]
```

Analysiert und speichert die Schlüsselverteilung der Tabelle. Während der Analyse ist die Tabelle mit einer Lesesperre gesperrt. Das funktioniert auf `MyISAM` und `BDB`-Tabellen.

Das ist äquivalent zum Laufenlassen von `myisamchk -a` auf die Tabelle.

MySQL benutzt die gespeicherte Schlüsselverteilung, um zu entscheiden, in welcher Reihenfolge Tabellen verknüpft werden sollen, wenn man eine Verknüpfung (Join) auf irgend etwas anderes als eine Konstante macht.

Der Befehl gibt eine Tabelle mit folgenden Spalten zurück:

Spalte	**Wert**
Table	Tabellenname.
Op	Immer ``analyze''.
Msg_type	`status`, `error`, `info` oder `warning`.
Msg_text	Die Meldung.

Sie können die gespeicherte Schlüsselverteilung mit dem `SHOW INDEX`-Befehl überprüfen. See Abschnitt 5.5.5.1, „Informationen über Datenbank, Tabellen, Spalten und Indexe abrufen“.

Wenn die Tabelle seit dem letzten `ANALYZE TABLE`-Befehl nicht geändert wurde, wird sie nicht noch einmal analysiert.

5.5.3. `FLUSH`-Syntax

```
FLUSH flush_option [,flush_option]
```

Wenn Sie einige der internen Caches, die MySQL benutzt, löschen wollen, benutzen Sie den `FLUSH`-Befehl. Um `FLUSH` ausführen zu können, müssen Sie die **RELOAD**-Berechtigung haben.

`flush_option` kann eine der folgenden sein:

`HOSTS`	Leert die Host-Cache-Tabellen. Sie sollten die Host-Tabellen flushen, wenn einige Ihrer Hosts die IP-Nummer ändern oder wenn Sie die Fehlermeldung `Host ... is blocked` erhalten. Wenn mehr als `max_connect_errors` Fehler in einer Zeile für einen gegebenen Host während der Verbindung zum MySQL-Server vorkommen, nimmt MySQL an, dass etwas nicht stimmt und blockiert den Host von weiteren Verbindungsversuchen. Wenn die Host-Tabellen geflusht werden, gestattet das dem Host, einen erneuten Verbindungsversuch zu machen. See Abschnitt A.2.4, „`Host '...' is blocked`-Fehler“. Sie können `mysqld` mit `-O max_connection_errors=999999999` starten, um diese Fehlermeldung zu vermeiden.
`LOGS`	Schließt alle Log-Dateien und öffnet sie danach wieder. Wenn Sie die Update-Log-Datei oder eine binäre Log-Datei ohne Erweiterung angegeben haben, wird die Erweiterungsnummer der Log-Datei um eins relativ zur vorherigen Datei hoch gezählt. Wenn Sie eine Erweiterung im Dateinamen benutzt haben, schließt MySQL die Update-Log-Datei und öffnet sie danach wieder. See Abschnitt 5.9.3, „Die Update-Log-Datei“. Das ist dasselbe, wie dem `mysqld`-Server das `SIGHUP`-Signal senden.
`PRIVILEGES`	Lädt die Berechtigungen aus den Berechtigungstabellen der `mysql`-Datenbank neu.
`TABLES`	Schließt alle offenen Tabellen und erzwingt, dass alle Tabellen in Benutzung geschlossen werden.
`[TABLE | TABLES] tabelle [,tabelle...]`	Flusht nur die angegebenen Tabellen.
`TABLES WITH READ LOCK`	Schließt alle offenen Tabellen und sperrt alle Tabellen aller Datenbanken mit einer Lesesperre, bis man `UNLOCK TABLES` ausführt. Das ist eine sehr bequeme Möglichkeit, Datensicherungen zu erzeugen, wenn Sie ein Dateisystem wie Veritas haben, das Schnappschüsse aufnehmen kann.
`STATUS`	Setzt alle Status-Variablen auf null zurück. Das sollte man nur benutzen, wenn man eine Anfrage debuggt.

Jeden der oben genannten Befehle können Sie auch mit dem `mysqladmin`-Dienstprogramm ausführen, indem Sie `flush-hosts`, `flush-logs`, `reload` oder `flush-tables`-Befehle eingeben.

Sehen Sie sich auch den `RESET`-Befehl an, der bei der Replikation benutzt wird. See Abschnitt 5.10.6, „SQL-Befehle in Bezug auf Replikation“.

5.5.4. `KILL`-Syntax

```
KILL Thread_id
```

Jede Verbindung zu `mysqld` läuft durch einen separaten Thread. Sie können sehen, welche Threads laufen, indem Sie den `SHOW PROCESSLIST`-Befehl ausführen, und einen Thread killen, indem Sie den `KILL Thread_id`-Befehl ausführen.

Wenn Sie die **process**-Berechtigung haben, können Sie alle Threads sehen und killen. Ansonsten können Sie nur Ihre eigenen Threads sehen und killen.

Sie können auch die `mysqladmin processlist`- und `mysqladmin kill`-Befehle benutzen, um Threads einzusehen und zu killen.

Wenn Sie ein `KILL` ausführen, wird ein Thread-spezifischer `kill flag` für den Thread gesetzt.

In den meisten Fällen kann es einige Zeit dauern, bis der Thread stirbt, weil der kill-Flag nur in bestimmten Intervallen geprüft wird:

- Bei `SELECT`-, `ORDER BY`- und `GROUP BY`-Schleifen wird der Flag geprüft, nachdem ein Block von Zeilen gelesen wurde. Wenn der kill-Flag gesetzt ist, wird das Statement abgebrochen.
- Bei `ALTER TABLE` wird der kill-Flag geprüft, bevor jeder Block von Zeilen aus der Original-Tabelle gelesen wird. Wenn der Flag gesetzt ist, wird der Befehl abgebrochen und die temporäre Tabelle wird gelöscht.
- Bei `UPDATE TABLE` und `DELETE TABLE` wird der kill-Flag geprüft, nachdem jeder Block gelesen wurde sowie nach jeder aktualisierten oder gelöschten Zeile. Wenn der Flag gesetzt ist, wird das Statement abgebrochen. Beachten Sie, dass die Änderungen nicht zurück gerollt (Rollback) werden, wenn Sie keine Transaktionen benutzen!
- `GET_LOCK()` wird mit `NULL` abgebrochen.
- Ein `INSERT DELAYED`-Thread flusht schnell alle Zeilen, die er im Speicher hat, und stirbt.
- Wenn der Thread im Tabellen-Lock-Handler ist (Status: `Locked`), wird die Tabellen-Sperre schnell abgebrochen.
- Wenn der Thread bei einem `write`-Aufruf auf freien Plattenplatz wartet, wird der Schreibvorgang mit einer Meldung, dass die Platte voll ist, abgebrochen.

5.5.5. `SHOW`-Syntax

```
   SHOW DATABASES [LIKE platzhalter]
oder SHOW [OPEN] TABLES [FROM datenbank] [LIKE platzhalter]
oder SHOW [FULL] COLUMNS FROM tabelle [FROM datenbank] [LIKE platzhalter]
oder SHOW INDEX FROM tabelle [FROM datenbank]
oder SHOW TABLE STATUS [FROM datenbank] [LIKE platzhalter]
oder SHOW STATUS [LIKE platzhalter]
oder SHOW VARIABLES [LIKE platzhalter]
oder SHOW LOGS
oder SHOW [FULL] PROCESSLIST
oder SHOW GRANTS FOR benutzer
oder SHOW CREATE TABLE tabelle
oder SHOW MASTER STATUS
oder SHOW MASTER LOGS
oder SHOW SLAVE STATUS
```

`SHOW` stellt Informationen über Datenbanken, Tabellen, Spalten oder Status-Informationen über den Server zur Verfügung. Wenn der `LIKE platzhalter`-Teil benutzt wird, kann die `platzhalter`-Zeichenkette eine Zeichenkette sein, die die SQL-‘`%`’- und ‘`_`’- Platzhalterzeichen benutzt.

5.5.5.1. Informationen über Datenbank, Tabellen, Spalten und Indexe abrufen

Sie können `datenbank.tabelle` als Alternative zur `tabelle FROM datenbank`-Syntax benutzen. Diese beiden Statements sind äquivalent:

```
mysql> SHOW INDEX FROM tabelle FROM datenbank;
mysql> SHOW INDEX FROM datenbank.tabelle;
```

`SHOW DATABASES` listet die Datenbanken auf dem MySQL-Server-Host auf. Diese Liste erhalten Sie auch mit dem `mysqlshow`-Befehl.

`SHOW TABLES` listet die Tabellen in einer gegebenen Datenbank auf. Sie erhalten diese Liste auch mit dem `mysqlshow datenbank`-Befehl.

HINWEIS: Wenn ein Benutzer keinerlei Berechtigungen für eine Tabelle hat, wird die Tabelle in der Ausgabe von `SHOW TABLES` oder `mysqlshow datenbank` nicht aufgeführt.

`SHOW OPEN TABLES` listet die Tabellen auf, die momentan im Tabellen-Cache geöffnet sind. See Abschnitt 6.4.6, „Wie MySQL Tabellen öffnet und schließt“. Das `Comment`-Feld zeigt an, wie oft die Tabelle gecachet (`cached`) und in Benutzung (`in_use`) ist.

`SHOW COLUMNS` listet die Spalten in einer gegebenen Tabelle auf. Wenn Sie die `FULL`-Option angeben, erhalten Sie auch die Berechtigungen, die Sie für jede Spalte besitzen. Wenn die Spaltentypen von dem abweichen, was Sie erwarten, nämlich, was Sie im `CREATE TABLE`-Statement angegeben haben, beachten Sie, dass MySQL manchmal Spaltentypen ändert. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“.

Das `DESCRIBE`-Statement gibt ähnliche Informationen wie `SHOW COLUMNS` aus. See Abschnitt 7.6.2, „`DESCRIBE`-Syntax (Informationen über Spalten erhalten)“.

`SHOW FIELDS` ist ein Synonym für `SHOW COLUMNS`. `SHOW KEYS` ist ein Synonym für `SHOW INDEX`. Sie können die Spalten oder Indexe einer Tabelle auch mit `mysqlshow Datenbanktabelle` oder `mysqlshow -k Datenbanktabelle` anzeigen.

`SHOW INDEX` gibt die Index-Informationen in einem Format aus, das dem `SQLStatistics`-Aufruf in ODBC stark ähnelt. Folgende Spalten werden zurückgegeben:

Spalte	Bedeutung
`Table`	Name der Tabelle.
`Non_unique`	0, wenn der Index keine Duplikate enthalten darf.
`Key_name`	Name des Indexes.
`Seq_in_index`	Spaltensequenznummer im Index, zählt ab 1.
`Column_name`	Spaltenname.
`Collation`	Wie die Spalte im Index sortiert ist. In MySQL können diese Werte ‘`A`’ (Ascending - aufsteigend) oder `NULL` (Not sorted - unsortiert) sein.
`Cardinality`	Anzahl der eindeutigen Werte im Index. Dieser Wert wird durch Laufenlassen von `isamchk -a` aktualisiert.
`Sub_part`	Anzahl der indizierten Zeichen, wenn die Spalte nur teilweise indiziert ist. `NULL`, wenn der gesamte Schlüssel indiziert ist.
`Comment`	Verschiedene Anmerkungen. Momentan teilt es nur mit, ob der Index FULLTEXT ist oder nicht.

Beachten Sie: Weil `Cardinality` basierend auf statistischen Werten gezählt wird, die als Ganzzahlen gespeichert sind, ist es nicht notwendigerweise bei kleinen Tabellen korrekt.

5.5.5.2. `SHOW TABLE STATUS`

```
SHOW TABLE STATUS [FROM datenbank] [LIKE platzhalter]
```

`SHOW TABLE STATUS` (neu in Version 3.23) funktioniert wie `SHOW STATUS`, zeigt aber viele weitere Informationen über jede Tabelle. Diese Liste erhalten Sie auch mit dem `mysqlshow --status datenbank`-Befehl. Folgende Spalten werden zurückgegeben:

Spalte	Bedeutung
`Name`	Name der Tabelle.
`Type`	Typ der Tabelle. See Kapitel 8, *MySQL-Tabellentypen*.
`Row_format`	Das Zeilenspeicherformat (fest, dynamisch oder komprimiert).
`Rows`	Anzahl der Zeilen.
`Avg_row_length`	Durchschnittliche Zeilenlänge.
`Data_length`	Länge der Daten-Datei.

`Max_data_length`	Maximale Länge der Daten-Datei.
`Index_length`	Länge der Index-Datei.
`Data_free`	Anzahl der zugewiesenen (allocated), aber nicht benutzten Bytes.
`Auto_increment`	Nächster autoincrement-Wert.
`Create_time`	Wann die Tabelle erzeugt wurde.
`Update_time`	Wann die Daten-Datei das letzte Mal aktualisiert wurde.
`Check_time`	Wann die Tabelle das letzte Mal geprüft wurde.
`Create_options`	Zusätzliche Optionen, die beim `CREATE TABLE` benutzt wurden.
`Comment`	Der Kommentar, der beim Erzeugen der Tabelle angegeben wurde (oder einige Informationen, warum MySQL nicht auf die Tabelleninformationen zugreifen konnte).

Bei `InnoDB`-Tabellen wird im Tabellenkommentar der freie Platz im Tablespace ausgegeben.

5.5.5.3. `SHOW STATUS`

`SHOW STATUS` zeigt Server-Status-Informationen an (wie `mysqladmin extended-status`). Die Ausgabe ähnelt der unten stehenden, obwohl Format und Anzahl der Zeilen wahrscheinlich abweichen:

```
+--------------------------+------------+
| Variable_name            | Value      |
+--------------------------+------------+
| Aborted_clients          | 0          |
| Aborted_connects         | 0          |
| Bytes_received           | 155372598  |
| Bytes_sent               | 1176560426 |
| Connections              | 30023      |
| Created_tmp_disk_tables  | 0          |
| Created_tmp_tables       | 8340       |
| Created_tmp_files        | 60         |
| Delayed_insert_Threads   | 0          |
| Delayed_writes           | 0          |
| Delayed_errors           | 0          |
| Flush_commands           | 1          |
| Handler_delete           | 462604     |
| Handler_read_first       | 105881     |
| Handler_read_key         | 27820558   |
| Handler_read_next        | 390681754  |
| Handler_read_prev        | 6022500    |
| Handler_read_rnd         | 30546748   |
| Handler_read_rnd_next    | 246216530  |
| Handler_update           | 16945404   |
| Handler_write            | 60356676   |
| Key_blocks_used          | 14955      |
| Key_read_requests        | 96854827   |
| Key_reads                | 162040     |
| Key_write_requests       | 7589728    |
| Key_writes               | 3813196    |
| Max_used_connections     | 0          |
| Not_flushed_key_blocks   | 0          |
| Not_flushed_delayed_rows | 0          |
| Open_tables              | 1          |
| Open_files               | 2          |
| Open_streams             | 0          |
| Opened_tables            | 44600      |
| Questions                | 2026873    |
| Select_full_join         | 0          |
| Select_full_range_join   | 0          |
| Select_range             | 99646      |
| Select_range_check       | 0          |
| Select_scan              | 30802      |
| Slave_running            | OFF        |
| Slave_open_temp_tables   | 0          |
| Slow_launch_threads      | 0          |
| Slow_queries             | 0          |
| Sort_merge_passes        | 30         |
| Sort_range               | 500        |
| Sort_rows                | 30296250   |
| Sort_scan                | 4650       |
| Table_locks_immediate    | 1920382    |
| Table_locks_waited       | 0          |
| Threads_cached           | 0          |
| Threads_created          | 30022      |
| Threads_connected        | 1          |
| Threads_running          | 1          |
| Uptime                   | 80380      |
+--------------------------+------------+
```

The status variables listed höher have the following Bedeutung:

Variable	**Bedeutung**

`Aborted_clients`	Anzahl der Verbindungen, die abgebrochen wurden, weil der Client starb, ohne die Verbindung ordnungsgemäß zu schließen. See Abschnitt A.2.9, „Kommunikationsfehler / Abgebrochene Verbindung“.
`Aborted_connects`	Anzahl der fehlgeschlagenen Versuche, sich mit dem MySQL-Server zu verbinden. See Abschnitt A.2.9, „Kommunikationsfehler / Abgebrochene Verbindung“.
`Bytes_received`	Anzahl der Bytes, die von allen Clients empfangen wurden.
`Bytes_sent`	Anzahl der Bytes, die an alle Clients gesendet wurden.
`Connections`	Anzahl der Verbindungsversuche zum MySQL-Server.
`Created_tmp_disk_tables`	Anzahl der (implizit) auf der Platte erzeugten temporären Tabellen bei der Ausführung von Statements.
`Created_tmp_tables`	Anzahl der (implizit) im Arbeitsspeicher erzeugten temporären Tabellen bei der Ausführung von Statements.
`Created_tmp_files`	Wie viele temporäre Dateien `mysqld` erzeugt hat.
`Delayed_insert_Threads`	Anzahl der verzögerten Insert-Handler-Threads in Benutzung.
`Delayed_writes`	Anzahl der Zeilen, die mit `INSERT DELAYED` geschrieben wurden.
`Delayed_errors`	Anzahl der Zeilen, die mit `INSERT DELAYED` geschrieben wurden, und bei denen irgend ein Fehler auftrat (wahrscheinlich `duplicate key`).
`Flush_commands`	Anzahl der ausgeführten `FLUSH`-Befehle.
`Handler_delete`	Wie oft eine Zeile aus einer Tabelle gelöscht wurde.
`Handler_read_first`	Wie oft der erste Eintrag aus einem Index gelesen wurde. Wenn dieser Wert hoch ist, legt das nahe, dass der Server viele komplette Index-Scans macht (zum Beispiel `SELECT spalte1 FROM foo`, unter der Annahme, dass spalte1 indiziert ist).
`Handler_read_key`	Anzahl der Anfragen, eine Zeile basierend auf einem Schlüssel zu lesen. Wenn dieser Wert hoch ist, ist das ein gutes Indiz dafür, dass Ihre Anfragen und Tabellen korrekt indiziert sind.
`Handler_read_next`	Anzahl der Anfragen, die nächste Zeile in der Reihenfolge des Schlüssels zu lesen. Dieser Wert wird herauf gezählt, wenn Sie eine Index-Spalte mit einer Bereichsbeschränkung (Limit) abfragen. Er wird ebenfalls herauf gezählt, wenn Sie einen Index-Scan durchführen.
`Handler_read_rnd`	Anzahl der Anfragen, eine Zeile basierend auf einer festen Position zu lesen. Dieser Wert wird hoch sein, wenn Sie viele Anfragen ausführen, die erfordern, dass das Ergebnis sortiert wird.
`Handler_read_rnd_next`	Anzahl der Anfragen, die nächste Zeile in der Daten-Datei zu lesen. Dieser Wert wird hoch sein, wenn Sie viele Tabellen-Scans durchführen. Im Allgemeinen weist das darauf hin, dass Ihre Tabellen nicht korrekt indiziert sind, oder dass Ihre Anfragen nicht so geschrieben sind, dass Sie Vorteile aus den Indexen ziehen, die Sie haben.
`Handler_update`	Anzahl der Anfragen, eine Zeile in einer Tabelle zu aktualisieren.
`Handler_write`	Anzahl der Anfragen, eine Zeile in eine Tabelle einzufügen.
`Key_blocks_used`	Die Anzahl der benutzten Blocks im Schlüssel-Cache.
`Key_read_requests`	Die Anzahl der Anfragen, einen Schlüssel-Block aus dem Cache zu lesen.
`Key_reads`	Die Anzahl physikalischer Lesezugriffen eines Schlüssel-Blocks von der Platte.
`Key_write_requests`	Die Anzahl der Anfragen, einen Schlüssel-Block in den Cache zu schreiben.
`Key_writes`	Die Anzahl physikalischer Schreibvorgänge eines Schlüssel-Blocks auf Platte.
`Max_used_connections`	Die höchste Anzahl von Verbindungen, die gleichzeitig in Benutzung sind.
`Not_flushed_key_blocks`	Schlüssel-Blöcke im Schlüssel-Cache, die verändert wurden, aber noch nicht auf die Platte zurück geschrieben (flush).
`Not_flushed_delayed_rows`	Anzahl der Zeilen, die in `INSERT DELAY`-Warteschleifen darauf warten, geschrieben zu werden.
`Open_tables`	Anzahl der offenen Tabellen.
`Open_files`	Anzahl der offenen Dateien.
`Open_streams`	Anzahl der offenen Streams (hauptsächlich zum Loggen benutzt).
`Opened_tables`	Anzahl der Tabellen, die geöffnet wurden.
`Select_full_join`	Anzahl der Joins ohne Schlüssel (sollte 0 sein).
`Select_full_range_join`	Anzahl der Joins, bei denen eine Bereichssuche auf die Referenztabelle statt fand.

`Select_range`	Anzahl der Joins, bei denen Bereiche auf die erste Tabelle benutzt wurden. (Es ist normalerweise unkritisch, wenn dieser Wert hoch ist.)
`Select_scan`	Anzahl der Joins, bei denen die erste Tabelle gescannt wurde.
`Select_range_check`	Anzahl der Joins ohne Schlüssel, bei denen nach jeder Zeile auf Schlüsselbenutzung geprüft wurde (sollte 0 sein).
`Questions`	Anzahl der Anfragen, die zum Server geschickt wurden.
`Slave_open_temp_tables`	Anzahl der temporären Tabellen, die momentan vom Slave-Thread geöffnet sind.
`Slow_launch_threads`	Anzahl der Threads, die länger als `slow_launch_time` brauchten, um sich zu verbinden.
`Slow_queries`	Anzahl der Anfragen, die länger als `long_query_time` benötigten. See Abschnitt 5.9.5, „Die Anfragen-Log-Datei für langsame Anfragen“.
`Sort_merge_passes`	Anzahl der Verschmelzungen (Merge), die von einem Sortiervorgang benötigt wurden. Wenn dieser Wert hoch ist, sollten Sie in Betracht ziehen, `sort_buffer` herauf zu setzen.
`Sort_range`	Anzahl der Sortiervorgänge, die mit Bereichen durchgeführt wurden.
`Sort_rows`	Anzahl der sortierten Zeilen.
`Sort_scan`	Anzahl der Sortiervorgänge, die durchgeführt wurden, indem die Tabelle gescannt wurde.
`Table_locks_immediate`	Wie oft eine Tabellensperre sofort erlangt wurde. Verfügbar nach Version 3.23.33.
`Table_locks_waited`	Wie oft eine Tabellensperre nicht sofort erlangt werden konnte und gewartet werden musst. Wenn dieser Wert hoch ist und Sie Performance-Probleme haben, sollten Sie zunächst Ihre Anfragen optimieren und dann entweder Ihre Tabelle(n) zerteilen oder Replikation benutzen. Verfügbar nach Version 3.23.33.
`Threads_cached`	Anzahl der Threads im Thread-Cache.
`Threads_connected`	Anzahl der momentan offenen Verbindungen.
`Threads_created`	Anzahl der Threads, die zur Handhabung von Verbindungen erzeugt wurden.
`Threads_running`	Anzahl der Threads, die nicht schlafen.
`Uptime`	Seit wie vielen Sekunden der Server hoch gefahren ist.

Einige Anmerkungen zum oben Aufgeführten:

- Wenn `Opened_tables` hoch ist, ist Ihre `table_cache`-Variable wahrscheinlich zu niedrig.
- Wenn `key_reads` hoch ist, ist Ihr `key_cache` wahrscheinlich zu klein. Die Cache-Zugriffsrate kann mit `key_reads` / `key_read_requests` berechnet werden.
- Wenn `Handler_read_rnd` hoch ist, haben Sie wahrscheinlich viele Anfragen, die MySQL zwingen, ganze Tabellen zu scannen, oder Sie haben Joins, die Schlüssel nicht richtig benutzen.
- Wenn `Threads_created` hoch ist, sollten Sie eventuell die `Thread_cache_size`-Variable herauf setzen.
- Wenn `Created_tmp_disk_tables` hoch ist, sollten Sie eventuell die `tmp_table_size`-Variable herauf setzen, damit temporäre Tabellen im Speicher erzeugt werden statt auf der Platte.

5.5.5.4. `SHOW VARIABLES`

```
SHOW VARIABLES [LIKE platzhalter]
```

`SHOW VARIABLES` zeigt die Werte einiger MySQL-Systemvariablen. Sie erhalten diese List auch mit dem `mysqladmin variables`-Befehl. Wenn die Vorgabewerte unpassend sind, können Sie die meisten dieser Variablen mit Kommandozeilenoptionen setzen, wenn Sie `mysqld` hoch fahren. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

Die Ausgabe ähnelt der unten stehenden, obwohl Format und Anzahl der Zeilen wahrscheinlich abweichen:

```
+-------------------------+--------------------------+
| Variable_name           | Value                    |
+-------------------------+--------------------------+
| ansi_mode               | OFF                      |
| back_log                | 50                       |
| basedir                 | /my/monty/               |
| bdb_cache_size          | 16777216                 |
```

```
| bdb_log_buffer_size     | 32768                       |
| bdb_home                | /my/monty/data/             |
| bdb_max_lock            | 10000                       |
| bdb_logdir              |                             |
| bdb_shared_data         | OFF                         |
| bdb_tmpdir              | /tmp/                       |
| binlog_cache_size       | 32768                       |
| concurrent_insert       | ON                          |
| connect_timeout         | 5                           |
| datadir                 | /my/monty/data/             |
| delay_key_write         | ON                          |
| delayed_insert_limit    | 100                         |
| delayed_insert_timeout  | 300                         |
| delayed_queue_size      | 1000                        |
| flush                   | OFF                         |
| flush_time              | 0                           |
| have_bdb                | YES                         |
| have_innodb             | YES                         |
| have_raid               | YES                         |
| have_openssl            | NO                          |
| init_file               |                             |
| interactive_timeout     | 28800                       |
| join_buffer_size        | 131072                      |
| key_buffer_size         | 16776192                    |
| language                | /my/monty/share/english/    |
| large_files_support     | ON                          |
| log                     | OFF                         |
| log_update              | OFF                         |
| log_bin                 | OFF                         |
| log_slave_updates       | OFF                         |
| long_query_time         | 10                          |
| low_priority_updates    | OFF                         |
| lower_case_table_names  | 0                           |
| max_allowed_packet      | 1048576                     |
| max_binlog_cache_size   | 4294967295                  |
| max_connections         | 100                         |
| max_connect_errors      | 10                          |
| max_delayed_threads     | 20                          |
| max_heap_table_size     | 16777216                    |
| max_join_size           | 4294967295                  |
| max_sort_length         | 1024                        |
| max_tmp_tables          | 32                          |
| max_write_lock_count    | 4294967295                  |
| myisam_recover_options  | DEFAULT                     |
| myisam_sort_buffer_size | 8388608                     |
| net_buffer_length       | 16384                       |
| net_read_timeout        | 30                          |
| net_retry_count         | 10                          |
| net_write_timeout       | 60                          |
| open_files_limit        | 0                           |
| pid_file                | /my/monty/data/donna.pid    |
| port                    | 3306                        |
| protocol_version        | 10                          |
| record_buffer           | 131072                      |
| query_buffer_size       | 0                           |
| safe_show_database      | OFF                         |
| server_id               | 0                           |
| skip_locking            | ON                          |
| skip_networking         | OFF                         |
| skip_show_database      | OFF                         |
| slow_launch_time        | 2                           |
| socket                  | /tmp/mysql.sock             |
| sort_buffer             | 2097116                     |
| table_cache             | 64                          |
| table_type              | MYISAM                      |
| Thread_cache_size       | 4                           |
| Thread_stack            | 65536                       |
| tmp_table_size          | 1048576                     |
| tmpdir                  | /tmp/                       |
| version                 | 3.23.29a-gamma-debug        |
| wait_timeout            | 28800                       |
+-------------------------+-----------------------------+
```

Jede Option ist unten beschrieben. Die Werte für Puffergrößen, Längen und Stack-Größen sind in Bytes angegeben. Sie können Wert mit den Suffixen '`K`' oder '`M`' angeben, um Kilobytes oder Megabytes zu kennzeichnen. `16M` zum Beispiel bedeutet 16 Megabytes. Bei den Suffixen spielt Groß-/Kleinschreibung keine Rolle, `16M` und `16m` sind äquivalent:

- `ansi_mode.`

  Ist `ON`, wenn `mysqld` mit `--ansi` gestartet wurde. See Abschnitt 2.7.2, „MySQL im ANSI-Modus laufen lassen“.

- `back_log`

  Die Anzahl unerledigter Verbindungsanforderung, die MySQL haben kann. Dies kommt ins Spiel, wenn der Haupt-Thread von MySQL **SEHR** viele Verbindungsanforderungen in sehr kurzer Zeit erhält. Dann dauert es etwas (wenngleich sehr kurz), damit der Haupt-Thread die Verbindung prüfen und einen neuen Thread starten kann. Der `back_log`-Wert zeigt an, wie viele Verbindungen während dieser kurzen Zeit gestapelt (gestackt) werden können, bevor MySQL für einen Moment aufhört, neue Anforderungen zu beantworten. Sie brauchen diesen Wert nur dann herauf setzen, wenn Sie eine große Zahl von Verbindungen

in kurzer Zeit erwarten.

Mit anderen Worten ist dieser Wert die Größe der Listen-Queue (Warteschlange) für herein kommende TCP/IP-Verbindungen. Ihr Betriebssystem hat seine eigene Beschränkung hinsichtlich der Größe dieser Queue. Die Handbuchseiten zum Unix-`listen(2)`-System sollten hier weitere Details haben. Sehen Sie in der Dokumentation Ihres Betriebssystems nach, wie hoch der Wert dieser Variablen maximal sein kann. Wenn Sie versuchen, `back_log` höher als die Begrenzung Ihres Betriebssystems zu setzen, ist das ineffektiv.

- `basedir`

  Der Wert der `--basedir`-Option.

- `bdb_cache_size`

  Der zugewiesene Puffer, um Index und Zeilen bei `BDB`-Tabellen zu cachen. Wenn Sie keine `BDB`-Tabellen benutzen, sollten Sie `mysqld` mit `--skip-bdb` starten, um für diesen Cache keinen Arbeitsspeicher zu verschwenden.

- `bdb_log_buffer_size`

  Der zugewiesene Puffer, um Index und Zeilen bei `BDB`-Tabellen zu cachen. Wenn Sie keine `BDB`-Tabellen benutzen, sollten Sie diesen Wert auf o setzen und `mysqld` mit `--skip-bdb` starten, um für diesen Cache keinen Arbeitsspeicher zu verschwenden.

- `bdb_home`

  Der Wert der `--bdb-home`-Option.

- `bdb_max_lock`

  Die maximale Anzahl von Sperren (Vorgabewert: 1000), die bei einer `BDB`-Tabelle aktiv sein können. Sie sollten diesen Wert herauf setzen, wenn Sie Fehler folgender Art bekommen: `bdb: Lock table is out of available locks` oder `Got error 12 from ...`, wenn Sie lange Transaktionen durchführen oder wenn `mysqld` viele Zeile untersuchen muss, um die Anfrage zu berechnen.

- `bdb_logdir`

  Der Wert der `--bdb-logdir`-Option.

- `bdb_shared_data`

  Ist `ON`, wenn Sie `--bdb-shared-data` benutzen.

- `bdb_tmpdir`

  Der Wert der `--bdb-tmpdir`-Option.

- `binlog_cache_size. Die Größe des Caches, in dem`

  SQL-Statements für das Binär-Log während einer Transaktion vorgehalten werden. Wenn Sie oft große, aus vielen Statements bestehende Transaktionen durchführen, können Sie diesen Wert herauf setzen, um mehr Performance zu erzielen. See Abschnitt 7.7.1, „`BEGIN/COMMIT/ROLLBACK`-Syntax“.

- `character_set`

  Der vorgabemäßige Zeichensatz.

- `character_sets`

  Die unterstützten Zeichensätze.

- `concurrent_inserts`

  Falls `ON` (Vorgabe), läßt MySQL `INSERT` auf `MyISAM`-Tabellen zu, auf die zur gleichen Zeit `SELECT`-Anfragen laufen. Sie können diese Option ausschalten, indem Sie `mysqld` mit `--safe` oder `--skip-new` starten.

- `connect_timeout`

  Die Anzahl von Sekunden, die der `mysqld`-Server auf ein Verbindungspaket wartet, bevor er mit `Bad handshake` antwortet.

- `datadir`

Der Wert der `--datadir`-Option.

- `delay_key_write`

  Falls angeschaltet (Vorgabe), akzeptiert MySQL die `delay_key_write`-Option von `CREATE TABLE`. Das heißt, dass der Schlüsselpuffer für Tabellen bei dieser Option nicht bei jeder Index-Aktualisierung auf Platte zurückgeschrieben (flush) wird, sondern nur, wenn eine Tabelle geschlossen wird. Das beschleunigt Schreibvorgänge auf Schlüssel ganz erheblich, aber Sie sollten eine automatische Prüfung aller Tabellen mit `myisamchk --fast --force` hinzufügen, wenn Sie diese Option benutzen. Beachten Sie: Wenn Sie `mysqld` mit der `--delay-key-write-for-all-tables`-Option startet, heißt das, dass alle Tabelle so behandelt werden, als wenn sie mit der `delay_key_write`-Option erzeugt worden wären. Sie können diesen Flag löschen, wenn Sie `mysqld` mit `--skip-new` oder `--safe-mode` starten.

- `delayed_insert_limit`

  Nachdem `delayed_insert_limit` Zeilen eingefügt wurden, prüft der `INSERT DELAYED`-Handler, ob noch irgend welche `SELECT`-Statements anhängig sind. Falls ja, wird deren Ausführung zugelassen, bevor weiter gemacht wird.

- `delayed_insert_timeout`

  Wie lange ein `INSERT DELAYED`-Thread auf `INSERT`-Statements warten soll, bevor abgebrochen wird.

- `delayed_queue_size`

  Welche Warteschleifen-(Queue)-Speichergröße (in Zeilen) für die Handhabung von `INSERT DELAYED` zugewiesen werden soll. Wenn die Queue voll ist, wartet jeder Client, der `INSERT DELAYED` ausführt, bis es wieder Platz in der Queue gibt.

- `flush`

  Ist `ON`, wenn Sie MySQL mit der `--flush`-Option gestartet haben.

- `flush_time`

  Wenn diese Variable auf einen Wert ungleich 0 gesetzt wird, dann werden alle `flush_time` Sekunden alle Tabelle geschlossen (um Ressourcen frei zu geben und Dinge auf Platte zurück zu schreiben). Diese Option empfehlen wir nur auf Windows 95, Windows 98 oder auf Systemen, auf denen Sie sehr wenige Ressourcen haben.

- `have_bdb`

  Ist `YES`, wenn `mysqld` Berkeley-DB-Tabellen unterstützt. Ist `DISABLED`, wenn `--skip-bdb` benutzt wird.

- `have_innodb`

  Ist `YES`, wenn `mysqld` InnoDB-Tabellen unterstützt. Ist `DISABLED`, wenn `--skip-innodb` benutzt wird.

- `have_raid`

  Ist `YES`, wenn `mysqld` die `RAID`-Option unterstützt.

- `have_openssl`

  Ist `YES`, wenn `mysqld` SSL (Verschlüsselung) auf dem Client-/Server-Protokoll unterstützt.

- `init_file`

  Der Name der Datei, die mit der `--init-file`-Option angegeben wurde, als Sie den Server starteten. Das ist eine Datei mit SQL-Statements, die der Server beim Start ausführen soll.

- `interactive_timeout`

  Die Anzahl von Sekunden, die der Server bei einer interaktiven Verbindung wartet, bis er sie schließt. Ein interaktiver Client ist definiert als Client, der die `CLIENT_INTERACTIVE`-Option für `mysql_real_connect()` benutzt. Siehe auch `wait_timeout`.

- `join_buffer_size`

  Die Größe des Puffers, der für volle Joins benutzt wird (Joins, die keine Indexe benutzen). Der Puffer wird einmal pro vollem Join zwischen zwei Tabellen zugewiesen. Setzen Sie diesen Wert herauf, um einen schnelleren vollen Join zu erhalten, wenn das Addieren von Indexen nicht möglich ist. (Normalerweise ist die beste Art, schnelle Joins zu erhalten, das Addieren von Indexen.)

- `key_buffer_size`

  Index-Blöcke werden gepuffert und von allen Threads geteilt. `key_buffer_size` ist die Größe des Puffers, der für Index-Blöcke benutzt wird.

  Setzen Sie diesen Wert herauf, um eine bessere Index-Handhabung zu erzielen (für alle Lesevorgänge und für mehrfache Schreibvorgänge), so weit, wie Sie es sich leisten können; 64 MB auf einer 256-MB-Maschine, auf der hauptsächlich MySQL läuft, ist ein gebräuchlicher Wert. Wenn Sie diesen Wert allerdings zu hoch setzen (mehr als 50% Ihres gesamten Arbeitsspeichers), kann es sein, dass Ihr System anfängt auszulagern (Paging), was SEHR langsam werden kann. Denken Sie daran, dass Sie Platzt für den Dateisystem-Cache des Betriebssystems lassen müssen, weil MySQL Daten-Lesen nicht cachet.

  Sie können die Performance des Schlüsselpuffers mit `show status` überprüfen und sich die Variablen `Key_read_requests`, `Key_reads`, `Key_write_requests` und `Key_writes` ansehen. Das Verhältnis `Key_reads/Key_read_request` sollte normalerweise < 0,01 sein. `Key_write/Key_write_requests` ist üblicherweise nahe 1, wenn Sie hauptsächlich Aktualisieren (Update) und Löschen (Delete) ausführen, kann aber sehr viel kleiner werden, wenn Sie tendenziell Aktualisierungen ausführen, die viele Zeilen gleichzeitig betreffen, oder wenn Sie `delay_key_write` benutzen. See Abschnitt 5.5.5, „SHOW-Syntax“.

  Um noch mehr Geschwindigkeit beim Schreiben vieler Zeilen auf einmal zu erhalten, benutzen Sie `LOCK TABLES`. See Abschnitt 7.7.2, „LOCK TABLES/UNLOCK TABLES-Syntax“.

- `language`

  Die Sprache, in der Fehlermeldungen ausgegeben werden.

- `large_file_support`

  Ob `mysqld` mit Optionen für die Unterstützung großer Dateien kompiliert wurde.

- `locked_in_memory`

  Ob `mysqld` mit `--memlock` in den Speicher gesperrt wurde.

- `log`

  Ob das Loggen aller Anfragen angeschaltet ist.

- `log_update`

  Ob das Update-Log angeschaltet ist.

- `log_bin`

  Ob das Binär-Log angeschaltet ist.

- `log_slave_updates`

  Ob Aktualisierungen vom Slave geloggt werden sollen.

- `long_query_time`

  Wenn eine Anfrage länger als diesen Wert (in Sekunden) benötigt, wird der `Slow_queries`-Zähler hoch gezählt. Wenn Sie `--log-slow-queries` benutzen, wird die Anfrage in die Slow-Query-Logdatei geschrieben. See Abschnitt 5.9.5, „Die Anfragen-Log-Datei für langsame Anfragen“.

- `lower_case_table_names`

  Wenn auf 1 gesetzt, werden Tabellennamen in Kleinschreibung auf Platte gespeichert. Tabellennamen sind dann unabhängig von der verwendeten Groß-/Kleinschreibung. See Abschnitt A.5.1, „Groß-/Kleinschreibung beim Suchen“.

- `max_allowed_packet`

  Die maximale Größe eine Pakets. Der Nachrichtenpuffer wird auf `net_buffer_length` Bytes Länge initialisiert, kann aber wenn nötig bis zu `max_allowed_packet` Bytes Groß werden. Der Vorgabewert ist klein, um große (möglicherweise falsche) Pakete abzufangen. Sie müssen diesen Wert erhöhen, wenn Sie große `BLOB`-Spalten verwenden. Er sollte so Groß sein wie die größte `BLOB`-Spalte, die Sie verwenden wollen. Das aktuelle Protokoll begrenzt `max_allowed_packet` auf 16 MB.

- `max_binlog_cache_size`

Wenn eine Transaktion aus mehreren Statements mehr als diese Speichermenge benötigt, erhält man den Fehler "Multi-Statement transaction required more than 'max_binlog_cache_size' bytes of storage".

- `max_binlog_size`

  Verfügbar nach Version 3.23.33. Wenn ein Schreibvorgang ins binäre (Replikations-) Log den angegebenen Wert übersteigt, werden die Logs rotiert. Sie können den Wert auf weniger als 1024 Bytes setzen oder auf mehr als 1 GB. Vorgabe ist 1 GB.

- `max_connections`

  Die Anzahl von Clients, die gleichzeitig verbunden sind. Wenn Sie diesen Wert hoch setzen, wird die Anzahl der Datei-Deskriptoren heraufgesetzt, die `mysqld` benötigt. Siehe weiter unten, Bemerkungen zu Beschränkungen bei Datei-Deskriptoren. See Abschnitt A.2.5, „`Too many connections`-Fehler“.

- `max_connect_errors`

  Wenn es mehr als diese Anzahl unterbrochener Verbindungen von einem Host gibt, wird dieser Host von weiteren Verbindungen abgeschnitten. Sie können diese Sperre mit dem `FLUSH HOSTS`-Befehl aufheben.

- `max_delayed_Threads`

  Nicht mehr als diese Anzahl von Threads zulassen, um `INSERT DELAYED`-Statements abzuarbeiten. Wenn Sie versuchen, Daten in eine neue Tabelle einzufügen, wenn alle `INSERT DELAYED`-Threads in Benutzung sind, wird die Zeile eingefügt, als ob das `DELAYED`-Attribut nicht angegeben wäre.

- `max_heap_table_size`

  Kein Erzeugen von Heap-Tabellen zulassen, die größer als dieser Wert sind.

- `max_join_size`

  Joins, die wahrscheinlich mehr als `max_join_size` Datensätze lesen werden, geben einen Fehler zurück. Setzen Sie diesen Wert, wenn Ihre Benutzer dazu neigen, Joins auszuführen, denen eine `WHERE`-Klausel fehlt und die daher lange Zeit benötigen und womöglich Millionen von Zeilen zurück geben.

- `max_sort_length`

  Die Anzahl von Bytes, die beim Sortieren von `BLOB`- oder `TEXT`-Werten benutzt werden (nur die ersten `max_sort_length` Bytes jedes Werts werden benutzt, der Rest wird ignoriert).

- `max_user_connections`

  Die maximale Anzahl aktiver Verbindungen für einen einzelnen Benutzer (0 = keine Beschränkung).

- `max_tmp_tables`

  (Diese Option macht bislang noch nichts.) Maximale Anzahl von temporären Tabellen, die ein Client zur selben Zeit offen halten darf.

- `max_write_lock_count`

  Nach dieser Anzahl Schreibsperren wird einigen Lesesperren erlaubt, zwischendurch zu laufen.

- `myisam_recover_options`

  Der Wert der `--myisam-recover`-Option.

- `myisam_sort_buffer_size`

  Der Puffer, der beim Sortieren des Indexes zugewiesen wird, wenn man ein `REPAIR` oder ausführt oder Indexe mit `CREATE INDEX` oder `ALTER TABLE` erzeugt.

- `myisam_max_extra_sort_file_size.`

  Wenn das Erzeugen der temporären Datei für schnelle Index-Erzeugung um diesen Wert größer sein würde als die Benutzung des Schlüssel-Caches, wird die Schlüssel-Cache-Methode bevorzugt. Wird hauptsächlich benutzt, um lange Zeichen-Schlüssel in großen Tabellen zu zwingen, die langsamere Schlüssel-Cache-Methode zu benutzen, um den Index zu erzeugen. **HINWEIS:** Dieser Parameter wird in Megabytes angegeben!

- `myisam_max_sort_file_size`

Die maximale Größe der temporären Datei, die MySQL benutzen darf, während es den Index erzeugt (während `REPAIR`, `ALTER TABLE` oder `LOAD DATA INFILE`). Wenn die Datei größer als dieser Wert würde, wird der Index über den Schlüssel-Cache erzeugt (was langsamer ist). **HINWEIS:** Dieser Parameter wird in Megabytes angegeben!

- `net_buffer_length`

  Der Kommunikationspuffer wird zwischen Anfragen auf diesen Wert zurück gesetzt. Normalerweise sollte das nicht geändert werden, aber wenn Sie sehr wenig Arbeitsspeicher haben, können Sie ihn auf die erwartete Größe einer Anfrage setzen (also die erwartete Länge von SQL-Statements, die von Clients gesendet werden. Wenn Statements diese Länge überschreiten, wird der Puffer automatisch vergrößert, bis zu `max_allowed_packet` Bytes.)

- `net_read_timeout`

  Anzahl von Sekunden, die auf weitere Daten von einer Verbindung gewartet wird, bevor das Lesen abgebrochen wird. Beachten Sie: Wenn keine Daten von einer Verbindung erwartet werden, ist der Timeout durch `write_timeout` definiert. Siehe auch `slave_read_timeout`.

- `net_retry_count`

  Wenn ein Lesevorgang auf einem Kommunikations-Port unterbrochen wird, wird so oft wie angegeben neu versucht, bevor aufgegeben wird. Dieser Wert sollte auf `FreeBSD` recht hoch sein, weil interne Unterbrechnungsanforderungen (Interrupts) an alle Threads gesendet werden.

- `net_write_timeout`

  Anzahl von Sekunden, die auf das Schreiben eines Blocks zu einer Verbindung gewartet wird, bis das Schreiben abgebrochen wird.

- `open_files_limit`

  Wenn dieser Wert ungleich 0 ist, benutzt `mysqld` Datei-Deskriptoren, die mit `setrlimit()` benutzt werden. Wenn dieser Wert gleich 0 ist, reserviert `mysqld` `max_connections * 5` oder `max_connections + table_cache * 2` (je nachdem, was größer ist) Anzahl von Dateien. Sie sollten diesen Wert herauf setzen, wenn `mysqld` Ihnen die Fehlermeldung 'Too many open files' gibt.

- `pid_file`

  Der Wert der `--pid-file`-Option.

- `port`

  Der Wert der `--port`-Option.

- `protocol_version`

  Die Protokoll-Version, die vom MySQL-Server benutzt wird.

- `record_buffer`

  Jeder Thread, der einen sequentiellen Scan ausführt, alloziert einen Puffer dieser Größe für jede Tabelle, die er scannt. Wenn Sie viele sequentielle Scans ausführen, sollten Sie diesen Wert herauf setzen.

- `record_rnd_buffer`

  Wenn Zeilen nach einem Sortiervorgang in sortierter Reihenfolge gelesen werden, werden die Zeilen aus diesem Puffer gelesen, um Suchvorgänge auf der Platte zu vermeiden. Wenn dieser Wert nicht gesetzt ist, wird er auf den Wert von `record_buffer` gesetzt.

- `query_buffer_size`

  Die anfängliche Zuweisung des Anfragen-Puffers. Wenn die meisten Ihrer Anfragen lang sind (zum Beispiel beim Einfügen von Blobs), sollten Sie diesen Wert herauf setzen!

- `safe_show_databases`

  Keine Datenbanken zeigen, wenn der Benutzer keinerlei Datenbank- oder Tabellen-Berechtigungen dafür hat. Das kann die Sicherheit erhöhen, wenn Sie sich Sorgen machen, dass Leute in der Lage sind zu sehen, welche Datenbanken andere Benutzer haben. Siehe auch `skip_show_databases`.

- `server_id`

Der Wert der `--server-id`-Option.

- `skip_locking`

  Ist `OFF`, wenn `mysqld` externes Sperren benutzt.

- `skip_networking`

  Ist `ON`, wenn nur lokale (Socket-) Verbindungen zugelassen sind.

- `skip_show_databases`

  Hält Leute davon ab, `SHOW DATABASES` zu benutzen, wenn sie keine the `PROCESS_PRIV`-Berechtigung haben. Das kann die Sicherheit erhöhen, wenn Sie sich Sorgen machen, dass Leute in der Lage sind zu sehen, welche Datenbanken andere Benutzer haben. Siehe auch `safe_show_databases`.

- `slave_read_timeout`

  Anzahl von Sekunden, die auf weitere Daten von einer Master-/Slave-Verbindung gewartet wird, bevor das Lesen abgebrochen wird.

- `slow_launch_time`

  Wenn das Erzeugen des Threads länger als dieser Wert (in Sekunden) dauert, word der `Slow_launch_threads`-Zähler herauf gezählt.

- `socket`

  Der Unix-Socket, der vom Server benutzt wird.

- `sort_buffer`

  Jeder Thread, der einen Sortierdurchgang durchführen muss, alloziert einen Puffer dieser Größe. Setzen Sie diesen Wert herauf, um schnellere `ORDER BY`- oder `GROUP BY`-Operationen zu erhalten. See Abschnitt A.4.4, „Wohin MySQL temporäre Dateien speichert“.

- `table_cache`

  Die Anzahl offener Tabellen für alle Threads. Wenn dieser Wert herauf gesetzt wird, erhöht sich die Anzahl von Datei-Deskriptoren, die `mysqld` benötigt. Sie können prüfen, ob Sie den Tabellen-Cache vergrößern müssen, indem Sie die `Opened_tables`-Variable prüfen. See Abschnitt 5.5.5, „`SHOW`-Syntax“. Wenn diese Variable sehr Groß ist und Sie `FLUSH TABLES` nicht oft brauchen (was lediglich alle Tabellen zwingt, geschlossen und wieder geöffnet zu werden), sollten Sie den Wert dieser Variablen herauf setzen.

  Wegen weiterer Informationen zum Tabellen-Cache sehen Sie unter Abschnitt 6.4.6, „Wie MySQL Tabellen öffnet und schließt“ nach.

- `table_type`

  Der vorgabemäßige Tabellentyp.

- `thread_cache_size`

  Wie viele Threads in einem Cache für weitere Benutzung offen gehalten werden sollen. Wenn eine Client die Verbindung schließt, werden die Threads des Clients in den Cache geschrieben, wenn es nicht mehr als `Thread_cache_size` Threads als vorher gibt. Alle neuen Threads werden zuerst aus dem Cache genommen und nur, wenn der Cache leer ist, wird ein neuer Thread erzeugt. Diese Variable kann hoch gesetzt werden, um die Performance zu verbessern, wenn Sie sehr viele neue Verbindungen haben. (Normalerweise führt das nicht zu namhafter Performance-Steigerung, wenn Sie eine gute Thread-Implementierung haben.) Wie effizient der aktuelle Thread-Cache für Sie ist, können Sie feststellen, indem Sie den Unterschied zwischen `Connections` und `Threads_created` betrachten.

- `thread_concurrency`

  On Solaris, `mysqld` will call `thr_setconcurrency()` mit this value. `thr_setconcurrency()` permits the Applikation to give the Threads System a hint für the desired Anzahl von Threads that should be run at the same time.

- `thread_stack`

  Die Stack-Größe jedes Threads. Viele der Beschränkungen, die durch den `crash-me`-Test festgestellt werden, hängen von diesem Wert ab. Der Vorgabewert ist Groß genug für normale Operationen. See Abschnitt 6.1.4, „Die MySQL-

Benchmark-Suite“.

- `timezone`

  Die Zeitzone für den Server.

- `tmp_table_size`

  Wenn eine temporäre Tabelle im Arbeitsspeicher diese Größe überschreitet, wandelt MySQL sie automatisch in eine `MyISAM`-Tabelle auf der Platte um. Setzen Sie den Wert von `tmp_table_size` herauf, wenn Sie viele fortgeschrittene `GROUP BY`-Anfragen und viel Arbeitsspeicher haben.

- `tmpdir`

  Das Verzeichnis, das für temporäre Dateien und temporäre Tabellen benutzt wird.

- `version`

  Die Versionsnummer des Servers.

- `wait_timeout`

  Die Anzahl von Sekunden, die der Server auf Aktivität auf einer Verbindung wartet, bevor er sie schließt. Siehe auch `interactive_timeout`.

Der Handbuchabschnitt, der das Tunen von MySQL beschreibt, enthält einige Informationen darüber, wie die oben aufgeführten Variablen getunt werden. See Abschnitt 6.5.2, „Serverparameter tunen“.

5.5.5.5. SHOW LOGS

`SHOW LOGS` zeigt Ihnen Statusinformationen über bestehende Log-Dateien. Momentan werden nur Informationen über Berkeley-DB-Log-Dateien angezeigt.

- `File` zeigt den vollen Pfad zur Log-Datei.
- `Type` zeigt den Typ der Log-Datei (`BDB` für Berkeley-DB-Log-Dateien).
- `Status` zeigt den Status der Log-Datei (`FREE`, wenn die Datei entfernt werden kann, oder `IN USE`, wenn die Datei vom Transaktions-Subsystem benötigt wird)

5.5.5.6. SHOW PROCESSLIST

`SHOW PROCESSLIST` zeigt, welche Threads laufen. Diese Information erhalten Sie auch mit dem `mysqladmin processlist`-Befehl. Wenn Sie die **process**-Berechtigung haben, können Sie alle Threads sehen. Ansonsten sehen Sie nur Ihre eigenen Threads. See Abschnitt 5.5.4, „`KILL`-Syntax“. Wenn Sie die `FULL`-Option nicht benutzen, werden nur die ersten 100 Zeichen jeder Anfrage gezeigt.

Dieser Befehl ist sehr nützlich, wenn Sie die 'too many connections'-Fehlermeldung erhalten und herausfinden wollen, was vor sich geht. MySQL reserviert eine zusätzliche Verbindung für einen Client mit der `Process_priv`-Berechtigung, um sicherzustellen, dass Sie sich jederzeit einloggen und das System prüfen können (unter der Annahme, dass Sie diese Berechtigung nicht allen Ihren Benutzern geben).

5.5.5.7. SHOW GRANTS

`SHOW GRANTS FOR benutzer` listet die `Grant`-Befehle auf, die abgesetzt werden müssen, um die Berechtigungen für einen Benutzer zu duplizieren. Beispiel:

```
mysql> SHOW GRANTS FOR root@localhost;
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
```

5.5.5.8. SHOW CREATE TABLE

Zeigt ein `CREATE TABLE`-Statement an, das die angegebene Tabelle erzeugt:

```
mysql> show create table tabelle\G
*************************** 1. row ***************************
       Table: tabelle
Create Table: CREATE TABLE tabelle (
  id int(11) default NULL auto_increment,
  s char(60) default NULL,
  PRIMARY KEY (id)
) TYPE=MyISAM
```

`SHOW CREATE TABLE` setzt Tabellen- und Spaltennamen gemäß der `SQL_QUOTE_SHOW_CREATE`-Option in Anführungszeichen. Abschnitt 6.5.6, „SET-Syntax“.

5.6. MySQL-Lokalisierung und internationaler Gebrauch

5.6.1. Der für Daten und Sortieren benutzte Zeichensatz

Vorgabemäßig benutzt MySQL den ISO-8859-1-(Latin1)-Zeichensatz, wobei nach schwedischer / finnischer Reihenfolge sortiert wird. Dieser Zeichensatz ist für die USA und Westeuropa geeignet.

Alle standardmäßigen MySQL-Binärdistributionen werden mit `--with-extra-charsets=complex` kompiliert. Das fügt allen Standard-Programmen Code hinzu, damit diese `latin1` und alle Multi-Byte-Zeichensätze in der Binärdatei handhaben können. Andere Zeichensätze werden bei Bedarf aus einer Zeichensatz-Definitionsdatei geladen.

Der Zeichensatz legt fest, welche Zeichen in Namen erlaubt sind und wie Dinge durch die `ORDER BY`- und `GROUP BY`-Klauseln des `SELECT`-Statements sortiert werden.

Sie können den Zeichensatz mit der `--default-character-set`-Option ändern, wenn Sie den Server starten. Die verfügbaren Zeichensätze hängen von den `--with-charset=charset`- und `--with-extra-charset= list-of-charset | complex | all`-Optionen für `configure` ab und den Zeichensatz-Konfigurationsdateien, die in `SHAREDIR/charsets/Index` aufgeführt sind. See Abschnitt 3.3.3, „Typische `configure`-Optionen“.

Wenn Sie den Zeichensatz ändern, wenn Sie MySQL laufen lassen (was eventuell auch die Sortierreihenfolge ändert), müssen Sie `myisamchk -r -q` über alle Tabellen laufen lassen. Ansonsten sind Ihre Indexe eventuell nicht richtig sortiert.

Wenn sich ein Client mit dem MySQL-Server verbindet, schickt der Server den vorgabemäßigen Zeichensatz, der in Benutzung ist, an den Client. Der Client schaltet für diese Verbindung auf den Gebrauch dieses Zeichensatzes um.

Man sollte bei einer SQL-Anfrage `mysql_real_escape_string()` benutzen, wenn man Zeichenketten escapet. `mysql_real_escape_string()` ist identisch mit der alten `mysql_escape_string()`-Funktion, ausser dass es die MySQL-Connection-Handle als ersten Parameter nimmt.

Wenn der Client mit anderen Pfaden kompiliert wird, als wo der Server installiert ist, und der Benutzer, der MySQL konfigurierte, nicht alle Zeichensätze in die MySQL-Binärdatei eingeschlossen hat, muss man für den Client festlegen, wo dieser die zusätzlichen Zeichensätze finden kann, die er benötigt, falls der Server mit einem anderen Zeichensatz läuft als der Client.

Das kann man in einer MySQL-Optionsdatei festlegen:

```
[client]
character-sets-dir=/usr/local/mysql/share/mysql/charsets
```

Wobei der Pfad auf das Verzeichnis zeigt, in dem sich die dynamischen MySQL-Zeichensätze befinden.

Man kann den Client zwingen, einen bestimmten Zeichensatz zu benutzen, indem man angibt:

```
[client]
default-character-set=character-set-name
```

Aber normalerweise wird das nie benötigt.

5.6.1.1. Deutscher Zeichensatz

Um eine deutsche Sortierreihenfolge zu erhalten, startet man `mysqld` mit `--default-character-set=latin_de`. Das ergibt die folgenden Kennzeichen:

Beim Sortieren und Vergleichen von Zeichenketten wird das folgende Mapping auf die Zeichenketten durchgeführt, bevor der Vergleich ausgeführt wird:

```
ä  ->  ae
ö  ->  oe
ü  ->  ue
ß  ->  ss
```

Alle Akzentzeichen werden in ihr Nicht-Akzent-Pendant in Großschreibung umgewandelt. Alle Buchstaben werden in Großschreibung umgewandelt.

Beim Zeichenkettenvergleich mit `LIKE` wird das Mapping von einem auf zwei Buchstaben nicht durchgeführt. Alle Buchstaben werden in Großschreibung umgewandelt. Akzente werden aus allen Buchstaben entfernt, mit folgenden Ausnahmen: Ü, ü, Ö, ö, Ä und ä.

5.6.2. Nicht englische Fehlermeldungen

`mysqld` kann Fehlermeldungen in folgenden Sprachen ausgeben: tschechisch, dänisch, niederländisch, englisch (die Vorgabe), estnisch, französisch, deutsch, griechisch, ungarisch, italienisch, japanisch, koreanisch, norwegisch, norwegisch-ny, polnisch, portugiesisch, rumänisch, russisch, slowakisch, spanisch und schwedisch.

Um `mysqld` mit einer bestimmten Sprache zu starten, benutzen Sie die `--language=sprache` oder `-L sprache`-Optionen. Beispiel:

```
shell> mysqld --language=german
```

oder:

```
shell> mysqld --language=/usr/local/share/german
```

Beachten Sie, dass alle Sprachnamen in Kleinschreibung angegeben werden.

Die Sprachdateien liegen (vorgabemäßig) in *`mysql_base_dir`*`/share/`*`language/`*.

Um die Fehlermeldungsdatei zu aktualisieren, editieren Sie die `errmsg.txt`-Datei und führen folgenden Befehl aus, um die `errmsg.sys`-Datei zu erzeugen:

```
shell> comp_err errmsg.txt errmsg.sys
```

Wenn Sie auf eine neuere Version von MySQL aktualisieren, denken Sie daran, Ihre Änderungen mit der neuen `errmsg.txt`-Datei zu wiederholen!

5.6.3. Einen neuen Zeichensatz hinzufügen

Um MySQL einen weiteren Zeichensatz hinzuzufügen, führen Sie folgende Prozedur durch:

Entscheiden Sie, ob der Zeichensatz einfach oder komplex ist. Wenn der Zeichensatz keine besonderen Zeichenkettenvergleichsroutinen zum Sortieren und keine Multi-Byte-Unterstützung benötigt, ist er einfach. Wenn er eines oder beide Features benötigt, ist er komplex.

`latin1` und `dänisch` zum Beispiel sind einfache Zeichensätze, wohingegen `big5` oder `tschechisch` komplexe Zeichensätze sind.

Im folgenden Abschnitt wird angenommen, dass Sie Ihren Zeichensatz `MEINSET` nennen.

Bei einem einfachen Zeichensatz machen Sie folgendes:

1. Fügen Sie MEINSET am Ende der `sql/share/charsets/Index`-Datei hinzu. Geben Sie ihm eine eindeutige Nummer.
2. Erzeugen Sie die Datei `sql/share/charsets/MEINSET.conf`. (Sie können hierfür als Grundlage `sql/share/charsets/latin1.conf` benutzen).

   Die Syntax für die Datei ist sehr einfach:

   - Kommentare fangen mit einem '#'-Zeichen an und gehen bis zum Ende der Zeile.
   - Wörter werden durch beliebige Mengen von Leerraum getrennt.
   - Bei der Definition des Zeichensatzes muss jedes Wort eine Zahl im hexadezimalen Format sein.
   - Das `ctype`-Array nimmt bis zu 257 Wörter auf. Die `to_lower`-, `to_upper`- und `sort_order`-Arrays nehmen danach jeweils bis zu 256 Wörter auf.

   See Abschnitt 5.6.4, „Die Zeichen-Definitions-Arrays“.

3. Fügen Sie den Zeichensatznamen den `CHARSETS_AVAILABLE`- und `COMPILED_CHARSETS`-Listen in `configure.in` hinzu.

4. Rekonfigurieren, rekompilieren und testen Sie.

Bei einem komplexen Zeichensatz machen Sie folgendes:

1. Erzeugen Sie die Datei `strings/ctype-MEINSET.c` in der MySQL-Quelldistribution.

2. Fügen Sie MEINSET am Ende der `sql/share/charsets/Index`-Datei hinzu. Weisen Sie ihm eine eindeutige Nummer zu.

3. Sehen Sie sich eine der bestehenden `ctype-*.c`-Dateien an, um zu sehen, was definiert werden muss, zum Beispiel `strings/ctype-big5.c`. Beachten Sie, dass die Arrays in Ihrer Datei Namen wie `ctype_MEINSET`, `to_lower_MEINSET` usw. haben müssen. Das entspricht den Arrays im einfachen Zeichensatz. See Abschnitt 5.6.4, „Die Zeichen-Definitions-Arrays“. Bei einem komplexen Zeichensatz

4. fügen Sie am Anfang der Datei einen speziellen Kommentar wie folgt ein:

```
/*
 * Dieser Kommentar wird von configure geparst, um ctype.c zu erzeugen,
 * also ändern Sie ihn nicht, wenn Sie nicht genau wissen, was Sie tun.
 *
 * .configure. number_MEINSET=MYNUMBER
 * .configure. strxfrm_multiply_MEINSET=N
 * .configure. mbmaxlen_MEINSET=N
 */
```

   Das `configure`-Programm benutzt diesen Kommentar, um den Zeichensatz automatisch in die MySQL-Bibliothek einzufügen.

   Die Zeilen mit strxfrm_multiply und mbmaxlen werden in den folgenden Abschnitten erläutert. Geben Sie diese nur dann ein, wenn Sie die Zeichenketten-Vergleichsfunktionen oder die Multi-Byte-Zeichensatzfunktionen benötigen.

5. Danach sollten Sie einige der folgenden Funktionen erzeugen:

   - `my_strncoll_MEINSET()`
   - `my_strcoll_MEINSET()`
   - `my_strxfrm_MEINSET()`
   - `my_like_range_MEINSET()`

   See Abschnitt 5.6.5, „Unterstützung für Zeichenketten-Vergleich“.

6. Fügen Sie den Zeichensatznamen den `CHARSETS_AVAILABLE`- und `COMPILED_CHARSETS`-Listen in `configure.in` hinzu.

7. Rekonfigurieren, rekompilieren und testen Sie.

Die Datei `sql/share/charsets/README` enthält einige weitere Anweisungen.

Wenn Sie wollen, dass der Zeichensatz in die MySQL-Distribution aufgenommen wird, senden Sie einen Patch an `<internals@lists.mysql.com>`.

5.6.4. Die Zeichen-Definitions-Arrays

`to_lower[]` und `to_upper[]` sind einfache Arrays, die die Buchstaben in Klein- und Großschreibung enthalten, die jedem Mitglied des Zeichensatzes entsprechen. Beispiel:

```
to_lower['A'] enthält 'a'
to_upper['a'] enthält 'A'
```

`sort_order[]` ist eine Map, die anzeigt, wie Buchstaben für Vergleichs- und Sortierzwecke geordnet werden sollten. Bei vielen Zeichensätzen ist das dasselbe wie `to_upper[]` (was bedeutet, dass das Sortieren ohne Berücksichtigung der Groß-/Kleinschreibung erfolgt). MySQL sortiert Buchstaben auf der Grundlage des Wertes von `sort_order[character]`. Wegen komplizierterer Sortierregeln sehen Sie die Erörterung zu Zeichenketten-Vergleichen unten an See Abschnitt 5.6.5, „Unterstützung für Zeichenketten-Vergleich“.

`ctype[]` ist ein Array von Bit-Werten, mit einem Element pro Zeichen. (Beachten Sie, dass `to_lower[]`, `to_upper[]` und `sort_order[]` durch den Buchstabenwert indiziert werden, aber `ctype[]` durch den Buchstabenwert + 1. Das ist aus Gründen der Abwärtskompatibilität notwendig, um EOF (Dateiende) handhaben zu können.)

Sie finden folgenden Bitmasken-Definitionen in `m_ctype.h`:

```
#define _U      01      /* Großschreibung */
#define _L      02      /* Kleinschreibung */
#define _N      04      /* Numerisch (Ziffer) */
#define _S      010     /* Leerzeichen */
#define _P      020     /* Punkt */
#define _C      040     /* Steuerungszeichen (Control) */
#define _B      0100    /* leer */
#define _X      0200    /* heXadezimale Ziffer */
```

Der `ctype[]`-Eintrag für jeden Buchstaben sollte die Vereinigungsmenge der betreffenden Bitmasken-Werte sein, die den Buchstaben beschreiben. `'A'` beispielsweise ist Buchstabe in Großschreibung (`_U`) und gleichzeitig eine hexadezimale Ziffer (`_X`), daher sollte `ctype['A'+1]` folgenden Wert erhalten:

```
_U + _X = 01 + 0200 = 0201
```

5.6.5. Unterstützung für Zeichenketten-Vergleich

Wenn die Sortierregeln Ihrer Sprache zu komplex sind, um durch die einfache `sort_order[]`-Tabelle gehandhabt zu werden, müssen Sie die Zeichenketten-Vergleichsfunktionen benutzen.

Zum jetzigen Zeitpunkt ist die beste Dokumentation hierüber die Zeichensätze, die bereits implementiert sind. Sehen Sie sich als Beispiele die Zeichensätze big5, tschechisch, gbk, sjis und tis160 an.

Sie müssen den `strxfrm_multiply_MEINSET=N`-Wert mit einem speziellen Kommentar am Anfang der Datei festlegen. `N` sollte auf das höchste Verhältnis gesetzt werden, auf das die Zeichenketten während `my_strxfrm_MEINSET` anwachsen können (es muss eine positive Ganzzahl sein).

5.6.6. Unterstützung für Multi-Byte-Zeichen

Wenn Sie Unterstützung für einen neuen Zeichensatz hinzufügen wollen, der Multi-Byte-Buchstaben enthält, müssen Sie die Multi-Byte-Zeichenfunktionen benutzen.

Zum jetzigen Zeitpunkt ist die beste Dokumentation hierüber die Zeichensätze, die bereits implementiert sind. Sehen Sie sich als Beispiele die Zeichensätze euc_kr, gb2312, gbk, sjis und ujis an. Diese sind in den `ctype-'charset'.c`-Dateien im `strings`-Verzeichnis implementiert.

Sie müssen den `mbmaxlen_MEINSET=N`-Wert in einem speziellen Kommentar am Anfang der Quelldatei angeben. `N` sollte auf die Größe in Bytes des größten Buchstabens im Zeichensatz gesetzt werden.

5.6.7. Probleme mit Zeichensätzen

Wenn Sie versuchen, einen Zeichensatz zu benutzen, der nicht in Ihre Binärdatei kompiliert ist, können Sie verschiedene Probleme bekommen:

- Ihr Programm hat einen falschen Pfad zum Speicherort der Zeichensätze. (Vorgabe ist `/usr/local/mysql/share/mysql/charsets`). Das kann durch die Benutzung der `--character-sets-dir`-Option für das fragliche Programm behoben werden.
- Der Zeichensatz ist ein Multi-Byte-Zeichensatz, der nicht dynamisch geladen werden kann. Wenn das der Fall ist, müssen Sie das Programm mit Unterstützung für diesen Zeichensatz neu kompilieren.
- Der Zeichensatz ist ein dynamischer Zeichensatz, aber Sie haben keine configure-Datei dafür. In diesem Fall müssen Sie die configure-Datei für den Zeichensatz aus einer neuen MySQL-Distribution installieren.
- Ihre `Index`-Datei enthält nicht den Namen für den Zeichensatz.

  ```
  ERROR 1105: File '/usr/local/share/mysql/charsets/?.conf' not found
  (Errcode: 2)
  ```

  In diesem Fall müssen Sie sich entweder eine neue `Index`-Datei holen oder den Namen jedes fehlenden Zeichensatzes von Hand eintragen.

Bei MyISAM-Tabellen können Sie den Zeichensatznamen und die Anzahl für eine Tabelle mit `myisamchk -dvv tabelle` prüfen.

5.7. Serverseitige Skripte und Dienstprogramme für MySQL

5.7.1. Überblick über serverseitige Programme und Dienstprogramme

Alle MySQL-Clients, die mittels der `mysqlclient`-Bibliothek mit dem Server kommunizieren, benutzen folgenden Umgebungsvariablen:

Name	Beschreibung
`MYSQL_UNIX_PORT`	Der vorgabemäßige Socket; benutzt für Verbindungen zu `localhost`
`MYSQL_TCP_PORT`	Der vorgabemäßige TCP/IP-Port
`MYSQL_PWD`	Das vorgabemäßige Passwort
`MYSQL_DEBUG`	Debug-Trace-Optionen beim Debuggen
`TMPDIR`	Das Verzeichnis, in dem temporäre Tabellen / Dateien erzeugt werden

Die Benutzung von `MYSQL_PWD` ist unsicher. See Abschnitt 5.2.7, „Verbinden mit dem MySQL-Server“.

Der `mysql`-Client benutzt die Datei, die in der `MYSQL_HISTFILE`-Umgebungsvariablen angegeben ist, um die Kommandozeilen-History zu speichern. Der Vorgabewert für die History-Datei ist `$HOME/.mysql_history`, wobei `$HOME` der Wert der `HOME`-Umgebungsvariablen ist. See Anhang F, *Umgebungsvariablen*.

Alle MySQL-Programme nehmen viele unterschiedliche Optionen auf. Jedes MySQL-Programm bietet jedoch eine `--help`-Option, die Sie benutzen können, um eine vollständige Beschreibung der unterschiedlichen Programmoptionen zu erhalten. Probieren Sie zum Beispiel `mysql --help` aus.

Sie können Vorgabeoptionen für alle Standard-Client-Programme mit einer Optionsdatei überschreiben. Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

Die unten stehende Liste beschreibt kurz die MySQL-Programme:

- `myisamchk`

  Dienstprogramm zur Beschreibung, Prüfung, Optimierung und Reparatur von MySQL-Tabellen. Weil `myisamchk` viele Funktionen hat, ist es in einem eigenen Kapitel beschrieben. See Kapitel 5, *MySQL-Datenbankadministration*.

- `make_binary_distribution`

  Macht ein Binär-Release eines kompilierten MySQL. Dieses könnte über FTP an `/pub/mysql/Incoming` oder an `Support.mysql.com` geschickt werden, damit andere MySQL-Benutzer es benutzen können.

- `msql2mysql`

  Ein Shell-Skript, das `mSQL`-Programme zu MySQL konvertiert. Es deckt nicht alle Fälle ab, ist aber hilfreich, um mit dem Konvertieren anzufangen.

- `mysqlaccess`

  Ein Skript, das die Zugriffsberechtigungen für eine Host-, Benutzer- und Datenbank-Kombination prüft.

- `mysqladmin`

  Dienstprogramm für die Durchführung von Verwaltungsoperationen wie Erzeugen und Löschen von Datenbanken, Neuladen der Berechtigungstabellen, Zurückschreiben von Tabellen auf Platte und Neuöffnen von Log-Dateien. `mysqladmin` kann auch benutzt werden, um Versionsnummer sowie Status- und Prozess-Informationen vom Server zu erhalten. See Abschnitt 5.8.3, „mysqladmin, Verwaltung eines MySQL-Servers“.

- `mysqlbug`

  Das MySQL-Bug-Bericht-Skript. Dieses Skript sollte immer benutzt werden, wenn Sie einen Bug-Bericht an die MySQL-Liste ausfüllen.

- `mysqld`

Der SQL-Daemon. Dieser sollte immer laufen.

- `mysqldump`

  Dumpt eine MySQL-Datenbank in eine Datei als SQL-Statements oder als Tabulator-separierte Textdateien. Verbesserte Freeware, ursprünglich von Igor Romanenko. See Abschnitt 5.8.5, „mysqldump, Tabellenstrukturen und -daten dumpen“.

- `mysqlimport`

  Importiert Textdateien in die jeweiligen Tabellen mittels `LOAD DATA INFILE`. See Abschnitt 5.8.7, „mysqlimport, Daten aus Textdateien importieren“.

- `mysqlshow`

  Zeigt Informationen über Datenbanken, Tabellen, Spalten und Indexe an.

- `mysql_install_db`

  Erzeugt die MySQL-Berechtigungstabellen mit vorgabemäßigen Berechtigungen. Dieses Skript wird gewöhnlich nur einmal ausgeführt, wenn Sie MySQL das erste Mal auf einem System installieren.

- `replace`

  Ein Dienstprogramm, das von `msql2mysql` benutzt wird, aber auch darüber hinaus benutzt werden kann. `replace` ändert Zeichenketten in Dateien oder auf der Standardeingabe. Benutzt eine finite Status-Maschine, um zuerst Übereinstimmung mit längeren Zeichenketten zu finden. Kann benutzt werden, um Zeichenketten umzudrehen. Der folgende Befehl zum Beispiel dreht `a` und `b` in den angegebenen Dateien um:

  ```
  shell> replace a b b a --Datei1 Datei2 ...
  ```

5.7.2. safe_mysqld, der Wrapper um mysqld

`safe_mysqld` ist die empfohlene Art, einen `mysqld`-Daemon unter Unix zu starten. `safe_mysqld` fügt einige Sicherheits-Features hinzu wie das Neustarten des Servers, wenn ein Fehler auftritt, und das Mitschreiben von Laufzeitinformationen in eine Log-Datei.

Wenn Sie nicht `--mysqld=#` oder `--mysqld-version=#` benutzen, benutzt `safe_mysqld` eine ausführbare Datei namens `mysqld-max`, wenn es diese gibt. Wenn nicht, startet `safe_mysqld` `mysqld`. Das macht es sehr einfach, `mysqld-max` anstelle von `mysqld` versuchsweise zu benutzen. Kopieren Sie einfach `mysqld-max` dorthin, wo `mysqld` liegt, und es wird benutzt werden.

Normalerweise sollte man das `safe_mysqld`-Skript nie editieren, sondern statt dessen die Optionen für `safe_mysqld` in den `[safe_mysqld]`-Abschnitt der `my.cnf`-Datei einfügen. `safe_mysqld` liest alle Optionen des `[mysqld]`-, `[server]`- und `[safe_mysqld]`-Abschnitts aus den Optionsdateien. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

Beachten Sie, dass alle Optionen auf der Kommandozeile für `safe_mysqld` an `mysqld` durchgereicht werden. Wenn Sie in `safe_mysqld` irgend welche Optionen benutzen wollen, die `mysqld` nicht unterstützt, müssen Sie diese in der Optionsdatei angeben.

Die meisten Optionen für `safe_mysqld` sind dieselben wie die Optionen für `mysqld`. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

`safe_mysqld` unterstützt folgende Optionen:

- `--basedir=pfad`, `--core-file-size=#`

  Größe der Core-Datei, die `mysqld` in der Lage sein sollte zu erzeugen. Wird an `ulimit -c` durchgereicht.

- `--datadir=pfad`, `--defaults-extra-file=pfad`, `--defaults-file=pfad`, `--err-log=pfad`, `--ledir=pfad`

  Pfad zu `mysqld`

- `--log=pfad`, `--mysqld=mysqld-version`

  Name der `mysqld`-Version im `ledir`-Verzeichnis, die Sie starten wollen.

- `--mysqld-version=version`

  Ähnlich wie `--mysqld=`, aber hier für nur das Suffix für `mysqld` angegeben. Wenn Sie zum Beispiel `--mysqld-version=max` benutzen, startet `safe_mysqld` die `ledir/mysqld-max`-Version. Wenn das Argument für `--mysqld-version` leer ist, wird `ledir/mysqld` benutzt.

- `--no-defaults`, `--open-files-limit=#`

  Anzahl der Dateien, die `mysqld` in der Lage sein sollte zu öffnen. Wird an `ulimit -n` durchgereicht. Beachten Sie, dass Sie `safe_mysqld` als Root starten müssen, damit dies korrekt funktioniert!

- `--pid-file=pfad`, `--port=#` , `--socket=pfad`, `--timezone=#`

  Setzt die Zeitzone (die `TZ`)-Variable auf den Wert dieses Parameters.

- `--user=#`

Das `safe_mysqld`-Skript ist so geschrieben, dass es normalerweise einen Server starten kann, der aus einer Quell- oder einer Binärversion von MySQL installiert wurde, selbst wenn diese den Server an etwas anderen Stellen installieren. `safe_mysqld` erwartet, dass eine der folgenden Bedingungen zutrifft:

- Server und Datenbanken liegen relativ zum Verzeichnis, aus dem `safe_mysqld` aufgerufen wird. `safe_mysqld` sucht unterhalb seines Arbeitsverzeichnisses nach `bin`- und `data`-Verzeichnissen (bei Binärdistributionen) oder nach `libexec`- und `var`-Verzeichnissen (bei Quelldistributionen). Diese Bedingung sollte zutreffen, wenn Sie `safe_mysqld` aus Ihrem MySQL-Installationsverzeichnis ausführen (zum Beispiel `/usr/local/mysql` bei einer Binärdistribution).
- Wenn Server und Datenbanken nicht relativ zum Arbeitsverzeichnis liegen, versucht `safe_mysqld`, sie anhand absoluter Pfadnamen zu finden. Typische Speicherort sind `/usr/local/libexec` und `/usr/local/var`. Die tatsächlichen Speicherorte werden festgelegt, wenn die Distribution gebaut wird, woher `safe_mysqld` kommt. Sie sollten korrekt sein, wenn MySQL an einem Standardort installiert wurde.

Weil `safe_mysqld` versucht, Server und Datenbanken relativ zum eigenen Arbeitsverzeichnis zu finden, können Sie eine Binärdistribution von MySQL irgendwo hin installieren, so lange Sie `safe_mysqld` aus dem MySQL-Installationsverzeichnis starten:

```
shell> cd mysql_installations_verzeichnis
shell> bin/safe_mysqld &
```

Wenn `safe_mysqld` fehlschlägt, selbst wenn es aus dem MySQL-Installationsverzeichnis aufgerufen wurde, können Sie es so ändern, dass es den Pfad zu `mysqld` und die Pfadnamen-Optionen benutzt, die auf Ihrem System korrekt sind. Beachten Sie, dass bei zukünftigen Aktualisierungen von MySQL Ihre veränderte Version von `safe_mysqld` überschrieben wird. Daher sollten Sie eine Kopie Ihrer editierten Version machen, damit Sie diese neu installieren können.

5.7.3. mysqld_multi, Programm zur Verwaltung mehrerer MySQL-Server

`mysqld_multi` ist für die Verwaltung mehrerer `mysqld`-Prozesse gedacht, die auf unterschiedlichen UNIX-Sockets und TCP/IP-Ports laufen.

Das Programm sucht nach Gruppe(n), die [mysqld#] benannt sind, in my.cnf (oder der angegebenen --config-file=...), wobei # jede positive Zahl ab 1 sein kann. Diese Gruppen sollten dieselben sein wie die übliche `[mysqld]`-Gruppe (zum Beispiel Optionen für mysqld, siehe ausführliche Informationen im Handbuch über diese Gruppe), aber mit denjenigen Port-, Socket- usw. Optionen, die für jeden separaten `mysqld`-Prozess gewünscht sind. Die Zahl im Gruppennamen hat eine andere Funktion: Sie kann benutzt werden, um bestimmte `mysqld`-Server zu starten, anzuhalten, oder Berichte über sie mit diesem Programm auszugeben. Unten stehen weitere Informationen zur Benutzung und zu den Optionen.

```
Benutzung: mysqld_multi [OPTIONS] {start|stop|report} [GNR,GNR,GNR...]
oder       mysqld_multi [OPTIONS] {start|stop|report} [GNR-GNR,GNR,GNR-GNR,...]
```

Die GNR oben bedeutet die Gruppennummer. Sie können jede GNR starten, anhalten oder Berichtsinformationen über sie ausgeben, oder über mehrere von ihnen zugleich. (Siehe --example) Die GNRs in der Liste können mit Komma getrennt oder mit Bindestrich kombiniert werden, wobei letzteres heißt, dass alle GNRs zwischen GNR1-GNR2 betroffen sind. Ohne GNR-Argument werden alle gefundenen Gruppen entweder gestartet, angehalten, oder es werden Berichtsinformationen über sie ausgegeben. Beachten Sie, dass Sie in der GNR-Liste keinen Leerraum haben dürfen. Alles nach Leerraum wird ignoriert.

`mysqld_multi` unterstützt folgende Optionen:

- `--config-file=...`

  Alternative config-Datei. HINWEIS: Das betrifft nicht die eigenen Optionen des Programms (Gruppe `[mysqld_multi]`), sondern nur die Gruppen [mysqld#]. Ohne diese Option wird alles aus der normalen my.cnfDatei heraus gesucht.

- `--example`

  Zeigt ein Beispiel einer config-Datei.

- `--help`

  Hilfetext ausgeben und beenden.

- `--log=...`

  Log-Datei. Name und voller Pfad zur Log-Datei. HINWEIS: Wenn es die Datei gibt, wird alles angehängt.

- `--mysqladmin=...`

  `mysqladmin`-Binärdatei, die zum Herunterfahren des Servers benutzt wird.

- `--mysqld=...`

  `mysqld`-Binärdatei, die benutzt wird. Beachten Sie, dass Sie auch `safe_mysqld` diese Option angeben können. Die Optionen werden an `mysqld` durchgereicht. Stellen Sie jedoch sicher, dass Sie `mysqld` in Ihrer Umgebungsvariablen `PATH` haben oder bearbeiten Sie `safe_mysqld`.

- `--no-log`

  An stdout ausgeben statt in die Log-Datei. Vorgabemäßig ist die Log-Datei angeschaltet.

- `--password=...`

  Passwort für Benutzer von `mysqladmin`.

- `--tcp-ip`

  Zu MySQL-Server(n) über den TCP/IP-Port statt über den UNIX-Socket verbinden. Das betrifft das Anhalten und Berichten. Wenn eine Socket-Datei fehlt, kann der Server trotzdem laufen, aber man kann nur über den TCP/IP-Port auf ihn zugreifen. Vorgabemäßig wird die Verbindung über den UNIX-Socket hergestellt.

- `--user=...`

  MySQL-Benutzer von `mysqladmin`.

- `--version`

  Versionsnummer ausgeben und beenden.

Einige Anmerkungen zu `mysqld_multi`:

- Stellen Sie sicher, dass der MySQL-Benutzer, der die `mysqld`-Dienste anhält (indem er zum Beispiel `mysqladmin` benutzt), dasselbe Passwort und denselben Benutzernamen für alle Daten-Verzeichnisse benutzt, auf die zugegriffen wird (zur 'mysql'-Datenbank). Stellen Sie ausserdem sicher, dass der Benutzer die 'Shutdown_priv'-Berechtigung hat! Wenn Sie viele Daten-Verzeichnisse und viele verschiedene 'mysql'-Datenbanken mit unterschiedlichen Passwörtern für den MySQL-'root'-Benutzer haben, sollten Sie einen allgemeinen 'multi_admin'-Benutzer anlegen, der dasselbe Passwort benutzt (siehe unten). Hier ein Beispiel dafür:

  ```
  shell> mysql -u root -S /tmp/mysql.sock -proot_password -e
  "GRANT SHUTDOWN ON *.* TO multi_admin@localhost IDENTIFIED BY 'multipass'"
  ```
  See Abschnitt 5.2.5, „Wie das Berechtigungssystem funktioniert“.

  Das oben Angegebene müssen Sie für jeden laufenden `mysqld` im Daten-Verzeichnis tun, das Sie haben (ändern Sie einfach den Socket, -S=...).

- `pid-file` ist sehr wichtig, wenn Sie `safe_mysqld` benutzen, um `mysqld` zu starten (zum Beispiel --mysqld=safe_mysqld). Jeder `mysqld` sollte seine eigene `pid-file` haben. Der Vorteil der Benutzung von `safe_mysqld` anstelle von `mysqld` direkt ist hierbei, dass `safe_mysqld` jeden `mysqld`-Prozess 'bewacht' und neu startet, falls ein `mysqld`-Prozess wegen eines Signals kill -9 fehlschlägt oder ähnliches (wenn beispielsweise Speicherzugriffsfehler auftreten, was bei MySQL natürlich nie passiert ;-). Beachten Sie bitte, dass es für das `safe_mysqld`-Skript eventuell erforderlich ist, es von einer bestimmten Stelle aus zu starten. Das heißt, dass Sie eventuell in ein bestimmtes Verzeichnis wechseln müssen,

bevor Sie `mysqld_multi` starten. Wenn Sie beim Starten Probleme haben, sehen Sie bitte im `safe_mysqld`-Skript nach. Überprüfen Sie insbesondere folgende Zeilen:

```
---------------------------------------------------------------------------
MY_PWD=`pwd` Check if we are starting this relative (for the binary
release) if test -d /data/mysql -a -f ./share/mysql/englisch/errmsg.sys
-a -x ./bin/mysqld
---------------------------------------------------------------------------
```

See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.

Der obige Test soll erfolgreich verlaufen, ansonsten können Sie Probleme bekommen.

- Vermeiden Sie Gefahren, die auftauchen, wenn Sie mehrere `mysqlds` im selben Daten-Verzeichnis starten. Benutzen Sie unterschiedlichen Daten-Verzeichnisse, es sei denn, Sie wissen **GENAU**, was Sie tun!
- Die Socket-Datei und der TCP/IP-Port müssen für jeden `mysqld` verschieden sein.
- Die erste und die fünfte `mysqld`-Gruppe wurden beim Beispiel absichtlich ausgelassen. Sie haben eventuell 'Lücken' in der config-Datei. Das gibt Ihnen mehr Flexibilität. Die Reihenfolge, in der die `mysqlds` gestartet oder angehalten werden, hängt von der Reihenfolge ab, in der sie in der config-Datei erscheinen.
- Wenn Sie auf eine bestimmte Gruppe verweisen wollen, wenn Sie GNR bei diesem Programm benutzen, nehmen Sie einfach die Nummer am Ende des Gruppennamens ([mysqld# <==).
- Eventuell sollten Sie die Option '--user' für `mysqld` benutzen, aber um das zu tun, müssen Sie root sein, wenn Sie das `mysqld_multi`-Skript starten. Wenn Sie die Option in der config-Datei haben, macht das nichts; Sie erhalten nur eine Warnmeldung, wenn Sie nicht der Superuser sind und die `mysqlds` unter **IHREM** UNIX-Account gestartet werden. **WICHTIG**: Stellen Sie sicher, dass die `pid-file` und das Daten-Verzeichnis für **DENJENIGEN** UNIX-Benutzer lesbar und schreibbar sind (und ausführbar im letzteren Fall), als der der spezifische `mysqld`-Prozess gestartet wird. Benutzen Sie hier **NICHT** den UNIX-root-Account, es sei denn, Sie wissen **GENAU**, was Sie tun!
- **SEHR WICHTIG**: Stellen Sie sicher, dass Sie die Bedeutung der Optionen verstehen, die an die `mysqlds` durchgereicht werden und **WARUM** Sie mehrere verschiedene `mysqld`-Prozesse haben wollen. Mehrere `mysqlds` in einem Daten-Verzeichnis starten **ergibt keine zusätzliche Performance** bei einem threaded System!

See Abschnitt 5.1.4, „Viele MySQL-Server auf derselben Maschine laufen lassen“.

Hier ist ein Beispiel einer config-Datei für `mysqld_multi`.

```
# Diese Datei sollte wahrscheinlich in Ihrem Heimatverzeichnis liegen (~/.my.cnf) oder in /etc/my.cnf
# Version 2.1 von Jani Tolonen

[mysqld_multi]
mysqld     = /usr/local/bin/safe_mysqld
mysqladmin = /usr/local/bin/mysqladmin
user       = multi_admin
password   = multipass

[mysqld2]
socket     = /tmp/mysql.sock2
port       = 3307
pid-file   = /usr/local/mysql/var2/hostname.pid2
datadir    = /usr/local/mysql/var2
language   = /usr/local/share/mysql/english
user       = john

[mysqld3]
socket     = /tmp/mysql.sock3
port       = 3308
pid-file   = /usr/local/mysql/var3/hostname.pid3
datadir    = /usr/local/mysql/var3
language   = /usr/local/share/mysql/swedish
user       = monty

[mysqld4]
socket     = /tmp/mysql.sock4
port       = 3309
pid-file   = /usr/local/mysql/var4/hostname.pid4
datadir    = /usr/local/mysql/var4
language   = /usr/local/share/mysql/estonian
user       = tonu

[mysqld6]
socket     = /tmp/mysql.sock6
port       = 3311
pid-file   = /usr/local/mysql/var6/hostname.pid6
datadir    = /usr/local/mysql/var6
language   = /usr/local/share/mysql/japanese
user       = jani
```

See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

5.7.4. myisampack, MySQL-Programm zum Erzeugen komprimierter Nur-Lese-Tabellen

`myisampack` wird benutzt, um MyISAM-Tabellen zu komprimieren. `pack_isam` wird benutzt, um ISAM-Tabellen zu komprimieren. Weil ISAM-Tabellen veraltet sind, wird hier nur `myisampack` erörtert, aber alles, was auf `myisampack` zutrifft, gilt auch für `pack_isam`.

`myisampack` funktioniert, indem jede Spalte in der Tabelle separat komprimiert wird. Die Informationen, die benötigt werden, um Spalten zu dekomprimieren, werden in den Arbeitsspeicher gelesen, wenn die Tabelle geöffnet wird. Das ergibt viel bessere Performance beim Zugriff auf einzelne Datensätze, denn man muss nur exakt einen Datensatz dekomprimieren, nicht einen viel größeren Block, wie das zum Beispiel bei der Benutzung von Stacker auf MS-DOS nötig ist. Üblicherweise komprimiert `myisampack` die Daten-Datei auf 40%-70%.

MySQL benutzt Speicher-Mapping (`mmap()`) auf komprimierte Tabellen und geht zu normalen Lesen / Schreiben von Dateien zurück, wenn `mmap()` nicht funktioniert.

Für `myisampack` gibt es momentan zwei Einschränkungen:

- Nach dem Komprimieren ist die Tabelle nur-lesbar.
- `myisampack` kann auch `BLOB`- oder `TEXT`-Spalten komprimieren. Das ältere `pack_isam` konnte das nicht.

Die Behebung dieser Einschränkungen steht mit niedrigen Priorität auf unserer TODO-Liste.

`myisampack` wird wie folgt aufgerufen:

```
shell> myisampack [options] Dateiname ...
```

Jeder Dateiname sollte der Name einer Index-(`.MYI`)-Datei sein. Wenn Sie nicht im Datenbank-Verzeichnis sind, müssen Sie den Pfadnamen zur Datei angeben. Die `.MYI` Erweiterung kann weggelassen werden.

`myisampack` unterstützt folgende Optionen:

- `-b, --backup`

  Stellt eine Datensicherung der Tabelle als `tabelle.OLD` her.

- `-#, --debug=debug_options`

  Debug-Log ausgeben. Die `debug_options`-Zeichenkette ist häufig `'d:t:o,filename'`.

- `-f, --force`

  Erzwingt die Komprimierung der Tabelle, selbst wenn sie dadurch größer wird oder die temporäre Datei existiert. `myisampack` erzeugt eine temporäre Datei namens `tabelle.TMD`, während es die Tabelle komprimiert. Wenn Sie `myisampack` killen, kann es sein, dass die `.TMD`-Datei nicht gelöscht wird. Normalerweise wird `myisampack` mit einer Fehlermeldung beendet, wenn es eine existierende `tabelle.TMD`-Datei findet. Mit `--force` packt `myisampack` die Tabelle trotzdem.

- `-?, --help`

  Hilfetext ausgeben und beenden.

- `-j große_tabelle, --join=große_tabelle`

  Verbindet alle Tabellen, die auf der Kommandozeile angegeben wurden, in eine einzige große Tabelle `große_tabelle`. Alle Tabellen, die kombiniert werden sollen, MÜSSEN identisch sein (dieselben Spaltennamen und -typen, dieselben Indexe usw.).

- `-p #, --packlength=#`

  Legt die Speichergröße der Datensatzlänge in Bytes fest. Der Wert sollte 1, 2 oder 3 sein. (`myisampack` speichert alle Zeilen mit Längenzeigern von 1, 2, oder 3 Bytes. In den meisten Fällen kann `myisampack` den richtigen Längenwert festlegen, bevor es anfängt, die Datei zu komprimieren. Während des Komprimierungsprozesses stellt es aber eventuell fest, dass es eine kürzere Länge hätte nehmen können. In diesem Fall gibt `myisampack` einen Hinweis aus, dass Sie beim nächsten Mal, wenn Sie dieselbe Datei packen, eine kürzere Datensatzlänge nehmen sollten.)

- `-s, --silent`

Schweigsamer Modus. Ausgaben erfolgen nur, wenn Fehler auftreten.

- `-t, --test`

  Tabelle nicht tatsächlich komprimieren, sondern nur testweise packen.

- `-T dir_name, --tmp_dir=dir_name`

  Das genannte Verzeichnis als Speicherort der temporären Tabelle benutzen.

- `-v, --verbose`

  Geschwätziger Modus. Informationen über den Fortschritt und das Komprimierungsergebnis ausgeben.

- `-V, --version`

  Versionsinformationen ausgeben und beenden.

- `-w, --wait`

  Warten und noch einmal versuchen, wenn die Tabelle in Benutzung ist. Wenn der `mysqld`-Server mit der `--skip-locking`-Option aufgerufen wurde, ist es keine gute Idee, `myisampack` aufzurufen, wenn die Tabelle während des Komprimierungsprozesses möglicherweise aktualisiert wird.

Die unten stehende Befehlssequenz zeigt eine typische Tabellen-Komprimierungssitzung:

```
shell> ls -l station.*
-rw-rw-r--   1 monty    my         994128 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          53248 Apr 17 19:00 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:   station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-02-02  3:06:43
Data records:             1192  Deleted blocks:              0
Datafile: Parts:          1192  Deleted data:                0
Datafile pointer (bytes):     2  Keyfile pointer (bytes):     2
Max datafile length:   54657023  Max keyfile length:   33554431
Recordlength:               834
Record format: Fixed length

table description:
Key Start Len Index   Type                     Root  Blocksize    Rec/key
1   2     4   unique  unsigned long             1024       1024          1
2   32    30  multip. text                     10240       1024          1

Field Start Length Type
1     1     1
2     2     4
3     6     4
4     10    1
5     11    20
6     31    1
7     32    30
8     62    35
9     97    35
10    132   35
11    167   4
12    171   16
13    187   35
14    222   4
15    226   16
16    242   20
17    262   20
18    282   20
19    302   30
20    332   4
21    336   4
22    340   1
23    341   8
24    349   8
25    357   8
26    365   2
27    367   2
28    369   4
29    373   4
30    377   1
31    378   2
32    380   8
33    388   4
34    392   4
35    396   4
```

```
36    400   4
37    404   1
38    405   4
39    409   4
40    413   4
41    417   4
42    421   4
43    425   4
44    429   20
45    449   30
46    479   1
47    480   1
48    481   79
49    560   79
50    639   79
51    718   79
52    797   8
53    805   1
54    806   1
55    807   20
56    827   4
57    831   4

shell> myisampack station.MYI
Compressing station.MYI: (1192 records)
- Calculating statistics

normal:     20  empty-space:      16  empty-zero:        12  empty-fill:  11
pre-space:   0  end-space:        12  table-lookups:      5  zero:         7
Original trees:  57  After join: 17
- Compressing file
87.14%

shell> ls -l station.*
-rw-rw-r--   1 monty    my         127874 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          55296 Apr 17 19:04 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-04-17 19:04:26
Data records:             1192  Deleted blocks:              0
Datafile: Parts:          1192  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     1
Max datafile length:   16777215  Max keyfile length:     131071
Recordlength:              834
Record format: Compressed

table description:
Key Start Len Index   Type                     Root  Blocksize    Rec/key
1   2     4   unique  unsigned long            10240       1024          1
2   32    30  multip. text                     54272       1024          1

Field Start Length Type                         Huff tree  Bits
1     1     1      constant                           1     0
2     2     4      zerofill(1)                        2     9
3     6     4      no zeros, zerofill(1)              2     9
4     10    1                                         3     9
5     11    20     table-lookup                       4     0
6     31    1                                         3     9
7     32    30     no endspace, not_always            5     9
8     62    35     no endspace, not_always, no empty  6     9
9     97    35     no empty                           7     9
10    132   35     no endspace, not_always, no empty  6     9
11    167   4      zerofill(1)                        2     9
12    171   16     no endspace, not_always, no empty  5     9
13    187   35     no endspace, not_always, no empty  6     9
14    222   4      zerofill(1)                        2     9
15    226   16     no endspace, not_always, no empty  5     9
16    242   20     no endspace, not_always            8     9
17    262   20     no endspace, no empty              8     9
18    282   20     no endspace, no empty              5     9
19    302   30     no endspace, no empty              6     9
20    332   4      always zero                        2     9
21    336   4      always zero                        2     9
22    340   1                                         3     9
23    341   8      table-lookup                       9     0
24    349   8      table-lookup                      10     0
25    357   8      always zero                        2     9
26    365   2                                         2     9
27    367   2      no zeros, zerofill(1)              2     9
28    369   4      no zeros, zerofill(1)              2     9
29    373   4      table-lookup                      11     0
30    377   1                                         3     9
31    378   2      no zeros, zerofill(1)              2     9
32    380   8      no zeros                           2     9
33    388   4      always zero                        2     9
34    392   4      table-lookup                      12     0
35    396   4      no zeros, zerofill(1)             13     9
36    400   4      no zeros, zerofill(1)              2     9
37    404   1                                         2     9
38    405   4      no zeros                           2     9
39    409   4      always zero                        2     9
40    413   4      no zeros                           2     9
41    417   4      always zero                        2     9
```

```
42    421   4      no zeros                              2     9
43    425   4      always zero                           2     9
44    429   20     no empty                              3     9
45    449   30     no empty                              3     9
46    479   1                                           14     4
47    480   1                                           14     4
48    481   79     no endspace, no empty                 15     9
49    560   79     no empty                              2     9
50    639   79     no empty                              2     9
51    718   79     no endspace                           16     9
52    797   8      no empty                              2     9
53    805   1                                           17     1
54    806   1                                           3     9
55    807   20     no empty                              3     9
56    827   4      no zeros, zerofill(2)                 2     9
57    831   4      no zeros, zerofill(1)                 2     9
```

Die Informationen, die `myisampack` ausgibt, sind unten beschrieben:

- `normal`

  Die Anzahl von Spalten, für die keine spezielle Komprimierung benutzt wird.

- `empty-space`

  Die Anzahl von Spalten, die Werte enthalten, die ausschließlich aus Leerzeichen bestehen. Diese Werte nehmen 1 Bit in Anspruch.

- `empty-zero`

  Die Anzahl von Spalten, die Werte enthalten, die nur aus binären Nullen bestehen. Diese Werte nehmen 1 Bit in Anspruch.

- `empty-fill`

  Die Anzahl von Ganzzahl-Spalten, die nicht den gesamten Bereich Ihres Typs einnehmen. Diese werden auf einen kleineren Typ geändert (eine `INTEGER`-Spalte kann zum Beispiel auf `MEDIUMINT` geändert werden).

- `pre-space`

  Die Anzahl von Dezimal-Spalten, die mit führenden Leerzeichen gespeichert sind. In diesem Fall enthält jeder Wert einen Zähler für die Anzahl führender Leerzeichen.

- `end-space`

  Die Anzahl von Spalten, die viele Leerzeichen am Ende enthalten. In diesem Fall enthält jeder Wert einen Zähler für die Anzahl von Leerzeichen am Ende.

- `table-lookup`

  Die Spalte hat nur eine kleine Anzahl verschiedener Werte, die in `ENUM` umgewandelt werden, bevor die Huffman-Kompression durchgeführt wird.

- `zero`

  Die Anzahl von Spalten, bei denen alle Werte 0 sind.

- `Original trees`

  Die anfängliche Anzahl von Huffman-Bäumen.

- `After join`

  Die Anzahl von unterschiedlichen Huffman-Bäumen, die übrig sind, nachdem Bäume zusammengefasst wurden, um etwas Header-Platz zu sparen.

Nachdem eine Tabelle komprimiert wurde, gibt `myisamchk -dvv` zusätzliche Informationen über jedes Feld aus:

- `Type`

  Der Feldtyp kann folgende Deskriptoren enthalten:

  - `constant`

Alle Zeilen haben denselben Wert.

- `no endspace`

  Kein Leerzeichen am Ende speichern.

- `no endspace, not_always`

  Kein Leerzeichen am Ende speichern und bei allen Werten keine Komprimierung für Leerzeichen am Ende durchführen.

- `no endspace, no empty`

  Kein Leerzeichen am Ende speichern. Keine leeren Werte speichern.

- `table-lookup`

  Die Spalte wurde zu `ENUM` umgewandelt.

- `zerofill(n)`

  Die wichtigsten `n` Bytes im Wert sind immer 0 und wurden nicht gespeichert.

- `no zeros`

  Keine Nullen speichern.

- `always zero`

  0-Werte sind in 1 Bit gespeichert.

- `Huff tree`

  Der Huffman-Baum, der zu dem Feld gehört.

- `Bits`

  Die Anzahl von Bits, die im Huffman-Baum benutzt werden.

Nachdem Sie `pack_isam`/`myisampack` laufen gelassen haben, müssen Sie `isamchk` / `myisamchk` laufen lassen, um den Index neu zu erzeugen. Zugleich können Sie die Index-Blöcke sortieren und die Statistiken erzeugen, die benötigt werden, damit der MySQL-Optimierer effizienter läuft:

```
myisamchk -rq --analyze --sort-index tabelle.MYI
isamchk   -rq --analyze --sort-index tabelle.ISM
```

Nachdem Sie die komprimierte Tabelle ins MySQL-Datenbank-Verzeichnis gespielt haben, müssen Sie `mysqladmin flush-tables` ausführen, um `mysqld` anzuweisen, die neue Tabelle zu benutzen.

Wenn Sie eine gepackte Tabelle entpacken wollen, können Sie das mit der `--unpack`-Option für `isamchk` oder `myisamchk` tun.

5.7.5. mysqld-max, ein erweiterter mysqld-Server

`mysqld-max` ist der MySQL-Server (`mysqld`), der mit folgenden configure-Optionen konfiguriert wurde:

Option	Kommentar
--with-server-suffix=-max	Zur `mysqld`-Versionszeichenkette ein Suffix hinzufügen.
--with-bdb	Unterstützung für Berkeley-DB-(BDB)-Tabellen
--with-innodb	Unterstützung für InnoDB-Tabellen.
CFLAGS=-DUSE_SYMDIR	Symbolische-Links-Unterstützung für Windows.

Sie finden die MySQL-max-Binärdateien unter http://www.mysql.com/downloads/mysql-max-3.23.html.

Die Windows-MySQL-3.23-Binärdistribution beinhaltet sowohl die Standard-`mysqld.exe`-Binärdatei als auch die `mysqld-max.exe`-Binärdatei. http://www.mysql.com/downloads/mysql-3.23.html. See Abschnitt 3.1.2, „Installation von MySQL unter

Windows“.

Beachten Sie, dass, weil InnoDB und Berkeley-DB nicht für alle Plattformen verfügbar sind, einige der `Max`-Binärdateien eventuell noch Unterstützung für diese beiden Typen haben. Sie können überprüfen, welche Tabellentypen unterstützt werden, indem Sie die folgende Anfrage ausführen:

```
mysql> show variables like "have_%";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_bdb      | YES   |
| have_innodb   | NO    |
| have_isam     | YES   |
| have_raid     | NO    |
| have_openssl  | NO    |
+---------------+-------+
```

Die Bedeutung dieser Werte ist:

Wert	**Bedeutung**.
YES	Die Option ist aktiviert und benutzbar.
NO	MySQL ist nicht mit Unterstützung für diese Option kompiliert.
DISABLED	Die xxxx-Option ist deaktiviert, weil `mysqld` mit `--skip-xxxx` gestartet wurde oder weil `mysqld` nicht mit allen notwendigen Optionen gestartet wurde, um die Option zu aktivieren. In diesem Fall sollte die `hostname.err`-Datei den Grund dafür enthalten, warum die Option deaktiviert wurde.

HINWEIS: Um InnoDB-Tabellen erzeugen zu können, **MÜSSEN** Sie Ihre Startoptionen editieren und zumindest die `innodb_data_file_path`-Option eingeben. See Abschnitt 8.5.2, „Mit InnoDB anfangen - Optionen“.

Um bessere Performance für BDB-Tabellen zu erzielen, sollten Sie auch für diese einige Konfigurationsoptionen angeben. See Abschnitt 8.6.3, „BDB-Startoptionen“.

`safe_mysqld` versucht automatisch, eine `mysqld`-Binärdatei mit dem `-max`-Präfix zu starten. Das macht es sehr einfach, eine andere `mysqld`-Binärdatei in einer bestehenden Installation auszutesten. Lassen Sie einfach `configure` mit den Optionen, die Sie wollen, laufen, und installieren Sie dann die neue `mysqld`-Binärdatei als `mysqld-max` im selben Verzeichnis, wo Ihre alte `mysqld`-Binärdatei liegt. See Abschnitt 5.7.2, „safe_mysqld, der Wrapper um mysqld“.

Der `mysqld-max`-RPM benutzt das oben erwähnte `safe_mysqld`-Feature. Er installiert nur die ausführbare Datei `mysqld-max` und `safe_mysqld` benutzt diese automatisch, wenn `safe_mysqld` neu gestartet wird.

Folgende Tabelle zeigt, welche Tabellentypen unsere Standard-**MySQL-Max**-Binärdateien beinhalten:

System	**BDB**	**InnoDB**
AIX 4.3	NEIN	JA
HP-UX 11.0	NEIN	JA
Linux-Alpha	NEIN	JA
Linux-Intel	JA	JA
Linux-Ia64	NEIN	JA
Solaris-intel	NEIN	JA
Solaris-sparc	JA	JA
Caldera (SCO) OSR5	JA	JA
UnixWare	JA	JA
Windows/NT	JA	JA

5.8. Clientseitige Skripte und Hilfsprogramme von MySQL

5.8.1. Überblick über die clientseitigen Skripte und Dienstprogramme

Alle MySQL-Clients, die mittels der `mysqlclient`-Bibliothek mit dem Server kommunizieren, benutzen folgende Umgebungsvariablen:

Name	**Beschreibung**

MYSQL_UNIX_PORT	Der vorgabemäßige Socket, benutzt für Verbindungen zu localhost
MYSQL_TCP_PORT	Der vorgabemäßige TCP/IP-Port
MYSQL_PWD	Das vorgabemäßige Passwort
MYSQL_DEBUG	Debug-Trace-Optionen beim Debuggen
TMPDIR	Das Verzeichnis, in dem temporäre Tabellen / Dateien erzeugt werden

Die Benutzung von MYSQL_PWD ist unsicher. See Abschnitt 5.2.7, „Verbinden mit dem MySQL-Server“.

Der mysql-Client benutzt die Datei in der MYSQL_HISTFILE- Umgebungsvariablen genannte Datei, um die Kommandozeilen-History zu speichern. Der Vorgabewert für die History-Datei ist $HOME/.mysql_history, wobei $HOME der Wert der HOME-Umgebungsvariablen ist. See Anhang F, *Umgebungsvariablen*.

Alle MySQL-Programme haben viele verschiedene Optionen. Jedes MySQL-Programm stellt jedoch ein --help-Option zur Verfügung, die Sie benutzen können, um eine vollständige Beschreibung der verschiedenen Optionen des Programms zu erhalten. Probieren Sie zum Beispiel mysql --help aus.

Sie können die vorgabemäßigen Optionen für alle Standard-Client-Programme mit einer Optionsdatei überschreiben. Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

Die unten stehende Liste beschreibt kurz die MySQL-Programme:

- myisamchk

  Dienstprogramm zur Beschreibung, Prüfung, Optimierung und Reparatur von MySQL-Tabellen. Weil myisamchk viele Funktionen hat, ist es in einem eigenen Kapitel beschrieben. See Kapitel 5, *MySQL-Datenbankadministration*.

- make_binary_distribution

  Macht ein Binär-Release eines kompilierten MySQL. Dieses könnte über FTP an /pub/mysql/Incoming oder an Support.mysql.com geschickt werden, damit andere MySQL-Benutzer es benutzen können.

- msql2mysql

  Ein Shell-Skript, das mSQL-Programme zu MySQL konvertiert. Es deckt nicht alle Fälle ab, ist aber hilfreich, um mit dem Konvertieren anzufangen.

- mysqlaccess

  Ein Skript, das die Zugriffsberechtigungen für eine Host-, Benutzer- und Datenbank-Kombination prüft.

- mysqladmin

  Dienstprogramm für die Durchführung von Verwaltungsoperationen wie Erzeugen und Löschen von Datenbanken, Neulanden der Berechtigungstabellen, Zurückschreiben von Tabellen auf Platte und Neuöffnen von Log-Dateien. mysqladmin kann auch benutzt werden, um Versionsnummer sowie Status- und Prozess-Informationen vom Server zu erhalten. See Abschnitt 5.8.3, „mysqladmin, Verwaltung eines MySQL-Servers“.

- mysqlbug

  Das MySQL-Bug-Bericht-Skript. Dieses Skript sollte immer benutzt werden, wenn Sie einen Bug-Bericht an die MySQL-Liste ausfüllen.

- mysqld

  Der SQL-Daemon. Dieser sollte immer laufen.

- mysqldump

  Dumpt eine MySQL-Datenbank in eine Datei als SQL-Statements oder als Tabulator-separierte Textdateien. Verbesserte Freeware, ursprünglich von Igor Romanenko. See Abschnitt 5.8.5, „mysqldump, Tabellenstrukturen und -daten dumpen“.

- mysqlimport

  Importiert Textdateien in die jeweiligen Tabellen mittels LOAD DATA INFILE. See Abschnitt 5.8.7, „mysqlimport, Daten aus Textdateien importieren“.

- mysqlshow

Zeigt Informationen über Datenbanken, Tabellen, Spalten und Indexe an.

- `mysql_install_db`

  Erzeugt die MySQL-Berechtigungstabellen mit vorgabemäßigen Berechtigungen. Dieses Skript wird gewöhnlich nur einmal ausgeführt, wenn Sie MySQL das erste Mal auf einem System installieren.

- `replace`

  Ein Dienstprogramm, das von `msql2mysql` benutzt wird, aber auch darüber hinaus benutzt werden kann. `replace` ändert Zeichenketten in Dateien oder auf der Standardeingabe. Benutzt eine finite Status-Maschine, um zuerst Übereinstimmung mit längeren Zeichenketten zu finden. Kann benutzt werden, um Zeichenketten umzudrehen. Der folgende Befehl zum Beispiel dreht `a` und `b` in den angegebenen Dateien um:

  ```
  shell> replace a b b a --Datei1 Datei2 ...
  ```

5.8.2. Das Kommandozeilen-Werkzeug

`mysql` ist eine einfache SQL-Shell (mit GNU `readline`-Fähigkeiten). Sie unterstützt interaktiven und nicht interaktiven Gebrauch. Wenn sie interaktiv benutzt wird, werden Anfrageergebnisse in einem ASCII-Tabellenformat ausgegeben. Wenn sie nicht interaktiv benutzt wird (zum Beispiel als Filter), wird das Ergebnis in Tabulator-separiertem Format ausgegeben. (Das Ausgabeformat kann mit den Kommandozeilenoptionen geändert werden.) Skripte können Sie einfach wie folgt laufen lassen:

```
shell> mysql datenbank < skript.sql > ausgabe.tab
```

Wenn Sie Probleme haben, die auf ungenügenden Speicher beim Client zurückzuführen sind, benutzen Sie die `--quick`-Option! Diese zwingt `mysql`, `mysql_use_result()` statt `mysql_store_result()` zu benutzen, um die Ergebnismenge zu holen.

Die Benutzung von `mysql` ist sehr einfach. Starten Sie es einfach wie folgt: `mysql datenbank` oder `mysql --user=benutzername --password=ihr_passwort datenbank`. Geben Sie ein SQL-Statement ein, beenden Sie es mit '`;`', '`\g`' oder '`\G`', und drücken Sie die Eingabetaste.

`mysql` unterstützt folgende Optionen:

- `-?, --help`

  Hilfetext ausgeben und beenden.

- `-A, --no-auto-rehash`

  Kein automatisches Rehashing. Man muss 'rehash' benutzen, um Tabellen- und Feld-Vervollständigung zu erhalten. Durch die Option wird mysql schneller gestartet.

- `-B, --batch`

  Ergebnisse mit einem Tabulator als Trennzeichen ausgeben, jede Tabellenzeile auf einer neuen Zeile. Keine History-Datei benutzen.

- `--character-sets-dir=...`

  Verzeichnis, in dem sich die Zeichensätze befinden.

- `-C, --compress`

  Im Client-Server-Protokoll Komprimierung benutzen.

- `-#, --debug[=...]`

  Debug loggen. Vorgabe ist 'd:t:o,/tmp/mysql.trace'.

- `-D, --database=...`

Datenbank, die benutzt werden soll. Hauptsächlich nützlich in der `my.cnf`-Datei.

- `--default-character-set=...`

  Den vorgabemäßigen Zeichensatz setzen.

- `-e, --execute=...`

  Befehl ausführen und beenden. (Ausgabe wie bei --batch)

- `-E, --vertical`

  Ausgabe einer Anfrage (Zeilen) vertikal darstellen. Ohne diese Option können Sie diese Ausgabe auch dadurch erzwingen, dass Sie Ihre Statements mit `\G` beenden.

- `-f, --force`

  Weitermachen, auch wenn ein SQL-Fehler auftritt.

- `-g, --no-named-commands`

  Benannte Befehle werden deaktiviert. Benutzen Sie nur die *-Form, oder benutzen Sie benannte Befehle nur bei Zeilen, die mit einem Semikolon enden. Ab Version 10.9 startet der Client vorgabemäßig mit ANGESCHALTETER Option! Wenn die -g-Option angeschaltet ist, funktionieren Befehle im Langformat jedoch immer noch von der ersten Zeile aus.

- `-G, --enable-named-commands`

  Benannte Befehle sind **angeschaltet**. Befehle im Langformat sind ebenso zugelassen wie die abgekürzten *-Befehle.

- `-i, --ignore-space`

  Leerzeichen nach Funktionsnamen ignorieren.

- `-h, --host=...`

  Connect to the given host.

- `-H, --html`

  HTML-Ausgabe produzieren.

- `-L, --skip-line-numbers`

  Bei Fehlern keine Zeilennummer ausgeben. Nützlich, wenn man mit Ergebnisdateien vergleichen will, die Fehlermeldungen enthalten.

- `--no-pager`

  Pager deaktivieren und nach stdout ausgeben. Siehe auch interaktive Hilfe (\h).

- `--no-tee`

  Ausgabedatei (Outfile) deaktivieren. Siehe auch interaktive Hilfe (\h).

- `-n, --unbuffered`

  Nach jeder Anfrage Buffer zurückschreiben (flush).

- `-N, --skip-column-names`

  In Ergebnissen keine Spaltennamen ausgeben.

- `-O, --set-variable var=option`

  Einer Variablen einen Wert zuweisen. `--help` listet Variablen auf.

- `-o, --one-database`

  Nur die vorgabemäßige Datenbank aktualisieren. Das ist nützlich, wenn man in der Update-Logdatei Aktualisierungen (Updates) auf eine andere Datenbank überspringen will.

- `--pager[=...]`

  Ausgabetyp. Vorgabe ist Ihre `ENV`-Variable `PAGER`. Gültige Pager sind less, more, cat [> Dateiname] usw. Siehe auch interaktive Hilfe (\h). Diese Option funktioniert nicht im Stapelmodus. Der Pager funktioniert nur unter UNIX.

- `-p[password], --password[=...]`

  Passwort, das für die Verbindung zum Server benutzt wird. Wenn das Passwort nicht auf der Kommandozeile angegeben wird, wird eine Eingabeaufforderung dafür ausgegeben. Beachten Sie: Wenn Sie die Kurzform `-p` benutzen, darf zwischen der Option und dem Passwort kein Leerzeichen stehen.

- `-P --port=...`

  TCP/IP-Portnummer, die für die Verbindung benutzt wird.

- `-q, --quick`

  Ergebnisse nicht cachen, Zeile für Zeile ausgeben. Das kann den Server verlangsamen, wenn die Ausgabe verschoben wird. Keine History-Datei benutzen.

- `-r, --raw`

  Spaltenwerte ohne Escape-Umwandlung schreiben. Benutzt für `--batch`.

- `-s, --silent`

  Schweigsamer sein.

- `-S --socket=...`

  Socket-Datei, die für die Verbindung benutzt wird.

- `-t --table`

  Ausgabe im Tabellenformat. Das ist die Vorgabe im Nicht-Stapelmodus.

- `-T, --debug-info`

  Beim Verlassen einige Debug-Informationen ausgeben.

- `--tee=...`

  Alles an die Ausgabedatei anhängen. Siehe auch interaktive Hilfe (\h). Funktioniert nicht im Stapelmodus.

- `-u, --user=#`

  Benutzer zum Einloggen, falls nicht der aktuelle UNIX-Benutzer.

- `-U, --safe-updates[=#], --i-am-a-dummy[=#]`

  Läßt nur `UPDATE` und `DELETE` zu, die Schlüssel benutzen. Siehe unten wegen weiterer Informationen über diese Option. Sie können diese Option zurücksetzen, wenn Sie sie in Ihrer `my.cnf`-Datei haben, indem Sie `--safe-updates=0` benutzen.

- `-v, --verbose`

  Geschwätzigere Ausgabe (-v -v -v ergibt das Tabellen-Ausgabeformat).

- `-V, --version`

  Versionsinformationen ausgeben und beenden.

- `-w, --wait`

  Wenn die Verbindung geschlossen wurde, warten und noch einmal versuchen, statt abzubrechen.

Mit `-O` oder `--set-variable` können Sie auch die folgenden Variablen setzen:

Variablenname	Vorgabe	Beschreibung
connect_timeout	0	Anzahl von Sekunden, bevor die Verbindung wegen Zeitüberschreitung abgebrochen wird
max_allowed_packet	16777216	Maximale Paketlänge, die zum Server gesendet bzw. von diesem empfangen wird
net_buffer_length	16384	Puffer für TCP/IP- und Socket-Kommunikation
select_limit	1000	Automatisches Limit für SELECT bei Benutzung von --i-am-a-dummy
max_join_size	1000000	Automatisches Limit für Zeilen in einem Join bei Benutzung von --i-am-a-dummy.

Wenn Sie 'help' auf der Kommandozeile eingeben, gibt `mysql` die Befehle aus, die es unterstützt:

```
mysql> help

MySQL-Befehle:
help    (\h)    Diesen Text anzeigen.
?       (\h)    Synonym für `help'.
clear   (\c)    Lösch-Befehl.
connect (\r)    Erneut mit dem Server verbinden. Optionale Argumente sind db und host.
edit    (\e)    Befehl mit $EDITOR editieren.
ego     (\G)    Befehl an den MySQL-Server schicken, Ergebnis vertikal anzeigen.
exit    (\q)    mysql beenden. Dasselbe wie quit.
go      (\g)    Befehl an den MySQL-Server schicken.
nopager (\n)    Pager deaktivieren, nach stdout ausgeben.
notee   (\t)    Nicht in die Ausgabedatei (Outfile) schreiben.
pager   (\P)    PAGER [auf_pager] setzen. Anfrageergebnisse über PAGER ausgeben.
print   (\p)    Aktuellen Befehl ausgeben.
quit    (\q)    mysql beenden.
rehash  (\#)    Vervollständigungs-Hash neu aufbauen.
source  (\.)    Eine SQLsSkriptdatei ausführen. Benötigt einen Dateinamen als Argument.
status  (\s)    Statusinformationen vom Server abrufen.
tee     (\T)    Ausgabedatei [auf_outfile] setzen. Alles an die angegebene Ausgabedatei anhängen.
use     (\u)    Eine andere Datenbankbenutzung. Benötigt Datenbanknamen als Argument.
```

Bei diesen Befehlen funktioniert PAGER nur unter UNIX.

Der `status`-Befehl gibt Ihnen einige Informationen über die Verbindung und den Server, den Sie benutzen. Wenn der Server im `--safe-updates`-Modus läuft, gibt `status` auch die Werte der `mysql`-Variablen aus, die Ihre Anfragen beeinflussen.

Eine nützliche Startoption für Anfänger (eingeführt in MySQL-Version 3.23.11) ist `--safe-updates` (oder `--i-am-a-dummy` für Benutzer, die irgendwann ein `DELETE FROM tabelle` eingeben, aber vergessen, die `WHERE`-Klausel) zu benutzen. Wenn Sie diese Option benutzen, schickt `mysql` beim Öffnen der Verbindung folgenden Befehl an den MySQL-Server:

```
SET SQL_SAFE_UPDATES=1,SQL_SELECT_LIMIT=#select_limit#,
    SQL_MAX_JOIN_SIZE=#max_join_size#"
```

Wobei `#select_limit#` und `#max_join_size#` Variablen sind, die auf der `mysql`-Kommandozeile gesetzt werden können.

See Abschnitt 6.5.6, „SET-Syntax“.

Die Auswirkung davon ist folgende:

- `UPDATE`- oder `DELETE`-Statements ohne Schlüsselbeschränkung im `WHERE`-Teil sind nicht zugelassen. Man kann jedoch ein `UPDATE/DELETE` durch die Benutzung von `LIMIT` erzwingen:

  ```
  UPDATE tabelle SET not_key_column=# WHERE not_key_column=# LIMIT 1;
  ```

- Alle großen Ergebnisse werden automatisch auf `#select_limit#` Zeilen begrenzt.
- `SELECT`'s, die wahrscheinlich mehr als `#max_join_size` Zeilenkombinationen durchgehen müssen, werden abgebrochen.

Einige nützliche Anmerkungen zum `mysql`-Client:

Einige Daten sind lesbarer, wenn sie vertikal angezeigt werden statt auf die übliche horizontale kastenähnliche Art. Langer Text zum Beispiel, der Zeilenumbrüche beinhaltet, ist bei vertikaler Ausgabe meist viel einfacher zu lesen.

```
mysql> select * from mails where length(txt) < 300 limit 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
    reply: monty@no.spam.com
  mail_to: "Thimble Smith" <tim@no.spam.com>
      sbj: UTF-8
      txt: >>>>> "Thimble" == Thimble Smith writes:

Thimble> Hi. Meines Erachtens eine gute Idee. Kennt sich jemand mit UTF-8
Thimble> oder Unicode aus? Ansonsten packe ich das auf meine TODO-Liste
Thimble> und warte, was passiert.

Ja, mach das bitte!

Regards,
Monty
    Datei: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

- Zum Mitloggen benutzen Sie die `tee`-Option. `tee` wird mit der Option `--tee=...` oder interaktiv auf der Kommandozeile mi dem Befehl `tee` gestartet. Alle Daten, die auf dem Bildschirm erscheinen, werden auch in die angegebene Datei geschrieben. Das kann auch für Debug-Zwecke sehr hilfreich sein. `tee` kann von der Kommandozeile aus mit dem Befehl `notee` deaktiviert werden. Wenn `tee` noch einmal eingegeben wird, wird wiederum mit dem Loggen begonnen. Ohne Parameter wird die vorherige Datei wiederum benutzt. Beachten Sie, dass `tee` die Ergebnisse nach jedem Befehl in die Datei zurückschreibt, direkt bevor die Kommandozeilenaufforderung für den nächsten Befehl erscheint.
- Das Durchstöbern oder Durchsuchen der Ergebnisse im interaktiven Modus in UNIX-less, -more oder einem ähnlichen Programm ist jetzt möglich mit der Option `--pager[=...]`. Ohne Argument aufgerufen sieht der `mysql`-Client in der Umgebungsvariablen PAGER nach und setzt `pager` auf diesen Wert. `pager` kann von der interaktiven Kommandozeile mit dem Befehl `pager` gestartet und mit `nopager` deaktiviert werden. Optional nimmt der Befehl ein Argument entgegen und setzt `pager` darauf. `pager` kann ohne Argument aufgerufen werden, aber das erfordert, dass die Option `--pager` benutzt wurde, ansonsten gibt `pager` in stdout aus. `pager` funktioniert nur unter UNIX, denn es benutzt die popen()-Funktion, die es unter Windows nicht gibt. Unter Windows kann statt dessen die `tee`-Option benutzt werden, wenngleich diese in manchen Situationen nicht ganz so handlich ist wie `pager`.
- Ein paar Tipps zu `pager`: Sie können es benutzen, um in eine Datei zu schreiben:

  ```
  mysql> pager cat > /tmp/log.txt
  ```

  Die Ergebnisse werden nur in eine Datei geschrieben. Sie können auch Optionen an Programme übergeben, die Sie mit `pager` zusammen benutzen wollen:

  ```
  mysql> pager less -n -i -S
  ```

  Beachten Sie hierbei die Option '-S'. Beim Durchstöbern der Ergebnisse werden Sie diese wahrscheinlich als sehr nützlich erachten. Probieren Sie dei Option mit horizontaler Ausgabe (Befehle enden mit '\g', oder ';') und mit vertikaler Ausgabe (Befehle enden mit '\G') aus. Manchmal ist ein sehr breites Ergebnis schwer am Bildschirm zu lesen. Mit der Option -S für less können Sie die Ergebnisse im interaktiven less von links nach rechts durchstöbern, wobei verhindert wird, dass Zeilen, die länger sind als Ihre Bildschirmbreite, in die nächste Zeile umgebrochen werden. Das kann ein Ergebnis sehr viel lesbarer

gestalten. Sie können den Modus im interaktiven less an- und abschalten, wenn Sie '-S' benutzen. Siehe 'h' für weitere Hilfe zu less.

- Zum Schluss (falls Sie das nicht schon aus den oben aufgeführten Beispielen heraus gefunden haben ;-) können Sie sehr komplexe Dinge tun, um die Ergebnisse zu handhaben. Folgendes würde die Ergebnisse beispielsweise an zwei verschiedene Dateien in zwei unterschiedlichen Verzeichnissen schicken, auf zwei unterschiedlichen Festplatten, die auf /dr1 und /dr2 gemountet sind, und dennoch die Ergebnisse über less am Bildschirm anzeigen:

```
mysql> pager cat | tee /dr1/tmp/res.txt | tee /dr2/tmp/res2.txt | less -n -i -S
```

- Sie können die obigen Funktionen auch kombinieren, indem Sie `tee` anschalten und `pager` auf 'less' setzen. Dann können Sie die Ergebnisse in 'less' durchstöbern und trotzdem wird alles zugleich an eine Datei angehängt. Der Unterschied zwischen `UNIX tee`, was mit `pager` benutzt wird, und dem im `mysql`-Client eingebauten `tee` ist, dass das eingebaute `tee` sogar dann funktioniert, wenn kein `UNIX tee` verfügbar ist. Darüber hinaus gibt das eingebaute `tee` alles, was mitgeloggt wird, auch am Bildschirm aus, wohingegen das `UNIX tee` in Verbindung mit `pager` nicht so viel mitloggt. Letztlich läßt sich das interaktive `tee` auch handlicher aus- und einschalten, wenn Sie teilweise mitloggen wollen, aber in der Lage sein, das Feature zwischendurch auszuschalten.

5.8.3. mysqladmin, Verwaltung eines MySQL-Servers

Ein Dienstprogramm, um Verwaltungsoperationen durchzuführen. Die Syntax ist::

```
shell> mysqladmin [OPTIONS] befehl [befehl-option] befehl ...
```

Sie erhalten eine Auflistung der Optionen, die Ihre `mysqladmin`-Version unterstützt, indem Sie `mysqladmin --help` ausführen.

Das aktuelle `mysqladmin` unterstützt folgende Befehle:

- `create datenbank`

  Eine neue Datenbank erzeugen.

- `drop datenbank`

  Eine Datenbank und alle ihre Tabellen löschen.

- `extended-status`

  Eine erweiterte Statusmeldung vom Server ausgeben.

- `flush-hosts`

  Alle gecacheten Hosts zurückschreiben (flush).

- `flush-logs`

  Alle Logs zurückschreiben (flush).

- `flush-tables`

  Alle Tabellen zurückschreiben (flush).

- `flush-privileges`

  Berechtigungstabellen neu laden (dasselbe wie reload).

- `kill id,id,...`

  MySQL-Threads killen.

- `password`

  Ein neues Passwort setzen. Altes Passwort zu neuem Passwort ändern.

- `ping`

  Überprüfen, ob mysqld lebt.

- `processlist`

  Auflistung aktiver Threads im Server.

- `reload`

  Berechtigungstabellen neu laden.

- `refresh`

  Alle Tabellen zurückschreiben (flush), Log-Dateien schließen und erneut öffnen.

- `shutdown`

  Server herunter fahren.

- `slave-start`

  Slave-Replikations-Thread starten.

- `slave-stop`

  Slave-Replikations-Thread anhalten.

- `status`

  Eine kurze Statusmeldung vom Server ausgeben.

- `variables`

  Verfügbare Variablen ausgeben.

- `version`

  Versionsinformation vom Server abrufen.

Alle Befehle können auf ihr eindeutiges Präfix abgekürzt werden. Beispiel:

```
shell> mysqladmin proc stat
+----+-------+-----------+----+-------------+------+-------+------+
| Id | User  | Host      | db | Command     | Time | State | Info |
+----+-------+-----------+----+-------------+------+-------+------+
| 6  | monty | localhost |    | Processlist | 0    |       |      |
+----+-------+-----------+----+-------------+------+-------+------+
Uptime: 10077  Threads: 1  Questions: 9  Slow queries: 0  Opens: 6  Flush tables: 1  Open tables: 2  Memory in use: 10
```

Das Ergebnis des `mysqladmin status`-Befehls hat folgende Spalten:

Uptime	Anzahl von Sekunden, seit der MySQL-Server hoch gefahren ist.
Threads	Anzahl aktiver Threads (Clients).
Questions	Anzahl von Questions von Clients, seit `mysqld` gestartet wurde.
Slow queries	Anfragen, die länger als `long_query_time` Sekunden benötigten. See Abschnitt 5.9.5, „Die Anfragen-Log-Datei für langsame Anfragen“.
Opens	Wie viele Tabellen `mysqld` geöffnet hat.
Flush Tables	Anzahl von `flush ...`-, `refresh`- und `reload`-Befehlen.
Open Tables	Anzahl der Tabellen, die gerade geöffnet sind.
Memory in use	Arbeitsspeicher, der direkt vom `mysqld`-Code beansprucht wird (nur verfügbar, wenn MySQL mit --with-debug=full kompiliert wurde).
Max memory used	Maximaler Arbeitsspeicher, der direkt vom `mysqld`-Code beansprucht wird (nur verfügbar, wenn MySQL mit --with-debug=full kompiliert wurde).

Wenn Sie `mysqladmin shutdown` auf einem Socket ausführen (mit anderen Worten, auf dem Computer, wo `mysqld` läuft), wartet `mysqladmin`, bis MySQL die `pid-file` entfernt hat, um sicherzustellen, dass der `mysqld`-Server korrekt angehalten wurde.

5.8.4. Benutzung von `mysqlcheck` für Tabellenwartung und Wiederherstellung

nach Abstürzen

Ab MySQL-Version 3.23.38 können Sie ein neues Prüf- und Reparatur-Werkzeug für `MyISAM`-Tabellen einsetzen. Der Unterschied zu `myisamchk` ist, dass `mysqlcheck` benutzt werden kann, wenn der `mysqld`-Server läuft, wohingegen `myisamchk` nur benutzt werden sollte, wenn er nicht läuft. Der Vorteil ist, dass Sie den Server zum Prüfen oder zur Reparatur Ihrer Tabellen nicht mehr herunter fahren müssen.

`mysqlcheck` benutzt die MySQL-Server-Befehle `CHECK`, `REPAIR`, `ANALYZE` und `OPTIMIZE` auf eine für den Benutzer bequeme Weise.

Es gibt drei alternative Möglichkeiten, `mysqlcheck` aufzurufen:

```
shell> mysqlcheck [OPTIONS] datenbank [tabellen]
shell> mysqlcheck [OPTIONS] --databases datenbank1 [datenbank2 datenbank3...]
shell> mysqlcheck [OPTIONS] --all-databases
```

Daher kann es hinsichtlich der Auswahl von Datenbanken und Tabellen ähnlich wie `mysqldump` benutzt werden.

`mysqlcheck` besitzt im Vergleich zu den anderen Clients ein besonderes Feature: Das vorgabemäßige Verhalten, Tabellen mit -c zu prüfen, kann geändert werden, indem die Binärdatei umbenannt wird. Wenn Sie nur ein Werkzeug haben wollen, das vorgabemäßig Tabellen repariert, kopieren Sie eine `mysqlcheck` mit einem neuen Namen auf Ihre Festplatte, nämlich `mysqlrepair`, oder legen alternativ einen symbolischen Link auf `mysqlrepair` und benennen den Link `mysqlrepair`. Wenn Sie jetzt `mysqlrepair` aufrufen, repariert es vorgabemäßig Tabellen.

Folgende Namen können Sie benutzen, um das vorgabemäßige Verhalten von `mysqlcheck` zu verändern:

```
mysqlrepair:   Vorgabe-Option: -r (reparieren)
mysqlanalyze:  Vorgabe-Option: -a (analysieren)
mysqloptimize: Vorgabe-Option: -o (optimieren)
```

Die verfügbaren Optionen für `mysqlcheck` sind hier aufgelistet. Bitte prüfen Sie mit `mysqlcheck --help`, welche davon Ihre Version unterstützt.

- `-A, --all-databases`

  Prüft alle Datenbanken. Das ist dasselbe wie --databases mit allen Datenbanken ausgewählt.

- `-1, --all-in-1`

  Statt für jede Tabelle eine Anfrage auszuführen, alle Anfragen in 1 Anfrage pro Datenbank ausführen. Tabellennamen stehen in einer durch Kommas getrennten Liste.

- `-a, --analyze`

  Analysiert die angegebene Tabelle.

- `--auto-repair`

  Wenn eine geprüfte Tabelle beschädigt ist, sie automatisch reparieren. Die Reparatur wird durchgeführt, nachdem alle Tabellen geprüft wurden, falls beschädigte gefunden wurden.

- `-#, --debug=...`

  Debug-Log-Datei ausgeben. Das ist häufig 'd:t:o,filename'.

- `--character-sets-dir=...`

  Verzeichnis, wo Zeichensätze gespeichert sind.

- `-c, --check`

  Tabelle auf Fehler prüfen.

- `-C, --check-only-changed`

  Nur die Tabellen prüfen, die seit der letzten Prüfung geändert wurden oder die nicht ordnungsgemäß geschlossen wurden.

- `--compress`

  Kompression im Client-Server-Protokoll benutzen.

- `-?, --help`

  Diese Nachricht ausgeben und beenden.

- `-B, --databases`

  Mehrere Datenbanken prüfen. Beachten Sie den Unterschied im Gebrauch: In diesem Fall werden keine Tabellen angegeben. Alle Namensargumente werden als Datenbanknamen erachtet.

- `--default-character-set=...`

  Setzt den vorgabemäßigen Zeichensatz.

- `-F, --fast`

  Nur Tabellen prüfen, die nicht ordnungsgemäß geschlossen wurden.

- `-f, --force`

  Fortfahren, auch wenn ein SQL-Fehler auftritt.

- `-e, --extended`

  Wenn Sie diese Option beim Prüfen von Tabellen benutzen, stellt das sicher, dass die Tabelle zu 100% konsistent ist, dauert aber sehr lange. Wenn Sie diese Option beim Reparieren von Tabellen benutzen, wird eine erweiterte Reparatur der Tabelle durchgeführt, was nicht nur sehr lange dauern kann, sondern auch viele 'Müll'-Zeilen produzieren kann!

- `-h, --host=...`

  Mit dem angegebenen Host verbinden.

- `-m, --medium-check`

  Schneller als extended-check, findet aber nur 99,99% aller Fehler. Sollte in den meisten Fällen genügen.

- `-o, --optimize`

  Tabelle optimieren.

- `-p, --password[=...]`

  Passwort, das bei der Verbindung zum Server benutzt werden soll. Wenn das Passwort nicht angegeben wird, wird vom Terminal eine Eingabeaufforderung präsentiert.

- `-P, --port=...`

  Portnummer, die für Verbindungen zum Server benutzt werden soll.

- `-q, --quick`

  Wenn Sie diese Option beim Prüfen von Tabellen benutzen, verhindert das, dass die Zeilen nach falschen Verknüpfungen (Links) durchgesehen werden (gescannt). Das ist die schnellste Prüfmethode. Wenn Sie diese Option beim Reparieren von Tabellen benutzen, wird versucht, nur den Index-Baum zu reparieren. Das ist die schnellste Reparaturmethode.

- `-r, --repair`

  Kann fast alles reparieren, ausser eindeutige Schlüssel, die nicht eindeutig sind.

- `-s, --silent`

  Nur Fehlermeldungen ausgeben.

- `-S, --socket=...`

  Socket-Datei, die für die Verbindung benutzt werden soll.

- `--tables`

  Option --databases (-B) überschreiben.

- `-u, --user=#`

Benutzer zum Einloggen, falls nicht der aktuelle Unix-Benutzer.

- `-v, --verbose`

  Informationen über die verschiedenen Phasen ausgeben.

- `-V, --version`

  Versionsinformationen ausgeben und beenden.

5.8.5. mysqldump, Tabellenstrukturen und -daten dumpen

Dienstprogramm, um eine Datenbank oder eine Sammlung von Datenbanken zu sichern oder um Daten auf einen anderen SQL-Server zu übertragen (nicht notwendigerweise ein MySQL-Server). Der Dump enthält SQL-Statements, um Datenbanken und Tabellen zu erzeugen und / oder Tabellen mit Daten zu füllen.

Wenn Sie eine Datensicherung auf dem Server machen, sollten Sie in Betracht ziehen, statt dessen `mysqlhotcopy` zu benutzen. See Abschnitt 5.8.6, „mysqlhotcopy, MySQL-Datenbanken und Tabellen kopieren“.

```
shell> mysqldump [OPTIONS] datenbank [tabellen]
OR     mysqldump [OPTIONS] --databases [OPTIONS] datenbank1 [datenbank2 datenbank3...]
OR     mysqldump [OPTIONS] --all-databases [OPTIONS]
```

Wenn Sie keine Tabellen angeben oder `--databases` bzw. `--all-databases` benutzen, wird die gesamte Datenbank (bzw. werden alle Datenbanken) gedumpt.

Sie erhalten eine Auflistung der Optionen, die Ihre Version von `mysqldump` unterstützt, indem Sie `mysqldump --help` eingeben.

Wenn Sie `mysqldump` ohne `--quick` oder `--opt` ausführen, beachten Sie, dass `mysqldump` die gesamte Ergebnismenge in den Arbeitsspeicher lädt, bevor das Ergebnis gedumpt wird. Das kann zu Problemen führen, wenn Sie eine große Datenbank dumpen.

Wenn Sie eine neue Version des `mysqldump`-Programms benutzen und einen Dump erzeugen, der in einen sehr alten MySQL-Server eingelesen werden soll, sollten Sie die `--opt`- und `-e`-Optionen nicht benutzen.

`mysqldump` unterstützt folgende Optionen:

- `--add-locks`

  Führt `LOCK TABLES` vor und `UNLOCK TABLE` nach jedem Tabellen-Dump durch (um schnelleres Einfügen in MySQL zu erreichen).

- `--add-drop-table`

  Ein `drop table` vor jedem `create`-Statement hinzufügen.

- `-A, --all-databases`

  Alle Datenbanken dumpen. Das ist dasselbe wie `--databases` mit allen Datenbanken ausgewählt.

- `-a, --all`

  Alle MySQL-spezifischen Optionen für `create` benutzen.

- `--allow-keywords`

  Erzeugung von Spaltennamen zulassen, die Schlüsselwörter sind. Das funktioniert, indem jedem Spaltenname der Tabellenname als Präfix angefügt wird.

- `-c, --complete-insert`

  Vollständige `insert`-Statements benutzen (mit Spaltennamen).

- `-C, --compress`

  Alle Informationen zwischen Client und Server komprimieren, wenn bei Kompression unterstützen.

- `-B, --databases`

  Mehrere Datenbanken prüfen. Beachten Sie den Unterschied im Gebrauch: In diesem Fall werden keine Tabellen angegeben. Alle Namensargumente werden als Datenbanknamen erachtet. Vor jeder Ausgabe einer neuen Datenbank wird `USE datenbank;` eingefügt.

- `--delayed`

  Zeilen mit dem `INSERT DELAYED`-Befehl einfügen.

- `-e, --extended-insert`

  Die neue mehrzeilige `INSERT`-Syntax benutzen. (Ergibt kompaktere und schnellere inserts-Statements.)

- `-#, --debug[=option_string]`

  Programmbenutzung tracen (für Debug-Zwecke).

- `--help`

  Hilfetext ausgeben und beenden.

- `--fields-terminated-by=...`, `--fields-enclosed-by=...`, `--fields-optionally-enclosed-by=...`, `--fields-escaped-by=...`, `--lines-terminated-by=...`

  Diese Optionen werden zusammen mit der `-T`-Option benutzt und haben dieselbe Bedeutung wie die entsprechenden Klauseln für `LOAD DATA INFILE`. See Abschnitt 7.4.9, „`LOAD DATA INFILE`-Syntax“.

- `-F, --flush-logs`

  Log-Datei im MySQL-Server zurückschreiben, bevor der Dump durchgeführt wird.

- `-f, --force,`

  Fortfahren, selbst wenn beim Dump einer Tabelle ein SQL-Fehler auftritt.

- `-h, --host=..`

  Daten auf dem MySQL-Server auf dem genannten Host dumpen. Der vorgabemäßige Host ist `localhost`.

- `-l, --lock-tables.`

  Alle Tabellen sperren, bevor mit dem Dump begonnen wird. Die Tabellen werden mit `READ LOCAL` gesperrt, um gleichzeitiges Einfügen zu erlauben (bei `MyISAM`-Tabellen).

- `-n, --no-create-db`

  'CREATE DATABASE /*!32312 IF NOT EXISTS*/ datenbank;' wird nicht in die Ausgabe gschrieben. Diese Zeile wird ansonsten hinzugefügt, wenn --databases oder --all-databases angegeben wurde.

- `-t, --no-create-info`

  Keine Tabellenerzeugungsinformation schreiben (das `CREATE TABLE`-Statement).

- `-d, --no-data`

  Keine Zeileninformationen für die Tabelle schreiben. Das ist sehr nützlich, wenn Sie lediglich einen Dump der Tabellenstruktur erzeugen wollen.

- `--opt`

  Dasselbe wie `--quick --add-drop-table --add-locks --extended-insert --lock-tables`. Das sollte den schnellstmöglichen Dump zum Einlesen in einen MySQL-Server ergeben.

- `-pihr_passwort, --password[=ihr_passwort]`

  Das Passwort, das für die Verbindung zum Server benutzt werden soll. Wenn Sie keinen ‘`=ihr_passwort`’-Teil angeben, zeigt `mysqldump` eine Eingabeaufforderung für Ihr Passwort.

- `-P port_num, --port=port_num`

  Die TCP/IP-Portnummer, die für die Verbindung zu einem Host benutzt werden soll. (Diese wird für Verbindungen zu Hosts

ausser `localhost` benutzt, für den Unix-Sockets benutzt werden.)

- `-q, --quick`

  Anfrage nicht puffern, sondern direkt zu stdout dumpen. Benutzt für die Durchführung `mysql_use_result()`.

- `-r, --result-file=...`

  Direkte Ausgabe in die angegebene Datei. Diese Ausgabe sollte bei MS-DOS benutzt werden, weil sie verhindert, dass das Zeichen für neue Zeile '\n' in '\n\r' (new line + carriage return) umgewandelt werden.

- `-S /pfad/zu/socket, --socket=/pfad/zu/socket`

  Die Socket-Datei, die für die Verbindung zu `localhost` benutzt werden soll (was der vorgabemäßige Host ist).

- `--tables`

  Überschreibt die Option --databases (-B).

- `-T, --tab=pfad-zu-einem-verzeichnis`

  Erzeugt eine `tabelle.sql`-Datei, die die SQL-CREATE-Befehle enthält, und eine `tabelle.txt`-Datei, die die Daten enthält, für jede angegebene Tabelle. **HINWEIS**: Das funktioniert nur, wenn `mysqldump` auf derselben Maschine läuft wie der `mysqld`-Daemon. Das Format der `.txt`-Datei hängt von den `--fields-xxx`- und `--lines--xxx`-Optionen ab.

- `-u benutzername, --user=benutzername`

  Der MySQL-Benutzername, der für die Verbindung zum Server benutzt werden soll. Der Vorgabewert ist Ihr Unix-Loginname.

- `-O var=option, --set-variable var=option`

  Den Wert einer Variablen setzen. Die möglichen Werte sind unten aufgeführt.

- `-v, --verbose`

  Geschwätziger Modus. Gibt mehr Informationen darüber aus, was das Programm tut.

- `-V, --version`

  Versionsinformationen ausgeben und beenden.

- `-w, --where='wo-bedingung'`

  Nur ausgewählte Datensätze dumpen. Beachten Sie, dass Anführungszeichen zwingend erforderlich sind:

  ```
  "--where=user='jimf'" "-wuserid>1" "-wuserid<1"
  ```

- `-O net_buffer_length=#, where # < 16M`

  Beim Erzeugen von mehrzeiligen insert-Statements (wie bei der Option `--extended-insert` oder `--opt`), erzeugt `mysqldump` Zeilen bis zur Länge von `net_buffer_length`. Wenn Sie diesen Wert herauf setzen, müssen Sie sicherstellen, dass die `max_allowed_packet`-Variable im MySQL-Server größer als `net_buffer_length` ist.

Der häufigste Gebrauch von `mysqldump` dient wahrscheinlich der Herstellung einer Datensicherung ganzer Datenbanken. See Abschnitt 5.4.1, „Datenbank-Datensicherungen“.

```
mysqldump --opt datenbank > datensicherung.sql
```

Diese können Sie zurück in MySQL einlesen mit:

```
mysql datenbank < datensicherung.sql
```

oder

```
mysql -e "source /pfad-zur-datensicherung/datensicherung.sql" datenbank
```

Ausserdem ist es sehr nützlich, um einen anderen MySQL-Server mit Informationen aus einer Datenbank zu füllen:

```
mysqldump --opt datenbank | mysql ---host=entfernter-host -C datenbank
```

Es ist möglich, mehrere Datenbanken mit einem Befehl zu dumpen:

```
mysqldump --databases datenbank1 [datenbank2 datenbank3...] > meine_datenbanken.sql
```

Wenn Sie alle Datenbanken dumpen wollen, benutzen Sie:

```
mysqldump --all-databases > alle_datenbanken.sql
```

5.8.6. mysqlhotcopy, MySQL-Datenbanken und Tabellen kopieren

`mysqlhotcopy` ist ein Perl-Skript, das `LOCK TABLES`, `FLUSH TABLES` und `cp` oder `scp` benutzt, um schnell eine Datensicherung einer Datenbank anzulegen. Es stellt die schnellste Möglichkeit dar, eine Sicherung einer Datenbank oder einzelner Tabellen durchzuführen, läuft aber nur auf derselben Maschine, auf der sich die Datenbankverzeichnisse befinden.

```
mysqlhotcopy datenbank [/pfad/zu/neuem_verzeichnis]
mysqlhotcopy datenbank_1 ... datenbank_n /pfad/zu/neuem_verzeichnis
mysqlhotcopy datenbank./regex/
```

`mysqlhotcopy` unterstützt folgende Optionen:

- `-?, --help`

  Hilfe ausgeben und beenden.

- `-u, --user=#`

  Benutzername zum Einloggen.

- `-p, --password=#`

  Passwort für die Verbindung zum Server.

- `-P, --port=#`

  Port zur Verbindung zum lokalen Server.

- `-S, --socket=#`

  Socket zur Verbindung zum lokalen Server.

- `--allowold`

  Nicht abbrechen, wenn das Ziel bereits existiert (sondern in _old umbenennen)

- `--keepold`

  Vorheriges (jetzt umbenanntes) Ziel nach dem Durchführen nicht löschen.

- `--noindices`

  Keine kompletten Index-Dateien in die Kopie einfügen, um die Datensicherung kleiner und schneller zu machen. Die Indexe können später mit `myisamchk -rq.` neu aufgebaut werden.

- `--method=#`

  Kopiermethode (`cp` oder `scp`).

- `-q, --quiet`

  Keine Meldungen ausgeben, ausser bei Fehlern.

- `--debug`

  Debug anschalten.

- `-n, --dryrun`

Aktionen berichten, ohne sie auszuführen.

- `--regexp=#`

  Alle Datenbanken mit übereinstimmenden regexp-Namen sichern.

- `--suffix=#`

  Suffix für Namen kopierter Datenbanken.

- `--checkpoint=#`

  Checkpoint-Eingang in angegebene datenbank.tabelle einfügen.

- `--flushlog`

  Log-Dateien zurückschreiben, sobald alle Tabellen gesperrt sind.

- `--tmpdir=#`

  Temporäres Verzeichnis (anstelle von /tmp).

Geben Sie `perldoc mysqlhotcopy` ein, um eine vollständigere Dokumentation von `mysqlhotcopy` zu erhalten.

`mysqlhotcopy` liest die Gruppen `[client]` und `[mysqlhotcopy]` aus den Optionsdateien.

Damit Sie `mysqlhotcopy` ausführen können, benötigen Sie Schreibrechte im Datensicherungsverzeichnis, `SELECT`-Berechtigung auf die Tabellen, die Sie kopieren wollen, und die MySQL-`Reload`-Berechtigung (damit Sie `FLUSH TABLES` ausführen können).

5.8.7. mysqlimport, Daten aus Textdateien importieren

`mysqlimport` stellt eine Kommandozeilen-Schnittstelle für das `LOAD DATA INFILE` SQL-Statement zur Verfügung. Die meisten Optionen für `mysqlimport` entsprechen denselben Optionen für `LOAD DATA INFILE`. See Abschnitt 7.4.9, „`LOAD DATA INFILE`-Syntax“.

`mysqlimport` wird wie folgt aufgerufen:

```
shell> mysqlimport [optionen] datenbank textdatei1 [textdatei2....]
```

Bei jeder Textdatei, die auf der Kommandozeile angegeben wird, entfernt `mysqlimport` jegliche Erweiterungen vom Dateinamen und benutzt das Ergebnis, um festzulegen, in welche Tabelle der Dateiinhalt importiert werden soll. Dateien namens `patient.txt`, `patient.text` und `patient` beispielsweise würden alle in eine Tabelle namens `patient` importiert werden.

`mysqlimport` unterstützt folgende Optionen:

- `-c, --columns=...`

  Diese Option nimmt ein durch Kommas getrennte Auflistung von Feldnamen als Argument entgegen. Die Feldliste wird benutzt, um einen korrekten `LOAD DATA INFILE`-Befehl zu erzeugen, der an MySQL durchgereicht wird. See Abschnitt 7.4.9, „`LOAD DATA INFILE`-Syntax“.

- `-C, --compress`

  Komprimiert alle Informationen zwischen Client und Server, wenn bei Kompression unterstützen.

- `-#, --debug[=option_string]`

  Programmbenutzung tracen (zum Debuggen).

- `-d, --delete`

  Tabelle leeren, bevor die Textdatei importiert wird.

- `--fields-terminated-by=...`, `--fields-enclosed-by=...`, `--fields-optionally-enclosed-by=...`, `--fields-escaped-by=...`, `--lines-terminated-by=...`

Diese Optionen haben dieselbe Bedeutung wie die entsprechenden Klauseln für `LOAD DATA INFILE`. See Abschnitt 7.4.9, „`LOAD DATA INFILE`-Syntax“.

- `-f, --force`

  Fehler ignorieren. Wenn beispielsweise eine Tabelle für eine Textdatei nicht existiert, mit den verbleibenden Dateien weitermachen. Ohne `--force` wird `mysqlimport` beendet, wenn die Tabelle nicht existiert.

- `--help`

  Hilfetext ausgeben und beenden.

- `-h host_name, --host=host_name`

  Daten in den MySQL-Server auf dem genannten Host importieren. Der vorgabemäßige Host ist `localhost`.

- `-i, --ignore`

  Siehe Beschreibung für die `--replace`-Option.

- `-l, --lock-tables`

  ALLE Tabellen für Schreibvorgänge sperren, bevor irgend welche Textdateien verarbeitet werden. Das stellt sich, dass alle Tabellen auf dem Server synchronisiert werden.

- `-L, --local`

  Liest Eingabedateien vom Client. Vorgabemäßig wird angenommen, dass Textdateien auf dem Server liegen, wenn Sie sich über `localhost` verbinden (was der vorgabemäßige Host ist).

- `-pihr_passwort, --password[=ihr_passwort]`

  Das Passwort, das für die Verbindung zum Server benutzt werden soll. Wenn Sie keinen ‘`=ihr_passwort`’-Teil angeben, zeigt `mysqlimport` eine Eingabeaufforderung für Ihr Passwort.

- `-P port_num, --port=port_num`

  Die TCP/IP-Portnummer, die für die Verbindung zu einem Host benutzt werden soll. (Diese wird für Verbindungen zu Hosts ausser `localhost` benutzt, für den Unix-Sockets benutzt werden.)

- `-r, --replace`

  Die `--replace`- und `--ignore`-Optionen steuern die Handhabung von Eingabe-Datensätzen, die bestehende Datensätze auf eindeutigen Schlüsseln duplizieren würden. Wenn Sie `--replace` angeben, werden bestehende Zeilen ersetzt, die denselben eindeutigen Schlüsselwert besitzen. Wenn Sie `--ignore` angeben, werden Zeilen, die eine bestehende Zeile duplizieren würden, übersprungen. Wenn Sie keine der beiden Optionen angeben, tritt ein Fehler auf, wenn ein doppelter Schlüsseleintrag gefunden wird, und der Rest der Textdatei wird ignoriert.

- `-s, --silent`

  Schweigsamer Modus. Ausgaben erfolgen nur, wenn Fehler auftreten.

- `-S /pfad/zu/socket, --socket=/pfad/zu/socket`

  Die Socket-Datei, die für die Verbindung zu `localhost` benutzt werden soll (der der vorgabemäßige Host ist).

- `-u benutzername, --user=benutzername`

  Der MySQL-Benutzername, der für die Verbindung zum Server benutzt werden soll. Der Vorgabewert ist Ihr Unix-Loginname.

- `-v, --verbose`

  Geschwätziger Modus. Mehr Informationen darüber ausgeben, was das Programm macht.

- `-V, --version`

  Versionsinformationen ausgeben und beenden.

Hier ist ein Beispiel für die Benutzung von `mysqlimport`:

```
$ mysql --version
mysql  Ver 9.33 Distrib 3.22.25, for pc-linux-gnu (i686)
$ uname -a
Linux xxx.com 2.2.5-15 #1 Mon Apr 19 22:21:09 EDT 1999 i586 unknown
$ mysql -e 'CREATE TABLE imptest(id INT, n VARCHAR(30))' test
$ ed
a
100     Max Sydow
101     Graf Dracula
.
w imptest.txt
32
q
$ od -c imptest.txt
0000000   1   0   0  \t   M   a   x       S   y   d   o   w  \n   1   0
0000020   1  \t   G r a f       D   r   a   c   u   l   a  \n
0000040
$ mysqlimport --local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
$ mysql -e 'SELECT * FROM imptest' test
+------+---------------+
| id   | n             |
+------+---------------+
|  100 | Max Sydow     |
|  101 | Graf Dracula  |
+------+---------------+
```

5.8.8. Datenbanken, Tabellen und Spalten anzeigen

`mysqlshow` wird benutzt, um schnell nachzusehen, welche Datenbanken, Tabellen und Tabellenspalten es gibt.

Mit dem `mysql`-Programm können Sie dieselben Information mit den `SHOW`-Befehlen erhalten. See Abschnitt 5.5.5, „SHOW-Syntax“.

`mysqlshow` wird wie folgt aufgerufen:

```
shell> mysqlshow [optionen] [datenbank [tabelle [spalte]]]
```

- Wenn keine Datenbank angegeben wird, werden alle passenden Datenbanken gezeigt.
- Wenn keine Tabelle angegeben wird, werden alle passenden Tabellen in der Datenbank gezeigt.
- Wenn keine Spalte angegeben wird, werden alle passenden Spalten und Spaltentypen in der Tabelle gezeigt.

Beachten Sie, dass Sie in neueren MySQL-Versionen nur die Datenbanken, Tabellen und Spalten sehen können, für die Sie irgend welche Berechtigungen haben.

Wenn das letzte Argument einen Shell- oder SQL-Platzhalter enthält (`*`, `?`, `%` oder `_`), wird nur das gezeigt, was dem Platzhalter entspricht. Das kann zu Verwirrung führen, wenn Sie Spalten einer Tabelle anzeigen, die einen Unterstrich (`_`) enthalten, weil Ihnen `mysqlshow` in diesem Fall nur die Tabellennamen zeigt, die dem Muster entsprechen. Das kann leicht durch Hinzufügen eines zusätzlichen `%` am Ende der Kommandozeile (als separates Argument) behoben werden.

5.8.9. perror, Erklärung der Fehler-Codes

`perror` wird benutzt, um Fehlermeldungen auszugeben. `perror` wird wie folgt aufgerufen:

```
shell> perror [optionen] [ERRORCODE [ERRORCODE...]]

For example:

shell> perror 64 79
Error code  64:  Machine ist not on the network
Error code  79:  Can not access a needed shared library
```

`perror` wird benutzt, um eine Beschreibung für einen Systemfehler-Code anzuzeigen, oder einen Fehler-Code des MyISAM/ISAM-Tabellen-Handlers. Die Fehlermeldungen sind hauptsächlich abhängig vom Betriebssystem.

5.8.10. Wie SQL-Befehle aus einer Textdatei laufen gelassen werden

Der `mysql`-Client wird typischerweise interaktiv benutzt, wie folgt:

```
shell> mysql datenbank
```

Es ist jedoch möglich, Ihre SQL-Befehle in eine Datei zu schreiben und `mysql` anzuweisen, ihre Eingaben aus dieser Datei zu

lesen. Um das zu tun, erzeugen Sie eine Textdatei `textdatei`, die die Befehle enthält, die Sie ausführen wollen. Dann rufen Sie `mysql` wie gezeigt auf:

```
shell> mysql datenbank < textdatei
```

Sie können Ihre Textdatei auch mit einem `USE datenbank`-Statement beginnen lassen. In diesem Fall ist es nicht notwendig, den Datenbanknamen auf der Kommandozeile anzugeben:

```
shell> mysql < textdatei
```

See Abschnitt 5.8, „Clientseitige Skripte und Hilfsprogramme von MySQL“.

5.9. Die MySQL-Log-Dateien

MySQL hat mehrere unterschiedliche Log-Dateien, die Ihnen helfen können herauszufinden, was innerhalb `mysqld` vor sich geht:

Die Fehler-Log-Datei	Probleme, die beim Start, beim Laufenlassen oder beim Anhalten von `mysqld` auftreten.
Die ISAM-Log-Datei	Loggt alle Änderungen in ISAM-Tabellen mit. Wird nur benutzt, um den ISAM-Code zu debuggen.
Die Anfragen-Log-Datei	Hergestellte Verbindungen und ausgeführte Anfragen.
Die Update-Log-Datei	Veraltet: Speichert Statements, die Daten verändern.
Die Binär-Log-Datei	Speichert alle Statements, die etwas ändern. Wird auch für Replikation benutzt.
Die Slow-Log-Datei	Speichert alle Anfragen, die länger als `long_query_time` zur Ausführung benötigten oder keine Indexe benutzten.

Alle Log-Dateien liegen im `mysqld` Daten-Verzeichnis. Sie können `mysqld` zwingen, die Log-Dateien neu zu öffnen (oder in manchen Fällen auf eine neue Log-Datei umzuschalten), indem Sie `FLUSH LOGS` ausführen. See Abschnitt 5.5.3, „`FLUSH`-Syntax“.

5.9.1. Die Fehler-Log-Datei

`mysqld` schreibt alle Fehler nach stderr, die das `safe_mysqld`-Skript in eine Datei namens `'hostname'.err` umleitet. (Unter Windows schreibt `mysqld` direkt in die Datei `\mysql\data\mysql.err`.)

Diese enthält Informationen, wann `mysqld` gestartet und angehalten wurde und zusätzlich jeden kritischen Fehler, der während der Laufzeit passierte. Wenn `mysqld` unerwartet stirbt und `safe_mysqld` ihn neu starten muss, schreibt `safe_mysqld` eine `restarted mysqld`-Zeile in diese Datei. Diese Log-Datei enthält auch Warnungen, wenn `mysqld` eine Tabelle bemerkt, die automatisch geprüft oder repariert werden muss.

Auf manchen Betriebssystemen enthält die Fehler-Log-Datei einen Stack-Trace, wo `mysqld` starb. Dieser kann benutzt werden, um herauszufinden, wo `mysqld` starb. See Abschnitt E.1.4, „Einen Stack-Trace benutzen“.

5.9.2. Die allgemeine Anfragen-Log-Datei

Wenn Sie wissen wollen, was innerhalb `mysqld` geschieht, sollten Sie ihn mit `--log[=file]` starten. Diese Option loggt alle Verbindungen und Anfragen in die Log-Datei (vorgabemäßig `'hostname'.log` benannt). Diese Log-Datei kann sehr nützlich sein, wenn Sie einen Fehler in einem Client vermuten und wissen wollen, was genau `mysqld` sich bei dem dachte, was es vom Client geschickt bekam.

Vorgabemäßig startet das `mysql.server`-Skript den MySQL-Server mit der `-l`-Option. Wenn Sie bessere Performance brauchen, wenn Sie MySQL in einer Produktionsumgebung starten, können Sie die `-l`-Option aus `mysql.server` entfernen oder sie zu `--log-binary` ändern.

Die Einträge in diese Log-Datei werden geschrieben, wenn `mysqld` die Anfragen erhält. Die Reihenfolge kann vor derjenigen abweichen, in der die Statements ausgeführt werden. Das steht im Gegensatz zur Update-Log-Datei und zur Binär-Log-Datei, bei denen geschrieben wird, nachdem die Anfrage ausgeführt wurde, aber bevor irgend welche Sperren aufgehoben werden.

5.9.3. Die Update-Log-Datei

HINWEIS: Die Update-Log-Datei wird durch die binäre Log-Datei ersetzt. See Abschnitt 5.9.4, „Die binäre Update-Log-Datei“. Mit dieser können Sie alles machen, was Sie mit der Update-Log-Datei machen können.

Wenn er mit der `--log-update[=datei]`-Option gestartet wird, schreibt `mysqld` eine Log-Datei, die alle SQL-Befehle enthält, die Daten aktualisieren. Wenn kein Dateiname angegeben wird, ist die Vorgabe der Name der Host-Maschine. Wenn ein Dateiname angegeben wird, der aber keine Pfadangabe enthält, wird die Datei ins Daten-Verzeichnis geschrieben. Wenn `datei`

keine Erweiterung hat, erzeugt `mysqld` eine Log-Datei, die er wie folgt benennt: `datei.###`, wobei `###` eine Zahl ist, die jedes Mal hochgezählt wird, wenn Sie `mysqladmin refresh` oder `mysqladmin flush-logs` oder das `FLUSH LOGS`-Statement ausführen, oder wenn Sie den Server neu starten.

HINWEIS: Damit das dargestellte Schema funktioniert, sollten Sie NICHT eigene Dateien mit demselben Dateinamen wie die Update-Log-Datei plus Erweiterungen, die als die hochgezählte Zahl betrachtet werden könnten, im Verzeichnis anlegen, das von der Update-Log-Datei benutzt wird!

Wenn Sie die `--log` oder `-l`-Optionen benutzen, schreibt `mysqld` eine allgemeine Log-Datei mit dem Dateinamen `hostname.log`. Neustarts und Refresh-Operationen führen dann nicht dazu, dass eine neue Log-Datei erzeugt wird (obwohl diese geschlossen und wieder geöffnet wird). In diesem Fall können Sie sie (unter Unix) wie folgt kopieren:

```
mv hostname.log hostname-old.log
mysqladmin flush-logs
cp hostname-old.log ins-datensicherungs-verzeichnis
rm hostname-old.log
```

Das Mitloggen mittels der Update-Log-Datei ist clever, weil es nur Statements loggt, die tatsächlich Daten aktualisieren. Wenn ein `UPDATE` oder ein `DELETE` mit einem `WHERE` keine passenden Zeilen findet, wird nichts in die Log-Datei geschrieben. Es werden sogar `UPDATE`-Statements übersprungen, die eine Spalte auf einen Wert setzen, die sie bereits hat.

Das Schreiben in die Update-Log-Datei wird unmittelbar durchgeführt, nachdem eine Anfrage fertig ist, aber bevor irgend welche Sperren aufgehoben sind oder irgendein Commit durchgeführt wurde. Das stellt sicher, dass die Log-Datei stets in der Reihenfolge der Ausführung mitschreibt.

Wenn Sie eine Datenbank von Update-Log-Datei-Dateien aktualisieren wollen, könnten Sie folgendes tun (angenommen, Ihre Update-Log-Dateien haben Namen der Form `datei.###`):

```
shell> ls -1 -t -r datei.[0-9]* | xargs cat | mysql
```

`ls` wird benutzt, um alle Log-Dateien in der richtigen Reihenfolge zu erhalten.

Das ist nützlich, wenn Sie Datensicherungsdateien nach einem Absturz zurückspielen müssen und die Aktualisierungen neu ausführen wollen, die zwischen der Zeit der Datensicherung und dem Absturz lagen.

5.9.4. Die binäre Update-Log-Datei

In Zukunft wird die Binär-Log-Datei die Update-Log-Datei ersetzen, daher empfehlen wir, dass Sie so bald wie möglich zu diesem Log-Format wechseln!

Die Binär-Log-Datei enthält alle Informationen, die im Update-Log verfügbar sind, in einem effizienteren Format. Sie enthält ausserdem Informationen darüber, wie lange jede Anfrage brauchte, die die Datenbank aktualisierte.

Die Binär-Log-Datei wird auch benutzt, wenn Sie einen Slave von einem Master replizieren. See Abschnitt 5.10, „Replikation bei MySQL“.

Mit der `--log-bin[=datei]`-Option gestartet, schreibt `mysqld` eine Log-Datei, die alle SQL-Befehle enthält, die Daten aktualisieren. Wenn kein Dateiname angegeben wird, ist die Vorgabe der Name der Host-Machine, gefolgt von `-bin`. Wenn der Dateiname angegeben wird, aber keine Pfadangabe enthält, wird die Datei ins Daten-Verzeichnis geschrieben.

Sie können folgende Optionen für `mysqld` benutzen, um zu beeinflussen, was in die Binär-Log-Datei geschrieben wird:

`binlog-do-db=datenbank`	Weist den Master an, Aktualisierungen für die angegebene Datenbank zu loggen und alle anderen, nicht explizit erwähnten, auszuschließen. (Beispiel: `binlog-do-db=eine_datenbank`)
`binlog-ignore-db=datenbank`	Weist den Master an, Aktualisierungen für die angegebene Datenbank nicht in die Binär-Log-Datei zu loggen (Beispiel: `binlog-ignore-db=eine_datenbank`).

`mysqld` hängt dem Binär-Log-Datei-Dateinamen eine Erweiterung an, die eine Zahl ist, die jedes Mal heraufgezählt wird, wenn Sie `mysqladmin refresh`, `mysqladmin flush-logs` oder ein `FLUSH LOGS`-Statement ausführen oder den Server neu starten.

Damit Sie feststellen können, welche verschiedenen Binär-Log-Datei-Dateien benutzt wurden, erzeugt `mysqld` auch eine Binär-Log-Index-Datei, die die Namen aller benutzten Binär-Log-Datei-Dateien enthält. Vorgabemäßig hat diese denselben Namen wie die Binär-Log-Datei, mit der Erweiterung `'.index'`. Sie können den Namen der Binär-Log-Index-Datei mit der `--log-bin-index=[filename]`-Option ändern.

Wenn Sie Replikation benutzen, sollten Sie keine alten Binär-Log-Dateien löschen, bis Sie sicher sind, dass kein Slave sie jemals

wieder benötigen wird. Eine Art, das zu tun, ist, einmal pro Tag `mysqladmin flush-logs` auszuführen, und danach alle Logs zu entfernen, die älter als 3 Tage sind.

Sie können die Binär-Log-Datei mit dem `mysqlbinlog`-Befehl untersuchen. Beispielsweise können Sie einen MySQL-Server wie folgt aus der Binär-Log-Datei aktualisieren:

```
mysqlbinlog log-file | mysql -h server_name
```

Sie können auch das `mysqlbinlog`-Programm benutzen, um die Binär-Log-Datei direkt von einem entfernten MySQL-Server zu lesen!

`mysqlbinlog --help` gibt Ihnen weitere Informationen zur Benutzung dieses Programms.

Wenn Sie `BEGIN [WORK]` oder `SET AUTOCOMMIT=0` verwenden, müssen Sie die MySQL-Binär-Log-Datei für Datensicherungen anstelle der alten Update-Log-Datei benutzen.

Das Loggen in die Binär-Log-Datei wird unmittelbar nach jeder Anfrage geschrieben, aber bevor irgend welche Sperren aufgehoben wurden oder irgend ein Commit durchgeführt wurde. Das stellt sicher, dass die Log-Datei in der Reihenfolge der Ausführung mitschreibt.

Alle Aktualisierungen (`UPDATE`, `DELETE` oder `INSERT`), die eine transaktionale Tabelle (like BDB-Tabellen) ändern, werden bis zu einem `COMMIT` gecachet. Jegliche Aktualisierungen auf eine nicht transaktionale Tabelle werden sofort in der Binär-Log-Datei gespeichert. Jedem Thread wird beim Start ein Puffer der Größe `binlog_cache_size` für die Pufferung von Anfragen zugewiesen. Wenn eine Anfrage größer als dieser ist, öffnet der Thread eine temporäre Datei, um den größeren Cache zu handhaben. Die temporäre Datei wird gelöscht, wenn der Thread beendet wird.

`max_binlog_cache_size` kann dazu benutzt werden, um die gesamte benutzte Größe zu begrenzen, und um eine Anfrage aus mehreren Transaktionen zu cachen.

Wenn Sie die Update- oder Binär-Log-Datei benutzen, funktionieren gleichzeitige Einfügeoperationen nicht im Zusammenhang mit `CREATE ... INSERT` und `INSERT ... SELECT`. Damit stellen Sie sicher, dass Sie eine exakte Kopie Ihrer Tabellen wieder herstellen können, indem Sie die Log-Datei auf eine Datensicherung anwenden.

5.9.5. Die Anfragen-Log-Datei für langsame Anfragen

Mit der `--log-slow-queries[=datei]`-Option gestartet schreibt `mysqld` eine Log-Datei, die alle SQL-Befehle enthält, die länger als `long_query_time` zur Ausführung brauchten. Die Zeit, um die anfänglichen Tabellensperren zu erhalten, wird nicht zur Ausführungszeit hinzugezählt.

Anfragen-Log-Datei für langsame Anfragen wird geschrieben, nachdem jede Anfrage ausgeführt wurde und nachdem alle Sperren aufgehoben wurden. Das kann von der Reihenfolge abweichen, in der die Statements ausgeführt wurden.

Wenn kein Dateiname angegeben wird, ist die Vorgabe der Name der Host-Maschine mit dem Suffix `-slow.log`. Wenn ein Dateiname angegeben wird, der aber keine Pfadangabe enthält, wird die Datei ins Daten-Verzeichnis geschrieben.

Die Anfragen-Log-Datei für langsame Anfragen kann benutzt werden, um Anfragen zu finden, die für die Ausführung lange Zeit benötigen und daher Kandidaten für Optimierungen sind, was bei einer großen Log-Datei allerdings eine schwierige Aufgabe werden kann. Sie können die Anfragen-Log-Datei für langsame Anfragen durch den `mysqldumpslow`-Befehl durchschleifen (pipen), um eine Zusammenfassung der Anfragen zu erhalten, die in der Log-Datei erscheinen.

Wenn Sie `--log-long-format` benutzen, erscheinen auch Anfragen, die keine Indexe benutzen. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

5.9.6. Wartung und Pflege der Log-Dateien

MySQL hat viele Log-Dateien, die es leicht machen festzustellen, was vor sich geht. See Abschnitt 5.9, „Die MySQL-Log-Dateien“. Von Zeit zu Zeit jedoch muss man hinter `MySQL` saubermachen, damit die Log-Dateien nicht zu viel Festplattenplatz in Anspruch nehmen.

Wenn Sie MySQL mit Log-Dateien benutzen, werden Sie von Zeit zu Zeit alte Log-Dateien entfernen wollen und MySQL mitteilen, in neue Dateien zu loggen. See Abschnitt 5.4.1, „Datenbank-Datensicherungen“.

Bei einer Linux-(`RedHat`)-Installation können Sie hierfür das `mysql-log-rotate`-Skript benutzen. Wenn Sie MySQL von einer RPM-Distribution installiert haben, sollte das Skript automatisch installiert worden sein. Beachten Sie, dass Sie damit vorsichtig umgehen sollten, wenn Sie die Log-Datei für Replikation benutzen!

Auf anderen Systemen müssen Sie selbst ein kurzes Skript installieren, dass Sie von `cron` starten können, um Log-Dateien zu handhaben.

Sie können MySQL zwingen, mit neuen Log-Dateien zu starten, indem Sie `mysqladmin flush-logs` oder den SQL-Befehl `FLUSH LOGS` benutzen. Wenn Sie MySQL-Version 3.21 benutzen, müssen Sie `mysqladmin refresh` benutzen.

Der obige Befehl macht folgendes:

- Wenn standardmäßiges Loggen (`--log`) oder Loggen langsamer Anfragen (`--log-slow-queries`) benutzt wird, wird die Log-Datei geschlossen und wieder geöffnet (`mysql.log` und `` `hostname`-slow.log `` als Vorgabe).
- Wenn Update-Logging (`--log-update`) benutzt wird, wird die Update-Log-Datei geschlossen und eine neue Log-Datei mit einer höheren Log-Zahl geöffnet.

Wenn Sie nur eine Update-Log-Datei benutzen, müssen Sie die Log-Dateien nur auf Platte zurückschreiben (flush) und dann die alten Update-Log-Datei-Dateien zu einer Datensicherungsdatei verschieben. Wenn Sie normales Loggen benutzen, können Sie etwas wie das Folgende tun:

```
shell> cd mysql-data-verzeichnis
shell> mv mysql.log mysql.old
shell> mysqladmin flush-logs
```

Und dann eine Datensicherung nehmen und `mysql.old` entfernen.

5.10. Replikation bei MySQL

Dieses Kapitel beschreibt die verschiedenen Replikationsfeatures in MySQL. Es dient als Referenz für die Optionen, die bei Replikation verfügbar sind. Sie erhalten eine Einführung in die Replikation und lernen, wie Sie sie implementieren. Am Ende des Kapitels werden einige häufige gestellte Fragen und die dazugehörigen Antworten aufgelistet sowie Beschreibungen der Probleme und wie man sie löst.

5.10.1. Einführung in die Replikation

Einweg-Replikation wird benutzt, um sowohl Stabilität als auch Geschwindigkeit zu steigern. Was Stabilität betrifft, haben Sie zwei Möglichkeiten und können zur Möglichkeit der Datensicherung zurückkehren, wenn Sie Probleme mit dem Master haben. Die Geschwindigkeitssteigerung wird dadurch erreicht, dass ein Teil der Anfragen, die nichts aktualisieren, an den Replikationsserver geschickt werden. Das funktioniert naturgemäß nur dann, wenn Anfragen, die nichts aktualisieren, überwiegen, aber das ist der Normalfall.

Ab Version 3.23.15 unterstützt MySQL intern Einweg-Replikation. Ein Server agiert als Master, der andere als Slave. Beachten Sie, dass ein Server beide Rolle - als Master und als Slave - in einem Paar spielen kann. Der Master hält eine Binär-Log-Datei der Aktualisierungen vor (see Abschnitt 5.9.4, „Die binäre Update-Log-Datei“) sowie eine Index-Datei für Binär-Log-Dateien, um hinsichtlich der Log-Rotation auf dem Laufenden zu bleiben. Der Slave informiert den Master beim Verbinden darüber, wo er seit der letzten erfolgreich durchgeführten Aktualisierung aufgehört hat, schließt zu den Aktualisierungen auf, blockiert danach und wartet darauf, dass ihn der Master über neue Aktualisierungen informiert.

Beachten Sie, dass alle Aktualisierungen auf eine Datenbank, die repliziert wird, durch den Master durchgeführt werden sollten!

Ein weiterer Vorteil von Replikation ist, dass man permanente (live) Datensicherungen vom System erhält, wenn man die Datensicherung auf dem Slave durchführt statt auf dem Master. See Abschnitt 5.4.1, „Datenbank-Datensicherungen“.

5.10.2. Replikationsimplementation

MySQL-Replikation basiert darauf, dass der Server alle Änderungen Ihrer Datenbank im Binär-Log verfolgt (Updates, Deletes usw.) (see Abschnitt 5.9.4, „Die binäre Update-Log-Datei“) und der oder die Slave-Server die gespeicherten Anfragen aus der Binär-Log-Datei des Masters lesen, so dass der Slave dieselben Anfragen auf seine Kopie der Daten ausführen kann.

Es ist **sehr wichtig** sich klarzumachen, dass die Binär-Log-Datei schlicht eine Aufzeichnung ist, die ab einem festen Zeitpunkt an startet (ab dem Moment, wo Sie Binär-Loggen starten). Alle Slaves, die Sie aufsetzen, benötigen Kopien aller Daten vom Master, wie Sie zu dem Zeitpunkt existierten, als Binär-Loggen auf dem Master aktiviert wurde. Wenn Sie Ihre Slaves mit Daten starten, die nicht mit dem übereinstimmen, was auf dem Master war, **als die Binär-Log-Datei gestartet wurde**, funktionieren Ihre Slaves womöglich nicht richtig.

Eine zukünftige Version (4.0) von MySQL wird die Notwendigkeit beseitigen, (eventuell große) Schnappschüsse von Daten für neue Slaves vorzuhalten, die Sie über die Live-Datensicherungs-Funktionalität aufsetzen wollen, wobei kein Sperren (Locking) erforderlich ist. Zu dieser Zeit ist es jedoch notwendig, alle Schreibzugriffe entweder mit einer globalen Lese-Sperre oder durch das Herunterfahren des Masters zu blockieren, während man einen Schnappschuss anlegt.

Sobald ein Slave korrekt konfiguriert ist und läuft, verbindet er sich einfach mit dem Master und wartet darauf, dass Aktualisierung ausgeführt werden. Wenn der Master abgeschaltet wird oder der Slave die Verbindung zum Master verliert, versucht er alle

`master-connect-retry` Sekunden, sich neu zu verbinden, bis er sich neu verbinden kann, und nimmt dann das Warten auf Aktualisierungen wieder auf.

Jeder Slave achtet darauf, wo er aufgehört hat. Der Master-Server weiß nicht, wie viele Slaves es gibt oder welche zu einem gegebenen Zeitpunkt auf aktuellem Stand sind.

Der nächste Abschnitt erläutert den Master-Slave-Einrichtungsprozess detaillierter.

5.10.3. Wie man Replikation aufsetzt

Unten findet sich eine kurze Beschreibung, wie Sie komplette Replikation auf Ihrem aktuellen MySQL-Server einrichten können. Es wird angenommen, dass Sie alle Ihre Datenbanken replizieren wollen und bislang Replikation noch nicht konfiguriert haben. Sie müssen Ihren Master-Server kurz herunter fahren, um die unten stehenden Schritte fertigzustellen.

1. Stellen Sie sicher, dass Sie eine aktuelle Version von MySQL auf dem Master und dem Slave oder den Slaves haben.

   Benutzen Sie Version 3.23.29 oder höher. Vorherige Releases benutzten ein anderes Binär-Log-Format und hatten Bugs, die in neueren Releases behoben wurden. Bitte berichten Sie keine Bugs, bevor Sie bestätigen können, dass das Problem im neuesten Release beobachtet werden kann.

2. Richten Sie einen speziellen Replikationsbenutzer auf dem Master mit der `FILE`-Berechtigung und Berechtigungen, sich von allen Slaves aus zu verbinden, ein. Wenn der Benutzer ausschließlich Replikation durchführt (was empfohlen wird), müssen Sie ihm keine zusätzlichen Berechtigungen geben.

   Erzeugen Sie zum Beispiel einen Benutzer namens `repl`, der auf Ihren Master von jedem Host aus zugreifen kann, mit folgendem Befehl:

   ```
   GRANT FILE ON *.* TO repl@"%" IDENTIFIED BY 'passwort';
   ```

3. Fahren Sie den MySQL-Master herunter:

   ```
   mysqladmin -u root -ppasswort shutdown
   ```

4. Machen Sie einen Schnappschuss aller Daten auf Ihrem Master-Server.

   Die einfachste Art, das (unter Unix) zu tun, ist, einfach **tar** zu benutzen, um ein Archiv Ihre gesamten Daten-Verzeichnisses zu erzeugen. Der genaue Speicherort Ihres Daten-Verzeichnisses hängt von Ihrer Installation ab.

   ```
   tar -cvf /tmp/mysql-snapshot.tar /pfad/zu/data-dir
   ```

   Windows-Benutzer können WinZip oder ähnliche Software benutzen, um ein Archiv des Daten-Verzeichnisses anzulegen.

5. In der Datei `my.cnf` für den Master fügen Sie `log-bin` und `server-id=eindeutige_nummer` zum `[mysqld]`-Abschnitt und hinzu und starten Sie den Server neu. Es ist sehr wichtig, dass die ID auf dem Slave sich von der ID auf dem Master unterscheidet. Denken Sie sich `server-id` als etwas, dass einer IP-Adresse ähnlich ist - es identifiziert in der Gemeinschaft der Replikationspartner die Server-Instanz eindeutig.

   ```
   [mysqld]
   log-bin
   server-id=1
   ```

6. Starten Sie den MySQL-Master neu.

7. Fügen Sie auf dem Slave oder den Slaves folgendes zur Datei `my.cnf` hinzu:

   ```
   master-host=hostname_des_masters
   master-user=replikations_benutzername
   master-password=replikations_benutzerpasswort
   master-port=TCP/IP-Port_für_master>
   server-id=eine_eindeutige_nummer_zwischen_2_und_2^32-1
   ```

   Ersetzen Sie die Beispielwerte durch etwas, was für Ihr System stimmig ist.

   `server-id` muss für jeden Partner, der an Replikation teilnimmt, unterschiedlich sein. Wenn Sie keine server-id angeben, wird sie auf 1 gesetzt, falls Sie `master-host` nicht definiert haben, ansonsten wird sie auf 2 gesetzt. Beachten Sie für den Fall, dass sie `server-id` weglassen, dass der Master Verbindungen von allen Slaves verweigert und die Slaves verweigern werden, sich mit dem Master zu verbinden. Daher ist das Weglassen der `server-id` nur dann eine gute Idee, wenn Sie es nur für Datensicherungen mit einer Binär-Log-Datei verwenden.

8. Kopieren Sie die Schnappschuss-Daten in Ihr Daten-Verzeichnis auf Ihrem Slave oder Ihren Slaves. Stellen Sie sicher, dass

die Berechtigungen auf die Dateien und Verzeichnisse korrekt sind. Der Benutzer, unter dem MySQL läuft, muss in der Lage sein, sie zu lesen und zu schreiben, genau wie auf dem Master.

9. Starten Sie den Slave oder die Slaves neu.

Nachdem Sie das Obige durchgeführt haben, sollten sich die Slaves mit dem Master verbinden können und alle Aktualisierungen mitbekommen, die nach der Aufnahme des Schnappschusses passieren.

Wenn Sie vergessen haben, die `server-id` für den Slave zu setzen, erhalten Sie folgenden Fehler in der Fehler-Log-Datei:

```
Warning: one should set server_id to a non-0 value if master_host ist set.
The server will not act as a slave.
```

Wenn Sie vergessen haben, selbiges für den Master zu machen, sind die Slaves nicht in der Lage, sich mit dem Master zu verbinden.

Wenn ein Slave aus irgend welchen Gründen nicht in der Lage ist zu replizieren, finden Sie Fehlermeldungen in der Fehler-Log-Datei auf dem Slave.

Sobald ein Slave repliziert, finden Sie eine Datei namens `master.info` im selben Verzeichnis, wo auch Ihre Fehler-Log-Datei liegt. Die `master.info`-Datei wird vom Slave benutzt, um auf dem Laufenden zu bleiben, wie viel der Binär-Log-Datei des Masters er bereits abgearbeitet hat. Sie sollten die Datei **NICHT** entfernen oder editieren, es sei denn, Sie wissen genau, was Sie tun. Selbst in diesem Fall sollten Sie vorzugsweise den `CHANGE MASTER TO`-Befehl benutzen.

5.10.4. Replikationsfeatures und bekannte Probleme

Unten steht eine Erläuterung dessen, was unterstützt wird und was nicht:

- Replikation läuft korrekt mit `AUTO_INCREMENT`-, `LAST_INSERT_ID`- und `TIMESTAMP`-Werten.
- `RAND()` bei Updates repliziert nicht korrekt. Benutzen Sie `RAND(ein_nicht_zufalls_ausdruck)`, wenn Sie Updates mit `RAND()` replizieren. Sie können zum Beispiel `UNIX_TIMESTAMP()` als Argument für `RAND()` benutzen.
- Sie müssen auf Master und Slave denselben Zeichensatz (`--default-character-set`) benutzen. Wenn nicht, erhalten Sie eventuell Fehler wegen doppelter Schlüsseleinträge (duplicate key) auf dem Slave, weil ein Schlüssel, der auf dem Master als eindeutig betrachtet wird, das in einem anderen Zeichensatz eventuell nicht ist.
- `LOAD DATA INFILE` wird korrekt gehandhabt, solange die Datei zur Zeit der Update-Ausführung noch auf dem Master-Server liegt. `LOAD LOCAL DATA INFILE` wird übersprungen.
- Aktualisierungsanfragen, die Benutzer-Variablen benutzen, sind (noch) nicht replikationssicher.
- `FLUSH`-Befehle werden nicht in der Binär-Log-Datei gespeichert und werden deswegen nicht auf den Slaves repliziert. Das stellt normalerweise kein Problem dar, weil `FLUSH` nichts ändert. In Bezug auf die `MySQL`-Berechtigungstabellen heißt das jedoch, dass Sie bei direkten Änderungen in diesen Tabellen ohne Benutzung des `GRANT`-Statements und der anschließenden Replikation der `MySQL`-Berechtigungs-Datenbank auf den Slaves `FLUSH PRIVILEGES` ausführen müssen, damit die neuen Berechtigungen wirksam werden.
- Temporäre Tabellen werden ab Version 3.23.29 korrekt repliziert, ausgenommen im Fall, dass Sie den Slave-Server schließen (nicht nur den Slave-Thread), Sie noch einige temporäre Tabellen offen haben und diese bei nachfolgenden Aktualisierungen benutzt werden. Um mit diesem Problem fertig zu werden, schließen Sie den Slave mit `SLAVE STOP` und prüfen dann die `Slave_open_temp_tables`-Variable, um zu sehen, ob Sie 0 ist. Dann führen Sie `mysqladmin shutdown` aus. Wenn die Variable nicht 0 ist, starten Sie den Slave-Thread neu mit `SLAVE START` und probieren es noch einmal. Zukünftig (ab Version 4.0) wird es eine sauberere Lösung geben. In früheren Versionen wurden temporäre Tabellen nicht korrekt repliziert. Wir empfehlen, dass Sie entweder auf eine neuere Version aktualisieren oder vor allen Anfragen mit temporären Tabellen `SET SQL_LOG_BIN=0` auf alle Clients ausführen.
- MySQL unterstütz nur einen Master und viele Slaves. In Version 4.x wird ein Abstimmungsalgorithmus eingebaut, der automatisch den Master umschaltet, wenn etwas mit dem aktuellen Master schief geht. Ausserdem werden wir 'Agenten'-Prozesse einführen, die bei der Lastverteilung helfen, indem sie SELECT-Anfragen an verschiedene Slaves senden.
- Ab Version 3.23.26 ist es sicher, Server in einer zirkulären Master-Slave-Beziehung mit angeschaltetem `log-slave-updates` zu verbinden. Beachten Sie jedoch, dass bei dieser Art von Einrichtung viele Anfrage nicht richtig funktionieren, es sei denn, Ihr Client-Code ist so geschrieben, dass er sich um mögliche Probleme kümmert, die durch Aktualisierungen auftreten können, die in unterschiedlicher Reihenfolge auf verschiedenen Servern laufen.

  Das bedeutet, dass Sie eine Einrichtung wie die folgende machen können:

  ```
  A -> B -> C -> A
  ```

Diese Einrichtung funktioniert nur dann, wenn Sie ausschließlich Aktualisierungen ausführen, die nicht zwischen den Tabellen zu Konflikten führen. Mit anderen Worten, wenn Sie Daten in A und C einfügen, sollten Sie nie eine Zeile in A einfügen, die zu einem Konflikt mit einem Schlüsselwert bei einem Zeilen-Einfügevorgang in C führt. Ebenfalls sollte Sie nie dieselben Zeilen auf zwei Servern einfügen, wenn die Reihenfolge, in der die Aktualisierungen durchgeführt werden, eine Rolle spielt.

Beachten Sie, dass sich das Log-Format in Version 3.23.26 geändert hat, so das Slaves vor Version 3.23.26 diese nicht lesen können.

- Wenn die Anfrage auf dem Slave zu einem Fehler führt, beendet sich der Slave-Thread und in der `.err`-Datei erscheint eine Meldung. Sie sollten sich dann manuell mit dem Slave verbinden, die Ursache des Fehlers beheben (zum Beispiel nicht existierende Tabellen) und dann den SQL-Befehl `SLAVE START` laufen lassen (verfügbar ab Version 3.23.16). In Version 3.23.15 müssen Sie den Server neu starten.
- Wenn die Verbindung zum Master verloren geht, versucht der Slave unmittelbar, sich neu zu verbinden, und wenn das fehlschlägt, alle `master-connect-retry` Sekunden (Vorgabe 60 Sekunden). Deswegen ist es sicher, den Master herunter zu fahren und dann nach einer Weile wieder hochzufahren. Der Slave ist auch in der Lage, mit Netzwerk-Verbindungsausfällen umzugehen.
- Den Slave (sauber) herunterzufahren ist ebenfalls sicher, weil er sich merkt, wo er aufgehört hat. Unsauberes Herunterfahren kann zu Problemen führen, insbesondere dann, wenn der Platten-Cache nicht synchronisiert wurde, als das System starb. Die Fehlertoleranz Ihres Systems wird stark verbessert, wenn Sie ein gutes UPS haben.
- Wenn der Master auf einem Port auf Anfragen wartet, der nicht Standard ist, müssen Sie diesen mit dem `master-port`-Parameter in `my.cnf` angeben.
- In Version 3.23.15 werden alle Tabellen und Datenbanken repliziert. Ab Version 3.23.16 können Sie die Replikation mit der `replicate-do-db`-Anweisung in `my.cnf` auf einen Satz von Datenbanken beschränken oder einen Satz von Datenbanken mit `replicate-ignore-db` ausschließen. Beachten Sie, dass es bis Version 3.23.23 einen Bug gab, so dass mit `LOAD DATA INFILE` nicht sauber umgegangen wurde, wenn Sie diesen Befehl in einer Datenbank ausführten, die von der Replikation ausgeschlossen war.
- Ab Version 3.23.16 schaltet `SET SQL_LOG_BIN = 0` Replikations-(Binär)-Loggen auf dem Master aus und `SET SQL_LOG_BIN = 1` schaltet es wieder an. Sie benötigen die process-Berechtigung, um das auszuführen.
- Ab Version 3.23.19, you can clean up stale Replikation leftovers when something goes wrong und you want a clean start mit `FLUSH MASTER` und `FLUSH SLAVE`-Befehle. In Version 3.23.26 we have renamed them to `RESET MASTER` und `RESET SLAVE` respectively to clarify what they do. The old `FLUSH` variants still work, though, for Kompatibilität.
- Ab Version 3.23.21, you can use `LOAD TABLE FROM MASTER` for network backup und to set up Replikation initially. We have recently received a Anzahl von bug reports concerning it that we are investigating, so we recommend that you use it only in testing until we make it mehr stable.
- Ab Version 3.23.23, you can change masters und adjust log position mit `CHANGE MASTER TO`.
- Ab Version 3.23.23, you tell the master that updates in certain Datenbanken should not be logged to the Binär-Log-Datei mit `binlog-ignore-db`.
- Ab Version 3.23.26 können Sie `replicate-rewrite-db` benutzen, um den Slave anzuweisen, Aktualisierungen einer Datenbank auf dem Master auf eine mit einem anderen Namen auf dem Slave anzuwenden.
- Ab Version 3.23.28 können Sie `PURGE MASTER LOGS TO 'log-name'` benutzen, um alte Log-Dateien loszuwerden, während der Slave läuft.

5.10.5. Replikationsoptionen in my.cnf

Wenn Sie Replikation benutzen, empfehlen wir, dass Sie MySQL-Version 3.23.30 oder höher benutzen. Ältere Versionen funktionieren, haben aber einige Bugs und fehlende Features.

Sowohl auf dem Master als auch auf dem Slave müssen Sie die `server-id`-Option benutzen. Diese setzt eine eindeutige Replikations-ID. Sie sollten einen eindeutigen Wert im Bereich zwischen 1 und 2^32-1 für jeden Master und Slave benutzen. Beispiel: `server-id=3`

In folgender Tabelle stehen die Optionen, die Sie für den **MASTER** benutzen können:

Option	Beschreibung
`log-bin=dateiname`	Schreibt in die binäre Update-Log-Datei am angegebenen Ort. Beachten Sie, dass, wenn

	Sie ihr einen Parameter mit einer Erweiterung angeben (zum Beispiel `log-bin=/mysql/logs/replikation.log`), Versionen bis zu 3.23.24 während der Replikation nicht richtig funktionieren, wen Sie `FLUSH LOGS` ausführen. Das Problem ist seit Version 3.23.25 behoben. Wenn Sie Log-Namen dieser Art benutzen, wird `FLUSH LOGS` auf dem binären Log ignoriert. Um das Log zu löschen, führen Sie `FLUSH MASTER` aus. Vergessen Sie dabei nicht, `FLUSH SLAVE` auf allen Slaves laufen zu lassen.Ab Version 3.23.26 sollten Sie `RESET MASTER` und `RESET SLAVE` benutzen.
`log-bin-index=dateiname`	Weil der Benutzer `FLUSH LOGS`-Befehle ausführen könnte, muss man wissen, welches Log momentan aktiv ist und welche in welcher Reihenfolge durch Log-Rotation herausgenommen wurden. Diese Informationen sind in der Binär-Log-Index-Datei gespeichert. Der Vorgabewert ist `hostname`.index. Beispiel: `log-bin-index=datenbank.index`.
`sql-bin-update-same`	Falls gesetzt, führt das Setzen von `SQL_LOG_BIN` auf einen Wert automatisch dazu, dass `SQL_LOG_UPDATE` auf denselben Wert gesetzt wird, und umgekehrt.
`binlog-do-db=datenbank`	Weist den Master an, Aktualisierung in die Binär-Log-Datei zu loggen, wenn die aktuelle Datenbank 'datenbank' ist. Alle anderen Datenbanken werden ignoriert. Beachten Sie bei der Benutzung, dass Sie sicherstellen sollten, dass Sie Aktualisierungen nur in der aktuellen Datenbank ausführen. Beispiel: `binlog-do-db=eine_datenbank`.
`binlog-ignore-db=datenbank`	Weist den Master an, das Aktualisierung der aktuellen Datenbank 'datenbank' nicht in der Binär-Log-Datei gespeichert werden sollen. ignoriert. Beachten Sie bei der Benutzung, dass Sie sicherstellen sollten, dass Sie Aktualisierungen nur in der aktuellen Datenbank ausführen. Beispiel: `binlog-ignore-db=eine_datenbank`

Folgende Tabelle enthält die Optionen, die Sie für **SLAVE** benutzen können:

Option	**Beschreibung**
`master-host=host`	Hostname des Masters oder IP-Adresse für Replikation. Falls nicht gesetzt, startet der Slave-Thread nicht. Beispiel: `master-host=datenbank-master.meinefirma.de`.
`master-user=benutzername`	Der Benutzer, den der Slave-Thread für Authentifizierung benutzt, wenn er sich mit dem Master verbindet. Der Benutzer muss die `FILE`-Berechtigung besitzen. Wenn der Master-Benutzer nicht gesetzt ist, wird Benutzer `test` angenommen. Beispiel: `master-user=steve`.
`master-password=passwort`	Das Passwort, das der Slave-Thread für Authentifizierung benutzt, wenn er sich mit dem Master verbindet. Wenn nicht gesetzt, wird ein leeres Passwort angenommen. Beispiel: `master-password=hund`.
`master-port=portnummer`	Der Port, auf dem der Master auf Verbindungen wartet. Wenn nicht gesetzt, wird die kompilierte Einstellung von `MYSQL_PORT` angenommen. Wenn Sie nicht an den `configure`-Optionen gedreht haben, sollte das 3306 sein. Beispiel: `master-port=3306`.
`master-connect-retry=sekunden`	Die Anzahl Sekunden, die der Slave-Thread schläft, bevor er wiederum versucht, sich mit dem Master zu verbinden, falls der Master herunter fuhr oder die Verbindung verloren ging. Vorgabewert ist 60. Beispiel: `master-connect-retry=60`.
`master-ssl`	Schaltet SSL an. Beispiel: `master-ssl`.
`master-ssl-key`	Der Name der SSL-Schlüsseldatei für den Master. Beispiel: `master-ssl-key=SSL/master-key.pem`.
`master-ssl-cert`	Der Dateiname des SSL-Zertifikats für den Master. Beispiel: `master-ssl-key=SSL/master-cert.pem`.
`master-info-file=dateiname`	Der Speicherort der Datei, die sich merkt, bis wohin der Master während des Replikationsprozesses verfolgt wurde. Vorgabewert ist master.info im data- Verzeichnis. Beispiel: `master-info-file=master.info`.
`replicate-do-table=datenbank.tabelle`	Weist den Slave-Thread an, die Replikation auf die angegebene Tabelle zu beschränken. Um mehr als eine Tabelle anzugeben, benutzen Sie die Anweisung mehrfach, einmal für jede Tabelle. Das funktioniert auch bei Datenbank-übergreifenden Aktualisierungen, im Gegensatz zu `replicate-do-db`. Beispiel: `replicate-do-table=eine_datenbank.eine_tabelle`.
`replicate-ignore-table=datenbank.tabelle`	Weist den Slave-Thread an, die angegebene Tabelle nicht zu replizieren. Um mehr als eine Tabelle anzugeben, die ignoriert werden soll, geben Sie die Anweisung mehrfach ein, einmal für jede Tabelle. Das funktioniert auch bei Datenbank-übergreifenden Aktualisierungen, im Gegensatz zu `replicate-ignore-db`. Beispiel: `replicate-ignore-table=eine_datenbank.eine_tabelle`.

`replicate-wild-do-table=datenbank.tabelle`	Weist den Slave-Thread an, die Replikation auf Tabellen zu beschränken, die dem angegebenen Platzhalter-Muster entsprechen. Um mehr als ein Tabellenmuster anzugeben, das ignoriert werden soll, geben Sie die Anweisung mehrfach ein, einmal für jedes Tabellenmuster. Das funktioniert auch bei Datenbank-übergreifenden Aktualisierungen. Beispiel: `replicate-wild-do-table=foo%.bar%` repliziert nur Aktualisierungen auf Tabellen in allen Datenbanken, die mit 'foo' anfangen und deren Tabellennamen mit 'bar' beginnen.
`replicate-wild-ignore-table=datenbank.tabelle`	Weist den Slave-Thread an, Tabellen nicht zu replizieren, die dem angegebenen Platzhalter-Muster entsprechen. Um mehr als ein Tabellenmuster anzugeben, das ignoriert werden soll, geben Sie die Anweisung mehrfach ein, einmal für jedes Tabellenmuster. Das funktioniert auch bei Datenbank-übergreifenden Aktualisierungen. Beispiel: `replicate-wild-ignore-table=foo%.bar%` aktualisiert keine Tabellen in Datenbanken, die mit 'foo' anfangen und deren Tabellennamen mit 'bar' beginnen.
`replicate-ignore-db=datenbank`	Weist den Slave-Thread an, die angegebene Datenbank nicht zu replizieren. Um mehr als eine Datenbank anzugeben, die ignoriert werden soll, geben Sie die Anweisung mehrfach ein, einmal für jede Datenbank. Diese Option funktioniert nicht für Datenbank-übergreifende Aktualisierungen. Wenn Sie Datenbank-übergreifende Aktualisierungen brauchen, stellen Sie sicher, dass Sie Version 3.23.28 oder höher verwenden und benutzen Sie `replicate-wild-ignore-table=datenbank.%` Beispiel: `replicate-ignore-db=eine_datenbank`.
`replicate-do-db=datenbank`	Weist den Slave-Thread an, die Replikation auf die angegebene Datenbank zu beschränken. Um mehr als eine Datenbank anzugeben, benutzen Sie die Anweisung mehrfach, einmal für jede Datenbank. Beachten Sie, dass das nicht funktioniert, wenn Sie Datenbank-übergreifende Anfragen wie `UPDATE eine_datenbank.eine_tabelle SET foo='bar'` ausführen, während Sie eine andere oder keine Datenbank ausgewählt haben. Wenn Sie Datenbank-übergreifende Aktualisierungen brauchen, stellen Sie sicher, dass Sie Version 3.23.28 oder höher verwenden und benutzen Sie `replicate-wild-do-table=datenbank.%`. Beispiel: `replicate-do-db=eine_datenbank`.
`log-slave-updates`	Weist den Slave an, Aktualisierungen vom Slave-Thread in die binäre Log-Datei zu schreiben. Ist vorgabemäßig ausgeschaltet. Sie müssen diese Option anschalten, wenn Sie planen, die Slave in eine zirkuläre Kette zu hängen ('Daisy-Chain').
`replicate-rewrite-db=von_name->zu_name`	Aktualisierungen auf eine Datenbank mit einem anderen Namen als dem Orginalnamen. Beispiel: `replicate-rewrite-db=master_datenbank->slave_datenbank`.
`skip-slave-start`	Weist den Slave-Server an, den Slave nicht beim Hochfahren zu starten. Der Benutzer kann ihn später mit `SLAVE START` starten.
`slave_read_timeout=#`	Anzahl von Sekunden, die der Slave auf weitere Daten vom Master wartet, bevor er abbricht.

5.10.6. SQL-Befehle in Bezug auf Replikation

Replikation kann über die SQL-Schnittstelle gesteuert werden. Hier eine Zusammenfassung der Befehle:

Befehl	**Beschreibung**
`SLAVE START`	Startet den Slave-Thread. (Slave)
`SLAVE STOP`	Hält den Slave-Thread an. (Slave)
`SET SQL_LOG_BIN=0`	Schaltet das Loggen in die Update-Log-Datei aus, wenn der Benutzer die process-Berechtigung hat. Wird ansonsten ignoriert. (Master)
`SET SQL_LOG_BIN=1`	Schaltet das Loggen in die Update-Log-Datei wieder an, wenn der Benutzer die process-Berechtigung hat. Wird ansonsten ignoriert. (Master)
`SET SQL_SLAVE_SKIP_COUNTER=n`	Die nächsten `n` Ereignisse vom Master ignorieren. Gilt nur, wenn der Slave-Thread nicht läuft, gibt ansonsten einen Fehler aus. Nützlich für den Ausgleich von Replikationsabweichungen.
`RESET MASTER`	Löscht alle Binär-Log-Dateien, die in der Index-Datei aufgeführt sind, und setzt die BinärLog-Index-Datei auf leer zurück. In Versionen vor 3.23.26 versions heißt dieser Befehl `FLUSH MASTER`. (Master)
`RESET SLAVE`	Führt dazu, dass der Slave seine Replikationsposition in den Master-Logs vergisst. In Versionen vor 3.23.26 versions heißt dieser Befehl `FLUSH SLAVE`. (Slave)
`LOAD TABLE tabelle FROM MASTER`	Lädt eine Kopie der Tabelle vom Master auf den Slave. (Slave)

`CHANGE MASTER TO master_def_list`	Ändert die Master-Parameters auf den Wert, der in `master_def_list` angegeben ist, und startet den Slave-Thread neu. `master_def_list` ist eine durch Kommas getrennte Liste `master_def`, wobei `master_def` eins der folgenden Elemente ist: `MASTER_HOST`, `MASTER_USER`, `MASTER_PASSWORD`, `MASTER_PORT`, `MASTER_CONNECT_RETRY`, `MASTER_LOG_FILE` oder `MASTER_LOG_POS`. Beispiel:
	`CHANGE MASTER TO`
	`  MASTER_HOST='master2.meinefirma.com',`
	`  MASTER_USER='replikation',`
	`  MASTER_PASSWORD='gro33esgeheimnis',`
	`  MASTER_PORT=3306,`
	`  MASTER_LOG_FILE='master2-bin.001',`
	`  MASTER_LOG_POS=4;`
	Sie müssen nur die Werte angeben, die geändert werden sollen. Die Werte, die Sie auslassen, bleiben dieselben, ausser wenn Sie den Host oder den Port ändern. In diesem Fall nimmt der Slave an, dass der Master ein anderer ist, weil Sie sich zu einem anderen Host oder über einen anderen Port verbinden. Daher treffen die alten Werte von Log und Position nicht mehr zu und werden automatisch auf eine leere Zeichenkette bzw. auf 0 zurück gesetzt (dem Startwert). Beachten Sie, dass sich der Slave beim Neustart an seinen alten Master erinnert. Falls das nicht wünschenswert ist, sollten Sie die `master.info`-Datei löschen, bevor Sie neu starten. Der Slave liest dann seinen Master aus der Datei `my.cnf` oder von der Kommandozeile. (Slave)

`SHOW MASTER STATUS`	Stellt Statusinformationen über die Binär-Log-Datei des Masters zur Verfügung. (Master)
`SHOW SLAVE STATUS`	Stellt Statusinformationen über die wichtigsten Parameter des Slave-Threads zur Verfügung. (Slave)
`SHOW MASTER LOGS`	Nur verfügbar ab Version 3.23.28. Listet die Binär-Log-Dateien auf dem Master auf. Sie sollten diesen Befehl vor `PURGE MASTER LOGS TO` benutzen, um herauszufinden, wie weit Sie gehen sollten.
`PURGE MASTER LOGS TO 'logname'`	Verfügbar ab Version 3.23.28. Löscht alle Replikations-Logs, die in der Index-Log-Datei aufgeführt sind, die vor dem angegebenen Log liegen und entfernt Sie aus dem Log-Index, so dass die angegebene Log-Datei nunmehr die erste wird. Beispiel: `PURGE MASTER LOGS TO 'mysql-bin.010'` Dieser Befehl macht nichts und schlägt mit einer Fehlermeldung fehl, wenn Sie einen aktiven Slave haben, der momentan eine der Log-Dateien liest, die Sie zu löschen versuchen. Wenn Sie jedoch einen schlafenden Slave haben und gerade eine der Log-Dateien löschen, die dieser Slave lesen will, wird der Slave nicht in der Lage sein zu replizieren, sobald er wach wird. Der Befehl kann sicher verwendet werden, während Slaves replizieren - Sie brauchen diese also nicht anhalten. Zuerst müssen Sie alle Slaves mit `SHOW SLAVE STATUS` überprüfen, um festzustellen, an welcher Log-Datei sie gerade sind, dann eine Auflistung aller Log-Dateien auf dem Master mit `SHOW MASTER LOGS` machen, die früheste davon herausfinden, an der noch ein Slave arbeitet (wenn alle Slaves aktuell sind, ist das die letzte Log-Datei auf der Liste), dann alle Logs, die Sie löschen wollen, sichern (optional), und schließlich bis zum Ziel-Log löschen.

5.10.7. Replikation - Häufig gestellte Fragen

Frage: Warum sehe ich manchmal mehr als eine `Binlog_Dump`-Thread auf dem Master, nachdem ich den Slave neu gestartet habe?

Antwort: `Binlog_Dump` ist ein kontinuierlicher Prozess, der folgendermaßen vom Server gehandhabt wird:

- Zu den Aktualisierungen aufschließen.
- Sobald keine Aktualisierungen mehr übrig sind, in den Zustand `pThread_cond_wait()` gehen, durch den er entweder durch eine Aktualisierung oder einen Kill erweckt werden kann.
- Beim Aufwachen den Grund dafür prüfen. Wenn er nicht sterben soll, mit der `Binlog_dump`-Schleife weitermachen.
- Wenn ein schwerer Fehler auftritt, wenn zum Beispiel ein toter Client entdeckt wird, die Schleife beenden.

Wenn der Slave-Thread also beim Slave anhält, bemerkt das der entsprechende `Binlog_Dump`-Thread auf dem Master solange nicht, bis zumindest eine Aktualisierung (oder ein Kill) auf den Master durchgeführt wird, was benötigt wird, um ihn von `pThread_cond_wait()` aufzuwecken. In der Zwischenzeit könnte der Slave bereits eine weitere Verbindung geöffnet haben,

die in einem weiteren `Binlog_Dump`-Thread resultiert.

Das beschriebene Problem sollten in Versionen ab 3.23.26 nicht auftreten. In Version 3.23.26 kam `server-id` für jeden Replikationsserver hinzu, und nun werden alle alten Zombie-Threads auf dem Master gekillt, wenn ein neuer Replikations-Thread sich vom selben Slave aus verbindet.

Frage: Wie rotiere ich Replikations-Logs?

Antwort: In Version 3.23.28 sollten Sie den `PURGE MASTER LOGS TO`-Befehl benutzen, nachdem festgestellt wurde, welche Logs gelöscht werden können und nachdem sie optional gesichert wurden. In früheren Versionen ist der Prozess sehr viel anstrengender und kann nicht sicher durchgeführt werden, ohne alle Slaves anzuhalten, falls Sie planen, Log-Namen wiederholt zu verwenden. Sie müssen die Slave-Threads anhalten, die Binär-Log-Index-Datei editieren, alle alten Logs löschen, den Master neu starten, die Slave-Threads neu starten und dann die alten Log-Dateien entfernen.

Frage: Wie aktualisiere ich bei einer laufenden Replikationseinrichtung?

Antwort: Wenn Sie vor Version 3.23.26 aktualisieren, sollten Sie nur die Master-Tabellen sperren, warten, bis die Slaves auf aktuellem Stand sind, und dann `FLUSH MASTER` auf dem Master und `FLUSH SLAVE` auf dem Slave laufen lassen, um die Logs zurückzusetzen, und danach neue Versionen des Masters und des Slaves neu starten. Beachten Sie, dass der Slave für einige Zeit heruntergefahren bleiben kann - weil der Master alle Aktualisierung loggt, wird der Slave in der Lage sein, auf den aktuellen Stand zu kommen, sobald er hoch gefahren ist und sich verbinden kann.

Nach Version 3.23.26 wurde das Replikationsprotokoll für Änderungen gesperrt, daher können Sie Masters und Slaves im laufenden Betrieb auf eine neuere 3.23-Version aktualisieren, und Sie können unterschiedliche Versionen von MySQL auf Slave und Master laufen haben, solange beide neuer als Version 3.23.26 sind.

Frage: Welche Dinge sollte ich beachten, wenn ich Zweiweg-Replikation aufsetze?

Antwort: MySQL-Replikation unterstützt derzeit kein Sperr-Protokoll zwischen Master und Slave, um die Atomizität einer verteilten (Cross-Server-) Aktualisierung zu gewährleisten. Mit anderen Worten ist es für einen Client A möglich, eine Aktualisierung zu Co-Master 1 zu machen. In der Zwischenzeit, bevor er sich an Co-Master 2 wendet, könnte Client B eine Aktualisierung auf Co-Master 2 machen, die dazu führt, dass die Aktualisierung von Client A anders funktioniert als auf Master 1. Wenn daher die Aktualisierung von Client A zu Co-Master 2 durchdringt, produziert das Tabellen, die anders sein werden als die auf Co-Master 1, selbst nachdem alle Aktualisierungen von Co-Master 2 ebenfalls durchgeführt wurden. Daher sollten sie keine zwei Server in einer Zweiweg-Replikation verketten, es sei denn, Sie können sicher sein, dass Ihre Aktualisierungen immer in bestimmter Reihenfolge ablaufen, oder indem Sie irgendwie in Ihrem Client-Code Vorkehrungen gegen Aktualisierung treffen, die nicht in der richtigen Reihenfolge sind.

Sie müssen sich auch darüber im Klaren sein, dass Zweiweg-Replikation Ihre Performance nicht wesentlich verbessert, falls überhaupt, sofern Aktualisierungen betroffen sind. Beide Server müssen ungefähr dieselbe Menge Aktualisierungen durchführen, was auch ein Server hätte tun können. Der einzige Unterschied liegt darin, dass es sehr viel weniger Lock-Contention gibt, weil die Aktualisierungen, die von einem anderen Server stammen, in einem Slave-Thread serialisiert werden. Dennoch kann der erzielte Vorteil durch Netzwerk-Verzögerungen konterkariert werden.

Frage: Wie kann ich Replikation benutzen, um die Performance meines Systems zu verbessern?

Antwort: Sie sollten einen Server als Master aufsetzen und alle Schreibvorgänge zu ihm lenken, und so viele Slaves wie möglich einrichten und die Lesevorgänge zwischen Master und Slaves verteilen. Ausserdem können Sie die Slaves mit `--skip-bdb`, `--low-priority-updates` und `--delay-key-write-for-all-tables` starten, um für die Slaves Geschwindigkeitsverbesserungen zu erzielen. In diesem Fall benutzt der Slave nicht transaktionale `MyISAM`-Tabellen anstelle von `BDB`-Tabellen, um mehr Geschwindigkeit zu erhalten.

Frage: Was muss ich in meinem Client-Code tun, damit dieser Performance-verbessernde Replikation nutzt?

Antwort: Wenn der Teil Ihres Codes, der für den Datenbankzugriff zuständig ist, korrekt abstrahiert / modularisiert ist, sollte die Konvertierung zur replizierten Einrichtung sehr glatt und einfach verlaufen: Ändern Sie die Implementation Ihres Datenbankzugriffs so, dass von irgend einem Slave oder dem Master gelesen und immer zum Master geschrieben wird. Wenn Ihr Code nicht diese Abstraktionsebene besitzt, ist die Einrichtung eines Replikationssystems ein guter Grund, ihn zu säubern. Sie könnten damit beginnen, eine Wrapper-Bibliothek oder ein Wrapper-Modul mit folgenden Funktionen zu benutzen:

- `safe_writer_connect()`
- `safe_reader_connect()`
- `safe_reader_query()`
- `safe_writer_query()`

`safe_` bedeutet, dass die Funktion sich um die Handhabung jeglicher Fehlerbedingungen kümmert.

Danach sollten Sie Ihren Client-Code so umwandeln, dass er die Wrapper-Bibliothek benutzt. Dieser Prozess kann anfangs etwas anstrengend und aufregend sein, wird sich aber auf lange Sicht lohnen. Alle Applikationen, die dem geschilderten Muster folgen, werden ebenfalls Nutzen aus der Master-/Slaves-Lösung ziehen. Der Code wird sich viel einfacher pflegen lassen und Optionen zur Problemlösung werden trivial sein. Sie brauchen einfach nur ein oder zwei Funktionen zu ändern, um zum Beispiel zu loggen, wie lang jede Anfrage dauerte, oder welche Anfrage unter Ihren Tausenden einen Fehler produzierte. Wenn Sie schon eine Menge Code geschrieben haben, wollen Sie den Umwandlungsprozess wahrscheinlich automatisieren. Hierfür können Sie zum Beispiel Monty's `replace`-Dienstprogramm benutzen, das der Standard-Distribution von MySQL beiliegt, oder Ihr eigenes Perl-Skript schreiben. Hoffentlich folgt Ihr Code irgend einem erkennbaren Muster - wenn nicht, ist es wahrscheinlich ohnehin besser, ihn neu zu schreiben, oder zumindest, ihn durchzugehen und manuell in ein Muster zu bringen.

Beachten Sie, dass Sie natürlich andere Namen für die Funktionen verwenden können. Wichtig ist, eine einheitliche Schnittstelle für Verbindungen zum Lesen, Verbindungen zum Schreiben, Durchführen von Lesevorgängen und Durchführung von Schreibvorgängen zu haben.

Frage: Wann und in welchem Umfang kann MySQL-Replikation die Performance meines Systems verbessern?

Antwort: MySQL-Replikation bringt die meisten Vorteile auf einem System mit häufigen Lesevorgängen und nicht so häufigen Schreibvorgängen. Theoretisch können Sie eine Einrichtung aus einem Master und vielen Slaves so skalieren, dass Sie solange Slaves hinzufügen, bis Sie entweder keine Netzwerk-Bandbreite mehr haben oder bis Ihre Aktualisierungslast so weit ansteigt, dass der Master sie nicht mehr handhaben kann.

Im festlegen zu können, wie viele Slaves sie haben können, bevor die zusätzlichen Vorteile aufgewogen werden, und um wieviel Sie die Performance Ihrer Site steigern können, müssen Sie Ihre Anfragenmuster kennen und empirisch (durch Benchmarks) festlegen, wie das Verhältnis zwischen dem Durchsatz von Lesevorgängen (pro Sekunde, oder `max_reads`) und Schreibvorgängen (`max_writes`) auf einem typischen Master und einem typischen Slave ist. Das unten stehende Beispiel zeigt Ihnen eine eher vereinfachte Berechnung, was Sie mit Replikation für ein imaginäres System erreichen können.

Nehmen wir an, Ihr Systemlast besteht aus 10% Schreibvorgängen und 90% Lesevorgängen, und Sie haben festgestellt, dass `max_reads` = 1200 - 2 * `max_writes` ist. Mit anderen Worten kann Ihr System 1.200 Lesevorgänge pro Sekunde ohne Schreibzugriffe ausführen, und der durchschnittliche Schreibvorgang ist zweimal so langsam wie der durchschnittliche Lesevorgang, und das Verhältnis ist linear. Nehmen wir ferner an, dass Ihr Master und Slave dieselbe Kapazität haben, und dass es N Slaves und 1 Master gibt. Dann gilt für jeden Server (Master oder Slave):

`lesen = 1200 - 2 * schreiben` (aus Benchmarks)

`lesen = 9 * schreiben / (N + 1)` (lesen aufgeteilt, aber schreiben geht an alle Server)

```
9 * schreiben / (N+1) + 2 * schreiben = 1200
```

```
schreiben = 1200/(2 + 9/(N+1)
```

Wenn also N = 0, was bedeutet, dass es keine Replikation gibt, kann Ihr System 1200 / 11, also etwa 109 Schreibvorgänge pro Sekunden handhaben (was heißt, dass Sie neunmal so viele Lesevorgänge haben, was der Natur Ihrer Applikation entspricht).

Wenn N = 1 ist, können Sie bis zu 184 Schreibvorgänge pro Sekunde haben.

Wenn N = 8 ist, können Sie bis zu 400 haben.

Wenn N = 17 ist, können Sie 480 haben.

Wenn schließlich N gegen unendlich geht (und Ihr Budget gegen negativ unendlich), können Sie sehr nahe an 600 Schreibvorgänge pro Sekunde kommen und damit den Systemdurchsatz um etwa den Faktor 5,5 erhöhen. Mit nur 8 Servern jedoch kommen Sie bereits auf eine Steigerung um fast den Faktor 4.

Beachten Sie, dass diese Berechnungen von unbegrenzter Netzwerk-Bandbreite ausgehen und verschiedene andere Faktoren vernachlässigen, die auf Ihrem System signifikant sein könnten. In vielen Fällen werden Sie nicht in der Lage sein, präzise vorauszusagen, was auf Ihrem System passieren wird, wenn Sie N Replikations-Slaves hinzufügen. Dennoch kann die Beantwortung folgender Fragen Ihnen helfen zu entscheiden, ob und wie viel (wenn überhaupt) Replikation bewirken kann, dass sich Ihre System-Performance verbessert:

- Was ist das Lese-/Schreibverhältnis auf Ihrem System?
- Wieviel zusätzliche Schreiblast kann ein Server handhaben, wenn Sie die Leselast verringern?
- Für wie viele Slave haben Sie Bandbreite auf Ihrem Netzwerk?

Frage: Wie kann ich Replikation benutzen, um Redundanz / hohe Verfügbarkeit zur Verfügung zu stellen?

Antwort: Mit den momentan verfügbaren Features würden Sie einen Master und einen Slave (nicht mehrere Slaves) aufsetzen und

ein Skript schreiben, das den Master beobachtet, um zu sehen, ob er hochgefahren ist, und Ihre Applikationen und die Slaves anweisen, den Master im Falle von Fehlschlägen zu ändern. Einige Vorschläge: set up a master und a slave (or several slaves) und write a Skript

- Um einem Slave mitzuteilen, den Master zu ändern, benutzen Sie den `CHANGE MASTER TO`-Befehl.
- Eine gute Möglichkeit, Ihre Applikationen darüber informiert zu halten, wo der Master ist, ist ein dynamischer DNS-Eintrag für den Master. Bei **bind** können Sie `nsupdate` benutzen, um Ihr DNS dynamisch zu aktualisieren.
- Sie sollten Ihre Slaves mit der `log-bin`-Option und ohne `log-slave-updates` laufen lassen. Auf diese Art wird der Slave bereit sein, ein Master zu werden, sobald Sie `STOP SLAVE` eingeben, `RESET MASTER` und `CHANGE MASTER TO` auf den anderen Slaves. Das wird auch dabei helfen, fehlgelaufene Aktualisierungen zu entdecken, die auf Grund von Fehlkonfiguration des Slaves passieren (im Idealfall sollten Sie die Zugriffsrechte so konfigurieren, dass kein Client einen Slave aktualisieren kann, ausser der Slave-Thread), in Kombination mit den Bugs in Ihren Client-Programmen (die den Slave nie direkt aktualisieren sollten).

Momentan arbeiten wir an der Integration eines Systems in MySQL, das automatisch den Master auswählt, aber bis es fertig ist, müssen Sie Ihre eigenen Beobachtungswerkzeuge schaffen.

5.10.8. Problemlösung bei Replikation

Wenn Sie den Anweisungen gefolgt sind und Ihre Replikationseinrichtung nicht funktioniert, beseitigen Sie zunächst die Möglichkeit von Benutzerfehlern, indem Sie folgendes prüfen:

- Loggt der Master in die Binär-Log-Datei? Prüfen Sie das mit `SHOW MASTER STATUS`. Wenn das der Fall ist, ist die `Position` nicht 0. Wenn nicht, überprüfen Sie, ob Sie dem Master die `log-bin`-Option angegeben und die `server-id` gesetzt haben.
- Läuft der Slave? Überprüfen Sie das mit `SHOW SLAVE STATUS`. Die Antwort steht in der `Slave_running`-Spalte. Wenn nicht, überprüfen Sie die Slave-Optionen und überprüfen Sie die Fehler-Log-Datei auf Meldungen.
- Wenn der Slave läuft, hat er eine Verbindung mit dem Master hergestellt? Führen Sie `SHOW PROCESSLIST` aus, finden Sie den Thread mit dem `System user`-Wert in der `User`-Spalte und `none` in der `Host`-Spalte und überprüfen Sie die `State`-Spalte. Wenn dort steht `connecting to master`, überprüfen Sie die Berechtigungen für den Replikations-Benutzer auf dem Master, den Master-Hostnamen, Ihre DNS-Einrichtung, ob der Master tatsächlich läuft, ob er durch den Slave erreichbar ist, und wenn all das in Ordnung zu sein scheint, lesen Sie die Fehler-Log-Dateien.
- Wenn der Slave lief, aber dann anhielt, schauen Sie in die Ausgabe von SHOW SLAVE STATUS und überprüfen Sie die Fehler-Log-Dateien. Das passiert üblicherweise, wenn eine Anfrage, die auf dem Master funktionierte, auf dem Slave fehlschlägt. Das sollte nie vorkommen, wenn Sie einen korrekten Schnappschuss des Masters aufgenommen haben und die Daten nie auf dem Slave ausserhalb des Slave-Threads verändern. Wenn das doch auftritt, ist es ein Bug und sollte berichtet werden.
- Wenn eine Anfrage, die auf dem Master funktionierte, nicht auf dem Slave läuft, und eine komplette Datenbank-Synchronisation (das richtige, was man tun sollte) nicht ratsam erscheint, versuchen Sie folgendes:
  - Überprüfen Sie zunächst, ob irgend ein 'streunender' Datensatz im Weg ist. Finden Sie heraus, wie das geschehen konnte, dann löschen Sie ihn und lassen `SLAVE START` laufen.
  - Wenn das Obige nicht funktioniert oder nicht zutrifft, versuchen Sie herauszufinden, ob es sicher ist, die Aktualisierung manuell durchzuführen (falls notwendig) und ignorieren Sie dann die nächste Anfrage vom Master.
  - Wenn Sie sich entschieden haben, dass Sie die nächste Anfrage überspringen können, führen Sie `SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START;` aus, um eine Anfrage zu überspringen, die kein auto_increment oder last_insert_id benutzt, ansonsten `SET SQL_SLAVE_SKIP_COUNTER=2; SLAVE START;`. Der Grund, warum auto_increment- / last_insert_id-Anfragen anders sind, liegt darin, dass für Sie zwei Ereignisse in der Binär-Log-Datei des Masters verzeichnet sind.
  - Wenn Sie sicher sind, dass der Slave perfekt mit dem Master synchronisiert gestartet ist, und dass niemand die fraglichen Tabellen ausserhalb des Slave-Threads aktualisiert hat, berichten Sie den Bug, damit wir die oben beschriebenen Tricks nicht noch einmal machen müssen.
- Stellen Sie sicher, dass es sich nicht um alten Bug handelt, indem Sie auf die aktuellste Version aktualisieren.
- Wenn alles Weitere fehlschlägt, lesen Sie die Fehler-Log-Dateien. Wenn diese Groß sind, führen Sie ein `grep -i slave /pfad/zu/your-log.err` auf dem Slave durch. Es gibt kein allgemeines Muster, nach dem man auf dem Master suchen könnte, weil die einzigen Fehler, die dieser mitschreibt, allgemeine Systemfehler sind - falls möglich, wird er Fehler an die Slaves senden, wenn etwas schief ging.

Wenn Sie sicher sind, dass es keine Benutzerfehler gibt und die Replikation immer noch nicht funktioniert oder nicht stabil ist, ist es an der Zeit, einen Bug-Bericht auszuarbeiten. Um dem Bug auf die Spur zu kommen, brauchen wir soviel Informationen von Ihnen wie möglich. Bitte nehmen Sie sich etwas Zeit und schreiben Sie einen guten Bug-Bericht. Im Idealfall hätten wir gerne einen Test-Fall in dem Format, das Sie im `mysql-test/t/rpl*`-Verzeichnis des Source-Baums finden. Wenn Sie einen solchen Test-Fall schicken, können Sie in den meisten Fällen ein Patch innerhalb von ein oder zwei Tagen erwarten. Diese Zeitspanne hängt allerdings von einer Anzahl weiterer Faktoren ab.

Die zweitbeste Option ist ein einfaches Programm mit leicht konfigurierbaren Verbindungsargumenten für Master und Slave, das das Problem auf Ihrem System veranschaulicht. Sie können dies in Perl oder C schreiben, abhängig davon, welche Sprache Sie besser beherrschen.

Wenn Sie den Bug auf eine der beiden oben beschriebenen Weisen demonstrieren können, benutzen Sie `mysqlbug`, um einen Bug-Bericht vorzubereiten, und schicken Sie ihn an `<bugs@lists.mysql.com>`. Wenn Sie ein 'Phantom' haben - ein Problem, das auftritt, aber nicht einfach reproduziert werden kann - tun Sie folgendes:

- Stellen Sie sicher, dass kein Benutzerfehler im Spiel ist. Beispielsweise könnte der Slave ausserhalb des Slave-Threads aktualisiert werden - dann sind die Daten nicht synchronisiert und Sie haben womöglich einen Fehler wegen doppelter Schlüsseleinträge bei Aktualisierungen, wobei der Slave-Thread dann anhält und darauf wartet, dass Sie die Tabellen manuell in Ordnung und in Synchronisation bringen.
- Lassen Sie den Slave mit `log-slave-updates` und `log-bin` laufen - das behält eine Log-Datei aller Aktualisierungen auf dem Slave bei.
- Sichern Sie alle Beweise, bevor Sie die Replikation zurück setzen. Wenn wir keine oder nur schemenhafte Informationen haben, brauchen wir eine Weile, um dem Problem auf den Grund zu gehen. Die Beweise, die Sie für uns sammeln sollten, sind:
  - Alle Binär-Log-Dateien auf dem Master.
  - Alle Binär-Log-Dateien auf dem Slave.
  - Die Ausgabe von `SHOW MASTER STATUS` auf dem Master zu der Zeit, als Sie das Problem entdeckten.
  - Die Ausgabe von `SHOW SLAVE STATUS` auf dem Master zu der Zeit, als Sie das Problem entdeckten.
  - Fehler-Log-Dateien auf Master und Slave.
- Benutzen Sie `mysqlbinlog`, um die Binär-Log-Dateien zu untersuchen. Folgendes sollte hilfreich sein, um eine Anfrage zu finden, die Probleme verursacht, zum Beispiel:

```
mysqlbinlog -j pos_from_slave_status /pfad/zu/log_from_slave_status | head
```

Sobald Sie die Beweise des Phantomproblems gesammelt haben, versuchen Sie zuerst, es in einen separaten Test-Fall zu isolieren. Berichten Sie dann das Problem an `<bugs@lists.mysql.com>`, wobei Sie soviel Informationen wie möglich mitschicken.

Kapitel 6. MySQL-Optimierung

Optimierung ist eine komplizierte Aufgabe, weil sie ein umfassendes Verständnis des gesamten Systems voraussetzt. Es ist möglich, einige lokale Optimierungen Ihres Systems oder Ihrer Applikation mit geringem Wissen durchzuführen. Je optimaler Sie allerdings Ihr System gestalten wollen, desto mehr müssen Sie darüber wissen.

Dieses Kapitel erklärt und gibt Beispiele für verschiedene Möglichkeiten, MySQL zu optimieren. Denken Sie allerdings daran, dass es immer noch zusätzliche Möglichkeiten gibt, das System noch schneller zu machen.

6.1. Überblick über Optimierung

Der wichtigste Teil, um ein System schnell zu machen, ist natürlich das grundlegende Design. Ausserdem müssen Sie wissen, welche Dinge Ihr System macht und was die Flaschenhälse sind.

Die wichtigsten Flaschenhälse sind:

- Suchvorgänge auf Festplatte. Die Festplatte benötigt Zeit, um ein Stück Daten zu finden. Bei modernen Festplatten (Stand: 1999) ist die mittlere Zugriffszeit üblicherweise weniger als 10 ms, daher können theoretisch etwa 1.000 Suchvorgänge pro Sekunde durchgeführt werden. Bei neueren Festplatten wird diese Zeit allmählich besser. Für einzelne Tabellen ist sie sehr schwer zu optimieren. Eine Möglichkeit, das zu optimieren, besteht darin, Daten auf mehr als eine Platte zu verteilen.
- Lesen von / Schreiben auf Festplatte. Wenn die Festplatte in der richtigen Position ist, um die Daten zu lesen, die wir brauchen, kann sie bei modernen Platten (Stand: 1999) etwas 10 bis 20 MB pro Sekunde heraus geben. Das ist leichter zu optimieren als Suchvorgänge, weil man von mehrfachen Festplatten parallel lesen kann.
- CPU-Zyklen. Wenn die Daten im Hauptspeicher sind (oder bereits dort waren), müssen sie verarbeitet werden, um das Ergebnis zu erhalten. Kleine Tabellen im Vergleich zum Arbeitsspeicher ist der Faktor, der am meisten begrenzt. Auf der anderen Seite ist Geschwindigkeit bei kleinen Tabellen üblicherweise nicht das Problem.
- Speicher-Bandbreite. Wenn der Prozessor mehr Daten braucht, als in den CPU-Cache passen, wird die Bandbreite des Hauptspeichers zum Flaschenhals. Auf den meisten Systemen ist das ein ungewöhnlicher Flaschenhals, aber man sollte sich dessen bewusst sein.

6.1.1. MySQL-Design-Einschränkungen

Weil MySQL extrem schnelles Tabellensperren beherrscht (mehrfache Leser / einzelne Schreiber), ist das größte verbleibende Problem eine Mischung aus einem laufenden Strom von Einfügevorgängen und langsamen Selects auf dieselbe Tabelle.

Wir glauben, dass diese Wahl auf einer sehr großen Anzahl von Systemen letztlich einen Gewinn darstellt. Auch dieser Fall ist üblicherweise dadurch zu lösen, dass man mehrfache Kopien der Tabelle vorhält, aber man benötigt mehr Anstrengung und Hardware.

Wir arbeiten auch an einigen Erweiterungen, um dieses Problem in Hinsicht auf einige häufige Applikationsnischen zu lösen.

6.1.2. Portabilität

Weil alle SQL-Server unterschiedliche Teile von SQL implementieren, ist es immer Arbeit, portable SQL-Applikationen zu schreiben. Bei sehr einfachen Selects und Inserts ist das sehr einfach, aber je mehr Sie brauchen, desto schwieriger wird es. Wenn Sie eine Applikation wollen, die bei vielen Datenbanken noch schnell läuft, wird es sogar noch schwieriger!

Um eine komplexe Applikation portabel zu machen, müssen Sie sich für eine Reihe von SQL-Servern entscheiden, mit denen sie funktionieren soll.

Weitere Informationen zu Benchmark-Ergebnissen finden Sie unter http://dev.mysql.com/tech-resources/benchmarks/.

Sie sollten zum Beispiel keine Spaltennamen benutzen, die länger als 10 Zeichen sind, wenn Sie auch Informix oder DB2 benutzen wollen.

Sowohl die MySQL-Benchmarks als auch die Crash-me-Programme sind sehr Datenbank-abhängig. Indem Sie einen Blick darauf werfen, wie wir damit umgegangen sind, bekommen Sie ein Gefühl dafür, was Sie in Ihrer Applikation schreiben müssen, damit diese Datenbank-unabhängig läuft. Die Benchmark-Tests selbst befinden sich im `sql-bench`-Verzeichnis der MySQL-Quelldistribution. Sie sind in Perl mit der DBI-Datenbank-Schnittstelle geschrieben (die den Zugriffsteil des Problems löst).

Siehe http://www.mysql.com/information/benchmarks.html wegen der Ergebnisse aus diesem Benchmark-Test.

Wie Sie an den Ergebnissen sehen, haben alle Datenbanken einige Schwachpunkte, das heißt, sie haben verschiedene Design-

Kompromisse, die zu unterschiedlichem Verhalten führen.

Wenn Sie nach Datenbank-Unabhängigkeit streben, müssen Sie ein gutes Gefühl für die Flaschenhälse jedes SQL-Servers bekommen. MySQL ist SEHR schnell beim Abrufen und Aktualisieren von Dingen, hat aber Probleme, wenn gleichzeitig langsame Leser / Schreiber auf dieselbe Tabelle zugreifen. Oracle hat ein großes Problem, wenn Sie versuchen, auf Zeilen zuzugreifen, der kürzlich aktualisiert wurden (solange, bis sie auf Platte zurückgeschrieben wurden). Transaktionale Datenbanken sind allgemein nicht sehr gut darin, Zusammenfassungstabellen aus Log-Tabellen zu erzeugen, weil in diesem Fall Sperren auf Zeilenebene fast nutzlos ist.

Um Ihre Applikation *wirklich* Datenbank-unabhängig zu machen, müssen Sie eine leicht erweiterbare Schnittstelle definieren, über die Sie Ihre Daten manipulieren. Weil auf den meisten Systemen C++ verfügbar ist, ist es sinnvoll, C++-Klassen als Schnittstellen zu den Datenbanken zu benutzen.

Wenn Sie irgend ein spezifisches Feature einer Datenbankbenutzung (wie den `REPLACE`-Befehl in MySQL), sollten Sie eine Methode für die anderen SQL-Server codieren, um dasselbe Feature (wenngleich langsamer) zu implementieren. Bei MySQL können Sie die `/*! */`-Syntax benutzen, um MySQL-spezifische Schlüsselwörter in einer Anfrage zu verwenden. Der Code innerhalb von `/**/` wird von den meisten anderen SQL-Servern als Kommentar behandelt (ignoriert).

Wenn WIRKLICH hohe Performance wichtiger als Exaktheit ist, wie bei einigen Web-Applikationen, besteht eine Möglichkeit darin, eine Applikationsebene zu erzeugen, die alle Ergebnisse cachet, um Ihnen noch höhere Performance zu bringen. Indem Sie alte Ergebnisse nach einer Weile 'auslaufen' lassen, können Sie den Cache in vernünftiger Weise 'frisch' halten. Das ist in Fällen extrem hoher Last recht nett, wobei Sie den Cache dynamisch vergrößern und die Verfallszeit (Expire Timeout) höher setzen können, bis wieder Normalauslastung eintritt.

In diesem Fall sollte die Tabellenerzeugungsinformation Informationen über die ursprüngliche Cache-Größe enthalten und darüber, wie oft die Tabelle normalerweise aktualisiert (refresh) werden sollte.

6.1.3. Wofür benutzen wir MySQL?

In der anfänglichen Phase der Entwicklung von MySQL wurden die Features von MySQL für unseren größten Kunden gemacht. Dieser macht Data-Warehousing für eine Reihe der größten Einzelhändler in Schweden.

Aus allen Verkaufsstellen erhalten wir wöchentliche Zusammenfassungen aller Bonuskarten-Transaktionen, und es wird erwartet, dass daraus nützliche Informationen für die Eigentümer der Verkaufsstellen zur Verfügung gestellt werden, damit diese herausfinden können, wie ihre Werbemaßnahmen ihre Kunden beeinflussen.

Die Datenmenge ist recht riesig (etwa 7 Millionen Zusammenfassungs-Transaktionen pro Monat), und wir haben Daten von 4 bis 10 Jahren, die wir dem Benutzer präsentieren müssen. Wir bekamen wöchentliche Anfragen von Kunden, die 'sofortigen' Zugriff auf neue Berichte aus diesen Daten wollten.

Die Lösung bestand darin, alle Informationen monatsweise in komprimierten 'Transaktions-' Tabellen zu speichern. Wir haben einen Satz einfacher Makros (ein Skript), die aus diesen Tabellen Zusammenfassungstabellen erzeugen, die nach verschiedenen Kriterien gruppiert sind (Produktgruppe, Kunden-ID, Verkaufsstelle usw.). Die Berichte sind Web-Seiten, die dynamisch durch ein kleines Perl-Skript erzeugt werden, das eine Web-Seite parst, die enthaltenen SQL-Statements ausführt und die Ergebnisse einfügt. Wir hätten statt dessen PHP oder mod_perl benutzt, aber diese waren damals noch nicht verfügbar.

Für grafische Darstellungen schrieben wir ein einfaches Werkzeug in `C`, das GIFs auf der Grundlage der Ergebnisse einer SQL-Anfrage erzeugen kann (nach einigem Verarbeiten des Ergebnisses). Dieses wird ebenfalls dynamisch durch ein Perl-Skript ausgeführt, das die `HTML`-Dateien parst.

In den meisten Fällen kann ein neuer Bericht einfach durch das Kopieren eines bestehenden Skripts und das Verändern der SQL-Anfrage darin erzeugt werden. In einigen Fällen müssen wir einer bestehenden Zusammenfassungstabelle weitere Felder hinzufügen oder eine neue generieren, aber auch das ist recht einfach, weil wir alle Transaktionstabellen auf Platte haben. (Momentan haben wir mindestens 50 GB an Transaktionstabellen und 200 GB weiterer Kundendaten.)

Wir lassen unsere Kunden auch direkt mit ODBC auf die Transaktionstabellen zugreifen, so dass fortgeschrittene Benutzer selbst mit den Daten experimentieren können.

Wir hatten mit der Handhabung keinerlei Probleme, auf einer recht bescheidenen Sun Ultra SPARCstation (2x200 MHz). Kürzlich haben wir einen unserer Server auf eine mit 2 Prozessoren bestückte 400 MHz-UltraSPARC erweitert und planen jetzt, Transaktionen auf Produktebene zu handhaben, was eine zehnfache Steigerung der Datenmenge bedeuten würde. Wir glauben, dass wir auch damit Schritt halten können, indem wir unseren Systemen einfach mehr Festplattenplatz hinzufügen.

Wir experimentieren auch mit Intel-Linux, um in der Lage zu sein, mehr CPU-Power preisgünstiger zu erhalten. Jetzt, wo wir das binäre portable Datenbankformat haben (neu seit Version 3.23), werden wir dieses für einige Teile der Applikation benutzen.

Unser anfängliches Gefühl sagt uns, dass Linux viel besser bei geringer bis mittlerer Last ist, während Solaris wegen der extremen Festplatten-Eingabe-/Ausgabe-Geschwindigkeit (Disk-IO) bei Hochlast besser ist, aber wir können noch nichts Endgültiges darüber aussagen. Nach einigen Diskussionen mit den Linux-Kernel-Entwicklern ist das eventuell ein Seiteneffekt von Linux, das dem Stapel-Job so viel Ressourcen gibt, dass die interaktive Performance sehr gering wird. Dadurch scheint die Maschine sehr langsam

und unempfänglich für Eingaben zu lassen, während große Stapel-Jobs abgearbeitet werden. Wir hoffen, dass dies in zukünftigen Linux-Kernels besser gehandhabt wird.

6.1.4. Die MySQL-Benchmark-Suite

Dieser Abschnitt sollte eine technische Beschreibung der MySQL- Benchmark-Suite (und von `crash-me`) enthalten, aber diese Beschreibung wurde noch nicht geschrieben. Momentan können Sie eine gute Idee über den Benchmark bekommen, wenn Sie einen Blick auf den Code und die Ergebnisse im `sql-bench`-Verzeichnis jeder MySQL-Quelldistribution werfen.

Diese Benchmark-Suite ist als Benchmark gedacht, der jedem Benutzer mitteilt, welche Dinge in einer gegebenen SQL-Implementation gut performen und welche schlecht.

Beachten Sie, dass dieser Benchmark single-threaded ist. Daher misst er die minimale Zeit der Operationen. In Zukunft planen wir, auch etliche multi-threaded Test hinzuzufügen.

Beispiele (die auf derselben NT-4.0-Maschine liefen):

2.000.000 Zeilen vom Index lesen	**Sekunden**	**Sekunden**
mysql	367	249
mysql_odbc	464	
db2_odbc	1206	
informix_odbc	121126	
ms-sql_odbc	1634	
oracle_odbc	20800	
solid_odbc	877	
sybase_odbc	17614	

350.768 Zeilen einfügen	**Sekunden**	**Sekunden**
mysql	381	206
mysql_odbc	619	
db2_odbc	3460	
informix_odbc	2692	
ms-sql_odbc	4012	
oracle_odbc	11291	
solid_odbc	1801	
sybase_odbc	4802	

Im obigen Test lief MySQL mit einem 8 MB Index-Cache.

Weitere Benchmark-Ergebnisse haben wir unter http://www.mysql.com/information/benchmarks.html gesammelt.

Beachten Sie, dass Oracle nicht beinhaltet ist, weil sie gebeten haben, entfernt zu werden. Alle Oracle-Benchmarks müssen von Oracle freigegeben werden! Wir glauben, dass das die Aussagefähigkeit von Oracle-Benchmarks **SEHR** zweifelhaft erscheinen läßt, weil alle obigen Benchmarks dafür da sind zu zeigen, was eine Standard-Installation bei einem einzelnen Client machen kann.

Um eine Benchmark-Suite laufen zu lassen, müssen Sie eine MySQL-Quelldistribution herunter laden, den Perl-DBI-Treiber und den Perl-DBD-Treiber für die gewünschte Datenbank installieren und dann folgendes tun:

```
cd sql-bench
perl run-all-tests --server=#
```

Wobei # einer der unterstützten Server ist. Sie erhalten eine Auflistung aller Optionen und unterstützten Server, indem Sie `run-all-tests --help` ausführen.

`Crash-me` versucht herauszufinden, welche Features eine Datenbank unterstützt und wo ihre Fähigkeiten und Einschränkungen sind, indem tatsächliche Anfragen ausgeführt werden. Beispielsweise stellt es fest:

- Welche Spaltentypen unterstützt werden.

- Wie viele Indexe unterstützt werden.
- Welche Funktionen unterstützt werden.
- Wie Groß eine Anfrage sein kann.
- Wie Groß eine `VARCHAR`-Spalte sein kann.

6.1.5. Wie Sie Ihre eigenen Benchmarks benutzen

Sie sollten Ihre Applikation und Datenbank auf jeden Fall einem Benchmark-Test unterziehen um herauszufinden, wo Flaschenhälse sind. Indem Sie einen Flaschenhals beseitigen (oder ihn durch ein 'Dummy-Modul' ersetzen), können Sie leicht den nächsten Flaschenhals herausfinden (usw.). Selbst wenn die insgesamte Performance für Ihre Applikation ausreichend ist, sollten Sie zumindest einen Plan für jeden Flaschenhals aufstellen und entscheiden, auf welche Weise dieser beseitigt werden soll, wenn Sie eines Tages die zusätzliche Performance benötigen.

Als Beispiel für ein portables Benchmark-Programm schauen Sie sich die MySQL-Benchmark-Suite an. See Abschnitt 6.1.4, „Die MySQL-Benchmark-Suite“. Sie können jedes Programm dieser Suite nehmen und es Ihren Bedürfnissen entsprechend abändern. Wenn Sie das tun, können Sie unterschiedliche Lösungen für Ihr Problem finden und testen, was bei Ihnen wirklich die schnellste Lösung ist.

Es ist häufig der Fall, dass Probleme nur dann auftreten, wenn das System unter schwerer Last läuft. Viele Kunden nahmen mit uns Kontakt auf, nachdem sie ein (getestetes) System in eine Produktionsumgebung stellten und Lastprobleme bekamen. Bei jedem dieser Fälle gab es bislang entweder Probleme mit dem Grund-Design (Tabellen-Scans laufen NICHT gut unter hoher Last) oder im Zusammenhang mit dem Betriebssystem / den Bibliotheken. Das meiste davon wäre **SEHR** viel einfacher zu beheben, wenn die Systeme nicht bereits in einer Produktionsumgebung liefen.

Um solcherlei Probleme zu vermeiden, sollten Sie einige Anstrengung darauf verwenden, Ihre gesamte Applikation unter der schlimmstmöglichen Last zu benchmarken! Hierfür können Sie Super Smack benutzen, das Sie hier erhalten: http://www.mysql.com/Downloads/super-smack/super-smack-1.0.tar.gz. Wie der Name nahelegt, kann es Ihr System auf die Knie zwingen, wenn Sie das wollen. Achten Sie daher darauf, es nur auf Entwicklungssystemen zu verwenden.

6.2. `SELECT`s und andere Anfragen optimieren

Zunächst etwas, das alle Anfragen betrifft: Je komplexer das Berechtigungssystem, das Sie einrichten, desto mehr Overhead bekommen Sie.

Falls Sie noch keinerlei `GRANT`-Statements ausgeführt haben, optimiert MySQL die Berechtigungsprüfung zum Teil. Wenn Sie also sehr hohe Zugriffszahlen haben, kann es einen Zeitvorteil darstellen, Grants zu vermeiden. Ansonsten können mehr Berechtigungsprüfungen in einem größeren Overhead resultieren.

Wenn Sie Probleme bei einer bestimmten MySQL-Funktion haben, können Sie den Zeitbedarf jederzeit wie folgt mit dem MySQL-Client feststellen:

```
mysql> select benchmark(1000000,1+1);
+------------------------+
| benchmark(1000000,1+1) |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.32 sec)
```

Das Ergebnis zeigt, dass MySQL 1.000.000 `+`-Operationen in 0,32 Sekunden auf einer `PentiumII-400MHz`-Maschine ausführen kann.

Alle MySQL-Funktionen sollten sehr optimiert sein, aber es mag einige Ausnahmen geben und `benchmark(schleifenzaehler,ausdruck)` ist ein großartiges Werkzeug, um herauszufinden, ob das das Problem bei Ihrer Anfrage darstellt.

6.2.1. `EXPLAIN`-Syntax (Informationen über ein `SELECT` erhalten)

```
     EXPLAIN tabelle
oder EXPLAIN SELECT select_optionen
```

`EXPLAIN tabelle` ist ein Synonym für `DESCRIBE tabelle` oder `SHOW COLUMNS FROM tabelle`.

Wenn Sie einem `SELECT`-Statement das Schlüsselwort `EXPLAIN` voran stellen, erklärt MySQL explains, wie er das `SELECT` ausführen würde, indem Informationen darüber gemacht werden, wie Tabellen verknüpft (Join) werden und in welcher Reihenfolge.

Mit der Hilfe von EXPLAIN können Sie erkennen, wo Sie Tabellen Indexe hinzufügen müssen, um ein schnelleres SELECT zu erhalten, das Indexe benutzt, um die Datensätze zu finden. Ausserdem sehen Sie, ob der Optimierer die Tabellen in optimaler Reihenfolge verknüpft. Um den Optimierer zu zwingen, eine spezielle Verknüpfungsreihenfolge bei einem SELECT-Statement einzuhalten, fügen Sie eine STRAIGHT_JOIN-Klausel hinzu.

Bei nicht einfachen Verknüpfungen (Joins) gibt EXPLAIN für jede Tabelle, die im SELECT-Statement benutzt wurde, eine Informationszeile zurück. Die Tabellen sind in der Reihenfolge aufgelistet, in der sie gelesen werden würden. MySQL löst alle Joins mit einer Single-Sweep-Multi-Join-Methode auf. Das bedeutet, dass MySQL eine Zeile aus der ersten Tabelle liest, dann die passende Zeile in der zweiten Tabelle sucht, dann in der dritten Tabelle usw. Wenn alle Tabellen verarbeitet wurden, gibt er die ausgewählten Spalten aus und geht rückwärts durch die Tabellenliste durch, bis eine Tabelle gefunden wird, bei der es weitere passende Zeilen gibt. Die nächste Zeile wird aus dieser Tabelle gelesen, und der Prozess fährt mit der nächsten Tabelle fort.

Die Ausgabe von EXPLAIN enthält folgende Spalten:

- table

  Die Tabelle, auf die sich die Ausgabezeile bezieht.

- type

  Der Join-Typ. Informationen über die verschiedenen Typen finden Sie weiter unten.

- possible_keys

  Die possible_keys-Spalte gibt an, welche Indexe MySQL verwenden konnte, um Zeilen in dieser Tabelle zu finden. Beachten Sie, dass diese Spalte völlig unabhängig von der Reihenfolge der Tabellen ist. Das heißt, dass einige der Schlüssel in possible_keys möglicherweise bei der tatsächlich verwendeten Tabellenreihenfolge nicht verwendbar sind.

  Wenn diese Spalte leer ist, gibt es keine relevanten Indexe. In diesem Fall können Sie die Performance Ihrer Anfrage womöglich verbessern, indem Sie die WHERE-Klausel untersuchen, um festzustellen, ob diese auf eine oder mehrere Spalten verweist, die zweckmäßigerweise indiziert werden sollten. Wenn das der Fall ist, erzeugen Sie einen entsprechenden Index und prüfen Sie die Anfrage noch einmal mit EXPLAIN. See Abschnitt 7.5.4, „ALTER TABLE-Syntax“.

  Um zu sehen, welche Indexe eine Tabelle hat, benutzen Sie SHOW INDEX FROM tabelle.

- key

  Die key-Spalte gibt den Schlüssel an, den MySQL tatsächlich benutzen wird. Der Schlüssel ist NULL, wenn kein Index gewählt wurde. Wenn MySQL den falschen Index wählt, können Sie ihn wahrscheinlich zwingen, einen anderen Index zu nehmen, indem Sie myisamchk --analyze oder Abschnitt 5.4.6.1, „Aufrufsyntax von myisamchk“ ausführen oder USE INDEX/IGNORE INDEX benutzen. See Abschnitt 7.4.1.1, „JOIN-Syntax“.

- key_len

  Die key_len-Spalte gibt die Länge des Schlüssels an, den MySQL benutzen wird. Die Länge ist NULL, wenn key NULL ist. Beachten Sie, dass Ihnen das angibt, wie viele Teile eines mehrteiligen Schlüssels MySQL tatsächlich benutzen wird.

- ref

  Die ref-Spalte zeigt an, welche Spalten oder Konstanten beim key benutzt werden, um Zeilen aus der Tabelle auszuwählen.

- rows

  die rows-Spalte gibt die Anzahl von Zeilen an, von denen MySQL annimmt, dass es sie untersuchen muss, um die Anfrage auszuführen.

- Extra

  Diese Spalte enthält zusätzliche Informationen darüber, wie MySQL die Anfrage auflösen wird. Folgende unterschiedliche Text-Zeichenketten können in dieser Spalte stehen:

  - Distinct

    MySQL wird die Suche nach weiteren Zeilen für die aktuelle Zeilenkombination nicht fortsetzen, nachdem er die erste passende Zeile gefunden hat.

  - Not exists

    MySQL war in der Lage, eine LEFT JOIN-Optimierung der Anfrage durchzuführen, und wird keine weiteren Spalten in dieser Tabelle für die vorherige Zeilenkombination mehr untersuchen, nachdem er eine Zeile gefunden hat, die den LEFT

`JOIN`-Kriterien entspricht.

Hier ist ein Beispiel dafür:

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.id=t2.id WHERE t2.id IS NULL;
```

Angenommen, `t2.id` ist mit `NOT NULL` definiert. In diesem Fall scannt MySQL `t1` und schlägt die Zeilen in `t2` über `t1.id` nach. Wenn MySQL eine übereinstimmende Zeile in `t2` findet, weiß er, dass `t2.id` nie `NULL` sein kann und scannt nicht durch den Rest der Zeilen in `t2`, die dieselbe `id` haben. Mit anderen Worten, für jede Zeile in `t1` muss MySQL nur ein einziges Mal in `t2` nachschlagen, unabhängig davon, wie viel übereinstimmende Zeilen es in `t2` gibt.

- `range checked for each record (index map: #)`

  MySQL hat keinen gut geeigneten Index zum Benutzen gefunden. Statt dessen wird er für jede Zeilenkombination in der vorherigen Tabelle eine Prüfung vornehmen, welchen Index er benutzen soll (falls überhaupt) und diesen Index benutzen, um Zeilen aus der Tabelle abzurufen. Das ist nicht sehr schnell, aber immer noch schneller, als einen Join ohne Index durchzuführen.

- `Using filesort`

  MySQL braucht einen zusätzlichen Durchgang, um herauszufinden, wie die Zeilen in sortierter Reihenfolge abgerufen werden sollen. Die Sortierung wird durchgeführt, indem in Abhängigkeit vom `join type` durch alle Zeilen durchgegangen wird und der Sortierschlüssel und Zeiger auf die Zeilen für alle Zeilen gespeichert wird, die dem `WHERE` entsprechen. Danach werden die Schlüssel sortiert. Schließlich werden die Zeilen in sortierter Reihenfolge abgerufen.

- `Using index`

  Die Spalteninformation wird aus der Tabelle abgerufen, indem nur Informationen aus dem Index-Baum benutzt werden, ohne dass zum Suchen zusätzlich in den tatsächlichen Zeilen gelesen werden muss. Das kann gemacht werden, wenn alle benutzten Spalten der Tabelle Teil desselben Indexes sind.

- `Using temporary`

  Um die Anfrage aufzulösen muss MySQL eine temporäre Tabelle erzeugen, die das Ergebnis enthält. Das passiert typischerweise, wenn Sie ein `ORDER BY` auf eine andere Spalte setzen als auf die, die Sie im `GROUP BY` angegeben haben.

- `Where used`

  Eine `WHERE`-Klausel wird benutzt, um zu begrenzen, bei welchen Zeilen auf Übereinstimmung in der nächsten Tabelle gesucht wird oder welche Zeilen an den Client geschickt werden. Wenn Sie diese Information nicht haben und die Tabelle vom Typ `ALL` oder `index` ast, ist vielleicht in Ihrer Anfrage etwas falsch (falls Sie nicht vorhaben, alle Zeilen aus der Tabelle zu holen / zu untersuchen).

Wenn Sie wollen, dass Ihre Anfragen so schnell wie möglich laufen, sollten Sie auf `Using filesort` und `Using temporary` achten.

Die verschiedenen Join-Typen sind unten aufgeführt, sortiert vom besten zum schlechtesten Typ:

- `system`

  Die Tabelle hat nur eine Zeile (= Systemtabelle). Das ist ein spezieller Fall des `const`-Join-Typs.

- `const`

  Die Tabelle hat höchsten eine übereinstimmende Zeile, die am Anfang der Anfrage gelesen werden wird. Weil es nur eine Zeile gibt, können Spaltenwerte in dieser Zeile vom Optimierer als Konstanten betrachtet werden. `const`-Tabellen sind sehr schnell, weil sie nur einmal gelesen werden!

- `eq_ref`

  Aus dieser Tabelle wird für jede Zeilenkombination der vorherigen Tabellen eine Zeile gelesen. Das ist der bestmögliche Join-Typ, ausgenommen die `const`-Typen. Er wird benutzt, wenn alle Teile eines Indexes vom Join benutzt werden und der Index `UNIQUE` oder ein `PRIMARY KEY` ist.

- `ref`

Alle Zeilen mit übereinstimmenden Index-Werten werden für jede Zeilenkombination der vorherigen Tabellen gelesen. `ref` wird benutzt, wenn der Join nur das am weitesten links stehende Präfix des Schlüssels benutzt, oder wenn der Schlüssel nicht `UNIQUE` oder ein `PRIMARY KEY` ist (mit anderen Worten, wenn der Join auf der Grundlage des Schlüsselwerts keine einzelne Zeile auswählen kann). Wenn der Schlüssel, der benutzt wird, nur mit einigen wenigen Zeilen übereinstimmt, ist dieser Join-Typ gut.

- `range`

  Nur Zeilen, die innerhalb eines angegebenen Bereichs sind, werden abrufen, wobei ein Index benutzt wird, um die Zeilen auszuwählen. Die `key`-Spalte gibt an, welcher Index benutzt wird. `key_len` enthält den längsten Schlüsselteil, der benutzt wurde. Die `ref`-Spalte ist für diesen Typ NULL.

- `index`

  Das ist dasselbe wie `ALL`, ausser dass nur der Index-Baum gescannt wird. Das ist üblicherweise schneller als `ALL`, weil die Index-Datei üblicherweise kleiner ist als die Daten-Datei.

- `ALL`

  Für jede Zeilenkombination der vorherigen Tabellen wird ein kompletter Tabellenscan durchgeführt. Das ist normalerweise nicht gut, wenn die Tabelle die erste Tabelle ist, die nicht als `const` gekennzeichnet ist, und üblicherweise **sehr** schlecht in allen anderen Fällen. Sie können `ALL` normalerweise vermeiden, indem Sie mehr Indexe hinzufügen, so dass die Zeile auf der Grundlage der Konstanten-Werte oder Spaltenwerte von früheren Tabellen abgerufen werden kann.

Sie erhalten einen guten Anhaltspunkt, wie gut ein Join ist, wenn Sie alle Werte in der `rows`-Spalte der `EXPLAIN`-Ausgabe multiplizieren. Das sollte grob aussagen, wie vielen Zeilen MySQL untersuchen muss, um die Anfrage auszuführen. Diese Anzahl wird auch benutzt, wenn Sie Anfragen mit der `max_join_size`-Variablen begrenzen. See Abschnitt 6.5.2, „Serverparameter tunen“.

Das folgende Beispiel zeigt, wie ein `JOIN` progressiv optimiert werden kann, indem die Informationen genutzt werden, die `EXPLAIN` bereit stellt.

Angenommen, Sie haben unten stehendes `SELECT`-Statement, das Sie mit `EXPLAIN` untersuchen:

```
EXPLAIN SELECT tt.TicketNumber, tt.TimeIn,
            tt.ProjectReference, tt.EstimatedShipDate,
            tt.ActualShipDate, tt.ClientID,
            tt.ServiceCodes, tt.RepetitiveID,
            tt.CurrentProcess, tt.CurrentDPPerson,
            tt.RecordVolume, tt.DPPrinted, et.COUNTRY,
            et_1.COUNTRY, do.CUSTNAME
        FROM tt, et, et AS et_1, do
        WHERE tt.SubmitTime IS NULL
            AND tt.ActualPC = et.EMPLOYID
            AND tt.AssignedPC = et_1.EMPLOYID
            AND tt.ClientID = do.CUSTNMBR;
```

Nehmen wir bei diesem Beispiel folgendes an:

- Die Spalten, die verglichen werden, wurden wie folgt deklariert:

Tabelle	Spalte	Spaltentyp
`tt`	`ActualPC`	`CHAR(10)`
`tt`	`AssignedPC`	`CHAR(10)`
`tt`	`ClientID`	`CHAR(10)`
`et`	`EMPLOYID`	`CHAR(15)`
`do`	`CUSTNMBR`	`CHAR(15)`

- Die Tabellen haben die unten stehenden Indexe:

Tabelle	Index
`tt`	`ActualPC`
`tt`	`AssignedPC`
`tt`	`ClientID`
`et`	`EMPLOYID` (primary key)

do	CUSTNMBR (primary key)

- Die tt.ActualPC-Werte sind nicht gleichmäßig verteilt.

Anfangs, bevor die Optimierung durchgeführt wurde, ergab das EXPLAIN-Statement folgende Informationen:

```
table type possible_keys               key  key_len ref  rows  Extra
et    ALL  PRIMARY                     NULL NULL    NULL 74
do    ALL  PRIMARY                     NULL NULL    NULL 2135
et_1  ALL  PRIMARY                     NULL NULL    NULL 74
tt    ALL  AssignedPC,ClientID,ActualPC NULL NULL    NULL 3872
      range checked for each record (key map: 35)
```

Weil type bei jeder Tabelle ALL ist, zeigt die Ausgabe, dass MySQL eine komplette Verknüpfung (Full Join) aller Tabellen durchführt! Das dauert recht lange, weil das Produkt der Zeilenanzahl in jeder Tabelle untersucht werden muss! In diesem Fall ist das 74 * 2.135 * 74 * 3.872 = 45.268.558.720 Zeilen. Wenn die Tabellen größer wären, können Sie sich vorstellen, wie lange das dauern würde.

Ein Problem liegt hier darin, dass MySQL (noch) keine Indexe auf Spalten effizient benutzen kann, wenn sie unterschiedlich deklariert sind. In diesem Zusammenhang sind VARCHAR und CHAR dasselbe, es sei denn, sie sind mit unterschiedlichen Längen deklariert. Weil tt.ActualPC als CHAR(10) und et.EMPLOYID als CHAR(15) deklariert ist, gibt eine Unstimmigkeit der Längen.

Um diese Ungleichheit der Spaltenlängen zu beheben, benutzen Sie ALTER TABLE, um ActualPC von 10 auf 15 Zeichen zu verlängern:

```
mysql> ALTER TABLE tt MODIFY ActualPC VARCHAR(15);
```

Jetzt sind tt.ActualPC und et.EMPLOYID beide VARCHAR(15). Eine erneute Ausführung des EXPLAIN-Statements ergibt dieses Ergebnis:

```
table type   possible_keys   key     key_len rew         rows    Extra
tt    ALL    AssignedPC,ClientID,ActualPC NULL NULL NULL 3872    where used
do    ALL    PRIMARY         NULL    NULL    NULL        2135
      range checked for each record (key map: 1)
et_1  ALL    PRIMARY         NULL    NULL    NULL        74
      range checked for each record (key map: 1)
et    eq_ref PRIMARY         PRIMARY 15      tt.ActualPC 1
```

Das ist nicht perfekt, aber viel besser (das Produkt der rows-Werte ist jetzt um einen Faktor 74 niedriger). Diese Version wird innerhalb von ein paar Sekunden ausgeführt.

Eine zweite Änderung kann durchgeführt werden, um die Unstimmigkeit der Spaltenlängen für die tt.AssignedPC = et_1.EMPLOYID- und tt.ClientID = do.CUSTNMBR-Vergleiche zu beheben:

```
mysql> ALTER TABLE tt MODIFY AssignedPC VARCHAR(15),
                      MODIFY ClientID   VARCHAR(15);
```

Jetzt ergibt EXPLAIN folgende Ausgabe:

```
table type   possible_keys   key     key_len rew           rows     Extra
et    ALL    PRIMARY         NULL    NULL    NULL          74
tt    rew    AssignedPC,ClientID,ActualPC ActualPC 15 et.EMPLOYID 52 where used
et_1  eq_ref PRIMARY         PRIMARY 15      tt.AssignedPC  1
do    eq_ref PRIMARY         PRIMARY 15      tt.ClientID    1
```

Das ist fast so gut, wie es überhaupt geht.

Das verbleibende Problem besteht darin, dass MySQL vorgabemäßig annimmt, dass die Werte in der tt.ActualPC-Spalte gleichmäßig verteilt sind, was in der tt-Tabelle nicht der Fall ist. Glücklicherweise ist es einfach, MySQL darüber zu informieren:

```
shell> myisamchk --analyze PFAD_ZU_MYSQL_DATENBANK/tt
shell> mysqladmin refresh
```

Jetzt ist der Join perfekt und EXPLAIN ergibt dieses Ergebnis:

```
table type   possible_keys   key     key_len ref           rows    Extra
tt    ALL    AssignedPC,ClientID,ActualPC NULL NULL NULL  3872    where used
et    eq_ref PRIMARY         PRIMARY 15      tt.ActualPC    1
et_1  eq_ref PRIMARY         PRIMARY 15      tt.AssignedPC  1
do    eq_ref PRIMARY         PRIMARY 15      tt.ClientID    1
```

Beachten Sie, dass die `rows`-Spalte in der Ausgabe von `EXPLAIN` eine gehobene Form von Vermutung des MySQL-Join-Optimierers ist. Um eine Anfrage zu optimieren, sollten Sie überprüfen, ob diese Zahlen der Wahrheit nahe kommen. Wenn nicht, erhalten Sie eventuell bessere Performance, wenn Sie `STRAIGHT_JOIN` in Ihrem `SELECT`-Statement benutzen und versuchen, die Tabellen in der `FROM`-Klausel in anderer Reihenfolge anzugeben.

6.2.2. Anfragen-Performance abschätzen

In den meisten Fällen können Sie die Performance schätzen, indem Sie Suchvorgänge auf Festplatte zählen. Bei kleinen Tabellen können Sie die Zeile üblicherweise mit 1 Festplatten-Suchvorgang finden (weil der Index wahrscheinlich im Cache ist). Bei größeren Tabellen können Sie schätzen, dass Sie (bei der Benutzung von B++-Baum-Indexen) brauchen werden: `log(zeilen_zahl) / log(index_block_laenge / 3 * 2 / (index_laenge + daten_zeiger_laenge)) + 1` Suchvorgänge, um die Zeile zu finden.

In MySQL ist ein Index-Block üblicherweise 1024 Bytes lang und der Daten-Zeiger üblicherweise 4 Bytes. Eine 500.000-Zeilen-Tabelle mit einer Indexlänge von 3 (medium integer) ergibt: `log(500.000)/log(1024/3*2/(3+4)) + 1` = 4 Suchvorgänge.

Da der obige Index etwa 500.000 * 7 * 3/2 = 5,2 MB benötigen würde (angenommen, dass die Index-Puffer zu 2/3 gefüllt sind, was ein typischer Wert sit), haben Sie wahrscheinlich viel vom Index im Arbeitsspeicher und werden wahrscheinlich nur 1 bis 2 Betriebssystem-Aufrufe benötigen, um Daten zu lesen, um die Zeile zu finden.

Bei Schreibvorgängen brauchen Sie jedoch 4 Suchanfragen (wie oben), um herauszufinden, wo der neue Index platziert wird, und normalerweise 2 Suchvorgänge, um den Index zu aktualisieren und die Zeile zu schreiben.

Beachten Sie, dass oben Gesagtes nicht bedeutet, dass Ihre Applikation allmählich mit N log N verfällt! Solange alles durch das Betriebssystem oder den SQL-Server gecachet wird, werden die Dinge nur marginal langsamer, wenn die Tabellen größer werden. Wenn die Daten zu Groß werden, um gecachet zu werden, werden die Dinge anfangen, viel langsamer zu laufen, bis Ihre Applikation schließlich komplett durch Suchvorgänge auf Festplatte ausgebremst wird (die mit N log N zunehmen). Um das zu vermeiden, vergrößern Sie den Index-Cache, wenn die Daten wachsen. See Abschnitt 6.5.2, „Serverparameter tunen“.

6.2.3. Geschwindigkeit von `SELECT`-Anfragen

Wenn Sie ein langsames `SELECT ... WHERE` schneller machen wollen, ist im Allgemeinen das erste, was zu prüfen ist, ob Sie einen Index hinzufügen können oder nicht. See Abschnitt 6.4.3, „Wie MySQL Indexe benutzt“. Alle Verweise (Reference) zwischen verschiedenen Tabellen sollten üblicherweise mit Indexen gemacht werden. Sie können den `EXPLAIN`-Befehl benutzen, um herauszufinden, welche Indexe für ein `SELECT` benutzt werden. See Abschnitt 6.2.1, „`EXPLAIN`-Syntax (Informationen über ein `SELECT` erhalten)“.

Einige allgemeine Tipps:

- Um MySQL zu helfen, Anfragen besser zu optimieren, lassen Sie `myisamchk --analyze` auf eine Tabelle laufen, nachdem sie mit relevanten Daten gefüllt wurde. Das aktualisiert einen Wert für jeden Index-Teil, der die durchschnittliche Anzahl von Zeilen angibt, die denselben Wert haben. (Bei eindeutigen Indexen ist das natürlich immer 1). MySQL benutzt diesen Wert, um zu entscheiden, welcher Index benutzt werden soll, wenn Sie zwei Tabellen mit einem 'nicht konstanten Ausdruck' verbinden. Sie können das Ergebnis nach dem Laufenlassen von `analyze` mit `SHOW INDEX FROM tabelle` überprüfen und die `Cardinality`-Spalte untersuchen.
- Um einen Index und Daten gemäß einem Index zu sortieren, benutzen Sie `myisamchk --sort-index --sort-records=1` (wenn Sie nach Index 1 sortieren wollen). Wenn Sie einen eindeutigen Index haben, von dem Sie alle Datensätze gemäß der Reihenfolge dieses Indexes lesen wollen, ist das eine gute Art, das schneller zu machen. Beachten Sie jedoch, dieses Sortieren nicht optimal geschrieben wird und bei einer großen Tabelle lange dauert!

6.2.4. Wie MySQL `WHERE`-Klauseln optimiert

Die `WHERE`-Optimierungen wurden hier in den `SELECT`-Teil aufgenommen, weil sie meist in Verbindung mit `SELECT` benutzt werden, aber dieselben Optimierungen treffen für `WHERE` bei `DELETE`- und `UPDATE`-Statements zu.

Beachten Sie auch, dass dieser Abschnitt nicht vollständig ist. MySQL führt viele Optimierungen durch und wir hatten noch keine Zeit, alle davon zu dokumentieren.

Einige der Optimierungen, die MySQL durchführt, sind unten aufgeführt:

- Entfernung unnötiger Klammern:

```
((a AND b) AND c OR (((a AND b) AND (c AND d))))
```

```
-> (a AND b AND c) OR (a AND b AND c AND d)
```

- Konstanten-'Falten' (Folding):

```
   (a<b AND b=c) AND a=5
-> b>5 AND b=c AND a=5
```

- Bedingungsentfernung bei Konstanten (notwendig wegen Konstanten-'Falten'):

```
   (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
-> B=5 OR B=6
```

- Konstante Ausdrücke, die von Indexen benutzt werden, werden nur einmal ausgewertet.
- `COUNT(*)` auf eine einzelne Tabelle ohne ein `WHERE` wird direkt aus der Tabelleninformation abgerufen. Das wird auch bei jeglichen `NOT NULL`-Ausdrücken gemacht, wenn diese nur für eine Tabelle benutzt werden.
- Früherkennung ungültiger Konstanten-Ausdrücke. MySQL stellt schnell fest, dass einige `SELECT`-Statements unmöglich sind, und gibt keine Zeilen zurück.
- `HAVING` wird mit `WHERE` vereinigt, wenn Sie `GROUP BY` oder Gruppen-Funktionen (`COUNT()`, `MIN()` usw.) nicht benutzen.
- Für jeden Sub-Join wird ein einfacheres `WHERE` konstruiert, um eine schnelle `WHERE`-Evaluierung für jeden Sub-Join zu erzielen, und auch, um Datensätze so bald wie möglich überspringen zu können.
- Alle Konstanten-Tabellen werden zuerst gelesen, vor jeder anderen Tabelle in der Anfrage. Eine Konstanten-Tabelle ist:
  - Eine leere Tabelle oder eine Tabelle mit 1 Zeile.
  - Eine Tabelle, die bei einer `WHERE`-Klausel auf einen `UNIQUE`-Index oder einen `PRIMARY KEY` benutzt wird, wobei alle Index-Teile mit konstanten Ausdrücken benutzt werden und die Index-Teile als `NOT NULL` definiert sind.

  Alle folgenden Tabellen werden als Konstanten-Tabellen benutzt:

```
mysql> SELECT * FROM t WHERE primary_key=1;
mysql> SELECT * FROM t1,t2
           WHERE t1.primary_key=1 AND t2.primary_key=t1.id;
```

- Die beste Join-Kombination, um Tabellen zu verknüpfen, wird gefunden, wenn man alle Möglichkeiten probiert. Wenn alle Spalten in `ORDER BY` und in `GROUP BY` aus derselben Tabelle stammen, wird diese Tabelle vorzugsweise vorn hingestellt, wenn verknüpft wird.
- Wenn es eine `ORDER BY`-Klausel und eine andere `GROUP BY`-Klausel gibt, oder wenn `ORDER BY` oder `GROUP BY` Spalten aus Tabellen enthält, die nicht aus der ersten Tabelle in der Join-Reihe stammen, wird eine temporäre Tabelle erzeugt.
- Wenn Sie `SQL_SMALL_RESULT` benutzen, benutzt MySQL eine temporäre Tabelle im Arbeitsspeicher.
- Jeder Tabellen-Index wird abgefragt und der beste Index, der weniger als 30% der Zeilen überspannt, wird benutzt. Wenn ein solcher Index nicht gefunden werden kann, wird ein schneller Tabellenscan benutzt.
- In einigen Fällen kann MySQL Zeilen vom Index lesen, ohne überhaupt in der Daten-Datei nachzuschlagen. Wenn alle Spalten, die vom Index benutzt werden, numerisch sind, wird nur der Index-Baum benutzt, um die Anfrage aufzulösen.
- Bevor jeder Datensatz herausgegeben wird, werden die, die nicht mit der `HAVING`-Klausel übereinstimmen, übersprungen.

Einige Beispiele von Anfragen, die sehr schnell sind:

```
mysql> SELECT COUNT(*) FROM tabelle;
mysql> SELECT MIN(schluessel_teil1),MAX(schluessel_teil1) FROM tabelle;
mysql> SELECT MAX(schluessel_teil2) FROM tabelle
           WHERE schluessel_teil_1=konstante;
mysql> SELECT ... FROM tabelle
           ORDER BY schluessel_teil1,schluessel_teil2,... LIMIT 10;
mysql> SELECT ... FROM tabelle
           ORDER BY schluessel_teil1 DESC,schluessel_teil2 DESC,... LIMIT 10;
```

Die folgenden Anfragen werden aufgelöst, indem nur der Index-Baum benutzt wird (unter der Annahme, dass die indizierten Spalten numerisch sind):

```
mysql> SELECT schluessel_teil1,schluessel_teil2 FROM tabelle WHERE schluessel_teil1=val;
```

```
mysql> SELECT COUNT(*) FROM tabelle
           WHERE schluessel_teil1=val1 AND schluessel_teil2=val2;
mysql> SELECT schluessel_teil2 FROM tabelle GROUP BY schluessel_teil1;
```

Die folgenden Anfragen benutzen Indexierung, um die Zeilen in sortierter Reihenfolge abzufragen, ohne einen separaten Sortierdurchgang:

```
mysql> SELECT ... FROM tabelle ORDER BY schluessel_teil1,schluessel_teil2,... ;
mysql> SELECT ... FROM tabelle ORDER BY schluessel_teil1 DESC,schluessel_teil2 DESC,... ;
```

6.2.5. Wie MySQL `DISTINCT` optimiert

`DISTINCT` wird für alle Spalten in `GROUP BY` umgewandelt, `DISTINCT` in Kombination mit `ORDER BY` benötigt in vielen Fällen ebenfalls eine temporäre Tabelle.

Wenn `LIMIT #` mit `DISTINCT` kombiniert wird, hält MySQL an, sobald er `#` eindeutige Zeilen findet.

Wenn Sie nicht Spalten aus allen benutzten Tabellen verwenden, hält MySQL mit dem Scannen der nicht benutzten Tabellen an, sobald er die erste Übereinstimmung gefunden hat.

```
SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;
```

Im Beispiel wird angenommen, dass t1 vor t2 benutzt wird (überprüfen Sie das mit `EXPLAIN`). In diesem Fall hört MySQL auf, von t2 zu lesen (für diese bestimmte Zeile in t1), sobald die erste Zeile in t2 gefunden wurde.

6.2.6. Wie MySQL `LEFT JOIN` optimiert

`A LEFT JOIN B` ist in MySQL wie folgt implementiert:

- Die Tabelle `B` wird als abhängig von Tabelle `A` und allen Tabellen, von denen `A` abhängig ist, gesetzt.
- Die Tabelle `A` wird als abhängig von allen Tabellen (ausser `B`) gesetzt, die in der `LEFT JOIN`-Bedingung aufgeführt sind.
- Alle `LEFT JOIN`-Bedingungen werden zu `WHERE`-Klausel verschoben.
- Alle Standard-Join-Optimierungen werden durchgeführt, mit der Ausnahme, dass eine Tabelle immer nach allen Tabellen gelesen wird, von denen sie abhängig ist. Wenn es eine zirkuläre Abhängigkeit gibt, gibt MySQL einen Fehler aus.
- Alle Standard-`WHERE`-Optimierungen werden durchgeführt.
- Wenn es eine Zeile in `A` gibt, die mit der `WHERE`-Klausel übereinstimmt, aber keine Zeile in `B`, die mit der `LEFT JOIN`-Bedingung übereinstimmt, wird eine zusätzliche Zeile für `B` erzeugt, deren Spalten alle auf `NULL` gesetzt sind.
- Wenn Sie `LEFT JOIN` benutzen, um Zeilen zu finden, die in einer Tabelle nicht existieren, und Sie folgendes im `WHERE`-Teil angeben: `spalten_name IS NULL`, wobei spalten_name eine Spalte ist, die als `NOT NULL` deklariert ist, hört MySQL mit der Suche nach weiteren Zeilen auf (für eine bestimmte Schlüsselkombination), nachdem er eine Zeile gefunden hat, die mit der `LEFT JOIN`-Bedingung übereinstimmt.

`RIGHT JOIN` ist analog zu `LEFT JOIN` implementiert.

Die Lese-Reihenfolge der Tabellen, die von `LEFT JOIN` und `STRAIGHT JOIN` erzwungen wird, hilft dem Optimierer (der berechnet, in welcher Reihenfolge die Tabellen verknüpft werden sollen), seine Arbeit schneller durchzuführen, weil weniger Tabellenvertauschungen überprüft werden müssen.

Beachten Sie, dass das oben Gesagte bedeutet, dass bei einer Anfrage des folgenden Typs:

```
SELECT * FROM a,b LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key) WHERE b.key=d.key
```

MySQL einen kompletten Scan von `b` durchführen wird, weil der `LEFT JOIN` erzwingt, dass diese vor `d` gelesen wird.

Das läßt sich in diesem Fall beheben, indem die Anfrage wie folgt geändert wird:

```
SELECT * FROM b,a LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key) WHERE b.key=d.key
```

6.2.7. Wie MySQL `LIMIT` optimiert

In einigen Fällen handhabt MySQL die Anfrage unterschiedlich, wenn Sie `LIMIT #` statt `HAVING` benutzen:

- Wenn Sie nur einige wenige Zeilen mit `LIMIT` auswählen, benutzt MySQL in einigen Fällen Indexe, wenn er ansonsten vorzugsweise einen vollständigen Tabellenscan durchführen würde.
- Wenn Sie `LIMIT #` mit `ORDER BY` benutzen, beendet MySQL das Sortieren, sobald er die ersten `#` Zeilen gefunden hat, anstatt die gesamte Tabelle zu sortieren.
- Wenn Sie `LIMIT #` mit `DISTINCT` kombinieren, hört MySQL auf, sobald er `#` eindeutige Zeilen gefunden hat.
- In einigen Fällen kann `GROUP BY` aufgelöst werden, indem der Schlüssel in der Reihenfolge gelesen wird (oder der Schlüssel sortiert wird) und danach Zusammenfassungen berechnet werden, bis sich der Schlüsselwert ändert. In diesem Fall berechnet `LIMIT #` keine unnötigen `GROUP BY`'s.
- Sobald MySQL die ersten `#` Zeilen an den Client geschickt hat, wird die Anfrage abgebrochen.
- `LIMIT 0` gibt immer schnell eine leere Ergebnismenge (empty set) zurück. Das ist nützlich, um die Anfrage zu überprüfen und die Spaltentypen der Ergebnisspalten zu erhalten.
- Die Größe der temporären Tabellen benutzt `LIMIT #`, um zu berechnen, wieviel Platz benötigt wird, um die Anfrage aufzulösen.

6.2.8. Geschwindigkeit von `INSERT`-Anfragen

Die Zeit, einen Datensatz einzufügen, besteht ungefähr aus:

- Verbindung: (3)
- Anfrage an den Server schicken: (2)
- Anfrage parsen: (2)
- Datensatz einfügen: (1 x Größe des Datensatzes)
- Indexe einfügen: (1 x Anzahl der Indexe)
- Schließen: (1)

Wobei die Zahlen in etwa proportional zur Gesamtzeit sind. Diese Berechnung zieht den anfänglichen Overhead, um Tabellen zu öffnen, nicht in Betracht (was einmal für jede gleichzeitig laufende Anfrage gemacht wird).

Die Größe der Tabelle verlangsamt das Einfügen von Indexen um N log N (B-Bäume).

Einige Möglichkeiten, die Geschwindigkeit von Einfügeoperationen zu steigern:

- Wenn Sie viele Zeilen vom selben Client aus zur gleichen Zeit einfügen, benutzen Sie mehrfache Werte (Liste) im `INSERT`-Statements. Das geht viel schneller (in manchen Fälle um Faktoren) als separate `INSERT`-Statements zu benutzen. Tunen Sie die `myisam_bulk_insert_tree_size`-Variable, um das sogar noch zu beschleunigen. See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“.
- Wenn Sie viele Zeilen von unterschiedlichen Clients aus einfügen, können Sie mehr Geschwindigkeit erzielen, wenn Sie das `INSERT DELAYED`-Statement benutzen. See Abschnitt 7.4.3, „`HANDLER`-Syntax“.
- Beachten Sie, dass Sie mit `MyISAM`-Tabellen Zeilen zur selben Zeit einfügen können, zu der `SELECT`s laufen, wenn es keine gelöschten Zeilen in den Tabellen gibt.
- Wenn Daten in eine Tabelle aus einer Textdatei eingeladen werden, benutzen Sie `LOAD DATA INFILE`. Das ist üblicherweise 20 mal schneller als viele `INSERT`-Statements zu benutzen. See Abschnitt 7.4.9, „`LOAD DATA INFILE`-Syntax“.
- Mit etwas zusätzlicher Mühe ist es möglich, `LOAD DATA INFILE` noch schneller laufen zu lassen, wenn die Tabelle viele Indexe hat. Gehen Sie wie folgt vor:

  1. Optional erzeugen Sie die Tabelle mit `CREATE TABLE`, zum Beispiel mit `mysql` oder über die Perl-DBI.
  2. Führen Sie ein `FLUSH TABLES`-Statement oder den Shell-Befehl `mysqladmin flush-tables` aus.
  3. Geben Sie `myisamchk --keys-used=0 -rq /pfad/zu/db/tabelle` ein. Dadurch entfernen Sie die Benutzung aller Indexe von der Tabelle.

4. Fügen Sie Daten in die Tabelle mit `LOAD DATA INFILE` ein. Dadurch werden keine Indexe aktualisiert, was deswegen sehr schnell läuft.

5. Wenn Sie in Zukunft nur noch aus der Tabelle lesen, benutzen Sie `myisampack`, um sie kleiner zu machen. See Abschnitt 8.1.2.3, „Kennzeichen komprimierter Tabellen“.

6. Erzeugen Sie die Indexe mit `myisamchk -r -q /pfad/zu/db/tabelle` neu. Hierdurch wird der Index-Baum im Speicher erzeugt, bevor er auf die Platte geschrieben wird, was viel schneller ist, weil viele Suchvorgänge auf Platte vermieden werden. Der sich ergebende Index-Baum ist ausserdem perfekt ausbalanciert.

7. Führen Sie ein `FLUSH TABLES`-Statement oder den Shell-Befehl `mysqladmin flush-tables` aus.

Diese Prozedur wird in Zukunft in `LOAD DATA INFILE` eingebaut werden.

Ab **MySQL 4.0** können Sie auch `ALTER TABLE tabelle DISABLE KEYS` anstelle von `myisamchk --keys-used=0 -rq /pfad/zu/db/tabelle` und `ALTER TABLE tabelle ENABLE KEYS` anstelle von `myisamchk -r -q /pfad/zu/db/tabelle` benutzen. Damit können Sie auch die `FLUSH TABLES`-Schritte überspringen.

- Sie können die Einfügegeschwindigkeit steigern, indem Sie Tabellen sperren:

```
mysql> LOCK TABLES a WRITE;
mysql> INSERT INTO a VALUES (1,23),(2,34),(4,33);
mysql> INSERT INTO a VALUES (8,26),(6,29);
mysql> UNLOCK TABLES;
```

Der hauptsächliche Geschwindigkeitsunterschied liegt darin, dass der Index-Puffer nur einmal auf Platte zurück geschrieben wird, nachdem alle `INSERT`-Statements fertig sind. Normalerweise würden die Index-Puffer so oft zurück geschrieben wie es `INSERT`-Statements gibt. Das Sperren wird nicht benötigt, wenn Sie alle Zeilen mit einem einzigen Statement einfügen können.

Durch das Sperren wird auch die Gesamtzeit von Tests auf mehrere Verbindungen gesenkt, aber die maximale Wartezeit für einige Threads wird erhöht (weil sie auf Sperren warten). Beispiel:

```
Thread 1 führt 1000 Einfügevorgänge durch.
Thread 2, 3 und 4 fügen 1 Einfügevorgang durch.
Thread 5 führt 1000 Einfügevorgänge durch.
```

Wenn Sie kein Sperren benutzen, sind die Threads 2, 3 und 4 vor 1 und 5 fertig. Wenn Sie Sperren benutzen, sind 2, 3 und 4 wahrscheinlich nicht vor 1 oder 5 fertig, aber die Gesamtzeit sollte etwa 40% geringer sein.

Weil `INSERT`-, `UPDATE`- und `DELETE`-Operationen in MySQL sehr schnell sind, erhalten Sie bessere Performance über alles, wenn Sie um alles herum Sperren hinzufügen, was mehr als etwa 5 Einfügeoperationen oder Aktualisierungen (Updates) in einer Zeile durchführt. Wenn Sie sehr viele Einfügeoperationen in einer Zeile durchführen, können Sie ein `LOCK TABLES` machen, gefolgt von einem gelegentlichen `UNLOCK TABLES` (etwa alle 1.000 Zeilen), um anderen Threads zu gestatten, auf die Tabelle zuzugreifen. Das Ergebnis wäre ebenfalls ein netter Geschwindigkeitsgewinn.

Natürlich ist `LOAD DATA INFILE` zum Einladen von Daten viel schneller.

Um sowohl für `LOAD DATA INFILE` als auch für `INSERT` mehr Geschwindigkeit zu erzielen, vergrößern Sie den Schlüssel-Puffer. See Abschnitt 6.5.2, „Serverparameter tunen“.

6.2.9. Geschwindigkeit von `UPDATE`-Anfragen

Update-Anfragen werden wie eine `SELECT`-Anfrage optimiert, mit dem zusätzlichen Overhead eines Schreibvorgangs. Die Geschwindigkeit des Schreibvorgangs hängt von der Größe der Daten und von der Anzahl der Indexe, die aktualisiert werden, ab. Indexe, die nicht geändert werden, werden nicht aktualisiert.

Eine weitere Möglichkeit, Aktualisierungen (Updates) schnell zu machen, ist, sie zu verzögern und dann später viele Aktualisierungen hintereinander zu machen. Viele Aktualisierungen hintereinander sind viel schneller als nur eine zugleich, wenn Sie die Tabelle sperren.

Beachten Sie, dass die Aktualisierung eines Datensatzes bei dynamischem Datensatzformat dazu führen kann, dass der Datensatz aufgespalten wird. Wenn Sie das oft durchführen, ist es daher sehr wichtig, gelegentlich `OPTIMIZE TABLE` auszuführen. See Abschnitt 5.5.1, „`OPTIMIZE TABLE`-Syntax“.

6.2.10. Geschwindigkeit von `DELETE`-Anfragen

Wenn Sie alle Zeilen in der Tabelle löschen wollen, sollten Sie `TRUNCATE TABLE tabelle` benutzen. See Abschnitt 7.4.7, „`TRUNCATE`-Syntax“.

Die Zeit, die für das Löschen eines Datensatzes benötigt wird, ist exakt proportional zur Anzahl der Indexe. Um Datensätze schneller zu löschen, können Sie die Größe des Index-Caches herauf setzen. See Abschnitt 6.5.2, „Serverparameter tunen“.

6.2.11. Weitere Optimierungstipps

Ungeordnete Liste von Tipps für schnellere Systeme:

- Benutzen Sie persistente Verbindungen zur Datenbank, um Verbindungs-Overhead zu vermeiden. Wenn Sie keine persistenten Verbindungen benutzen können und viele neue Verbindungen zur Datenbank aufmachen, sollten Sie den Wert der `Thread_cache_size`-Variablen ändern. See Abschnitt 6.5.2, „Serverparameter tunen“.
- Überprüfen Sie immer, dass alle Ihre Anfragen tatsächlich die Indexe benutzen, die Sie in den Tabellen erzeugt haben. In MySQL kann man das mit dem `EXPLAIN`-Befehl tun. See Explain: (Handbuch) Explain.
- Versuchen Sie, komplexe `SELECT`-Anfragen auf Tabellen zu vermeiden, die viel aktualisiert werden, um Probleme mit Tabellensperren zu vermeiden.
- Die neuen `MyISAM`-Tabellen können Zeilen in eine Tabelle ohne gelöschte Zeile zur gleichen Zeit einfügen, wie eine andere Tabelle aus ihr liest. Wenn das für Sie wichtig ist, sollten Sie Methoden in Betracht ziehen, bei denen Sie keine Zeilen löschen müssen, oder `OPTIMIZE TABLE` laufen lassen, nachdem Sie viele Zeilen gelöscht haben.
- Benutzen Sie `ALTER TABLE ... ORDER BY ausdruck1,ausdruck2,...`, wenn Sie Zeilen zumeist in der Reihenfolge ausdruck1,ausdruck2,... abrufen. Wenn Sie diese Option nach großen Änderungen in der Tabelle nutzen, erzielen Sie eventuell höhere Performance.
- In einigen Fällen kann es sinnvoll sein, eine Spalte einzuführen, die auf der Grundlage von Informationen aus anderen Spalten 'gehashed' ist. Wenn diese Spalte kurz und halbwegs eindeutig ist, kann das schneller sein als ein großer Index auf mehrere Spalten. In MySQL ist es sehr einfach, eine solche zusätzliche Spalte zu benutzen: `SELECT * FROM tabelle WHERE hash=MD5(concat(spalte1,spalte2)) AND spalte_1='constant' AND spalte_2='constant'`
- Bei Tabellen, die sich viel ändern, sollten Sie versuchen, alle `VARCHAR`- oder `BLOB`-Spalten zu vermeiden. Sonst erhalten Sie dynamische Zeilenlängen, sobald Sie eine einzige `VARCHAR`- oder `BLOB`-Spalte verwenden. See Kapitel 8, *MySQL-Tabellentypen*.
- Normalerweise nützt es nichts, eine Tabelle in verschiedene Tabellen aufzuteilen, nur weil die Zeile 'viel' werden. Um auf eine Zeile zuzugreifen, ist das wichtigste, was die Performance betrifft, der Suchvorgang nach dem ersten Byte der Zeile auf der Platte. Nachdem die Daten gefunden wurden, können die meisten neuen Platten die gesamte Zeile für die meisten Applikationen schnell genug lesen. Der einzige Fall, wo es wirklich etwas ausmacht, wenn eine Tabelle aufgeteilt wird, ist, wenn die Tabelle dynamische Zeilenlänge hat (siehe oben), was nicht in eine feste Zeilenlänge umgewandelt werden kann, oder wenn Sie die Tabelle sehr oft scannen müssen, die meisten der Spalten hierfür aber nicht benötigen. See Kapitel 8, *MySQL-Tabellentypen*.
- Wenn Sie sehr oft etwas auf der Grundlage von Informationen aus sehr vielen Zeilen berechnen müssen (zum Beispiel Dinge zählen), ist es wahrscheinlich besser, eine neue Tabelle einzuführen und den Zähler in Echtzeit zu aktualisieren. Eine Aktualisierung des Typs `UPDATE tabelle set zaehler=zaehler+1 where index_spalte=konstante` ist sehr schnell!

  Das ist sehr wichtig, wenn Sie Datenbanken wie MySQL benutzen, die nur Tabellensperren haben (viele Leser / einzelne Schreiber). Bei den meisten sonstigen Datenbanken ergibt das ebenfalls bessere Performance, weil der Zeilensperr-Manager weniger zu tun haben wird.
- Wenn Sie Statistiken aus großen Log-Tabellen gewinnen wollen, benutzen Sie Zusammenfassungstabellen, statt die gesamte Tabelle zu scannen. Die Wartung der Zusammenfassungen sollte wesentlich leichter sein, als die Statistiken 'live' zu generieren. Es ist viel schneller, neue Zusammenfassungstabellen aus den Logs zu erzeugen, wenn sich Dinge ändern (abhängig von Geschäftsentscheidungen) als eine laufende Applikation ändern zu müssen!
- Wenn möglich sollte man Berichte als 'live' oder 'statistisch' klassifizieren, wobei die Daten, die für statistische Berichte benötigt werden, nur auf der Grundlage von Zusammenfassungstabellen erzeugt werden, die aus den eigentlichen Daten generiert werden.
- Ziehen Sie Vorteile aus der Tatsache, dass Spalten Vorgabewerte haben. Fügen Sie nur dann explizit Werte ein, wenn der einzufügende Wert vom Vorgabewert abweicht. Das verringert das Parsen, das MySQL durchführen muss, und erhöht die Einfügegeschwindigkeit.
- In einigen Fällen ist es bequem, Daten zu komprimieren und in einem Blob zu speichern. In diesem Fall müssen Sie in Ihrer Applikation etwas zusätzlichen Code unterbringen, um die Dinge im Blob zu packen bzw. zu entpacken. Das kann aber in

manchen Phasen etliches an Zugriffen einsparen. Das ist praktisch, wenn Sie Daten haben, die mit einer statischen Tabellenstruktur nicht konform sind.

- Normalerweise sollten Sie versuchen, alle Daten nicht redundant zu halten (was sich in der Datenbanktheorie dritte Normalform nennt). Scheuen Sie sich aber nicht davor, Dinge zu duplizieren oder Zusammenfassungstabellen zu erzeugen, wenn Sie dies brauchen, um mehr Geschwindigkeit zu erzielen.

- Gespeicherte Prozeduren (Stored Procedures) oder UDF (user defined functions, benutzerdefinierte Funktionen) sind eine gute Möglichkeit, mehr Performance zu erzielen. Sie sollten jedoch immer eine andere (langsamere) Möglichkeit parat haben, wenn Sie eine Datenbank benutzen, die gespeicherte Prozeduren nicht unterstützt.

- Man erreicht immer etwas, wenn man Anfragen / Antworten in der Applikation cachet und versucht, viele Einfüge- oder Aktualisierungsvorgänge zugleich durchzuführen. Wenn Ihre Datenbank Tabellensperren unterstützt (wie MySQL und Oracle), sollte das dazu führen, dass der Index-Cache nur einmal auf Platte zurück geschrieben wird, nachdem alles Einfügen / Aktualisieren ausgeführt ist.

- Benutzen Sie `INSERT /*! DELAYED */`, wenn Sie nicht wissen brauchen, wann Ihre Daten geschrieben werden. Das erhöht die Geschwindigkeit, weil viele Datensätze mit einem einzige Festplattenschreibzugriff geschrieben werden können.

- Benutzten Sie `INSERT /*! LOW_PRIORITY */`, wenn Sie wollen, dass Ihre Selects höhere Priorität haben.

- Benutzen Sie `SELECT /*! HIGH_PRIORITY */`, um zu bewirken, dass Selects in der Wartereihe nach vorn springen. Das heißt, der Select wird sogar dann durchgeführt, wenn jemand darauf wartet, etwas zu schreiben.

- Benutzen Sie das mehrzeilige `INSERT`-Statement, um viele Zeilen mit einem SQL-Befehl zu speichern (viele SQL-Server unterstützen das).

- Benutzen Sie `LOAD DATA INFILE`, um größere Datenmengen zu laden. Das ist schneller als normale Einfügevorgänge und wird noch schneller, wenn `myisamchk` in `mysqld` integriert wird.

- Benutzen Sie `AUTO_INCREMENT`-Spalten, um eindeutige Werte zu erzeugen.

- Benutzen Sie gelegentlich `OPTIMIZE TABLE`, um Fragmentierungen zu vermeiden, wenn Sie das dynamische Tabellenformat verwenden.

  See Abschnitt 5.5.1, „`OPTIMIZE TABLE`-Syntax“.

- Benutzen Sie - wenn möglich - `HEAP`-Tabellen, um mehr Geschwindigkeit zu erzielen. See Kapitel 8, *MySQL-Tabellentypen*.

- Bei einer normalen Webserver-Konfiguration sollten Bilder als separate Dateien gespeichert werden. Das heißt, speichern Sie nur einen Verweis zur Datei in der Datenbank. Der Hauptgrund ist, dass normale Webserver viel besser darin sind, Dateien zu cachen als Datenbankinhalte. Daher ist es viel einfacher, ein schnelles System zu bekommen, wenn Sie Dateien benutzen.

- Benutzen Sie für nicht kritische Daten, auf die oft zugegriffen wird, Tabellen im Arbeitsspeicher (zum Beispiel Informationen über die Banner, die Benutzern ohne Cookies zuletzt präsentiert wurden).

- Spalten mit identischen Informationen in unterschiedlichen Tabellen sollten identisch deklariert sein und identische Namen haben. Vor Version 3.23 konnte man ansonsten langsame Joins erhalten.

  Versuchen Sie, die Namen einfach zu halten (benutzen Sie `name` anstelle von `kunde_name` in der Kundentabelle). Um Namen für andere SQL-Server portabel zu halten, sollten Sie sie kürzer als 18 Zeichen halten.

- Wenn Sie WIRKLICH hohe Geschwindigkeit brauchen, sollten Sie einen Blick auf die Low-Level-Schnittstellen zur Datenspeicherung werfen, die die unterschiedlichen SQL-Server unterstützen! Wenn Sie zum Beispiel auf `MyISAM` direkt zugreifen, erhalten Sie eine Geschwindigkeitssteigerung um den Faktor 2 bis 5, im Vergleich zur Benutzung der SQL-Schnittstelle. Um das durchführen zu können, müssen die Daten auf demselben Server liegen wie die Applikation und üblicherweise sollte auf sie nur von einem Prozess zugegriffen werden (weil externes Dateisperren reichlich langsam ist). Man könnte die oben genannten Probleme beseitigen, indem Low-Level-`MyISAM`-Befehle in den MySQL-Server eingebaut werden (das wäre eine einfache Möglichkeit, bei Bedarf mehr Performance zu erlangen). Indem die Datenbankshnittstelle sorgfältig entworfen wird, sollte es recht einfach sein, diese Arten von Optimierung zu unterstützen.

- In vielen Fällen ist es schneller, auf Daten aus einer Datenbank (mit einer direkten Verbindung) als über eine Textdatei zuzugreifen, schon deshalb, weil die Datenbank wahrscheinlich kompakter ist als die Textdatei (wenn Sie numerische Daten benutzen) und hierdurch weniger Festplattenzugriffe erforderlich sind. Ausserdem wird Code eingespart, weil Sie Ihre Textdateien nicht parsen müssen, um Zeilen- und Spaltenbegrenzungen zu finden.

- Ausserdem können Sie Replikation benutzen, um die Geschwindigkeit zu steigern. See Abschnitt 5.10, „Replikation bei MySQL“.

- Wenn eine Tabelle mit `DELAY_KEY_WRITE=1` deklariert wird, werden Aktualisierungen auf Indexe schneller, weil diese nicht auf Platte geschrieben werden, bis die Datei geschlossen wird. Der Nachteil ist, dass Sie auf diesen Tabellen `myisamchk`

laufen lassen sollten, bevor Sie `mysqld` starten, um sicherzustellen, dass diese in Ordnung sind, falls irgend etwas `mysqld` mittendrin killt. Weil die Schlüssel-Informationen jederzeit aus den Daten erzeugt werden können, sollten Sie durch `DELAY_KEY_WRITE` nichts verlieren.

6.3. Sperren (Locking)

6.3.1. Wie MySQL Tabellen sperrt

Im Anhang finden Sie eine Erörterung zu den unterschiedlichen Sperrmethoden. See Abschnitt E.4, „Sperrmethoden“.

Jedes Sperren in MySQL ist blockierungsfrei. Das wird erreicht, indem alle Sperren zugleich am Anfang einer Anfrage angefordert werden, und indem Tabellen immer in derselben Reihenfolge gesperrt werden.

The Sperrmethode, die MySQL für `WRITE`-Sperren benutzt, funktioniert wie folgt:

- Falls es keine Sperren auf die Tabelle gibt, wird eine Schreibsperre gemacht.
- Ansonsten wird die Sperranforderung in die Schreibsperren-Warteschlange eingereiht.

Die Sperrmethode, die MySQL für `READ`Sperren benutzt, funktioniert wie folgt:

- Falls es keine Schreibsperren auf die Tabelle gibt, wird eine Lesesperre gemacht.
- Ansonsten wird die Sperranforderung in die Lesesperren-Warteschlange eingereiht.

Wenn eine Sperre aufgehoben wird, wird die Sperren den Threads in der Schreibsperren-Warteschlange verfügbar gemacht, danach den Threads in der Lesesperren-Warteschlange.

Das bedeutet, wenn Sie viele Aktualisierungen auf eine Tabelle haben, warten `SELECT`-Statements, bis es keine Aktualisierungen mehr gibt.

Um das für den Fall zu umgehen, dass es viele `INSERT`- und-`SELECT`-Operationen auf eine Tabelle gibt, können Sie Zeilen in eine temporäre Tabelle einfügen und die echte Tabelle gelegentlich aus den Daten der temporäre Tabelle aktualisieren.

Das machen Sie wie folgt:

```
mysql> LOCK TABLES echte_tabelle WRITE, einfuege_tabelle WRITE;
mysql> insert into echte_tabelle select * von einfuege_tabelle;
mysql> TRUNCATE TABLE einfuege_tabelle;
mysql> UNLOCK TABLES;
```

Sie können bei `INSERT`, `UPDATE` oder `DELETE` die `LOW_PRIORITY`-Option oder bei `SELECT` die `HIGH_PRIORITY`-Option benutzen, wenn Sie dem Abruf von Daten in bestimmten Fällen Priorität einräumen wollen. Sie können auch `mysqld` mit `--low-priority-updates` starten, um dasselbe Verhalten zu erreichen.

Die Benutzung von `SQL_BUFFER_RESULT` kann ebenfalls helfen, Tabellensperren kürzer zu machen. See Abschnitt 7.4.1, „`SELECT`-Syntax“.

Sie können auch den Sperr-Code in `mysys/thr_lock.c` ändern, um eine einzige Warteschlagen zu benutzen. In diesem Fall haben Schreibsperren und Lesesperren dieselbe Priorität, was bei einigen Applikationen eventuell hilfreich ist.

6.3.2. Themen, die Tabellensperren betreffen

Der Tabellensperren-Code in MySQL ist blockierungsfrei.

MySQL benutzt Tabellensperren (anstelle von Zeilensperren oder Spaltensperren) für alle Tabellentypen ausser `BDB`-Tabellen, um eine sehr hohe Sperrgeschwindigkeit zu erzielen. Bei großen Tabellen ist Tabellensperren bei den meisten Applikationen VIEL besser als Zeilensperren, aber es gibt natürlich ein paar Fallstricke.

Bei `BDB`- und `InnoDB`-Tabellen benutzt MySQL Tabellensperren, wenn Sie die Tabelle explizit mit `LOCK TABLES` sperren oder einen Befehl ausführen, der jede Zeile in der Tabelle ändern wird, wie `ALTER TABLE`. Bei diesen Tabellentypen empfehlen wir, `LOCK TABLES` überhaupt nicht zu benutzen.

Ab MySQL-Version 3.23.7 können Sie Zeilen in `MyISAM`-Tabellen zur gleichen Zeit einfügen, während andere Threads aus der Tabelle lesen. Beachten Sie, dass das momentan nur funktioniert, wenn es zu der Zeit, zu der das Einfügen vorgenommen wird,

keine durch gelöschte Zeilen verursachte Löcher in der Tabelle gibt. Wenn alle Löcher mit neuen Daten gefüllt wurden, werden gleichzeitige Einfügevorgänge automatisch wieder aktiviert.

Tabellensperren ermöglicht, dass viele Threads gleichzeitig aus einer Tabelle lesen, aber bevor ein Thread in die Tabelle schreiben kann, muss er zunächst exklusiven Zugriff erhalten. Während der Aktualisierung müssen andere Threads, die auf diese Tabelle zugreifen wollen, warten, bis die Aktualisierung fertig ist.

Weil Aktualisierung von Tabellen normalerweise als wichtiger erachtet werden als `SELECT`, erhalten alle Statements, die eine Tabelle aktualisieren, eine höhere Priorität als Statements, die Informationen aus der Tabelle abrufen. Das sollte sicherstellen, dass Aktualisierungen nicht 'verhungern', wenn viele große Anfragen auf eine bestimmte Tabelle durchgeführt werden. (Sie können das ändern, indem Sie bei dem Statement, dass die Aktualisierung durchführt, LOW_PRIORITY verwenden, oder beim `SELECT`-Statement `HIGH_PRIORITY`.)

Ab MySQL-Version 3.23.7 können Sie die `max_write_lock_count`-Variable benutzen, um MySQL zu zwingen, temporär allen `SELECT`-Statements, die auf eine Tabelle warten, nach einer bestimmten Anzahl von Einfügevorgängen auf eine Tabelle höhere Priorität einzuräumen.

Tabellensperren ist jedoch bei folgendem Szenario nicht sehr gut:

- Ein Client führt ein `SELECT` aus, das lange Zeit läuft.
- Ein anderer Client führt danach ein `UPDATE` auf die benutzte Tabelle aus. Dieser Client wartet, bis das `SELECT` fertig ist.
- Ein weiterer Client führt ein weiteres `SELECT`-Statement auf dieselbe Tabelle aus. Weil `UPDATE` höhere Priorität als `SELECT` hat, wartet dieses `SELECT`, bis das `UPDATE` fertig ist. Es wartet auch darauf, dass das erste `SELECT` fertig ist!
- Ein Thread wartet bei etwas wie `Platte voll`. In diesem Fall warten alle anderen Threads, die auf die problemverursachende Tabelle zugreifen wollen, bis mehr Speicher verfügbar gemacht wurde.

Mögliche Lösungen dieses Problems sind:

- Versuchen Sie, `SELECT`-Statements schneller ablaufen zu lassen. Hierfür müssen Sie eventuell Zusammenfassungstabellen erzeugen.
- Starten Sie `mysqld` mit `--low-priority-updates`. Das gibt allen Statements, die eine Tabelle aktualisieren (ändern), geringere Priorität als einem `SELECT`-Statement. Im vorstehenden Szenario würde das `SELECT`-Statement vor dem `INSERT`-Statement ausgeführt werden.
- Sie können auch einem bestimmten `INSERT`-, `UPDATE`- oder `DELETE`-Statement mit dem `LOW_PRIORITY`-Attribut geringere Priorität geben.
- Starten Sie `mysqld` mit einem niedrigen Wert für **max_write_lock_count**, um `READ`-Sperren nach einer bestimmten Anzahl von `WRITE`-Sperren zu ermöglichen.
- Sie können festlegen, dass alle Aktualisierungen von einem bestimmten Thread mit niedriger Priorität ausgeführt werden, indem Sie den SQL-Befehl `SET SQL_LOW_PRIORITY_UPDATES=1` benutzen. See Abschnitt 6.5.6, „SET-Syntax“.
- Sie können mit dem `HIGH_PRIORITY`-Attribut festlegen, dass ein bestimmtes `SELECT` sehr wichtig ist. See Abschnitt 7.4.1, „SELECT-Syntax“.
- Wenn Sie Probleme mit `INSERT` in Kombination mit `SELECT` haben, stellen Sie auf die neuen `MyISAM`-Tabellen um, weil diese gleichzeitige `SELECT`s und `INSERT`s unterstützen.
- Wenn Sie hauptsächlich `INSERT`- und `SELECT`-Statements mischen, wird das `DELAYED`-Attribut für `INSERT` wahrscheinlich Ihre Probleme lösen. See Abschnitt 7.4.3, „HANDLER-Syntax“.
- Wenn Sie Probleme mit `SELECT` und `DELETE` haben, mag die `LIMIT`-Option für `DELETE` helfen. See Abschnitt 7.4.6, „DELETE-Syntax“.

6.4. Optimierung der Datenbank-Struktur

6.4.1. MySQL-Datenbank-Design-Überlegungen

MySQL speichert Zeilendaten und Indexdaten in separaten Dateien. Viele (fast alle) anderen Datenbanken vermischen Zeilen- und Indexdaten in derselben Datei. Wir glauben, dass die Wahl, die MySQL getroffen hat, für einen sehr weiten Bereich moderner Systeme besser ist.

Eine weitere Möglichkeit, Zeilendaten zu speichern, besteht darin, die Information für jede Spalten in einem separaten Bereich zu halten (Beispiele sind SDBM und Focus). Das verursacht Performance-Einbussen für jede Anfrage, die auf mehr als eine Spalte zugreift. Weil das so schnell schlechter wird, wenn auf mehr als eine Spalte zugegriffen wird, glauben wir, dass dieses Modell für Mehrzweck-Datenbanken nicht gut ist.

Der häufigere Fall ist, dass Index und Daten zusammen gespeichert sind (wie bei Oracle, Sybase usw.). In diesem Fall befindet sich die Zeileninformation auf der Leaf-Page des Indexes. Das Gute daran ist, dass man sich damit - abhängig davon, wie gut der Index gecachet ist - einen Festplatten-Lesezugriff spart. Das Schlechte an diesem Layout sind folgende Dinge:

- Tabellenscannen geht viel langsamer, weil man durch alle Indexe lesen muss, um an die Daten zu kommen.
- Man kann nicht nur die Index-Tabelle benutzen, um Daten einer Anfrage abzurufen.
- Man verliert viel Speicherplatz, weil man Indexe von den Nodes duplizieren muss (weil man die Zeile nicht in den Nodes speichern kann).
- Löschvorgänge werden die Tabelle im Zeitablauf zersetzen (weil Indexe in Nodes üblicherweise bei Löschvorgängen nicht aktualisiert werden).
- Ist es schwieriger, NUR die Index-Daten zu cachen.

6.4.2. Wie Sie Ihre Daten so klein wie möglich bekommen

Eine der grundlegendsten Optimierungen besteht darin, Ihre Daten (und Indexe) dazu zu bekommen, dass sie möglichst wenige Platz auf der Platte (und im Arbeitsspeicher) benutzen. Das kann zu gewaltigen Verbesserungen führen, weil Lesezugriffe von der Platte schneller ablaufen und normalerweise weniger Hauptspeicher benutzt wird. Das Indexieren nimmt darüber hinaus weniger Ressourcen in Anspruch, wenn es auf kleinere Spalten durchgeführt wird.

MySQL unterstützt viele verschiedene Tabellentypen und Zeilenformate. Wenn Sie das richtige Tabellenformat benutzen, kann Ihnen das große Performance-Gewinne bringen. See Kapitel 8, *MySQL-Tabellentypen*.

Sie erhalten bessere Performance auf eine Tabelle und minimieren den benötigten Speicherplatz, wenn Sie die unten aufgeführten Techniken verwenden:

- Benutzen Sie die effizientesten (kleinsten) möglichen Typen. MySQL hat viele spezialisierte Typen, die Plattenplatz und Arbeitsspeicher sparen.
- Benutzen Sie - falls möglich - die kleineren Ganzzahl-Typen, um kleinere Tabellen zu erhalten. `MEDIUMINT` zum Beispiel ist oft besser als `INT`.
- Deklarieren Sie Spalten - falls möglich - als `NOT NULL`. Das macht alles schneller und Sie sparen ein Bit pro Spalte. Beachten Sie, dass, wenn Sie wirklich `NULL` in Ihrer Applikation benötigen, Sie dieses natürlich benutzen sollten. Vermeiden Sie nur, einfach alle Spalten vorgabemäßig auf `NULL` zu haben.
- Wenn Sie keine Spalten variabler Länge haben (`VARCHAR`, `TEXT` oder `BLOB`-Spalten), wird ein Festgrößenformat benutzt. Das ist schneller, mag aber leider etwas Speicherplatz verschwenden. See Abschnitt 8.1.2, „MyISAM-Tabellenformate“.
- Der primäre Index einer Tabelle sollte so kurz wie möglich sein. Das macht die Identifikation einer Zeile schnell und effizient.
- Bei jeder Tabelle müssen Sie entscheiden, welche Speicher- / Index-Methode benutzt werden soll. See Kapitel 8, *MySQL-Tabellentypen*.
- Erzeugen Sie nur die Indexe, die Sie tatsächlich brauchen. Indexe sind gut für das Abfragen von Daten, aber schlecht, wenn Sie Dinge schnell speichern müssen. Wenn Sie meist auf eine Tabelle zugreifen, indem Sie nach einer Kombination von Spalten suchen, legen Sie einen Index auf diese. Der erste Index-Teil sollte die meistbenutzte Spalte sein. Wenn Sie IMMER viele Spalten benutzen, sollten Sie die Spalte zuerst benutzen, die mehr Duplikate hat, um eine bessere Kompression des Indexes zu erzielen.
- Wenn es sehr wahrscheinlich ist, dass eine Spalte ein eindeutiges Präfix auf der ersten Anzahl von Zeichen hat, ist es besser, nur dieses Präfix zu indexieren. MySQL unterstützt einen Index auf einem Teil einer Zeichen-Spalte. Kürzere Indexe sind nicht nur schneller, weil sie weniger Plattenplatz brauchen, sondern auch, weil Sie mehr Treffer im Index-Cache erhalten und daher weniger Festplattenzugriffe benötigen. See Abschnitt 6.5.2, „Serverparameter tunen“.
- Unter manchen Umständen kann es vorteilhaft sein, eine Tabelle zu teilen, die sehr oft gescannt wird. Das gilt insbesondere, wenn diese ein dynamisches Tabellenformat hat und es möglich ist, durch die Zerlegung eine kleinere Tabelle mit statischem Format zu erhalten, die benutzt werden kann, um die relevanten Zeilen zu finden.

6.4.3. Wie MySQL Indexe benutzt

Indexe werden benutzt, um Zeilen mit einem bestimmten Spaltenwert schnell zu finden. Ohne Index müsste MySQL mit dem ersten Datensatz anfangen und dann durch die gesamte Tabelle lesen, bis er die relevanten Zeilen findet. Je größer die Tabelle, desto mehr Zeit kostet das. Wenn die Tabellen für die infrage kommenden Zeilen einen Index hat, kann MySQL schnell eine Position bekommen, um mitten in der Daten-Datei loszusuchen, ohne alle Daten zu betrachten. Wenn eine Tabelle 1.000 Zeilen hat, ist das mindestens 100 mal schneller als sequentielles Lesen. Wenn Sie jedoch auf fast alle 1.000 Zeilen zugreifen müssen, geht sequentielles Lesen schneller, weil man mehrfache Festplattenzugriffe einspart.

Alle MySQL-Indexe (`PRIMARY`, `UNIQUE` und `INDEX`) sind in B-Bäumen gespeichert. Zeichenketten werden automatisch präfix-komprimiert, ebenfalls werden Leerzeichen am Ende komprimiert.

See Abschnitt 7.5.7, „`CREATE INDEX`-Syntax“.

Indexe werden benutzt, um:

- Schnell die Zeilen zu finden, die mit einer `WHERE`-Klausel übereinstimmen.
- Zeilen aus anderen Tabellen abzurufen, wenn Sie Joins durchführen.
- Den `MAX()`- oder `MIN()`-Wert für eine spezielle indizierte Spalte zu finden. Das wird durch einen Präprozessor optimiert, der überprüft, ob Sie `WHERE` schluessel_teil_# = constant auf allen Schlüsselteilen < N verwenden. In diesem Fall führt MySQL ein einzige Schlüsselnachschlagen durch und ersetzt den `MIN()`-Ausdruck mit einer Konstanten. Wenn alle Ausdrücke durch Konstanten ersetzt sind, gibt die Anfrage sofort ein Ergebnis zurück:

  ```
  SELECT MIN(schluessel_teil2),MAX(schluessel_teil2) FROM tabelle where schluessel_teil1=10
  ```

- Eine Tabelle zu sortieren oder zu gruppieren, wenn das Sortieren oder Gruppieren mit dem am weitesten links stehenden Präfix eines benutzbaren Schlüssels durchgeführt wird (zum Beispiel `ORDER BY schluessel_teil_1,schluessel_teil_2`). Der Schlüssel wird in umgekehrter Reihenfolge gelesen, wenn allen Schlüsselteilen `DESC` folgt.

  Der Index kann auch benutzt werden, selbst wenn `ORDER BY` nicht exakt mit dem Index übereinstimmt, solange alle unbenutzten Indexteile und alle zusätzlichen `ORDER BY`-Spalten Konstanten in der `WHERE`-Klausel sind. Folgende Anfragen werden einen Index benutzen, um den `ORDER BY`-Teil aufzulösen:

  ```
  SELECT * FROM foo ORDER BY schluessel_teil1,schluessel_teil2,schluessel_teil3;
  SELECT * FROM foo WHERE spalte=konstante ORDER BY spalte, schluessel_teil1;
  SELECT * FROM foo WHERE schluessel_teil1=konstante GROUP BY schluessel_teil2;
  ```

- In einigen Fällen kann eine Anfrage so optimiert werden, dass Sie Werte abruft, ohne in der Daten-Datei nachzuschlagen. Wenn alle benutzten Spalten einer Tabelle numerisch sind und ein ganz links stehendes Präfix für einen Schlüssel ergeben, können die Werte mit größerer Geschwindigkeit aus dem Index-Baum abgerufen werden:

  ```
  SELECT schluessel_teil3 FROM tabelle WHERE schluessel_teil1=1
  ```

Angenommen, Sie führen folgendes `SELECT`-Statement aus:

```
mysql> SELECT * FROM tabelle WHERE spalte1=val1 AND spalte2=val2;
```

Wenn es einen mehrspaltigen Index auf `spalte1` und `spalte2` gibt, können die entsprechenden Zeilen direkt geholt werden. Wenn es separate einspaltige Indexe auf `spalte1` und `spalte2` gibt, versucht der Optimierer, den restriktivsten Index zu finden, indem er entscheidet, welcher Index weniger Zeilen finden wird, und diesen Index dann benutzen, um Zeilen abzurufen.

Wenn die Tabelle einen mehrspaltigen Index hat, kann jedes Präfix auf der linken Seite vom Optimierer verwendet werden, um Zeilen zu finden. Wenn Sie zum Beispiel einen dreispaltigen Index auf `(spalte1,spalte2,spalte3)` haben, haben Sie Suchmöglichkeiten auf `(spalte1)`, `(spalte1,spalte2)` und `(spalte1,spalte2,spalte3)` indiziert.

MySQL kann keinen teilweisen Index verwenden, wenn die Spalten kein ganz linkes Präfix des Indexes bilden. Angenommen, Sie haben folgende `SELECT`-Statements:

```
mysql> SELECT * FROM tabelle WHERE spalte1=wert1;
mysql> SELECT * FROM tabelle WHERE spalte2=wert2;
mysql> SELECT * FROM tabelle WHERE spalte2=wert2 AND spalte3=wert3;
```

Wenn es einen Index auf `(spalte1,spalte2,spalte3)` gibt, benutzt nur die erste der drei Anfragen den Index. Die zweite und dritte Anfrage umfassen indizierte Spalten, aber `(spalte2)` und `(spalte2,spalte3)` sind nicht die ganz linken Präfixe von `(spalte1,spalte2,spalte3)`.

MySQL benutzt Indexe auch für `LIKE`-Vergleiche, wenn das Argument für `LIKE` eine Zeichenketten-Konstante ist, die nicht mit einem Platzhalterzeichen anfängt. Die folgenden `SELECT`-Statements zum Beispiel benutzen Indexe:

```
mysql> select * from tabelle where schluessel_spalte LIKE "Patrick%";
mysql> select * from tabelle where schluessel_spalte LIKE "Pat%_ck%";
```

Im ersten Statement werden nur Zeilen mit `"Patrick" <= schluessel_spalte < "Patricl"` berücksichtigt. Im zweiten Statement werden nur Zeilen mit `"Pat" <= schluessel_spalte < "Pau"` berücksichtigt.

Die folgenden `SELECT`-Statements benutzen keine Indexe:

```
mysql> select * from tabelle where schluessel_spalte LIKE "%Patrick%";
mysql> select * from tabelle where schluessel_spalte LIKE andere_spalte;
```

Im ersten Statement fängt der `LIKE`-Wert mit einem Platzhalterzeichen an. Im zweiten Statement ist der `LIKE`-Wert keine Konstante.

Suchen mit `spalte IS NULL` benutzt Indexe, wenn spalte ein Index ist.

MySQL benutzt normalerweise den Index, der die geringste Anzahl von Zeilen findet. Ein Index wird benutzt für Spalten, die Sie mit folgenden Operatoren vergleichen: `=`, `>`, `>=`, `<`, `<=`, `BETWEEN` und einem `LIKE` ohne Platzhalter-Präfix wie `'etwas%'`.

Jeder Index, der nicht alle `AND`-Ebenen in der `WHERE`-Klausel umfasst, wird nicht benutzt, um die Anfrage zu optimieren. Mit anderen Worte: Um einen Index benutzen zu können, muss ein Präfix des Indexes in jeder `AND`-Gruppe benutzt werden.

Die folgenden `WHERE`-Klauseln benutzen Indexe:

```
... WHERE index_teil1=1 AND index_teil2=2 AND andere_spalte=3
... WHERE index=1 OR A=10 AND index=2      /* index = 1 OR index = 2 */
... WHERE index_teil1='hello' AND index_teil_3=5
          /* optimiert "index_teil1='hello'" */
... WHERE index1=1 AND index2=2 OR index1=3 AND index3=3;
          /* kann den Index auf index1 benutzen, aber nicht auf index2 oder index 3 */
```

Die folgenden `WHERE`-Klauseln benutzen **KEINE** Indexe:

```
... WHERE index_teil2=1 AND index_teil3=2  /* index_teil_1 wird nicht benutzt */
... WHERE index=1 OR A=10                  /* Index wird nicht in beiden AND-Teilen benutzt */
... WHERE index_teil1=1 OR index_teil2=10  /* Kein Index umfasst alle Zeilen */
```

Beachten Sie, dass MySQL in manchen Fällen keinen Index benutzt, selbst wenn einer verfügbar wäre. Einige solcher Fälle sind hier aufgeführt:

- Wenn die Benutzung des Indexes erfordern würde, dass MySQL auf mehr als 30% der Zeilen in der Tabelle zugreift. (In diesem Fall ist ein Tabellenscan wahrscheinlich viel schneller, weil dieser weniger Festplattenzugriffe braucht.) Beachten Sie, dass MySQL den Index dennoch benutzt, wenn eine Anfrage `LIMIT` benutzt, um nur ein paar Zeilen abzufragen, weil er dann schneller die wenigen Zeilen im Ergebnis finden kann.

6.4.4. Spalten-Indexe

Alle MySQL-Spaltentypen können indiziert werden. Die Benutzung von Indexen auf den relevanten Spalten ist die beste Art, die Performance von `SELECT`-Operationen zu verbessern.

Die maximale Anzahl von Schlüsseln und die maximale Index-Länge ist durch den Tabellen-Handler vorgegeben. See Kapitel 8, *MySQL-Tabellentypen*. Bei allen Tabellen-Handlern können Sie zumindest 16 Schlüssel und eine Gesamtindexlänge von zumindest 256 Bytes haben.

Bei `CHAR`- und `VARCHAR`-Spalten können Sie ein Präfix einer Spalte indexieren. Das ist viel schneller und erfordert weniger Plattenspeicher als das Indexieren einer ganzen Spalte. Die Syntax, die im `CREATE TABLE`-Statement benutzt wird, um ein Spaltenpräfix zu indexieren, sieht wie folgt aus:

```
KEY index_name (spalten_name(laenge))
```

Das unten stehende Beispiel erzeugt einen Index auf die ersten 10 Zeichen der `name`-Spalte:

```
mysql> CREATE TABLE test (
           name CHAR(200) NOT NULL,
           KEY index_name (name(10)));
```

Bei BLOB- und TEXT-Spalten müssen Sie ein Präfix der Spalte indexieren. Sie können nicht die gesamte Spalte indexieren.

Ab MySQL-Version 3.23.23 können Sie auch spezielle **FULLTEXT**-Indexe erzeugen. Sie werden für die Volltextsuche benutzt. Nur der MyISAM-Tabellentyp unterstützt FULLTEXT-Indexe. Sie können nur auf VARCHAR- und TEXT-Spalten erzeugt werden. Die Indexierung erfolgt immer über die gesamte Spalte; teilweises Indexieren wird nicht unterstützt. Siehe Abschnitt 7.8, „MySQL-Volltextsuche“ für Details.

6.4.5. Mehrspaltige Indexe

MySQL kann Indexe auf mehrfache Spalten erzeugen. Ein Index darf aus bis zu 15 Spalten bestehen. (Auf CHAR- und VARCHAR-Spalten können Sie auch ein Präfix der Spalte als Teil eines Indexes benutzen).

Ein mehrspaltiger Index kann als sortiertes Array betrachtet werden, das Werte enthält, die durch die Verkettung der Werte der indizierten Spalten erzeugt werden.

MySQL benutzt mehrspaltige Indexe in einer Art, dass Anfragen schnell werden, wenn Sie eine bekannte Menge für die erste Spalte des Indexes in einer WHERE-Klausel angeben, selbst wenn Sie keine Werte für die anderen Spalten angeben.

Angenommen, einen Tabelle wurde wie folgt erzeugt:

```
mysql> CREATE TABLE test (
           id INT NOT NULL,
           nachname CHAR(30) NOT NULL,
           vorname CHAR(30) NOT NULL,
           PRIMARY KEY (id),
           INDEX name (nachname,vorname));
```

Dann ist der Index name ein Index über nachname und vorname. Der Index wird für Anfragen benutzt, die Werte in einem bekannten Bereich für nachname angeben, oder sowohl für nachname als auch für und vorname. Daher wird der name-Index in folgenden Anfragen benutzt:

```
mysql> SELECT * FROM test WHERE nachname="Widenius";

mysql> SELECT * FROM test WHERE nachname="Widenius"
                          AND vorname="Michael";

mysql> SELECT * FROM test WHERE nachname="Widenius"
                          AND (vorname="Michael" OR vorname="Monty");

mysql> SELECT * FROM test WHERE nachname="Widenius"
                          AND vorname >="M" AND vorname < "N";
```

In folgenden Anfragen wird der name-Index jedoch NICHT benutzt:

```
mysql> SELECT * FROM test WHERE vorname="Michael";

mysql> SELECT * FROM test WHERE nachname="Widenius"
                          OR vorname="Michael";
```

Weitere Informationen über die Art, wie MySQL Indexe benutzt, um die Anfragen-Performance zu verbessern, finden Sie unter Abschnitt 6.4.3, „Wie MySQL Indexe benutzt“.

6.4.6. Wie MySQL Tabellen öffnet und schließt

table_cache, max_connections und max_tmp_tables beeinflussen die maximale Anzahl von Dateien, die der Server offen halten kann. Wenn Sie einen oder mehrere dieser Werte erhöhen, können Sie an eine Begrenzung stoßen, die durch Ihr Betriebssystem in Bezug auf die Anzahl offener Datei-Deskriptoren pro Prozess festgelegt wird. Diese Begrenzung kann man jedoch auf vielen Systemen erhöhen. Sehen Sie im Handbuch Ihres Betriebssystems nach, wie man das macht, weil die Methode, wie die Begrenzung geändert wird, sich von System zu System stark unterscheidet.

table_cache ist verwandt mit max_connections. Für 200 gleichzeitig laufende Verbindungen sollten Sie zum Beispiel einen Tabellen-Cache von mindestens 200 * n haben, wobei n die maximale Anzahl von Tabellen in einem Join ist. Zusätzlich müssen Sie einige externe Datei-Deskriptoren für temporäre Tabellen und Dateien reservieren.

Stellen Sie sicher, dass Ihr Betriebssystem die Anzahl offener Datei-Deskriptoren handhaben kann, die durch die table_cache-Einstellung impliziert wird. Wenn table_cache zu hoch gesetzt wird, hat MySQL eventuell keine Datei-Deskriptoren mehr und verweigert Verbindungen, führt keine Anfragen mehr aus und läuft sehr unzuverlässig. Beachten Sie auch, dass der MyISAM-Tabellen-Handler zwei Datei-Deskriptoren für jede einzelne offene Tabelle benötigt. Sie können die Anzahl von Datei-Deskriptoren, die für MySQL verfügbar sind, in der --open-files-limit=#-Startoption angeben. See Abschnitt A.2.16, „File Not Found“.

Der Cache offener Tabellen kann bis auf table_cache anwachsen (Vorgabewert 64; das kann mit der -O Tabellen-Cache=#-Option für mysqld geändert werden). Eine Tabelle wird nie geschlossen, ausser wen der Cache voll ist und ein anderer

Thread versucht, eine Tabelle zu öffnen, oder wenn Sie `mysqladmin refresh` oder `mysqladmin flush-tables` benutzen.

Wenn sich der Tabellen-Cache füllt, benutzt der Server folgenden Prozedur, um einen Cache-Eintrag für die Benutzung zu finden:

- Tabellen, die momentan nicht in Benutzung sind, werden freigegeben, in der Reihenfolge der kürzlich am wenigsten benutzten Tabellen.
- Wenn der Cache voll ist und keine Tabellen freigegeben werden können, aber eine neue Tabelle geöffnet werden muss, wird der Cache temporär wie benötigt vergrößert.
- Wenn der Cache gerade im Zustand temporärer Erweiterung ist und eine Tabelle vom Zustand benutzt in den Zustand nicht benutzt wechselt, wird die Tabelle geschlossen und vom Cache freigesetzt.

Eine Tabelle wird für jeden gleichzeitigen Zugriff geöffnet. Das bedeutet, dass die Tabelle zweimal geöffnet werden muss, wenn Sie zwei Threads haben, die auf dieselbe Tabelle zugreifen oder einen Thread, der auf die Tabelle zweimal in derselben Anfrage zugreift (mit `AS`). Das erste öffnen jeder Tabelle benötigt nur einen Datei-Deskriptor. Der zusätzliche Deskriptor wird für die Index-Datei benötigt; dieser Deskriptor wird mit allen Threads geteilt (shared).

Wenn Sie eine Tabelle mit dem `HANDLER tabelle OPEN`-Statement öffnen, wird dem Thread ein dediziertes Tabellenobjekt zugewiesen. Diese Tabellenobjekt wird nicht mit anderen Threads geteilt und wird solange nicht geschlossen, bis der Thread `HANDLER tabelle CLOSE` aufruft oder stirbt. See Abschnitt 7.4.3, „HANDLER-Syntax“.

Sie können prüfen, ob Ihr Tabellen-Cache zu klein ist, indem Sie die mysqld-Variable `opened_tables` ansehen. Wenn diese recht Groß ist, selbst wenn Sie nicht viele `FLUSH TABLES` ausgeführt haben, sollten Sie Ihren Tabellen-Cache vergrößern. See Abschnitt 5.5.5.3, „SHOW STATUS“.

6.4.7. Nachteile der Erzeugung großer Mengen von Tabellen in derselben Datenbank

Wenn Sie viele Dateien in einem Verzeichnis haben, werden open-, close- und create-Operationen langsam. Wenn Sie ein `SELECT`-Statements auf viele unterschiedliche Tabellen ausführen, gibt es ein bisschen Overhead, wenn der Tabellen-Cache voll ist, weil für jede Tabelle, die geöffnet wird, eine andere geschlossen werden muss. Sie können diese Overhead verringern, indem Sie den Tabellen-Cache größer machen.

6.4.8. Warum gibt es so viele offene Tabellen?

Wenn Sie `mysqladmin status` ausführen, werden Sie etwa folgendes sehen:

```
Uptime: 426 Running Threads: 1 Questions: 11082 Reloads: 1 Open Tables: 12
```

Das kann etwas verwirrend sein, wenn Sie nur 6 Tabellen haben.

MySQL ist multi-threaded, daher kann er viele Anfragen auf dieselbe Tabelle simultan verarbeiten. Um das Problem zu minimieren, dass zwei Threads verschiedene Zustände in Bezug auf dieselbe Datei haben, wird die Tabelle unabhängig für jeden gleichzeitigen Thread geöffnet. Das benötigt etwas Arbeitsspeicher und einen externen Datei-Deskriptor für die Daten-Datei. Der Index-Datei-Deskriptor wird mit allen Threads geteilt.

6.5. Optimierung des MySQL-Servers

6.5.1. System / Kompilierzeitpunkt und Tuning der Startparameter

Wir fangen mit den Dingen auf Systemebene an, weil einige dieser Entscheidungen sehr früh getroffen werden müssen. In anderen Fällen mag ein kurzer Blick auf diesen Teil ausreichen, weil er nicht so wichtig für große Verbesserungen ist. Es ist jedoch immer nett, ein Gefühl dafür zu bekommen, wie viel man gewinnen kann, wenn man Dinge auf dieser Ebene ändert.

Es ist wirklich wichtig, dass vorgabemäßige Betriebssystem zu kennen! Um das meiste aus Mehrprozessor-Maschinen herauszuholen, sollte man Solaris benutzen (weil die Threads wirklich gut funktionieren) oder Linux (weil der 2.2-Kernel wirklich gute Mehrprozessor-Unterstützung bietet). Linux hat auf 32-Bit-Maschinen vorgabemäßig eine Dateigrößenbeschränkung von 2 GB. Das wird hoffentlich bald behoben, wenn neue Dateisysteme herausgebracht werden (XFS/Reiserfs). Wenn Sie dringen Unterstützung für größere Datei als 2 GB auf Linux-Intel-32-Bit benötigen, sollten Sie den LFS-Patch für das ext2-Dateisystem holen.

Weil wir MySQL noch nicht auf allzu vielen Plattformen in einer Produktionsumgebung getestet haben, empfehlen wir, dass Sie Ihre geplante Plattform testen, bevor Sie sich dafür entscheiden.

Weitere Tipps:

- Wenn Sie genug Arbeitsspeicher haben, könnten Sie alle Swap-Geräte entfernen. Einige Betriebssysteme benutzen in bestimmten Zusammenhängen ein Swap-Gerät, selbst wenn Sie freien Arbeitsspeicher haben.
- Benutzen Sie die `--skip-locking`-MySQL-Option, um externe Sperren zu vermeiden. Beachten Sie, dass das die Funktionalität von MySQL nicht tangiert, solange Sie nur einen Server laufen lassen. Denken Sie lediglich daran, den Server herunterzufahren (oder die relevanten Teile zu sperren), bevor Sie `myisamchk` laufen lassen. Auf manchen Systemen ist diese Umschaltung zwingend erforderlich, weil externes Sperren in keinem Fall funktioniert.

  Die `--skip-locking`-Option ist vorgabemäßig angeschaltet, wenn Sie mit MIT-pThreads kompilieren, weil `flock()` von MIT-pThreads nicht vollständig auf allen Plattformen unterstützt wird. Auch für Linux ist es vorgabemäßig angeschaltet, weil Linux-Dateisperren bis jetzt nicht zuverlässig funktionieren.

  Der einzige Fall, wo Sie `--skip-locking` nicht benutzen können, sit, wenn Sie mehrfache MySQL-*Server* (nicht Clients) auf denselben Daten laufen lassen, oder wenn Sie `myisamchk` auf eine Tabelle ausführen, ohne zuerst die `mysqld`-Server-Tabellen auf Platte zurückzuschreiben und zu sperren.

  Sie können immer noch `LOCK TABLES` / `UNLOCK TABLES` benutzen, selbst wenn Sie `--skip-locking` benutzen.

6.5.2. Serverparameter tunen

Sie erhalten die Puffer-Größen, die der `mysqld`-Server benutzt, mit diesem Befehl:

```
shell> mysqld --help
```

Dieser Befehl erzeugt eine Auflistung aller `mysqld`-Optionen und konfigurierbaren Variablen. Die Ausgabe enthält die Vorgabewerte und sieht etwa wie folgt aus:

```
Possible variables for option --set-variable (-O) are:
back_log               current value: 5
bdb_cache_size         current value: 1048540
binlog_cache_size      current_value: 32768
connect_timeout        current value: 5
delayed_insert_timeout  current value: 300
delayed_insert_limit   current value: 100
delayed_queue_size     current value: 1000
flush_time             current value: 0
interactive_timeout    current value: 28800
join_buffer_size       current value: 131072
key_buffer_size        current value: 1048540
lower_case_tabelles  current value: 0
long_query_time        current value: 10
max_allowed_packet     current value: 1048576
max_binlog_cache_size current_value: 4294967295
max_connections        current value: 100
max_connect_errors     current value: 10
max_delayed_threads    current value: 20
max_heap_table_size    current value: 16777216
max_join_size          current value: 4294967295
max_sort_length        current value: 1024
max_tmp_tables         current value: 32
max_write_lock_count   current value: 4294967295
myisam_sort_buffer_size  current value: 8388608
net_buffer_length      current value: 16384
net_retry_count        current value: 10
net_read_timeout       current value: 30
net_write_timeout      current value: 60
query_buffer_size      current value: 0
record_buffer          current value: 131072
record_rnd_buffer      current value: 131072
slow_launch_time       current value: 2
sort_buffer            current value: 2097116
table_cache            current value: 64
thread_concurrency     current value: 10
tmp_table_size         current value: 1048576
thread_stack           current value: 131072
wait_timeout           current value: 28800
```

Wenn aktuell ein `mysqld`-Server läuft, können Sie feststellen, welche Werte er für die Variablen tatsächlich benutzt, wenn Sie diesen Befehl ausführen:

```
shell> mysqladmin variables
```

Sie finden eine komplette Beschreibung aller Variablen im `SHOW VARIABLES`-Abschnitt dieses Handbuchs. See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“.

Wenn Sie `SHOW STATUS` eingeben, können Sie einige statistische Informationen des Servers sehen. See Abschnitt 5.5.5.3, „SHOW STATUS“.

MySQL benutzt Algorithmen, die sehr skalierbar sind, daher können Sie üblicherweise mit sehr wenig Arbeitsspeicher fahren. Wenn Sie MySQL jedoch mehr Speicher geben, erzielen Sie damit normalerweise auch bessere Performance.

Wenn Sie einen MySQL-Server tunen, sind die zwei wichtigsten Variablen `key_buffer_size` und `table_cache`. Sie sollten zunächst sicher sein, dass diese beiden richtig gesetzt sind, bevor Sie versuchen, irgend eine der anderen Variablen zu ändern.

Wenn Sie viel Arbeitsspeicher haben (>= 256 MB) und viele Tabellen und maximale Performance bei einer mäßigen Anzahl von Clients haben wollen, sollten Sie etwas wie das Folgende benutzen:

```
shell> safe_mysqld -O key_buffer=64M -O table_cache=256 \
           -O sort_buffer=4M -O record_buffer=1M &
```

Wenn Sie nur 128 MB und nur wenige Tabellen haben, aber viele Sortiervorgänge durchführen, können Sie etwas wie das Folgende benutzen:

```
shell> safe_mysqld -O key_buffer=16M -O sort_buffer=1M
```

Wenn Sie wenig Arbeitsspeicher und viele Verbindungen haben, können Sie etwas wie das Folgende benutzen:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=100k \
           -O record_buffer=100k &
```

Oder sogar:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=16k \
           -O table_cache=32 -O record_buffer=8k -O net_buffer=1K &
```

Wenn Sie `GROUP BY` oder `ORDER BY` auf Dateien anwenden, die größer als Ihr verfügbarer Arbeitsspeicher sind, sollten Sie den Wert von `record_rnd_buffer` heraufsetzen, um das Lesen von Zeilen nach Sortiervorgängen zu beschleunigen.

Wenn Sie MySQL installiert haben, enthält das `Support-files`-Verzeichnis einige unterschiedliche `my.cnf`-Beispiel-Dateien: `my-huge.cnf`, `my-large.cnf`, `my-medium.cnf` und `my-small.cnf`. Diese können Sie als Grundlage nehmen, um Ihr System zu optimieren.

Wenn es sehr viele Verbindungen gibt, können ``Swapping-Probleme" auftauchen, wen Sie `mysqld` nicht so konfiguriert haben, dass er für jede Verbindung sehr wenig Speicher benutzt. `mysqld` bringt natürlich bessere Leistungsdaten, wenn Sie genug Speicher für alle Verbindungen haben.

Beachten Sie, dass Änderungen einer Option für `mysqld` sich nur auf diese Instanz des Servers auswirken.

Um die Auswirkung einer Parameteränderung zu sehen, geben Sie folgendes ein:

```
shell> mysqld -O key_buffer=32m --help
```

Stellen Sie sicher, dass die `--help`-Option zuletzt kommt, ansonsten wird die Auswirkung jeglicher Optionen, die danach auf der Kommandozeile kommen, in der Ausgabe nicht gezeigt. output.

6.5.3. Wie Kompilieren und Linken die Geschwindigkeit von MySQL beeinflusst

Die meisten der folgenden Tests wurden mit den MySQL-Benchmarks unter Linux durchgeführt, aber sie sollten einen guten Anhaltspunkt für andere Betriebssysteme und Auslastungen geben.

Sie erhalten die schnellste ausführbare Datei, wenn Sie mit `-static` linken.

Unter Linux erhalten Sie den schnellsten Code, wenn Sie mit `pgcc` und `-O3` kompilieren. Um `sql_yacc.cc` mit diesen Optionen zu kompilieren, brauchen Sie etwa 200 MB Arbeitsspeicher, weil `gcc/pgcc` viel Speicher benötigt, um alle Funktionen inline zu machen. Sie sollten beim Konfigurieren von MySQL auch `CXX=gcc` setzen, um das Einschließen der `libstdc++`-Bibliothek zu vermeiden (die nicht benötigt wird). Beachten Sie, dass bei einigen Version von `pgcc` der erzeugte Code nur auf echten Pentium-Prozessoren läuft, selbst wenn Sie in den Compiler-Optionen angeben, dass Sie wollen, dass der Code auf alle Prozessoren vom Typ x586 läuft (wie AMD).

Einfach durch die Benutzung eines besseren Compilers und / oder besserer Compiler-Optionen können Sie eine 10-30%-ige Geschwindigkeitssteigerung in Ihrer Applikation erhalten. Das ist besonders wichtig, wenn Sie den SQL-Server selbst kompilieren!

Wir haben sowohl Cygnus CodeFusion als auch Fujitsu-Compiler getestet, aber es stellte sich heraus, dass keiner von beiden ausreichend Bug-frei war, damit MySQL mit angeschalteten Optimierungen kompiliert werden konnte.

Wenn Sie MySQL kompilieren, sollten Sie nur Unterstützung für die Zeichensätze einschließen, die Sie benutzen werden (Option `--with-charset=xxx`). Die Standard-MySQL-Binärdistributionen werden mit Unterstützung für alle Zeichensätze kompiliert.

Hier ist eine Auflistung einiger Messungen, die wir durchgeführt haben:

- Wenn Sie `pgcc` benutzen und alles mit `-O6` kompilieren, ist der `mysqld`-Server 1% schneller als mit `gcc` 2.95.2.
- Wenn Sie dynamisch linken (ohne `-static`), ist das Ergebnis unter Linux 13% langsamer. Beachten Sie, dass Sie dennoch dynamisch gelinkte MySQL-Bibliotheken benutzen können. Nur beim Server ist das kritisch in Bezug auf Performance.
- Wenn Sie Ihre `mysqld`-Binärdatei mit `strip libexec/mysqld` strippen, ist die resultierende Binärdatei bis zu 4% schneller.
- Wenn Sie sich über TCP/IP statt über Unix-Sockets verbinden, ist das auf demselben Computer 7,5% langsamer. (Wenn Sie sich zu `localhost` verbinden, benutzt MySQL vorgabemäßig Sockets.)
- Wenn Sie sich über TCP/IP von einem anderen Computer über ein 100-MBit-Ethernet verbinden, ist das 8% bis 11% langsamer.
- Wenn Sie mit `--with-debug=full` kompilieren, verlangsamen sich die meisten Anfragen um 20%, manche Anfragen jedoch werden wesentlich langsamer (der MySQL-Benchmarks zeigte 35%). Wenn Sie `--with-debug` benutzen, beträgt die Verlangsamung nur 15%. Wenn Sie eine `mysqld`-Version, die mit `--with-debug=full` kompiliert wurde, mit `--skip-safemalloc` starten, ist die Geschwindigkeit etwa dasselbe, als wenn Sie mit `--with-debug` konfigurieren.
- Auf einer Sun SPARCstation 20 ist SunPro C++ 4.2 5% schneller als `gcc` 2.95.2.
- Das Kompilieren mit `gcc` 2.95.2 für ultrasparc mit der Option `-mcpu=v8 -Wa,-xarch=v8plusa` ergibt 4% mehr Performance.
- Auf Solaris 2.5.1 sind MIT-pThreads 8% bis 12% langsamer als Solaris-native Threads, auf einem Einprozessorsystem. Bei mehr Last / Prozessoren sollte der Unterschied größer werden.
- Laufenlassen mit `--log-bin` macht **MySQL** 1% langsamer.
- Wenn beim Kompilieren unter Linux-x86 mit gcc keine Frame-Pointers `-fomit-frame-pointer` oder `-fomit-frame-pointer -ffixed-ebp` verwendet werden, ist `mysqld` 1% bis 4% schneller.

Die MySQL-Linux-Distribution, die von MySQL AB zur Verfügung gestellt wird, wurde früher mit `pgcc` kompiliert, aber wir mussten zum normalen gcc zurück gehen, weil es einen Bug in `pgcc` gibt, der Code erzeugt, der nicht auf AMD läuft. Wir werden gcc solange benutzen, bis dieser Bug behoben ist. Bis dahin können Sie, falls Sie keine AMD-Maschine haben, eine schnellere Binärdatei erhalten, wenn Sie mit `pgcc` kompilieren. Die Standard-MySQL-Linux-Binärdatei wird statisch gelinkt, um sie schneller und portierbarer zu machen.

6.5.4. Wie MySQL Speicher benutzt

Die unten stehende Liste zeigt einige Möglichkeiten, wie der `mysqld`-Server Speicher benutzt. Wo es zutrifft, wird der Name der für die Speicherbenutzung relevanten Servervariablen angegeben.

- Der Schlüssel-Puffer (Variable `key_buffer_size`) wird von allen Threads geteilt. Andere Puffer, die vom Server benutzt werden, werden bei Bedarf zugewiesen. See Abschnitt 6.5.2, „Serverparameter tunen“.
- Jede Verbindung benutzt etwas Thread-spezifischen Platz: Einen Stack (Vorgabe 64 KB, Variable `thread_stack`), einen Verbindungspuffer (Variable `net_buffer_length`) und a Ergebnispuffer (Variable `net_buffer_length`). Die Verbindungspuffer und Ergebnispuffer werden bei Bedarf dynamisch bis zu `max_allowed_packet` vergrößert. Wenn eine Anfrage läuft, wird auch eine Kopie der aktuellen Anfragezeichenkette zugewiesen.
- Alle Threads teilen sich denselben grundlegenden Speicher.
- Nur die komprimierten ISAM- / MyISAM-Tabellen werden Speicher-gemappt. Das liegt daran, dass der 32-Bit-Adressraum von 4 GB für die meisten großen Tabellen nicht Groß genug ist. Wenn Systeme mit 64-Bit-Adressraum gebräuchlicher werden, werden wir vielleicht eine allgemeine Unterstützung für Speicher-Mapping hinzufügen.
- Jeder Anfrage, die einen sequentiellen Scan über eine Tabelle durchführt, wird ein Lesepuffer zugewiesen (Variable `record_buffer`).

- Wenn Zeilen in 'zufälliger' Reihenfolge gelesen werden (zum Beispiel nach einem Sortiervorgang), wird ein Zufalls-Lesepuffer zugewiesen, um Suchvorgänge auf Festplatte zu vermeiden. (Variable `record_rnd_buffer`).

- Alle Joins werden in einem Durchgang durchgeführt und die meisten Joins können sogar ohne Benutzung einer temporären Tabelle durchgeführt werden. Die meisten temporären Tabellen sind Speicher-basierende (HEAP-) Tabellen. Temporäre Tabellen mit großer Datensatzlänge (berechnet als Summe aller Spaltenlängen) oder die `BLOB`-Spalten enthalten, werden auf Festplatte gespeichert.

  Ein Problem in MySQL-Versionen vor Version 3.23.2 ist, dass Sie den Fehler `The table tabelle is full` erhalten, wenn die Größe der HEAP-Tabelle `tmp_table_size` überschreitet. In neueren Versionen wird dies so gehandhabt, dass die Speicher-basierende (HEAP-) Tabelle bei Bedarf automatisch in eine Festplatten-basierende Tabelle (MyISAM) umgewandelt wird. Um das Problem zu umgehen, können Sie die Größe von temporären Tabellen durch Setzen der `tmp_table_size`-Option für `mysqld` ändern, oder durch Setzen der SQL-Option `SQL_BIG_TABLES` im Client-Programm. See Abschnitt 6.5.6, „SET-Syntax“. In MySQL-Version 3.20 war die maximale Größe der temporären Tabelle `record_buffer*16`. Wenn Sie also diese Version benutzen, müssen Sie den Wert von `record_buffer` herauf setzen. Sie können `mysqld` auch mit der `--big-tables`-Option starten, um temporäre Tabellen immer auf Festplatte zu speichern. Das wird jedoch die Geschwindigkeit vieler komplizierter Anfragen beeinflussen.

- Den meisten Sortier-Anfragen werden ein Sortierpuffer und 0 bis 2 temporäre Dateien zugewiesen, abhängig von der Größe der Ergebnismenge. See Abschnitt A.4.4, „Wohin MySQL temporäre Dateien speichert“.

- Fast alles Parsen und Berechnen wird in einem lokalen Speicherbereich durchgeführt. Für kleine Sachen wird kein Speicher-Overhead benötigt, und das normale, langsame Zuweisen und Freimachen von Speicher wird vermieden. Speicher wird nur für unerwartet lange Zeichenketten zugewiesen (das wird mit `malloc()` und `free()` gemacht).

- Jede Index-Datei wird einmal geöffnet. Die Daten-Datei wird einmal für jeden gleichzeitig laufenden Thread geöffnet. Für jeden gleichzeitigen Thread wird eine Tabellenstruktur, Spaltenstrukturen für jede Spalte und ein Puffer der Größe `3 * n` zugewiesen, wobei `n` die maximale Zeilenlänge ist (`BLOB`-Spalten werden nicht mitgerechnet). Eine `BLOB`-Spalte benutzt 5 bis 8 Bytes plus die Länge der `BLOB`-Daten. Der `ISAM`- / `MyISAM`-Tabellen-Handler benutzt einen zusätzlichen Zeilenpuffer für internen Gebrauch.

- Bei jeder Tabelle, die `BLOB`-Spalten enthält, wird ein Puffer dynamisch vergrößert, um größere `BLOB`-Werte einzulesen. Wenn Sie eine Tabelle scannen, wird ein Puffer so Groß wie der größte `BLOB`-Wert zugewiesen.

- Tabellen-Handler für alle Tabellen in Benutzung werden in einem Cache gespeichert und als FIFO verwaltet. Normalerweise hat der Cache 64 Einträge. Wenn eine Tabelle gleichzeitig von zwei laufenden Threads benutzt wurde, enthält der Cache zwei Einträge für die Tabelle. See Abschnitt 6.4.6, „Wie MySQL Tabellen öffnet und schließt“.

- Ein `mysqladmin flush-tables`-Befehl schließt alle Tabellen, die nicht in Benutzung sind, und kennzeichnet alle Tabellen in Benutzung als zu schließen, sobald der aktuell ausführende Thread fertig ist. Das setzt effektiv den meisten benutzten Speicher frei.

`ps` und andere System-Status-Programme berichten vielleicht, dass `mysqld` viel Arbeitsspeicher benutzt. Das kann durch Thread-Stacks auf verschiedenen Speicheradressen verursacht werden. `ps` der Solaris-Version zum Beispiel zählt den unbenutzten Speicher zwischen Stacks zum benutzten Speicher hinzu. Das können Sie bestätigen, wenn Sie den verfügbaren Swap mit `swap -s` überprüfen. Wir haben `mysqld` mit kommerziellen Memory-Leak-Detektoren getestet, daher sollte es keine Memory-Leaks geben.

6.5.5. Wie MySQL DNS benutzt

Wenn sich ein neuer Thread mit `mysqld` verbindet, erzeugt `mysqld` einen neuen Thread, um die Anfrage zu handhaben. Dieser Thread prüft zuerst, ob der Hostname im Hostnamen-Cache ist. Falls nicht, ruft der Thread `gethostbyaddr_r()` und `gethostbyname_r()` auf, um den Hostname aufzulösen.

Wenn das Betriebssystem die oben genannten Thread-sicheren Aufrufe nicht unterstützt, sperrt der Thread ein Mutex und ruft statt dessen `gethostbyaddr()` und `gethostbyname()` auf. Beachten Sie, dass in diesem Fall kein anderer Thread andere Hostnamen auflösen kann, die nicht im Hostnamen-Cache sind, bis der erste Thread fertig ist.

Sie können das DNS-Nachschlagen von Hostnamen (DNS-Lookup) abschalten, indem Sie `mysqld` mit `--skip-name-resolve` starten. In diesem Fall können Sie jedoch in den MySQL-Berechtigungstabellen nur IP-Nummern verwenden.

Wenn Sie ein sehr langsames DNS und viele Hosts haben, können Sie mehr Performance erzielen, wenn Sie entweder das DNS-Nachschlagen von Hostnamen (DNS-Lookup) abschalten (mit `--skip-name-resolve`) oder `HOST_CACHE_SIZE` (Vorgabe: 128) erhöhen und `mysqld` neu kompilieren.

Sie können den Hostnamen-Cache mit `--skip-host-cache` abschalten. Sie können den Hostnamen-Cache mit `FLUSH HOSTS` oder `mysqladmin flush-hosts` löschen.

Wenn Sie keine Verbindungen über TCP/IP zulassen wollen, starten Sie mysqld mit --skip-networking.

6.5.6. SET-Syntax

```
SET [OPTION] SQL_VALUE_OPTION= wert, ...
```

SET OPTION setzt verschiedene Optionen, die die Arbeitsweise des Servers oder Ihrer Clients beeinflussen. Jede Option, die Sie setzen, bleibt in Kraft, bis die aktuelle Sitzung beendet wird, oder bis Sie die Option auf einen anderen Wert setzen.

- `characterset zeichensatz_name | DEFAULT`

  Das mappt alle Zeichenketten von und zum Client auf das angegebene Mapping. Momentan ist die einzige Option für zeichensatz_name cp1251_koi8, aber Sie können leicht neue Mappings hinzufügen, indem Sie die sql/convert.cc-Datei in der MySQL-Quelldistribution editieren. Das vorgabemäßige Mapping kann durch Setzen des zeichensatz_name-Werts auf DEFAULT wieder hergestellt werden.

  Beachten Sie, dass sich die Syntax für das Setzen der characterset-Option von der Syntax für das Setzen anderer Optionen unterscheidet.

- `PASSWORD = PASSWORD('ein_passwort')`

  Setzt das Passwort für den aktuellen Benutzer. Jeder nicht anonyme Benutzer kann sein eigenes Passwort ändern!

- `PASSWORD FOR benutzer = PASSWORD('ein_passwort')`

  Setzt das Passwort für einen bestimmten Benutzer auf dem aktuellen Server-Host. Das kann nur ein Benutzer mit Zugriff auf die mysql-Datenbank tun. Der Benutzer sollte im user@hostname-Format eingegeben werden, wobei user und hostname exakt so sind, wie sie in den User- und Host-Spalten des mysql.user-Tabelleneintrags aufgelistet sind. Wenn Sie zum Beispiel in den Spalten User und Host die Einträge 'bob' und '%.loc.gov' haben wollen, schreiben Sie:

  ```
  mysql> SET PASSWORD FOR bob@"%.loc.gov" = PASSWORD("newpass");
  oder
  mysql> UPDATE mysql.user SET password=PASSWORD("newpass") where user="bob' und host="%.loc.gov";
  ```

- `SQL_AUTO_IS_NULL = 0 | 1`

  Falls auf 1 gesetzt (Vorgabe), wird mit folgendem Konstrukt die letzte eingefügte Zeile einer Tabelle mit einer auto_increment-Zeile gefunden: WHERE auto_increment_spalte IS NULL. Das wird von einigen ODBC-Programme wie Access benutzt.

- `AUTOCOMMIT= 0 | 1`

  Falls auf 1 gesetzt, werden alle Änderungen einer Tabelle auf einmal durchgeführt. Um eine Transaktion aus mehreren Befehlen anzufangen, müssen Sie das BEGIN-Statement benutzen. See Abschnitt 7.7.1, „BEGIN/COMMIT/ROLLBACK-Syntax“. Falls auf 0 gesetzt, müssen Sie COMMIT / ROLLBACK benutzen, um diese Transaktion zu akzeptieren / zu widerrufen. See Abschnitt 7.7.1, „BEGIN/COMMIT/ROLLBACK-Syntax“. Beachten Sie, dass MySQL nach dem Umschalten vom AUTOCOMMIT-Modus zum AUTOCOMMIT-Modus ein automatisches COMMIT auf alle offenen Transaktionen durchführt.

- `SQL_BIG_TABLES = 0 | 1`

  Falls auf 1 gesetzt, werden alle temporären Tabellen auf Platte statt im Arbeitsspeicher gespeichert. Das ist etwas langsamer, aber Sie erhalten nicht den Fehler The table tabelle is full, wenn Sie große SELECT-Operationen ausführen, die eine große temporäre Tabelle erfordern. Der Vorgabewert für eine neue Verbindung ist 0 (das heißt, temporäre Tabellen im Arbeitsspeicher benutzen).

- `SQL_BIG_SELECTS = 0 | 1`

  Falls auf 0 gesetzt, bricht MySQL ab, wenn ein SELECT versucht wird, das wahrscheinlich sehr lange dauern wird. Das ist nützlich, wenn ein unratsames WHERE-Statement abgesetzt wurde. Ein große Anfrage ist definiert als ein SELECT, das wahrscheinlich mehr als max_join_size Zeilen untersuchen muss. Der Vorgabewert für eine neue Verbindung ist 1 (was alle SELECT-Statements zuläßt).

- `SQL_BUFFER_RESULT = 0 | 1`

  SQL_BUFFER_RESULT erzwingt, dass das Ergebnis von SELECT's in eine temporäre Tabelle geschrieben wird. Das hilft MySQL, die Tabellensperren frühzeitig aufzuheben, und ist hilfreich in Fällen, wo es lange dauert, das Ergebnis an den Client zu senden.

- `SQL_LOW_PRIORITY_UPDATES = 0 | 1`

  Falls auf `1` gesetzt, warten alle `INSERT`-, `UPDATE`-, `DELETE`- und `LOCK TABLE WRITE`-Statements, bis es kein anhängiges `SELECT` oder `LOCK TABLE READ` für die betroffene Tabelle gibt.

- `SQL_MAX_JOIN_SIZE = wert | DEFAULT`

  Nicht zulassen, dass `SELECT`s, die wahrscheinlich mehr als `value` Zeilenkombinationen untersuchen müssen, ausgeführt werden. Wenn Sie diesen Wert setzen, können Sie `SELECT`s abfangen, bei denen Schlüssel nicht korrekt verwendet werden und die wahrscheinlich sehr lange dauern. Wenn dieser Wert auf etwas anderes als `DEFAULT` gesetzt wird, wird der `SQL_BIG_SELECTS`-Flag zurückgesetzt. Wenn Sie den `SQL_BIG_SELECTS`-Flag wieder setzen, wird die `SQL_MAX_JOIN_SIZE`-Variable ignoriert. Sie können für diese Variable einen Vorgabewert setzen, wenn Sie `mysqld` mit `-O max_join_size=#` starten.

- `SQL_SAFE_UPDATES = 0 | 1`

  Falls auf `1` gesetzt, bricht MySQL ab, wenn ein `UPDATE` oder `DELETE` versucht wird, das keinen Schlüssel oder kein `LIMIT` in der `WHERE`-Klausel benutzt. Das ermöglicht das Abfangen falscher Aktualisierungen, wenn SQL-Befehle von Hand eingegeben werden.

- `SQL_SELECT_LIMIT = wert | DEFAULT`

  Die maximale Anzahl von Datensätzen, die von `SELECT`-Statements zurückgegeben werden. Wenn ein `SELECT` eine `LIMIT`-Klausel hat, hat das `LIMIT` Vorrang vor dem Wert von `SQL_SELECT_LIMIT`. Der Vorgabewert für eine neue Verbindung ist ``unbegrenzt.'' Wenn Sie diese Begrenzung geändert haben, kann der Vorgabewert wieder hergestellt werden, indem Sie einen `SQL_SELECT_LIMIT`-Wert von `DEFAULT` verwenden.

- `SQL_LOG_OFF = 0 | 1`

  Falls auf `1` gesetzt, wird für diesen Client kein Loggen ins Standard-Log durchgeführt, wenn der Client die **process**-Berechtigung hat. Das betrifft nicht die Update-Log-Datei!

- `SQL_LOG_UPDATE = 0 | 1`

  Falls auf `0` gesetzt, wird für diesen Client kein Loggen in die Update-Log-Datei durchgeführt, wenn der Client die **process**-Berechtigung hat. Das betrifft nicht das Standard-Log!

- `SQL_QUOTE_SHOW_CREATE = 0 | 1`

  Falls auf `1` gesetzt, setzt `SHOW CREATE TABLE` Tabellen- und Spaltennamen in Anführungszeichen. Das ist vorgabemäßig **angeschaltet**, damit Replikation von Tabellen mit merkwürdigen Spaltennamen funktioniert. Abschnitt 5.5.5.8, „`SHOW CREATE TABLE`“.

- `TIMESTAMP = zeitstempel_wert | DEFAULT`

  Setzt die Zeit für diesen Client. Das wird benutzt, um den Original-Zeitstempel zu erhalten, wenn sie die Update-Log-Datei benutzen, um Zeilen wiederherzustellen. `zeitstempel_wert` sollte ein UNIX-Epoche-Zeitstempel sein, kein MySQL-Zeitstempel.

- `LAST_INSERT_ID = #`

  Setzt den Wert, der von `LAST_INSERT_ID()` zurückgegeben wird. Dieser wird in der Update-Log-Datei gespeichert, wenn Sie `LAST_INSERT_ID()` in einem Befehl benutzen, der eine Tabelle aktualisiert.

- `INSERT_ID = #`

  Setzt den Wert, der von einem folgenden `INSERT`- oder `ALTER TABLE`-Befehl benutzt wird, wenn ein `AUTO_INCREMENT`-Wert eingefügt wird. Das wird hauptsächlich zusammen mit der Update-Log-Datei benutzt.

6.6. Festplatte, Anmerkungen

- Wie bereits erwähnt sind Suchvorgänge auf der Festplatte ein großer Performance-Flaschenhals. Die Probleme werden mehr und mehr deutlich, wenn die Datenmenge wächst, so dass effizientes Caching unmöglich wird. Bei großen Datenbanken, in denen Sie auf Daten mehr oder weniger zufällig zugreifen, können Sie sicher davon ausgehen, dass Sie zumindest eine Plattenzugriff brauchen, um zu lesen, und eine Reihe weiterer Plattenzugriffe, um Dinge zu schreiben. Um dieses Problem zu minimieren, benutzen Sie Platten mit geringen Zugriffszeiten!

- Erhöhen Sie die Anzahl verfügbarer Festplattenscheiben (und verringern Sie dadurch den Such-Overhead), indem Sie entweder

Dateien auf andere Platten symbolisch verknüpfen (SymLink) oder die Platten 'stripen'.

- **Using Symbolische Links**

  Das bedeutet, dass Sie die Index- und / oder Daten-Datei(en) aus dem normalen Daten-Verzeichnis auf eine andere Festplatte verknüpfen (die auch 'gestriped' sein kann). Das macht sowohl den Suchvorgang als auch die Lesezeiten besser (wenn die Platten nicht für andere Dinge benutzt werden). See Abschnitt 6.6.1, „Symbolische Links benutzen“.

- **Stripen**

  'Stripen' heißt, dass Sie viele Festplatten haben und den ersten Block auf die erste Platte legen, den zweiten Block auf die zweite Platte und den n-ten Block auf die n-te Platte usw. Das bedeutet, wenn Ihre normale Datengröße weniger als die Stripe-Größe ist (oder perfekt passt), dass Sie wesentlich bessere Performance erhalten. Beachten Sie, dass Stripen sehr stark vom Betriebssystem und von der Stripe-Größe abhängig ist. Machen Sie Benchmark-Tests Ihrer Applikation mit unterschiedlichen Stripe-Größen. See Abschnitt 6.1.5, „Wie Sie Ihre eigenen Benchmarks benutzen“.

  Beachten Sie, dass der Geschwindigkeitsunterschied für das Stripen **sehr** stark vom Parameter abhängig ist. Abhängig davon, wie Sie den Stripe-Parameter setzen und von der Anzahl von Festplatten erhalten Sie Unterschiede in der Größenordnung von Faktoren. Beachten Sie, dass Sie entscheiden müssen, ob Sie für zufällige oder sequentielle Zugriffe optimieren.

- Aus Gründen der Zuverlässigkeit sollten sie vielleicht RAID 0 + 1 nehmen (Stripen + Spiegeln), doch in diesem Fall brauchen Sie 2 * n Laufwerke, um n Datenlaufwerke zu haben. Das ist wahrscheinlich die beste Option, wenn Sie genug Geld dafür haben! Sie müssen jedoch eventuell zusätzlich in Software für die Verwaltung von Volumes investieren, um das effizient zu handhaben.

- Eine gute Option ist es, nicht ganz so wichtige Daten (die wieder hergestellt werden können) auf RAID-0-Platten zu halten, während wirklich wichtige Daten (wie Host-Informationen und Log-Dateien) auf einer RAID-0+1- oder RAID-N-Platte gehalten werden. RAID-N kann ein Problem darstellen, wenn Sie viele Schreibzugriffe haben, weil Zeit benötigt wird, die Paritätsbits zu aktualisieren.

- Sie können auch den Parameter für das Dateisystem setzen, das die Datenbank benutzt. Eine einfache Änderung ist, das Dateisystem mit der noatime-Option zu mounten. Das bringt es dazu, das Aktualisieren der letzten Zugriffszeit in der Inode zu überspringen und vermeidet dadurch einige Platten-Suchzugriffe.

- Unter Linux können Sie viel mehr Performance erhalten (bis zu 100% unter Last ist nicht ungewöhnlich), wenn Sie hdpram benutzen, um die Schnittstelle Ihrer Festplatte zu konfigurieren! Das folgende Beispiel sollte recht gute hdparm-Optionen für MySQL (und wahrscheinlich viele andere Applikationen) darstellen:

  ```
  hdparm -m 16 -d 1
  ```

  Beachten Sie, dass Performance und Zuverlässigkeit beim oben Genannten von Ihrer Hardware abhängen, daher empfehlen wir sehr, dass Sie Ihr System gründlich testen, nachdem Sie `hdparm` benutzt haben! Sehen Sie in der Handbuchseite (ManPage) von `hdparm` nach weiteren Informationen! Wenn `hdparm` nicht vernünftig benutzt wird, kann das Ergebnis eine Beschädigung des Dateisystems sein. Machen Sie eine Datensicherung von allem, bevor Sie experimentieren!

- Auf vielen Betriebssystemen können Sie die Platten mit dem 'async'-Flag mounten, um das Dateisystem auf asynchrone Aktualisierung zu setzen. Wenn Ihr Computer ausreichend stabil ist, sollte Ihnen das mehr Performance geben, ohne zu viel Zuverlässigkeit zu opfern. (Dieser Flag ist unter Linux vorgabemäßig angeschaltet.)

- Wenn Sie nicht wissen müssen, wann auf eine Datei zuletzt zugegriffen wurden (was auf einem Datenbank-Server nicht wirklich nötig ist), können Sie Ihr Dateisystem mit dem noatime-Flag mounten.

6.6.1. Symbolische Links benutzen

Sie können Tabellen und Datenbanken vom Datenbank-Verzeichnis an andere Stellen verschieben und sie mit symbolischen Links auf neue Speicherorte ersetzen. Das könnten Sie zum Beispiel tun, um eine Datenbank auf ein Dateisystem mit mehr freiem Speicherplatz zu verlagern oder um die Geschwindigkeit Ihres System durch Verteilen Ihrer Tabellen auf unterschiedliche Platten zu steigern.

Die empfohlene Art, das zu tun, ist, nur Datenbanken auf unterschiedliche Platten per SymLink zu verknüpfen, und das bei Tabellen nur im Notfall zu tun.

6.6.1.1. Benutzung symbolischer Links für Datenbanken

Um eine Datenbank per SymLink zu verknüpfen, legt man zuerst ein Verzeichnis auf einer Platte mit freiem Speicherplatz an und erzeugt dann einen SymLink vom MySQL-Datenbank-Verzeichnis aus darauf:

```
shell> mkdir /dr1/datenbanken/test
shell> ln -s /dr1/datenbanken/test mysqld-datadir
```

MySQL unterstützt nicht das Verknüpfen eines Verzeichnisses zu mehrfachen Datenbanken. Wenn Sie ein Datenbank-Verzeichnis mit einem symbolischen Link ersetzen, funktioniert das solange gut, wie Sie keinen symbolischen Link zwischen Datenbanken machen. Angenommen, Sie haben eine Datenbank `datenbank1` unter dem MySQL-Daten-Verzeichnis und machen dann einen Symlink `datenbank2`, der auf `datenbank1` zeigt:

```
shell> cd /pfad/zu/datadir
shell> ln -s datenbank1 datenbank2
```

Jetzt erscheint für jede Tabelle `tabelle_a` in `datenbank1` auch eine Tabelle `tabelle_a` in `datenbank2`. Wenn ein Thread `datenbank1.tabelle_a` aktualisiert und ein anderer Thread `datenbank2.tabelle_a` aktualisiert, gibt es Probleme.

Wenn Sie das wirklich brauchen, müssen Sie folgenden Code in `mysys/mf_format.c` ändern:

```
if (flag & 32 || (!lstat(to,&stat_buff) && S_ISLNK(stat_buff.st_mode)))
```

zu:

```
if (1)
```

Unter Windows können Sie interne symbolische Links auf Verzeichnisse benutzen, indem Sie MySQL mit `-DUSE_SYMDIR` kompilieren. Das erlaubt Ihnen, verschiedene Datenbanken auf verschiedene Platte zu legen. See Abschnitt 3.6.2.5, „Daten auf verschiedenen Platten unter Win32 aufteilen“.

6.6.1.2. Benutzung symbolischer Links für Tabellen

Vor MySQL 4.0 konnten Sie Tabellen nicht per SymLink verknüpfen, wenn Sie nicht sehr sorgfältig dabei vorgingen. Das Problem liegt darin, dass bei `ALTER TABLE`, `REPAIR TABLE` oder `OPTIMIZE TABLE` auf eine per Symlink verknüpfte Datei die SymLinks entfernt und durch die Original-Dateien verknüpft werden. Das geschieht, weil beim obigen Befehl eine temporäre Datei im Datenbank-Verzeichnis erzeugt wird, und wenn der Befehl ausgeführt ist, die Original-Datei durch die temporäre Datei ersetzt wird.

Sie sollten Tabellen auf Systemen, die keinen vollständig funktionierenden `realpath()`-Aufruf haben, nicht per SymLink verknüpfen. (Zumindest Linux und Solaris unterstützen `realpath()`.)

In MySQL 4.0 werden Symlinks nur für `MyISAM`-Tabellen vollständig unterstützt. Bei anderen Tabellentypen erhalten Sie wahrscheinlich merkwürdige Probleme, wenn Sie einen der obigen Befehle ausführen.

Die Handhabung symbolischer Links in MySQL 4.0 funktioniert auf folgende Art (das gilt meist nur für `MyISAM`-Tabellen):

- Im Daten-Verzeichnis liegen immer die Tabellendefinitionsdatei und die Daten-/Index-Dateien.
- Sie können die Index-Datei und die Daten-Datei unabhängig voneinander auf unterschiedliche Verzeichnisse per SymLink verknüpfen.
- Das Erzeugen der SymLinks kann durch das Betriebssystem (wenn `mysqld` nicht läuft) oder mit dem `INDEX/DATA directory="pfad-zum-verzeichnis"`-Befehl in `CREATE TABLE` durchgeführt werden. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.
- `myisamchk` ersetzt keinen Symlink mit der Index-/Datendatei, sondern arbeitet direkt mit den Dateien, auf die die SymLinks verweisen. Jegliche temporäre Dateien werden im selben Verzeichnis erzeugt, wo die Daten-/Index-Datei ist.
- Wenn Sie eine Tabelle löschen, die Symlinks benutzt, werden sowohl der Symlink als auch die Datei, auf die der SymLink zeigt, gelöscht. Das ist ein guter Grund dafür, `mysqld` NICHT als Root laufen zu lassen und niemandem zu erlauben, Schreibzugriff auf die MySQL-Datenbankverzeichnisse zu haben.
- Wenn Sie eine Tabelle mit `ALTER TABLE RENAME` umbenennen und nicht die Datenbank ändern, wird der Symlink im Datenbank-Verzeichnis auf den neuen Namen umbenannt und die Daten-/Index-Datei wird entsprechend umbenannt.
- Wenn Sie `ALTER TABLE RENAME` benutzen, um eine Tabelle in eine andere Datenbank zu verschieben, wird die Tabelle in das andere Datenbank-Verzeichnis verschoben und die alten SymLinks und die Dateien, auf die sie zeigen, werden gelöscht.
- Wenn Sie keine Symlinks benutzen, sollten Sie die `--skip-symlink`-Option für `mysqld` benutzen, damit niemand eine Datei ausserhalb des `mysqld` Daten-Verzeichnisses löschen oder umbenennen kann.

Dinge, die noch nicht unterstützt werden:

- `ALTER TABLE` ignoriert alle `INDEX/DATA directory="pfad"`-Optionen.
- `CREATE TABLE` berichtet nicht, wenn eine Tabelle symbolische Links hat.
- `mysqldump` gibt die Information über symbolische Links nicht in der Ausgabe aus.
- `BACKUP TABLE` und `RESTORE TABLE` respektieren keine symbolischen Links.

Kapitel 7. MySQL-Sprachreferenz

MySQL hat eine sehr komplexe, aber intuitive und leicht zu erlernende SQL-Schnittstelle. Dieses Kapitel beschreibt die verschiedenen Befehle, Typen und Funktionen, die Sie kennen müssen, um MySQL effizient und effektiv zu benutzen. Dieses Kapitel dient auch als Referenz für die gesamte in MySQL beinhaltete Funktionalität. Um dieses Kapitel effektiv zu nutzen, sollten Sie unter den verschiedenen Stichworten nachschlagen.

7.1. Sprachstruktur

7.1.1. Literale: Wie Zeichenketten und Zahlen geschrieben werden

Dieser Abschnitt beschreibt die verschiedenen Arten, in MySQL Zeichenketten und Zahlen zu schreiben. Ebenfalls enthalten sind die verschiedenen Nuancen und Fallstricke, in denen man sich bei den grundlegenden Datentypen von MySQL verfangen kann.

7.1.1.1. Zeichenketten

Eine Zeichenkette ist eine Folge von Zeichen, die entweder von Apostrophs (einfachen Anführungszeichen, '') oder (doppelten) Anführungszeichen ("") umgeben ist (nur einfache Anführungszeichen, wenn Sie MySQL im ANSI-Modus laufen lassen). Beispiele:

```
'eine Zeichenkette'
"eine weitere Zeichenkette"
```

Innerhalb einer Zeichenkette haben bestimmte Folgen eine spezielle Bedeutung. Jede dieser Folgen fängt mit einem Backslash (\) an, bekannt als *Fluchtzeichen (Escape-Zeichen)*. MySQL erkennt folgende Flucht-Folgen (Escape-Folgen):

- `\0`

  Ein ASCII-0- (`NUL`) Zeichen.

- `\'`

  Ein Apostroph- ('') Zeichen.

- `\"`

  Ein Anführungszeichen ("").

- `\b`

  Ein Rückschritt- (Backspace-) Zeichen.

- `\n`

  Ein Neue-Zeile- (Newline-) Zeichen.

- `\r`

  Ein Wagenrücklauf- (carriage return) Zeichen.

- `\t`

  Ein Tabulator-Zeichen.

- `\z`

  ASCII(26) (Steuerung-Z). Dieses Zeichen kann kodiert werden, um das Problem zu umgehen, dass ASCII(26) unter Windows für Dateiende (END-OF-FILE) steht. (ASCII(26) verursacht Probleme, wenn Sie `mysql Datenbank < Dateiname` benutzen.)

- `\\`

  Ein Backslash- (‘`\`’) Zeichen.

- `\%`

  Ein ‘`%`’-Zeichen. Dieses wird benutzt, um nach literalen Instanzen von ‘`%`’ in Zusammenhängen zu suchen, wo ‘`%`’ ansonsten als Platzhalterzeichen interpretiert werden würde. See Abschnitt 7.3.2.1, „Zeichenketten-Vergleichsfunktionen“.

- `\_`

  Ein ‘`_`’-Zeichen. Dieses wird benutzt, um nach literalen Instanzen von ‘`_`’ in Zusammenhängen zu suchen, wo ‘`_`’ ansonsten als Platzhalterzeichen interpretiert werden würde. See Abschnitt 7.3.2.1, „Zeichenketten-Vergleichsfunktionen“.

Beachten Sie, dass bei der Benutzung von ‘`\%`’ oder ‘`\_`’ in einigen Zeichenketten-Zusammenhängen diese die Zeichenketten ‘`\%`’ und ‘`\_`’ und nicht ‘`%`’ und ‘`_`’ zurückgeben.

Es gibt verschiedene Möglichkeiten, Anführungszeichen innerhalb einer Zeichenkette zu schreiben:

- Ein ‘`'`’ innerhalb einer Zeichenkette, die mit ‘`'`’ begrenzt wird, kann als ‘`''`’ geschrieben werden.
- Ein ‘`"`’ innerhalb einer Zeichenkette, die ‘`"`’ begrenzt wird, kann als ‘`""`’ geschrieben werden.
- Sie können dem Anführungszeichen ein Fluchtzeichen (Escape-Zeichen) (‘`\`’) voranstellen.
- Ein ‘`'`’ innerhalb einer Zeichenkette, die mit ‘`"`’ begrenzt wird, braucht keine spezielle Behandlung und muss nicht verdoppelt oder escapet werden. In gleicher Weise benötigt ‘`"`’ innerhalb einer Zeichenkette, die mit ‘`'`’ begrenzt wird, keine spezielle Behandlung.

Die unten stehenden `SELECT`-Statements zeigen, wie Quoten und Escapen funktionieren:

```
mysql> SELECT 'hello', '"hello"', '""hello""', 'hel''lo', '\'hello';
+-------+---------+-----------+--------+--------+
| hello | "hello" | ""hello"" | hel'lo | 'hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT "hello", "'hello'", "''hello''", "hel""lo", "\"hello";
+-------+---------+-----------+--------+--------+
| hello | 'hello' | ''hello'' | hel"lo | "hello |
+-------+---------+-----------+--------+--------+

mysql> SELECT "Das\nsind\nvier\nZeilen";
+--------------------+
| Das
sind
vier
Zeilen |
+--------------------+
```

Wenn Sie Binärdaten in eine `BLOB`-Spalte einfügen, müssen folgende Zeichen durch Flucht-Folgen repräsentiert werden:

- `NUL`

  ASCII 0. Dieses geben Sie als ‘`\0`’ ein (ein Backslash und ein ASCII-‘`0`’-Zeichen).

- `\`

  ASCII 92, Backslash. Das geben Sie als ‘`\\`’ ein.

- `'`

  ASCII 39, Apostroph. Das geben Sie als ‘`\'`’ ein.

- `"`

  ASCII 34, Anführungszeichen. Das geben Sie als ‘`\"`’ ein.

Wenn Sie C-Code schreiben, können Sie die C-API-Funktion `mysql_escape_string()` für Fluchtzeichen (Escape-Zeichen)

für das INSERT-Statement benutzen. See Abschnitt 9.4.2, „C-API-Funktionsüberblick“. In Perl können Sie die quote-Methode des DBI-Pakets benutzen, um Sonderzeichen in die korrekten Flucht-Folgen umzuwandeln. See Abschnitt 9.2.2, „Die DBI-Schnittstelle“.

Sie sollten auf jede Zeichenkette, die eins der oben erwähnten Sonderzeichen enthalten könnte, eine der Flucht-Funktionen anwenden!

7.1.1.2. Zahlen

Ganzzahlen werden als Folge von Ziffern repräsentiert. Fließkommazahlen benutzen ‘.’ als Dezimalseparator. Jedem Zahlentyp kann ‘-’ vorangestellt werden, um einen negativen Wert anzuzeigen.

Beispiele gültiger Ganzzahlen:

```
1221
0
-32
```

Beispiele gültiger Fließkommazahlen:

```
294.42
-32032.6809e+10
148.00
```

Eine Ganzzahl kann in einem Fließkomma-Zusammenhang benutzt werden, sie wird dann als die äquivalente Fließkommazahl interpretiert.

7.1.1.3. Hexadezimale Werte

MySQL unterstützt hexadezimale Werte. In Zahlen-Zusammenhängen funktionieren diese wie eine Ganzzahl (64-Bit-Genauigkeit). Im Zeichenketten-Zusammenhang funktionieren sie wie eine binäre Zeichenkette, wobei jedes Paar hexadezimaler Ziffern in ein Zeichen umgewandelt wird:

```
mysql> SELECT x'FF'
       -> 255
mysql> SELECT 0xa+0;
       -> 10
mysql> select 0x5061756c;
       -> Paul
```

Die x'hexadezimale_zeichenkette'-Syntax (neu in Version 4.0) basiert auf ANSI-SQL. Die 0x-Syntax basiert auf ODBC. Hexadezimale Zeichenketten werden oft von ODBC benutzt, um Werte für BLOB-Spalten anzugeben.

7.1.1.4. NULL-Werte

Der NULL-Wert bedeutet ``keine Daten" und unterscheidet sich von Werten wie 0 bei numerischen Typen oder der leeren Zeichenkette bei Zeichenkettentypen. See Abschnitt A.5.3, „Probleme mit NULL-Werten“.

NULL kann durch \N repräsentiert werden, wenn Sie die Textdatei-Import- oder Exportformate (LOAD DATA INFILE, SELECT ... INTO OUTFILE) benutzen. See Abschnitt 7.4.9, „LOAD DATA INFILE-Syntax“.

7.1.2. Datenbank-, Tabellen-, Index-, Spalten- und Alias-Namen

Datenbank-, Tabellen-, Index-, Spalten- und Alias-Namen folgen in MySQL alle denselben Regeln.

Beachten Sie, dass sich die Regeln ab MySQL-Version 3.23.6 geändert haben, als das Quoten von Bezeichnern (für Datenbank-, Tabellen- und Spaltennamen) eingeführt wurde, mit ‘``’. ‘"’ funktioniert ebenfalls, um Bezeichner zu quoten, wenn Sie im ANSI-Modus fahren. See Abschnitt 2.7.2, „MySQL im ANSI-Modus laufen lassen“.

Bezeichner	Maximale Länge	Erlaubte Zeichen
Datenbank	64	Jedes Zeichen, dass für ein Verzeichnis erlaubt ist, ausser ‘/’ oder ‘.’.
Tabelle	64	Jedes Zeichen, dass für einen Dateinamen erlaubt ist, ausser ‘/’ oder ‘.’.
Spalte	64	Alle Zeichen.
Alias	255	Alle Zeichen.

Hinzuzufügen ist, dass Sie ASCII(0), ASCII(255) oder das Quote-Zeichen in einem Bezeichner nicht verwenden dürfen.

Beachten Sie, dass, falls der Bezeichner ein reserviertes Wort ist oder Sonderzeichen enthält, er bei der Benutzung immer in `

angegeben sein muss:

```
SELECT * from `select` where `select`.id > 100;
```

In vorherigen Versionen von MySQL sind die Namensregeln wie folgt:

- Ein Name muss aus alphanumerischen Zeichen des aktuellen Zeichensatzes bestehen und darf darüber hinaus '_' und '$' enthalten. Der vorgabemäßige Zeichensatz ist ISO-8859-1 Latin1; dass kann durch die `--default-character-set`-Option für `mysqld` geändert werden. See Abschnitt 5.6.1, „Der für Daten und Sortieren benutzte Zeichensatz".
- Ein Name kann mit jedem Zeichen anfangen, das in einem Namen erlaubt ist. Insbesondere kann ein Name auch mit einer Zahl anfangen (das ist in vielen anderen Datenbanksystemen anders!). Jedoch kann ein Namen nicht *nur* aus Zahlen bestehen.
- Sie können das '`.`'-Zeichen in Namen nicht benutzen, weil es benutzt wird, um das Format zu erweitern, mit dem man auf Spalten verweisen kann (siehe unten).

Es wird empfohlen, dass Sie keine Namen wie `1e` verwenden, weil ein Ausdruck wie `1e+1` mehrdeutig ist. Er kann als der Ausdruck `1e + 1` oder als die Zahl `1e+1` interpretiert werden.

In MySQL können Sie in folgender Form auf Spalten verweisen:

Spaltenverweis	Bedeutung
`spalten_name`	Spalte des Namens `spalten_name` einer beliebigen, in der Anfrage verwendeten Tabelle.
`tabelle.spalten_name`	Spalte des Namens `spalten_name` der Tabelle `tabelle` der aktuellen Datenbank.
`datenbank.tabelle.spalten_name`	Spalte des Namens `spalten_name` der Tabelle `tabelle` der Datenbank `datenbank`. Diese Form ist ab MySQL-Version 3.22 verfügbar.
`` `spalte` ``	Eine Spalte, die ein reserviertes Wort ist oder Sonderzeichen enthält.

Das `tabelle`- oder `datenbank.tabelle`-Präfix müssen Sie bei einem Spaltenverweis in einem Statement nicht angeben, es sei denn, der Verweis wäre ansonsten doppeldeutig. Nehmen Sie zum Beispiel an, die Tabellen `t1` und `t2` enthielten beide jeweils eine Spalte `c` und Sie verweisen auf `c` in einem `SELECT`-Statement, das sowohl `t1` als auch `t2` benutzt. In diesem Fall ist `c` mehrdeutig, weil es innerhalb der im Statement benutzten Tabellen nicht eindeutig ist. Daher müssen Sie angeben, welche Tabelle Sie meinen, indem Sie `t1.c` oder `t2.c` schreiben. Ähnliches gilt, wenn Sie aus einer Tabelle `t` in Datenbank `datenbank1` und von eine Tabelle `t` in Datenbank `datenbank2` abrufen. Dann müssen Sie auf Spalten in diesen Tabellen als `datenbank1.t.spalten_name` und `datenbank2.t.spalten_name` verweisen.

Die Syntax `.tabelle` bedeutet die Tabelle `tabelle` in der aktuellen Datenbank. Diese Syntax wird aus Gründen der ODBC-Kompatibilität akzeptiert, weil einige ODBC-Programme Tabellenname ein '`.`'-Zeichen voranstellen.

7.1.3. Groß-/Kleinschreibung in Namen

In MySQL entsprechen Datenbanken und Tabellen Verzeichnissen und Dateien innerhalb dieser Verzeichnisse. Folglich hängt die Groß-/Kleinschreibung davon ab, wie das zugrunde liegende Betriebssystem die Groß-/Kleinschreibung von Datenbank- und Tabellennamen festlegt. Das bedeutet, dass Datenbank- und Tabellennamen unter Unix von der Groß-/Kleinschreibung abhängen und unter Windows nicht. See Abschnitt 2.7.3, „MySQL-Erweiterungen zu ANSI SQL92".

HINWEIS: Obwohl die Groß-/Kleinschreibung für Datenbank- und Tabellennamen unter Windows keine Rolle spielt, sollten Sie nicht auf eine angegebene Datenbank oder Tabelle innerhalb derselben Anfrage mit unterschiedlicher Schreibweise verweisen. Folgende Anfrage würde nicht funktionieren, weil sie auf eine Tabelle sowohl mit `meine_tabelle` als auch mit `MEINE_TABELLE` verweist:

```
mysql> SELECT * FROM meine_tabelle WHERE MEINE_TABELLE.spalte=1;
```

Spaltennamen hängen in keinem Fall von der verwendeten Groß-/Kleinschreibung ab.

Aliase auf Tabellen hängen von der Groß-/Kleinschreibung ab. Folgende Anfrage würde nicht funktionieren, weil sie auf den Alias sowohl mit `a` als auch mit `A` verweist:

```
mysql> SELECT spalten_name FROM tabelle AS a
           WHERE a.spalten_name = 1 OR A.spalten_name = 2;
```

Aliase auf Spalten hängen nicht von der verwendeten Groß-/Kleinschreibung ab.

Wenn Sie Probleme damit haben, sich an die Schreibweise von Tabellennamen zu erinnern, halten Sie sich an eine durchgehende Konvention. Benutzen Sie zum Beispiel bei der Erzeugung von Datenbanken und Tabellen Kleinschreibung in Namen.

Eine Möglichkeit, dieses Problem zu vermeiden, ist, `mysqld` mit `-O lower_case_tabelles=1` zu starten. Vorgabemäßig ist diese Option 1 unter Windows und 0 unter Unix.

Wenn `lower_case_tabelles` 1 ist, wandelt MySQL alle Tabellennamen in Kleinschreibung um, sowohl beim Speichern als auch beim Nachschlagen. Wenn Sie diese Option ändern, beachten Sie, dass Sie zuerst Ihre alten Tabellennamen in Kleinschreibung umwandeln müssen, bevor Sie `mysqld` starten.

7.1.4. Benutzer-Variablen

MySQL unterstützt Thread-spezifische Variablen mit der `@variablename`-Syntax. Eine Variable kann aus alphanumerischen Zeichen des aktuellen Zeichensatzes sowie aus '_', '$' und '.' bestehen. Der vorgabemäßige Zeichensatz ist ISO-8859-1 Latin1; das kann mit der `--default-character-set`-Option für `mysqld` geändert werden. See Abschnitt 5.6.1, „Der für Daten und Sortieren benutzte Zeichensatz“.

Variablen müssen nicht initialisiert werden. Sie enthalten vorgabemäßig `NULL` und können Ganzzahl-, Real- oder Zeichenketten-Werte speichern. Alle Variablen für einen Thread werden automatisch freigegeben, wenn der Thread beendet wird.

Sie können eine Variable mit der `SET`-Syntax setzen:

```
SET @variable= { ganzzahl_ausdruck | realzahl_ausdruck | zeichenketten_ausdruck } [,@variable= ...].
```

Sie können eine Variable in einem Ausdruck auch mit der `@variable:=expr`-Syntax setzen:

```
select @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;
+----------------------+------+------+------+
| @t1:=(@t2:=1)+@t3:=4 | @t1  | @t2  | @t3  |
+----------------------+------+------+------+
|                    5 |    5 |    1 |    4 |
+----------------------+------+------+------+
```

(Wir mussten hier die `:=`-Syntax benutzen, weil `=` für Vergleiche reserviert ist.)

Benutzer-Variablen können benutzt werden, wo Ausdrücke erlaubt sind. Beachten Sie, dass das momentan keine Zusammenhänge einschließt, in denen explizit Zahlen erforderlich sind, wie in der `LIMIT`-Klausel eines `SELECT`-Statements oder der `IGNORE Anzahl LINES`-Klausel eines `LOAD DATA`-Statements.

HINWEIS: In einem `SELECT`-Statement wird jeder Ausdruck erst dann ausgewertet, wenn er an den Client geschickt wird. Das heißt, dass Sie in der `HAVING`-, `GROUP BY`- oder `ORDER BY`-Klausel nicht auf einen Ausdruck verweisen können, der Variablen beinhaltet, die nicht im `SELECT`-Teil gesetzt wurden. Folgendes Statement zum Beispiel funktioniert erwartungsgemäß NICHT:

```
SELECT (@aa:=id) AS a, (@aa+3) AS b FROM tabelle HAVING b=5;
```

Der Grund ist, dass `@aa` nicht den Wert der aktuellen Zeile enthält, sondern den Wert von `id` der vorher akzeptierten Zeile.

7.1.5. Kommentar-Syntax

Der MySQL-Server die Kommentar-Stile `# bis Zeilenende`, `-- bis Zeilenende` und `/* mittendrin oder mehrzeilig */`:

```
mysql> select 1+1;     # Dieser Kommentar geht bis zum Zeilenende
mysql> select 1+1;     -- Dieser Kommentar geht bis zum Zeilenende
mysql> select 1 /* Das ist ein Kommentar mittendrin */ + 1;
mysql> select 1+
/*
Das ist ein
mehrzeiliger
Kommentar
*/
1;
```

Beachten Sie, dass Sie beim Kommentarstil `--` mindestens ein Leerzeichen hinter `--` setzen müssen!

Obwohl der Server die Kommentar-Syntax wie beschrieben versteht, gibt es einige Einschränkungen in der Art, wie der `mysql`-Client `/* ... */`-Kommentare parst:

- Einfache und doppelte Anführungszeichen werden genommen, um den Anfang einer Zeichenkette zu bestimmen, selbst innerhalb eines Kommentars. Wenn die Zeichenkette nicht durch ein zweites Anführungszeichen innerhalb des Kommentars abgeschlossen wird, bemerkt der Parser nicht, dass der Kommentar zuende ist. Wenn Sie `mysql` interaktiv ausführen, sehen

Sie, dass `mysql` verwirrt ist, weil sich die Eingabeaufforderung von `mysql>` zu to `'>` oder `">` ändert.

- Ein Semikolon wird genommen, um das Ende des aktuellen SQL-Statements kenntlich zu machen. Alles Folgende wird als Anfang des nächsten Statements aufgefasst.

Diese Einschränkungen gelten sowohl, wenn Sie `mysql` interaktiv ausführen und wenn Sie Befehle in eine Datei schreiben und `mysql` mit `mysql < some-file` anweisen, seine Eingaben aus dieser Datei zu lesen.

MySQL unterstützt nicht den ANSI-SQL-Kommentarstil '`--`' ohne nachfolgendes Leerzeichen. See Abschnitt 2.7.4.8, „'`--`' als Beginn eines Kommentars“.

7.1.6. Ist MySQL pingelig hinsichtlich reservierter Wörter?

Ein häufiges Problem rührt daher, dass versucht wird, eine Tabelle mit Spaltennamen zu erzeugen, den die Namen von Datentypen oder in MySQL eingebauten Funktionen entsprechen, wie `TIMESTAMP` oder `GROUP`. Sie dürfen das tun (beispielsweise ist `ABS` ein zulässiger Spaltenname), aber es sind dann keine Leerzeichen zwischen einem Funktionsname und der ‘`(`’ erlaubt, wenn Sie Funktionen benutzen, deren Namen auch Spaltennamen sind.

Folgende Wörter sind in MySQL explizit reserviert. Die meisten davon sind in ANSI-SQL92 als Spalten- und / oder Tabellennamen verboten (zum Beispiel `group`). Einige wenige sind reserviert, weil MySQL sie benötigt und (momentan) einen `yacc`-Parser benutzt:

ADD	ALL	ALTER
ANALYZE	AND	AS
ASC	BEFORE	BETWEEN
BIGINT	BINARY	BLOB
BOTH	BY	CASCADE
CASE	CHANGE	CHAR
CHARACTER	CHECK	COLLATE
COLUMN	COLUMNS	CONSTRAINT
CONVERT	CREATE	CROSS
CURRENT_DATE	CURRENT_TIME	CURRENT_TIMESTAMP
CURRENT_USER	DATABASE	DATABASES
DAY_HOUR	DAY_MICROSECOND	DAY_MINUTE
DAY_SECOND	DEC	DECIMAL
DEFAULT	DELAYED	DELETE
DESC	DESCRIBE	DISTINCT
DISTINCTROW	DIV	DOUBLE
DROP	DUAL	ELSE
ENCLOSED	ESCAPED	EXISTS
EXPLAIN	FALSE	FIELDS
FLOAT	FLOAT4	FLOAT8
FOR	FORCE	FOREIGN
FROM	FULLTEXT	GRANT
GROUP	HAVING	HIGH_PRIORITY
HOUR_MICROSECOND	HOUR_MINUTE	HOUR_SECOND
IF	IGNORE	IN
INDEX	INFILE	INNER
INSERT	INT	INT1
INT2	INT3	INT4
INT8	INTEGER	INTERVAL
INTO	IS	JOIN
KEY	KEYS	KILL
LEADING	LEFT	LIKE

LIMIT	LINES	LOAD
LOCALTIME	LOCALTIMESTAMP	LOCK
LONG	LONGBLOB	LONGTEXT
LOW_PRIORITY	MATCH	MEDIUMBLOB
MEDIUMINT	MEDIUMTEXT	MIDDLEINT
MINUTE_MICROSECOND	MINUTE_SECOND	MOD
NATURAL	NOT	NO_WRITE_TO_BINLOG
NULL	NUMERIC	ON
OPTIMIZE	OPTION	OPTIONALLY
OR	ORDER	OUTER
OUTFILE	PRECISION	PRIMARY
PRIVILEGES	PROCEDURE	PURGE
READ	REAL	REFERENCES
REGEXP	RENAME	REPLACE
REQUIRE	RESTRICT	REVOKE
RIGHT	RLIKE	SECOND_MICROSECOND
SELECT	SEPARATOR	SET
SHOW	SMALLINT	SONAME
SPATIAL	SQL_BIG_RESULT	SQL_CALC_FOUND_ROWS
SQL_SMALL_RESULT	SSL	STARTING
STRAIGHT_JOIN	TABLE	TABLES
TERMINATED	THEN	TINYBLOB
TINYINT	TINYTEXT	TO
TRAILING	TRUE	UNION
UNIQUE	UNLOCK	UNSIGNED
UPDATE	USAGE	USE
USING	UTC_DATE	UTC_TIME
UTC_TIMESTAMP	VALUES	VARBINARY
VARCHAR	VARCHARACTER	VARYING
WHEN	WHERE	WITH
WRITE	XOR	YEAR_MONTH
ZEROFILL		

Folgende reservierte Wörter sind neu in MySQL 4.0:

CHECK	FORCE	LOCALTIME
LOCALTIMESTAMP	REQUIRE	SQL_CALC_FOUND_ROWS
SSL	XOR	

Folgende Symbole (aus der obigen Tabelle) sind von ANSI-SQL verboten, aber von MySQL als Spalten- und Tabellennamen zugelassen. Der Grund ist, dass einige davon sehr natürliche Namen sind und viele Leute diese bereits in Benutzung haben.

- ACTION
- BIT
- DATE
- ENUM
- NO

- `TEXT`
- `TIME`
- `TIMESTAMP`

7.2. Spaltentypen

MySQL unterstützt eine Reihe von Spaltentypen, die in drei Kategorien eingeteilt werden können: numerische Typen, Datums- und Zeit-Typen und Zeichenketten-Typen. Dieser Abschnitt gibt zuerst einen Überblick über die verfügbaren Typen und fasst den Speicherbedarf jedes Spaltentyps zusammen. Danach folgt eine detaillierter Beschreibung der Eigenschaften der Typen jeder Kategorie. Die detailliertere Beschreibung sollte wegen zusätzlicher Informationen über bestimmte Spaltentypen herangezogen werden, wie zu den erlaubten Formaten, in denen Sie Werte festlegen können.

Die von MySQL unterstützten Spaltentypen sind unten aufgeführt. Folgende Code-Buchstaben werden in der Beschreibung benutzt:

- `M`

  Gibt die maximale Anzeigebreite an. Die größte erlaubte Anzeigebreite ist 255.

- `D`

  Trifft auf Fließkomma-Typen zu und bezeichnet die Anzahl von Ziffern nach dem Dezimalpunkt. Der größte mögliche Wert ist 30, aber er sollte nicht größer sein als `M`-2.

Eckige Klammern (‘`[`’ und ‘`]`’) geben Teile der Typ-Festlegung an, die optional sind.

Wenn Sie `ZEROFILL` für eine Spalte angeben, beachten Sie, dass MySQL der Spalte automatisch ein `UNSIGNED`-Attribut hinzufügt.

- `TINYINT[(M)] [UNSIGNED] [ZEROFILL]`

  Eine sehr kleine Ganzzahl. Der vorzeichenbehaftete Bereich ist `-128` bis `127`. Der vorzeichenlose Bereich ist `0` to `255`.

- `SMALLINT[(M)] [UNSIGNED] [ZEROFILL]`

  Eine kleine Ganzzahl. Der vorzeichenbehaftete Bereich ist `-32768` bis `32767`. Der vorzeichenlose Bereich ist `0` bis `65535`.

- `MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]`

  A Ganzzahl mittlerer Größe. Der vorzeichenbehaftete Bereich ist `-8388608` bis `8388607`. Der vorzeichenlose Bereich ist `0` bis `16777215`.

- `INT[(M)] [UNSIGNED] [ZEROFILL]`

  Eine Ganzzahl normaler Größe. Der vorzeichenbehaftete Bereich ist `-2147483648` bis `2147483647`. Der vorzeichenlose Bereich ist `0` bis `4294967295`.

- `INTEGER[(M)] [UNSIGNED] [ZEROFILL]`

  Ein Synonym für `INT`.

- `BIGINT[(M)] [UNSIGNED] [ZEROFILL]`

  Eine große Ganzzahl. Der vorzeichenbehaftete Bereich ist `-9223372036854775808` bis `9223372036854775807`. Der vorzeichenlose Bereich ist `0` bis `18446744073709551615`.

  Einiger Dinge sollten Sie sich bei `BIGINT`-Spalten bewusst sein:

  - Weil alle arithmetischen Berechnungen mit vorzeichenbehafteten `BIGINT`- oder `DOUBLE`-Werten durchgeführt werden, sollten Sie keine vorzeichenlosen Ganzzahlen größer als `9223372036854775807` (63 Bits) benutzen, ausser bei Bit-Funktionen! Wenn Sie das doch tun, können einige der letzten Ziffern im Ergebnis falsch sein, weil Rundungsfehler beim Umwandeln von `BIGINT` in `DOUBLE` auftreten.

    MySQL 4.0 kann `BIGINT` in folgenden Fällen handhaben:

    - Benutzen Sie Ganzzahlen, um große vorzeichenlose Wert in einer `BIGINT`-Spalte zu speichern.
    - Bei `MIN(große_ganzzahl_spalte)` und `MAX(große_ganzzahl_spalte)`.
    - Bei der Benutzung der Operatoren (`+`, `-`, `*` usw.), wenn beide Operanden Ganzzahlen sind.
  - Sie können immer einen genauen Ganzzahlwert in einer `BIGINT`-Spalte speichern, wenn Sie sie als Zeichenkette speichern, denn in diesem Fall wird diese nicht zwischendurch als Double dargestellt.
  - '`-`', '`+`' und '`*`' benutzen arithmetische `BIGINT`-Berechnungen, wenn beide Argumente `INTEGER`-Werte sind! Das heißt, wenn Sie zwei Ganzzahlen multiplizieren (oder Ergebnisse von Funktionen, die Ganzzahlen zurückgeben), erhalten Sie vielleicht unerwartete Ergebnisse, wenn das Ergebnis größer als `9223372036854775807` ist.

- `FLOAT(genauigkeit) [ZEROFILL]`

  Eine Fließkommazahl. Kann nicht vorzeichenlos sein. `genauigkeit` ist `<=24` bei einer Fließkommazahl einfacher Genauigkeit und zwischen 25 und 53 bei einer Fließkommazahl doppelter Genauigkeit. Diese Typen sind wie die unten beschriebenen `FLOAT` und `DOUBLE`-Typen. `FLOAT(X)` hat denselben Wertebereich wie die entsprechenden `FLOAT`- und `DOUBLE`-Typen, jedoch ist die Anzeigebreite und die Anzahl der Dezimalstellen undefiniert.

  In MySQL-Version 3.23 ist das ein echter Fließkommawert. In früheren MySQL-Versionen hat `FLOAT(genauigkeit)` immer 2 Dezimalstellen.

  Beachten Sie, dass bei der Benutzung von `FLOAT` unerwartete Probleme auftreten können, weil alle Berechnungen in MySQL mit doppelter Genauigkeit durchgeführt werden. See Abschnitt A.5.6, „Probleme bei keinen übereinstimmenden Zeilen lösen“.

  Diese Syntax steht wegen der ODBC-Kompatibilität zur Verfügung.

- `FLOAT[(M,D)] [ZEROFILL]`

  Eine kleine Fließkommazahl (einfacher Genauigkeit). Kann nicht vorzeichenlos sein. Der Wertebereich umfasst `-3.402823466E+38` bis `-1.175494351E-38`, `0` und `1.175494351E-38` bis `3.402823466E+38`. M ist die Anzeigebreite und D ist die Anzahl von Dezimalstellen. `FLOAT` ohne Argument oder mit einem Argument <= 24 steht für eine Fließkommazahl einfacher Genauigkeit.

- `DOUBLE[(M,D)] [ZEROFILL]`

  Eine normal große Fließkommazahl (doppelter Genauigkeit). Kann nicht vorzeichenlos sein. Der Wertebereich umfasst `-1.7976931348623157E+308` bis `-2.2250738585072014E-308`, `0` und `2.2250738585072014E-308` bis `1.7976931348623157E+308`. M ist die Anzeigebreite und D ist die Anzahl von Dezimalstellen. `DOUBLE` ohne Argument oder `FLOAT(X)` mit 25 <= X <= 53 steht für eine Fließkommazahl doppelter Genauigkeit.

- `DOUBLE PRECISION[(M,D)] [ZEROFILL]`, `REAL[(M,D)] [ZEROFILL]`

  Synonyme für `DOUBLE`.

- `DECIMAL[(M[,D])] [ZEROFILL]`

  Eine unkomprimierte Fließkommazahl. Kann nicht vorzeichenlos sein. Verhält sich wie eine `CHAR`-Spalte: ``Unkomprimiert" bedeutet, dass die Zahl als Zeichenkette gespeichert wird, wobei ein Zeichen für jede Ziffer des Wertes steht. Der Dezimalpunkt und, bei negativen Zahlen, das '`-`'-Zeichen, werden in M nicht mitgezählt (aber hierfür wird Platz reserviert). Wenn `D` 0 ist, haben Werte keinen Dezimalpunkt oder Bruchteil. Der maximale Wertebereich von `DECIMAL`-Werte ist derselbe wie für `DOUBLE`, aber der tatsächliche Wertebereich einer gegebenen `DECIMAL`-Spalte kann durch die Auswahl von `M` und `D` eingeschränkt sein.

  Wenn `D` weggelassen wird, wird es auf 0 gesetzt. Wenn `M` ausgelassen wird, wird es auf 10 gesetzt.

  Beachten Sie, dass in MySQL-Version 3.22 das `M`-Argument den Platz für das Vorzeichen und den Dezimalpunkt beinhaltete!

- `NUMERIC(M,D) [ZEROFILL]`

  Synonym für `DECIMAL`.

- `DATE`

  Ein Datum. Der unterstützte Wertebereich ist `'1000-01-01'` bis `'9999-12-31'`. MySQL zeigt `DATE`-Werte im `'YYYY-`

MM-DD'-Format an, gestattet jedoch, DATE-Spalten Werte entweder als Zeichenketten oder als Zahlen zuzuweisen. See Abschnitt 7.2.2.2, „Die DATETIME-, DATE- und TIMESTAMP-Typen“.

- DATETIME

  Eine Datums-/Zeit-Kombination. Der unterstützte Wertebereich ist '1000-01-01 00:00:00' bis '9999-12-31 23:59:59'. MySQL zeigt DATETIME-Werte im 'YYYY-MM-DD HH:MM:SS'-Format an, gestattet jedoch, DATETIME-Spalten Werte entweder als Zeichenketten oder als Zahlen zuzuweisen. See Abschnitt 7.2.2.2, „Die DATETIME-, DATE- und TIMESTAMP-Typen“.

- TIMESTAMP[(M)]

  Ein Zeitstempel. Der Wertebereich ist '1970-01-01 00:00:00' bis irgendwann im Jahr 2037. MySQL zeigt TIMESTAMP-Werte im YYYYMMDDHHMMSS-, YYMMDDHHMMSS-, YYYYMMDD- oder YYMMDD-Format an, abhängig davon, ob M 14 (oder fehlend), 12, 8 oder 6 ist, gestattet aber, dass Sie TIMESTAMP-Spalten Werte entweder als Zeichenketten oder als Zahlen zuweisen. Eine TIMESTAMP-Spalte ist nützlich, um Datum und Zeit einer INSERT- oder UPDATE-Operation zu speichern, weil sie automatisch auf das Datum und die Zeit der jüngsten Operation gesetzt wird, wenn Sie nicht selbst einen Wert zuweisen. Sie können sie auch auf das aktuelle Datum und die aktuelle Zeit setzen, indem Sie einen NULL-Wert zuweisen. See Abschnitt 7.2.2, „Datums- und Zeit-Typen“.

  Ein TIMESTAMP wird immer mit 4 Bytes gespeichert. Das M-Argument betrifft nur die Anzeige der TIMESTAMP-Spalte.

  Beachten Sie, dass TIMESTAMP(X)-Spalten, bei denen X 8 oder 14 ist, als Zahlen interpretiert werden, während andere TIMESTAMP(X)-Spalten als Zeichenketten interpretiert werden. Das soll lediglich sicherstellen, dass Sie Tabellen mit diesen Typen verlässlich dumpen und wiederherstellen können! See Abschnitt 7.2.2.2, „Die DATETIME-, DATE- und TIMESTAMP-Typen“.

- TIME

  Ein Zeit-Typ. Der Wertebereich ist '-838:59:59' bis '838:59:59'. MySQL zeigt TIME-Werte im 'HH:MM:SS'-Format an, gestattet aber, TIME-Spalten Werte entweder als Zeichenketten oder als Zahlen zuweisen. See Abschnitt 7.2.2.3, „Der TIME-Typ“.

- YEAR[(2|4)]

  Ein Jahr in 2- oder 4-Ziffernformat (Vorgabe ist 4-Ziffern). Die zulässigen Werte reichen von 1901 bis 2155 sowie 0000 im 4-Ziffern-Jahresformat, und von 1970 bis 2069 beim 2-Ziffernformat (70 bis 69). MySQL zeigt YEAR-Werte im YYYY-Format an, gestattet aber, YEAR-Spalten Werte entweder als Zeichenketten oder als Zahlen zuweisen. (Der YEAR-Typ ist neu seit MySQL-Version 3.22.). See Abschnitt 7.2.2.4, „Der YEAR-Typ“.

- [NATIONAL] CHAR(M) [BINARY]

  Eine Zeichenkette fester Länge, die beim Speichern rechts stets mit Leerzeichen bis zur angegebenen Länge aufgefüllt wird. Der Wertebereich von M ist 1 bis 255 Zeichen. Leerzeichen am Ende werden beim Abruf des Wertes entfernt. CHAR-Werte werden nach dem vorgabemäßigen Zeichensatz ohne Berücksichtigung der Groß-/Kleinschreibung sortiert und verglichen, es sei denn, dass Schlüsselwort BINARY wird angegeben.

  NATIONAL CHAR (Kurzform NCHAR) ist die Art, wie ANSI-SQL bei einer CHAR-Spalte festlegt, dass der vorgabemäßige Zeichensatz verwendet werden soll. Das ist der Vorgabewert in MySQL.

  CHAR ist eine Abkürzung für CHARACTER.

  MySQL erlaubt das Anlegen einer Spalte des Typs CHAR(0). Das ist hauptsächlich nützlich, wenn Sie mit alten Applikationen kompatibel sein müssen, die auf die Existenz einer Spalte vertrauen, den Wert aber nicht tatsächlich benutzen. Es ist ebenfalls nett, um eine Spalte anzulegen, die nur 2 Werte annehmen kann: Eine CHAR(0), die nicht als NOT NULL definiert ist, belegt nur 1 Bit und kann 2 Werte annehmen: NULL oder "". See Abschnitt 7.2.3.1, „Die CHAR- und VARCHAR-Typen“.

- [NATIONAL] VARCHAR(M) [BINARY]

  Eine Zeichenkette variabler Länge. **HINWEIS:** Leerzeichen am Ende werden bei der Speicherung des Wertes entfernt (das unterscheidet den Typ von der ANSI-SQL-Spezifikation). Der Wertebereich von M ist 1 bis 255 Zeichen. VARCHAR-Werte werden nach dem vorgabemäßigen Zeichensatz ohne Berücksichtigung der Groß-/Kleinschreibung sortiert und verglichen, es sei denn, dass Schlüsselwort BINARY wird angegeben. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“.

  VARCHAR ist eine Abkürzung für CHARACTER VARYING. See Abschnitt 7.2.3.1, „Die CHAR- und VARCHAR-Typen“.

- TINYBLOB, TINYTEXT

Eine BLOB- oder TEXT-Spalte mit einer maximalen Länge von 255 (2^8 - 1) Zeichen. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“. See Abschnitt 7.2.3.2, „Die BLOB- und TEXT-Typen“.

- BLOB, TEXT

  Eine BLOB- oder TEXT-Spalte mit einer maximalen Länge von 65535 (2^16 - 1) Zeichen. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“. See Abschnitt 7.2.3.2, „Die BLOB- und TEXT-Typen“.

- MEDIUMBLOB, MEDIUMTEXT

  Eine BLOB- oder TEXT-Spalte mit einer maximalen Länge von 16777215 (2^24 - 1) Zeichen. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“. See Abschnitt 7.2.3.2, „Die BLOB- und TEXT-Typen“.

- LONGBLOB, LONGTEXT

  Eine BLOB- oder TEXT-Spalte mit einer maximalen Länge von 4294967295 (2^32 - 1) Zeichen. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“. Beachten Sie, dass Sie nicht den gesamten Wertebereich dieses Typs benutzen können, weil das Client-Server-Protokoll und MyISAM-Tabellen momentan eine Beschränkungen auf 16 MB pro Kommunikationspaket / Tabellenzeile haben. See Abschnitt 7.2.3.2, „Die BLOB- und TEXT-Typen“.

- ENUM('wert1','wert2',...)

  An Aufzählung. Ein Zeichenkettenobjekt, das nur einen Wert haben kann, der aus den Auflistungswerten 'wert1', 'wert2', ..., NULL oder dem speziellen ""-Fehlerwert ausgewählt wird. Eine ENUM kann maximal 65535 unterschiedliche Werte haben. See Abschnitt 7.2.3.3, „Der ENUM-Typ“.

- SET('wert1','wert2',...)

  Eine Reihe. Ein Zeichenkettenobjekt, das 0 oder mehr Werte haben kann, von denen jeder aus den Auflistungswerten 'wert1', 'wert2', ... ausgewählt werden muss. Eine SET kann maximal 64 Elemente haben. See Abschnitt 7.2.3.4, „Der SET-Typ“.

7.2.1. Numerische Typen

MySQL unterstützt alle numerischen Typen von ANSI/ISO-SQL92. Diese Typen beinhalten die exakten numerischen Datentypen (NUMERIC, DECIMAL, INTEGER und SMALLINT) sowie die näherungsweisen numerischen Datentypen (FLOAT, REAL und DOUBLE PRECISION). Das Schlüsselwort INT ist ein Synonym für INTEGER und das Schlüsselwort DEC ist ein Synonym für DECIMAL.

Die NUMERIC- und DECIMAL-Typen sind in MySQL als derselbe Typ implementiert, wie es vom SQL92-Standard zugelassen ist. Sie werden für Werte benutzt, bei denen es wichtig ist, die exakte Genauigkeit zu bewahren, zum Beispiel bei monetären Daten. Wenn Sie eine Spalte mit einem dieser Typen deklarieren, können Genauigkeit und Bereich festgelegt werden (und werden das üblicherweise auch). Beispiel:

```
    gehalt DECIMAL(9,2)
```

In diesem Beispiel repräsentiert 9 (genauigkeit) die Anzahl signifikanter Dezimalziffern, die für Werte gespeichert werden, und 2 (bereich) repräsentiert die Anzahl von Ziffern, die nach dem Dezimalpunkt gespeichert werden. In diesem Fall liegt der Wertebereich, der in der gehalt-Spalte gespeichert werden kann, deswegen zwischen -9999999.99 und 9999999.99. (MySQL kann tatsächlich Zahlen bis zu 9999999.99 in dieser Spalte speichern, weil er nicht das Vorzeichen für positive Zahlen speichern muss).

In ANSI/ISO-SQL92 ist die Syntax DECIMAL(p) äquivalent zu DECIMAL(p,0). Gleichermaßen ist die Syntax DECIMAL äquivalent zu DECIMAL(p,0), wobei es der Implementation überlassen bleibt, den Wert von p festzulegen. MySQL unterstützt momentan keine dieser abweichenden Formen der DECIMAL- / NUMERIC-Datentypen. Das ist im Allgemeinen kein ernstes Problem, weil der hauptsächliche Nutzen dieser Typen darin liegt, sowohl Genauigkeit als auch Bereich explizit steuern zu können.

DECIMAL- und NUMERIC-Werte sind als Zeichenketten gespeichert statt als Fließkommazahlen, um die dezimale Genauigkeit dieser Werte zu bewahren. Ein Zeichen wird benutzt für jede Ziffer des Werts, den Dezimalpunkt (wenn bereich > 0) und das ‘-’-Zeichen (für negative Zahlen). Wenn bereich 0 ist, enthalten DECIMAL- und NUMERIC-Werte weder Dezimalpunkt noch Bruchteil.

Der maximale Wertebereich von DECIMAL- und NUMERIC-Werten ist derselbe wie für DOUBLE, aber der tatsächliche Wertebereich einer gegebenen DECIMAL- oder NUMERIC-Spalte kann durch genauigkeit oder bereich für eine gegebene Spalte beschränkt werden. Wenn einer solchen Spalte ein Wert mit mehr Ziffern nach dem Dezimalpunkt zugewiesen wird, als

durch bereich zugelassen, wird der Wert auf diesen bereich gerundet. Wenn einer DECIMAL- oder NUMERIC-Spalte ein Wert zugewiesen wird, dessen Größe den Wertebereich überschreitet, der von der festgelegten (oder vorgabemäßigen) genauigkeit und bereich festgelegt wird, speichert MySQL den Wert des entsprechenden Endpunkts des Wertebereichs.

Als Erweiterung zum ANSI/ISO-SQL92-Standard unterstützt MySQL auch die Ganzzahltypen TINYINT, MEDIUMINT und BIGINT, wie oben aufgelistet. Ein andere Erweiterung wird von MySQL unterstützt, um optional die Anzeigebreite eines Ganzzahlwerts in Klammern festzulegen, die auf das Basis-Schlüsselwort des Typs folgen (zum Beispiel INT(4)). Die optionale Breitenspezifizierung wird benutzt, um die Anzeige von Werten, deren Breite geringer ist als für die Spalte festgelegt, linksseitig mit Leerzeichen aufzufüllen. Das begrenzt allerdings nicht den Wertebereich, der in der Spalte gespeichert werden kann, noch die Anzahl von Ziffern, die bei Werten angezeigt werden, die die angegebene Breite für die Spalte überschreiten. In Verbindung mit dem optionalen Erweiterungsattribut ZEROFILL wird - statt vorgabemäßig mit Leerzeichen - mit Nullen aufgefüllt. Bei einer Spalte zum Beispiel, die als INT(5) ZEROFILL deklariert wurde, wird 4 als 00004 dargestellt. Beachten Sie, dass Werte in einer Ganzzahlspalte, die größer sind als die Anzeigebreite, Probleme bei der Erzeugung temporärer Tabellen für einige komplizierte Joins durch MySQL auftreten können, weil MySQL in diesen Fällen darauf vertraut, dass die Daten in die Original-Spaltenbreite passten.

Alle Ganzzahl-Typen können ein optionales (Nicht-Standard-) Attribut UNSIGNED haben. Vorzeichenlose Werte können dafür benutzt werden, nur positive Zahlen in einer Spalte zuzulassen, wenn Sie eine Wertebereich brauchen, der etwas größer ausfällt.

Der FLOAT-Typ wird benutzt, um näherungsweise numerische Datentypen zu repräsentieren. Der ANSI/ISO-SQL92-Standard erlaubt eine optionale Festlegung der Genauigkeit (aber nicht den Wertebereich des Exponenten) in Bits, gefolgt vom Schlüsselwort FLOAT in Klammern. Die MySQL-Implementation unterstützt ebenfalls diese optionale Genauigkeitsfestlegung. Wenn das Schlüsselwort FLOAT für einen Spaltentyp ohne Genauigkeitsfestlegung benutzt wird, benutzt MySQL 4 Bytes, um die Werte zu speichern. Eine abweichende Syntax wird ebenfalls unterstützt, wobei zwei Zahlen in Klammern dem FLOAT-Schlüsselwort folgen. Mit dieser Option legt die erste Zahl wie gehabt den Speicherbedarf für den Wert in Bytes fest, und die zweite Zahl legt die Anzahl von Ziffern fest, die nach dem Dezimalpunkt gespeichert und angezeigt werden sollen (wie bei DECIMAL und NUMERIC). Wenn MySQL in einer solchen Spalte einen Wert mit mehr Dezimalziffern nach dem Dezimalpunkt speichern soll als für die Spalte festgelegt, wird der Wert beim Speichern gerundet, um die zusätzlichen Ziffern zu entfernen.

Die REAL- und DOUBLE PRECISION-Typen akzeptieren keine Genauigkeitsfestlegungen. Als Erweiterung zum ANSI/ISO-SQL92-Standard erkennt MySQL DOUBLE als ein Synonym für den DOUBLE PRECISION-Typ. Im Gegensatz zur Anforderung des Standard, dass die Genauigkeit für REAL kleiner sein muss als die für DOUBLE PRECISION, implementiert MySQL beide als 8-Byte-Fließkommawerte doppelter Genauigkeit (wenn er nicht im ``ANSI-Modus" läuft). Für maximale Portabilität sollte Code, der die Speicherung näherungsweiser numerischer Daten erfordert, FLOAT oder DOUBLE PRECISION ohne Festlegung der Genauigkeit oder Anzahl von Dezimalstellen benutzen.

Wenn ein Wert in einer numerischen Spalte gespeichert werden soll, der ausserhalb des erlaubten Wertebereichs des Spaltentyps ist, schneidet MySQL den Wert auf den entsprechenden Endpunkt des Wertebereichs ab und speichert statt dessen diesen Wert.

Der Wertebereich einer INT-Spalte ist zum Beispiel -2147483648 bis 2147483647. Wenn Sie versuchen, -9999999999 in eine INT-Spalte einzufügen, wird der Wert auf den unteren Endpunkt des Bereichs abgeschnitten, und es wird -2147483648 gespeichert. Gleichermaßen wird beim Einfügen in eine solche Spalte nicht 9999999999, sondern 2147483647 gespeichert.

Wenn die INT-Spalte UNSIGNED ist, ist die Größe des Wertebereichs dieselbe, aber ihre Endpunkte verschieben sich zu 0 und 4294967295. Wenn Sie versuchen, -9999999999 bzw. 9999999999 zu speichern, werden die in der Spalte gespeicherten Werte statt dessen zu 0 bzw. 4294967296.

Umwandlungen, die aufgrund von Abschneiden geschehen, werden als ``Warnungen" bei ALTER TABLE, LOAD DATA INFILE, UPDATE und in mehrzeiligen INSERT-Statements berichtet.

7.2.2. Datums- und Zeit-Typen

Die Datums- und Zeit-Typen sind DATETIME, DATE, TIMESTAMP, TIME und YEAR. Jeder dieser Typen hat einen zulässigen Wertebereich sowie einen ``0"-Wert, der benutzt wird, wenn Sie einen wirklich unzulässigen Wert speichern. Beachten Sie, dass MySQL es zuläßt, dass Sie bestimmte 'nicht ganz' zulässige Datumswerte speichern, zum Beispiel 1999-11-31. Der Grund hierfür ist, dass wir meinen, dass es in der Verantwortung der Applikation liegt, Datumsüberprüfungen vorzunehmen, und nicht beim SQL-Server. Um Datumsprüfungen 'schnell' zu machen, überprüft MySQL nur, dass der Monat im Bereich 0 bis 12 liegt und der Tag im Bereich 0 bis 31. Diese Bereiche sind deshalb so definiert, weil es MySQL zuläßt, dass Sie in einer DATE- oder DATETIME-Spalte Datumsangaben speichern, bei denen der Tag oder Monat-Tag 0 sind. Das ist extrem nützlich für Applikationen, die einen Geburtstag speichern müssen, dessen exaktes Datum unbekannt ist. In diesem Fall können Sie einfach Datumsangaben wie 1999-00-00 oder 1999-01-00 speichern. (Sie können nicht erwarten, von Funktionen wie DATE_SUB() oder DATE_ADD für solche Datumsangaben korrekte Werte zu erhalten.)

Einige allgemeine Überlegungen, die man im Kopf behalten sollte, wenn man mit Datums- und Zeit-Typen arbeitet:

- MySQL ruft Werte für einen gegebenen Datums- oder Zeit-Typ in einem Standard-Format ab, versucht aber, eine Vielzahl von Formaten zu interpretieren, die Sie bereit stellen (wenn Sie zum Beispiel einen Wert angeben, der zugewiesen oder mit einem Datums- oder Zeit-Typ verglichen werden soll). Dennoch werden nur die in den folgenden Abschnitten beschriebenen Formate unterstützt. Es wird davon ausgegangen, dass Sie zulässige Werte bereitstellen; und es können unvorhersehbare Ergebnisse

zustande kommen, wenn Sie Werte in anderen Formaten angeben.

- Obwohl MySQL versucht, Werte in verschiedenen Formaten zu interpretieren, erwartet er immer, dass der Jahresanteil von Datumswerten ganz links steht. Datumsangaben müssen in der Reihenfolge Jahr - Monat - Tag gemacht werden (zum Beispiel `'98-09-04'`) statt in der Reihenfolge Monat - Tag - Jahr oder Tag - Monat - Jahr, die anderswo häufig gebraucht werden (zum Beispiel `'09-04-98'`, `'04-09-98'`).

- MySQL wandelt einen Datums- oder Zeitwert automatisch in eine Zahl um, wenn der Wert in einem numerischen Zusammenhang benutzt wird, und umgekehrt.

- Wenn MySQL auf einen Datums- oder Zeitwert trifft, der ausserhalb des Wertebereichs oder in sonstiger Weise für den Typ nicht zulässig ist (siehe Anfang dieses Abschnitts), wird der Wert zum ``0''-Wert dieses Typs umgewandelt. (Die Ausnahme ist, dass `TIME`-Werte ausserhalb des Wertebereichs auf den entsprechenden Endpunkt des `TIME`-Wertebereichs abgeschnitten werden.) Die unten stehende Tabelle zeigt das Format des ``0''-Werts für jeden Typ:

Spaltentyp	**``0''-Wert**
`DATETIME`	`'0000-00-00 00:00:00'`
`DATE`	`'0000-00-00'`
`TIMESTAMP`	`00000000000000` (Länge abhängig von der Anzeigebreite)
`TIME`	`'00:00:00'`
`YEAR`	`0000`

- Die ``0''-Werte sind speziell, aber Sie können diese explizit speichern oder auf sie verweisen, indem Sie die in der Tabelle dargestellten Werte benutzen. Sie können das auch mit den Werten `'0'` oder `0` machen, die leichter zu schreiben sind.

- ``0''-Datums- oder -Zeitwerte, die über **MyODBC** benutzt werden, werden in **MyODBC**-Version 2.50.12 und höher automatisch in `NULL` umgewandelt, weil ODBC solche Werte nicht handhaben kann.

7.2.2.1. Jahr-2000-Probleme und Datumstypen

MySQL selbst ist Jahr-2000-konform (Jahr-2000-sicher, see Abschnitt 2.2.4, „Jahr-2000-Konformität“), aber Eingabewerte, die an MySQL übergeben werden, sind das möglicherweise nicht. Jede Eingabe von Jahreswerten mit 2 Ziffern ist mehrdeutig, weil das Jahrhundert unbekannt ist. Solche Werte müssen in 4-stellige Form umgedeutet werden, weil MySQL Jahre intern mit 4 Ziffern speichert.

Bei `DATETIME`-, `DATE`-, `TIMESTAMP`- und `YEAR`-Typen interpretiert MySQL Datumsangaben mit mehrdeutigen Jahreswerten nach folgenden Regeln:

- Jahreswerte im Bereich `00 bis 69` werden in `2000 bis 2069` umgewandelt.

- Jahreswerte im Bereich `70 bis 99` werden in `1970 bis 1999` umgewandelt.

Denken Sie daran, dass diese Regeln nur eine vernünftige Schätzung dessen bedeuten, was die Daten tatsächlich darstellen sollen. Wenn die von MySQL benutzten Heuristiken keine korrekten Werte ergeben, müssen Sie eindeutige Eingaben in Form 4-stelliger Jahreswerte bereit stellen.

`ORDER BY` sortiert 2-stellige `YEAR/DATE/DATETIME`-Typen korrekt.

Beachten Sie, dass einige Funktionen wie `MIN()` und `MAX()` ein `TIMESTAMP / DATE` in eine Zahl umwandeln. Das heißt, dass ein Zeitstempel mit einer 2-stelligen Jahresangabe bei diesen Funktionen nicht korrekt funktioniert. Das kann in diesem Fall dadurch behoben werden, dass der `TIMESTAMP / DATE` in ein 4-stelliges Jahresformat umgewandelt wird, oder etwas wie `MIN(DATE_ADD(zeitstempel,INTERVAL 0 DAYS))` benutzt wird.

7.2.2.2. Die `DATETIME`-, `DATE`- und `TIMESTAMP`-Typen

Die `DATETIME`-, `DATE`- und `TIMESTAMP`-Typen sind verwandt. Dieser Abschnitt beschreibt ihre Charakteristiken, wo sie sich ähnlich sind und wo sie sich unterscheiden.

Der `DATETIME`-Typ wird benutzt, wenn Sie Werte brauchen, die sowohl Datums- als auch Zeitinformationen beinhalten. MySQL ruft `DATETIME`-Werte ab und zeigt sie an im `'YYYY-MM-DD HH:MM:SS'`-Format. Der unterstützte Wertebereich ist `'1000-01-01 00:00:00'` bis `'9999-12-31 23:59:59'`. (``Unterstützt'' heißt, dass frühere Werte zwar funktionieren können, dass es aber keine Garantie dafür gibt.)

Der `DATE`-Typ wird benutzt, wenn Sie nur einen Datumswert brauchen, ohne Zeitanteil. MySQL ruft `DATE`-Werte ab und zeigt sie

an im 'YYYY-MM-DD'-Format. Der unterstützte Wertebereich ist '1000-01-01' bis '9999-12-31'.

Der TIMESTAMP-Typ ist ein Typ, den Sie dafür benutzen können, um INSERT- oder UPDATE-Operationen mit dem aktuellen Datum und der aktuellen Zeit zu stempeln. Wenn Sie mehrfache TIMESTAMP-Spalten haben, wird nur die erste automatisch aktualisiert.

Die automatische Aktualisierung der TIMESTAMP-Spalte geschieht unter einer der folgenden Bedingungen:

- Die Spalte wird in einem INSERT- oder LOAD DATA INFILE-Statement nicht explizit angegeben.
- Die Spalte wird in einem UPDATE-Statement nicht explizit angegeben, aber ein anderer Spaltenwert ändert sich. (Beachten Sie, dass ein UPDATE, das eine Spalte auf einen Wert setzt, den diese bereits hat, nicht dazu führt, dass die TIMESTAMP-Spalte aktualisiert wird, weil MySQL das Aktualisieren in einem solchen Fall auf Effizienzgründen ignoriert.)
- Wenn Sie die TIMESTAMP-Spalte explizit auf NULL setzen.

TIMESTAMP-Spalten abgesehen von der ersten können ebenfalls auf das aktuelle Datum und die aktuelle Zeit gesetzt werden. Setzen Sie die Spalte einfach auf NULL oder auf NOW().

Sie können jede TIMESTAMP-Spalte auf einen Wert setzen, der vom aktuellen Datum und der aktuellen Zeit abweicht, indem Sie sie explizit auf den gewünschten Wert setzen. Das gilt sogar für die erste TIMESTAMP-Spalte. Sie können diese Eigenschaft benutzen, wenn Sie einen TIMESTAMP auf das aktuelle Datum und die aktuelle Zeit setzen wollen, wenn Sie eine Zeile erzeugen, nicht aber, wenn die Zeile später aktualisiert wird:

- Lassen Sie MySQL die Spalte setzen, wenn die Zeile erzeugt wird. Das initialisiert sie auf das aktuelle Datum und die aktuelle Zeit.
- Wenn Sie nachfolgende Aktualisierungen anderer Spalten in der Zeile durchführen, setzen Sie die TIMESTAMP-Spalte explizit auf ihren aktuellen Wert.

Auf der anderen Seite finden Sie vielleicht mindestens so einfach, eine DATETIME-Spalte zu benutzen, die Sie auf NOW() initialisieren, wenn die Zeile erzeugt wird, und die Sie bei nachfolgenden Aktualisierungen nicht anfassen.

TIMESTAMP-Werte haben einen Wertebereich von 1970 bis irgendwann im Jahr 2037, bei einer Auflösung von einer Sekunde. Werte werden als Zahlen angezeigt.

Das Format, in dem MySQL TIMESTAMP-Werte abruft und anzeigt, hängt von der Anzeigebreite ab, wie in der obigen Tabelle dargestellt. Das `volle' TIMESTAMP-Format ist 14 Ziffern, aber TIMESTAMP-Spalten können mit kürzeren Anzeigebreiten angelegt werden:

Spaltentyp	Anzeigeformat
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Alle TIMESTAMP-Spalten haben dieselbe Speichergröße, unabhängig von der Anzeigebreite. Die gebräuchlichsten Anzeigebreiten sind 6, 8, 12 und 14. Sie können zur Zeit der Tabellenerzeugung beliebige Anzeigebreiten festlegen, aber Werte von 0 oder größer als 14 werden auf 14 gesetzt. Ungerade Werte im Bereich von 1 bis 13 werden auf die nächst höhere gerade Zahl gesetzt.

Sie können DATETIME-, DATE- und TIMESTAMP-Werte mit folgenden Formaten festlegen:

- Als eine Zeichenkette im 'YYYY-MM-DD HH:MM:SS'- oder 'YY-MM-DD HH:MM:SS'-Format. Eine ``entspannte" Syntax ist zugelassen - jedes Satzzeichen kann als Begrenzer zwischen Datumsanteilen oder Zeitanteilen verwendet werden. Beispielsweise sind '98-12-31 11:30:45', '98.12.31 11+30+45', '98/12/31 11*30*45' und '98@12@31 11^30^45' äquivalent.
- Als eine Zeichenkette im 'YYYY-MM-DD'- oder 'YY-MM-DD'-Format. Auch hier ist eine ``entspannte" Syntax zugelassen. Beispielsweise sind '98-12-31', '98.12.31', '98/12/31' und '98@12@31' äquivalent.

- Als eine Zeichenkette ohne Begrenzer im `'YYYYMMDDHHMMSS'`- oder `'YYMMDDHHMMSS'`-Format, vorausgesetzt, die Zeichenkette ergibt als Datum einen Sinn. `'19970523091528'` und `'970523091528'` beispielsweise werden als `'1997-05-23 09:15:28'` interpretiert, aber `'971122129015'` ist unzulässig (es hat einen Minutenanteil, der keinen Sinn ergibt) und wird in `'0000-00-00 00:00:00'` umgewandelt.
- Als eine Zeichenkette ohne Begrenzer im `'YYYYMMDD'`- oder `'YYMMDD'`-Format, vorausgesetzt, die Zeichenkette ergibt als Datum einen Sinn. `'19970523'` und `'970523'` werden als `'1997-05-23'` interpretiert, aber `'971332'` ist unzulässig (es hat einen Monatsanteil und einen Tagesanteil, der keinen Sinn ergibt) und wird in `'0000-00-00'` umgewandelt.
- Als eine Zahl im `YYYYMMDDHHMMSS`- oder `YYMMDDHHMMSS`-Format, vorausgesetzt, die Zahl ergibt als Datum einen Sinn. `19830905132800` und `830905132800` zum Beispiel werden als `'1983-09-05 13:28:00'` interpretiert.
- Als eine Zahl im `YYYYMMDD`- oder `YYMMDD`-Format, vorausgesetzt, die Zahl ergibt als Datum einen Sinn. `19830905` und `830905` zum Beispiel werden als `'1983-09-05'` interpretiert.
- Als Ergebnis einer Funktion, die einen Wert zurückgibt, der in einem `DATETIME`-, `DATE`- oder `TIMESTAMP`-Zusammenhang einen Sinn ergibt, wie `NOW()` oder `CURRENT_DATE`.

Unzulässige `DATETIME`-, `DATE`- oder `TIMESTAMP`-Werte werden in den ``0''-Wert des jeweiligen Typs umgewandelt (`'0000-00-00 00:00:00'`, `'0000-00-00'` oder `00000000000000`).

Bei Werten, die als Zeichenketten angegeben werden, die Begrenzer für Datumsanteile enthalten, ist es nicht notwendig, zwei Ziffern für Monats- oder Tageswerte anzugeben, die weniger als `10` sind. `'1979-6-9'` ist dasselbe wie `'1979-06-09'`. Gleichermaßen ist es bei Zeichenketten, die Begrenzer für Zeitanteile enthalten, nicht notwendig, zwei Ziffern für Stunden-, Monats- oder Sekundenwerte anzugeben, die weniger als `10` sind. `'1979-10-30 1:2:3'` ist dasselbe wie `'1979-10-30 01:02:03'`.

Werte, die als Zahlen angegeben sind, sollten 6, 8, 12 oder 14 Ziffern lang sein. Wenn die Zahl 8 oder 14 Ziffern lang ist, wird angenommen, dass sie im `YYYYMMDD`- oder `YYYYMMDDHHMMSS`-Format ist und dass das Jahr durch die ersten 4 Ziffern angegeben wird. Wenn die Zahl 6 oder 12 Ziffern lang ist, wird angenommen, dass sie im `YYMMDD`- oder `YYMMDDHHMMSS`-Format ist und dass das Jahr durch die ersten 2 Ziffern angegeben wird. Zahlen, die nicht diesen Längen entsprechen, werden interpretiert, als ob sie mit führenden Nullen auf die nächst mögliche Länge gebracht worden wären.

Werte, die als nicht begrenzte Zeichenketten angegeben werden, werden interpretiert, indem ihre Länge als gegeben angenommen wird. Wenn die Zeichenkette 8 oder 14 Zeichen lang ist, wird angenommen, dass das Jahr durch die ersten 4 Zeichen angegeben wird. Ansonsten wird angenommen, dass das Jahr durch die ersten 2 Zeichen angegeben wird. Die Zeichenkette wird von links nach rechts interpretiert, um die Jahres-, Monats-, Tages-, Stunden- und Sekundenwerte zu finden, für so viele Anteile, wie in der Zeichenkette vorkommen. Das bedeutet, dass Sie keine Zeichenketten benutzen sollten, die weniger als 6 Zeichen haben. Wenn Sie zum Beispiel `'9903'` angeben, in der Annahme, dass das März 1999 darstellt, werden Sie feststellen, dass MySQL einen ``0''-Datumswert in Ihre Tabelle einfügt. Das liegt daran, dass die Jahres- und Monatswerte `99` und `03` sind, aber der Tagesanteil fehlt (0), so dass der Wert kein zulässiges Datum darstellt.

`TIMESTAMP`-Spalten speichern zulässige Werte mit der vollen Genauigkeit, mit der der Wert angegeben wurde, unabhängig von der Anzeigebreite. Das hat mehrere Auswirkungen:

- Geben Sie immer Jahr, Monat und Tag an, selbst wenn Ihre Spaltentypen `TIMESTAMP(4)` oder `TIMESTAMP(2)` sind. Ansonsten wäre der Wert kein zulässiges Datum und `0` würde gespeichert werden.
- Wenn Sie `ALTER TABLE` benutzen, um eine enge `TIMESTAMP`-Spalte breiter zu machen, werden Informationen angezeigt, die vorher ``versteckt'' waren.
- Gleichermaßen führt das Verengen einer `TIMESTAMP`-Spalte nicht dazu, dass Informationen verloren gehen, ausser in dem Sinn, dass weniger Informationen dargestellt werden, wenn die Werte angezeigt werden.
- Obwohl `TIMESTAMP`-Werte mit voller Genauigkeit gespeichert werden, ist die einzige Funktion, die direkt mit dem zugrunde liegenden gespeicherten Wert arbeitet, `UNIX_TIMESTAMP()`. Alle anderen Funktionen arbeiten mit dem formatierten, abgerufenen Wert. Das bedeutet, Sie können keine Funktionen wie `HOUR()` oder `SECOND()` benutzen, wenn nicht auch der relevante Teil des `TIMESTAMP`-Werts im formatierten Werte enthalten ist. Wenn zum Beispiel der `HH`-Teil einer `TIMESTAMP`-Spalte nicht angezeigt wird, wenn die Anzeigebreite nicht mindestens 10 beträgt, wird der Versuch, `HOUR()` auf kürzere `TIMESTAMP`-Werte anzuwenden, unsinnige Ergebnisse erzeugen.

Bis zu einem gewissen Grad können Sie einem Objekt eines Datumstyp Werte eines anderen Datumstyps zuweisen. Jedoch kann eine Änderung des Wertes oder ein Informationsverlust eintreten:

- Wenn Sie einem `DATETIME`- oder `TIMESTAMP`-Objekt einen `DATE`-Wert zuweisen, wird der Zeitanteil im Ergebniswert auf `'00:00:00'` gesetzt, weil der `DATE`-Wert keine Zeitinformationen enthält.

- Wenn Sie einem `DATE`-Objekt einen `DATETIME`- oder `TIMESTAMP`-Wert zuweisen, wird der Zeitanteil des Ergebniswerts gelöscht, weil der `DATE`-Typ keine Zeitinformationen speichert.
- Denken Sie daran, dass `DATETIME`-, `DATE`- und `TIMESTAMP`-Werte zwar in denselben Formaten angegeben werden können, dass die Typen jedoch nicht alle denselben Wertebereich haben. `TIMESTAMP`-Werte zum Beispiel können nicht früher als `1970` oder später als `2037` sein. Das bedeutet, dass ein Datum wie `'1968-01-01'`, was als `DATETIME` oder `DATE`-Wert zulässig wäre, kein gültiger `TIMESTAMP`-Wert ist und in `0` umgewandelt wird, wenn er einem solchen Objekt zugewiesen wird.

Seien Sie auf der Hut vor Fallstricken, wenn Sie Datumswerte angeben:

- Das entspannte Format läßt Werte als Zeichenketten zu, die täuschen können. Ein Wert wie `'10:11:12'` zum Beispiel sieht wegen des '`:`'-Begrenzers wie ein Zeitwert aus, wird er aber in einem Datums-Zusammenhang benutzt, wird er als das Datum `'2010-11-12'` interpretiert. Der Wert `'10:45:15'` wird in `'0000-00-00'` umgewandelt, weil `'45'` kein zulässiger Monat ist.
- Jahreswerte, die als zwei Ziffern angegeben werden, sind mehrdeutig, weil das Jahrhundert unbekannt ist. unknown. MySQL interpretiert 2-stellige Jahreswerte nach folgenden Regeln:
  - Jahreswerte im Bereich `00 bis 69` werden in `2000 bis 2069` umgewandelt.
  - Jahreswerte im Bereich `70 bis 99` werden in `1970 bis 1999` umgewandelt.

7.2.2.3. Der `TIME`-Typ

MySQL ruft `TIME`-Werte ab und zeigt sie an im `'HH:MM:SS'`-Format (oder `'HHH:MM:SS'`-Format für große Stundenwerte). `TIME`-Werte rangieren von `'-838:59:59'` bis `'838:59:59'`. Der Grund dafür, dass der Stundenanteil so Groß sein kann, liegt darin, dass der `TIME`-Typ nicht nur benutzt werden kann, um die Tageszeit zu repräsentieren (wobei die Stunden weniger als 24 sein müssen), sondern auch abgelaufene Zeit oder ein Zeitintervall zwischen zwei Ereignissen (was viel größer als 24 Stunden oder sogar negativ sein kann).

Sie können `TIME`-Werte in einer Vielzahl von Formaten angeben:

- Als eine Zeichenkette im `'D HH:MM:SS.bruchteil'`-Format. (Beachten Sie, dass MySQL bislang nicht den Bruchteil für die TIME-Spalte speichert.) Man kann auch folgende ``entspannte" Syntax benutzen:

  `HH:MM:SS.bruchteil`, `HH:MM:SS`, `HH:MM`, `D HH:MM:SS`, `D HH:MM`, `D HH` oder `SS`. Hierbei ist `D` Tage zwischen 0 und 33.
- Als eine Zeichenkette ohne Begrenzer im `'HHMMSS'`-Format, vorausgesetzt, dass diese als Zeitangabe einen Sinn ergibt. `'101112'` zum Beispiel wird als `'10:11:12'` interpretiert, aber `'109712'` ist unzulässig (es hat einen Minutenanteil, der keinen Sinn ergibt) und wird in `'00:00:00'` umgewandelt.
- Als eine Zahl im `HHMMSS`-Format, vorausgesetzt, dass diese als Zeitangabe einen Sinn ergibt. `101112` zum Beispiel wird als `'10:11:12'` interpretiert. Folgende alternativen Formate werden ebenfalls verstanden: `SS`, `MMSS`, `HHMMSS`, `HHMMSS.bruchteil`. Beachten Sie, dass MySQL bislang noch nicht den Bruchteil speichert.
- Als Ergebnis einer Funktion, die einen Wert zurück gibt, der in einem `TIME`-Zusammenhang akzeptabel ist, wie `CURRENT_TIME`.

Bei `TIME`-Werten, die als Zeichenketten angegeben sind, die einen Begrenzer für den Zeitanteil beinhalten, ist es nicht notwendig, zwei Ziffern für Stunden-, Minuten- oder Sekunden-Werte anzugeben, die weniger als `10` sind. `'8:3:2'` ist dasselbe wie `'08:03:02'`.

Seien Sie vorsichtig damit, einer `TIME`-Spalte ``kurze" `TIME`-Werte zuzuweisen. Ohne Semikolon interpretiert MySQL Werte unter der Annahme, dass die am weitesten rechts stehenden Ziffern Sekunden repräsentieren. (MySQL interpretiert `TIME`-Werte als vergangene Zeit statt als Tageszeit.) Sie könnten zum Beispiel denken, dass `'1112'` und `1112` `'11:12:00'` bedeuten (12 Minuten nach 11 Uhr), aber MySQL interpretiert sie als `'00:11:12'` (11 Minuten, 12 Sekunden). Gleichermaßen wird `'12'` und `12` als `'00:00:12'` interpretiert. `TIME`-Werte mit Semikolon werden statt dessen immer als Tageszeit interpretiert. Das heißt, `'11:12'` bedeutet `'11:12:00'`, nicht `'00:11:12'`.

Werte, die ausserhalb des `TIME`-Wertebereichs liegen, ansonsten aber zulässig sind, werden auf den entsprechenden Endpunkt des Wertebereichs abgeschnitten. `'-850:00:00'` bzw. `'850:00:00'` werden in `'-838:59:59'` bzw. `'838:59:59'` umgewandelt.

Unzulässige `TIME`-Werte werden in `'00:00:00'` umgewandelt. Beachten Sie, dass es keine Möglichkeit gibt zu unterscheiden,

wenn ein Wert von `'00:00:00'` in einer Tabelle gespeichert ist, ob dieser originär als `'00:00:00'` eingegeben wurde oder ob es ein unzulässiger Wert war, weil `'00:00:00'` selbst ein zulässiger `TIME`-Wert ist.

7.2.2.4. Der YEAR-Typ

Der `YEAR`-Typ ist ein 1-Byte-Typ, der für die Darstellung von Jahren benutzt wird.

MySQL ruft `YEAR`-Werte ab und speichert sie im `YYYY`-Format. Der Wertebereich ist `1901` bis `2155`.

Sie können `YEAR`-Werte in einer Vielzahl von Formaten angeben:

- Als vierstellige Zeichenkette im Wertebereich von `'1901'` bis `'2155'`.
- Als vierstellige Zahl im Wertebereich von `1901` bis `2155`.
- Als zweistellige Zeichenkette im Wertebereich von `'00'` bis `'99'`. Werte in den Bereichen von `'00'` bis `'69'` und `'70'` bis `'99'` werden in `YEAR`-Werte in den Bereichen von `2000` bis `2069` und `1970` bis `1999` umgewandelt.
- Als zweistellige Zahl im Wertebereich von `1` bis `99`. Werte in den Bereichen von `1` bis `69` und `70` bis `99` werden in `YEAR`-Werte in den Bereichen von `2001` bis `2069` und `1970` bis `1999` umgewandelt. Beachten Sie, dass der Wertebereich für zweistellige Zahlen sich geringfügig vom Wertebereich für zweistellige Zeichenketten unterscheidet, weil Sie 0 nicht direkt als Zahl eingeben können und sie dann als `2000` interpretiert wird. Sie *müssen* sie als Zeichenkette `'0'` oder `'00'` angeben, oder sie wird als `0000` interpretiert.
- Als Ergebnis einer Funktion, die einen Wert zurück gibt, der in einem `YEAR`-Zusammenhang akzeptabel ist, wie `NOW()`.

Unzulässige `YEAR`-Werte werden in `0000` umgewandelt.

7.2.3. Zeichenketten-Typen

Die Zeichenketten-Typen sind `CHAR`, `VARCHAR`, `BLOB`, `TEXT`, `ENUM` und `SET`. Dieser Abschnitt beschreibt, wie diese Typen funktionieren, ihren Speicherbedarf und wie sie in Anfragen benutzt werden.

7.2.3.1. Die CHAR- und VARCHAR-Typen

Die `CHAR`- und `VARCHAR`-Typen sind ähnlich, unterscheiden sich aber in der Art, wie sie gespeichert und abgerufen werden.

Die Länge einer `CHAR`-Spalte wird auf die Länge festgelegt, die Sie bei der Erzeugung der Tabelle angeben. Die Länge kann zwischen 1 und 255 variieren. (Ab MySQL-Version 3.23 kann die Länge zwischen 0 und 255 liegen.) Wenn `CHAR`-Werte gespeichert werden, werden sie am rechten Ende bis zur festgelegten Länge mit Leerzeichen aufgefüllt. Wenn `CHAR`-Werte abgerufen werden, werden die Leerzeichen am Ende entfernt.

Werte in `VARCHAR`-Spalten sind Zeichenketten variabler Länge. Sie können eine `VARCHAR`-Spalte mit jeder Länge zwischen 1 und 255 deklarieren, genau wie für `CHAR`-Spalten. Im Gegensatz zu `CHAR` werden `VARCHAR`-Werte jedoch nur mit so vielen Zeichen wie nötig gespeichert, plus 1 Byte, um die Länge zu speichern. Die Werte werden nicht aufgefüllt; statt dessen werden Leerzeichen am Ende beim Speichern entfernt. (Diese Entfernung von Leerzeichen weicht von der ANSI-SQL-Spezifikation ab.)

Wenn Sie einer `CHAR`- oder `VARCHAR`-Spalte einen Wert zuweisen, der die maximale Spaltenlänge überschreitet, wird der Wert so zurecht geschnitten, das er passt.

Die unten stehende Tabelle stellt die Unterschiede zwischen den beiden Spaltentypen dar, indem das Ergebnis der Speicherung unterschiedlicher Zeichenkettenwerte in `CHAR(4)`- und `VARCHAR(4)`-Spalten gezeigt wird:

Wert	`CHAR(4)`	**Speicherbedarf**	`VARCHAR(4)`	**Speicherbedarf**
`''`	`'  '`	4 Bytes	`''`	1 Byte
`'ab'`	`'ab '`	4 Bytes	`'ab'`	3 Bytes
`'abcd'`	`'abcd'`	4 Bytes	`'abcd'`	5 Bytes
`'abcdefgh'`	`'abcd'`	4 Bytes	`'abcd'`	5 Bytes

Die Werte, die aus den `CHAR(4)`- und `VARCHAR(4)`-Spalten abgerufen werden, sind in jedem Fall gleich, weil Leerzeichen am Ende von `CHAR`-Spalten beim Abruf entfernt werden.

Werte in `CHAR`- und `VARCHAR`-Spalten werden unabhängig von der Groß-/Kleinschreibung sortiert und verglichen, es sei denn, beim Erzeugen der Tabelle wurde das `BINARY`-Attribut festgelegt. Das `BINARY`-Attribut bedeutet, dass Spaltenwerte abhängig von der Groß-/Kleinschreibung in Übereinstimmung mit der ASCII-Reihenfolge der Maschine sortiert und verglichen werden, auf der der MySQL-Server läuft. `BINARY` beeinflusst nicht, wie die Spalte gespeichert oder abgerufen wird.

Das BINARY-Attribut ist 'klebrig', das heißt, dass der gesamte Ausdruck als ein BINARY-Wert verglichen wird, sobald eine BINARY-Spalte im Ausdruck benutzt wird.

MySQL ändert eventuell 'still' den Typ von CHAR- oder VARCHAR-Spalten bei der Tabellenerzeugung.

See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“.

7.2.3.2. Die BLOB- und TEXT-Typen

Ein BLOB ist großes Binärobjekt (Binary Large OBject), das eine variable Menge von Daten enthalten kann. Die vier BLOB-Typen TINYBLOB, BLOB, MEDIUMBLOB und LONGBLOB unterscheiden sich nur hinsichtlich der maximalen Länge der Werte, die sie aufnehmen können. See Abschnitt 7.2.6, „Speicherbedarf von Spaltentypen“.

Die vier TEXT-Typen TINYTEXT, TEXT, MEDIUMTEXT und LONGTEXT entsprechen den vier BLOB-Typen und haben dieselben maximalen Längen und denselben Speicherbedarf. Der einzige Unterschied zwischen BLOB- und TEXT-Typen ist, dass beim Sortieren und Vergleichen bei BLOB-Werten Groß-/Kleinschreibung berücksichtigt wird, bei TEXT-Werten dagegen nicht. Mit anderen Worten ist ein TEXT ein BLOB ohne Berücksichtigung der Groß-/Kleinschreibung.

Wenn Sie einer BLOB- oder TEXT-Spalte einen Wert zuweisen, der die maximale Länge des Spaltentyps überschreitet, wird der Wert so zurecht geschnitten, dass er passt.

In fast jeder Hinsicht können Sie eine TEXT-Spalte als eine VARCHAR-Spalte betrachten, die so Groß sein kann, wie Sie wollen. Gleichermaßen können Sie eine BLOB-Spalte als eine VARCHAR BINARY-Spalte betrachten. Die Unterschiede sind:

- Seit MySQL-Version 3.23.2 können Sie Indexe auf BLOB- und TEXT-Spalten anlegen. Ältere Versionen von MySQL unterstützten das nicht.
- Leerzeichen am Ende werden beim Speichern von BLOB- und TEXT-Spalten nicht wie bei VARCHAR-Spalten entfernt.
- BLOB- und TEXT-Spalten können keine DEFAULT-Werte haben.

MyODBC definiert BLOB-Werte als LONGVARBINARY und TEXT-Werte als LONGVARCHAR.

Weil BLOB- und TEXT-Werte extrem lang sein können, treffen Sie bei der Benutzung eventuell auf Beschränkungen:

- Wenn Sie GROUP BY oder ORDER BY für BLOB- oder TEXT-Spalten benutzen wollen, müssen Sie den Spaltenwert in ein Objekt fester Länge umwandeln. Standardmäßig wird das mit der SUBSTRING-Funktion gemacht. Beispiel:

```
mysql> select kommentar from tabelle,substring(kommentar,20) as substr
       ORDER BY substr;
```

  Wenn Sie das nicht tun, werden nur die ersten max_sort_length Bytes der Spalte beim Sortieren benutzt. Der Vorgabewert von max_sort_length ist 1024; dieser Wert kann mit der -O-Option geändert werden, wenn der mysqld-Server gestartet wird. Sie können auf einen Ausdruck, der BLOB- oder TEXT-Werte enthält, gruppieren, indem Sie die Spaltenposition angeben oder ein Alias benutzen:

```
mysql> select id,substring(blob_spalte,1,100) from tabelle
           GROUP BY 2;
mysql> select id,substring(blob_spalte,1,100) as b from tabelle
           GROUP BY b;
```

- Die maximale Größe eines BLOB- oder TEXT-Objekts wird durch seinen Typ festgelegt, aber der größte Wert, den Sie tatsächlich zwischen Client und Server übertragen können, wird von der Menge verfügbaren Arbeitsspeichers und der Größe des Kommunikationspuffers festgelegt. Sie können die Nachrichtenpuffergröße ändern, müssen das aber auf beiden Seiten, also beim Client und beim Server, tun. See Abschnitt 6.5.2, „Serverparameter tunen“.

Beachten Sie, dass intern jeder BLOB- oder TEXT-Wert durch ein separat zugewiesenes Objekt dargestellt wird. Das steht im Gegensatz zu allen anderen Spaltentypen, für die Speicherplatz einmal pro Spalte zugewiesen wird, wenn die Tabelle geöffnet wird.

7.2.3.3. Der ENUM-Typ

Ein ENUM ist ein Zeichenketten-Objekt, dessen Wert normalerweise aus einer Liste zulässiger Werte ausgesucht wird, die explizit bei der Spaltenspezifizierung bei der Tabellenerzeugung aufgezählt werden.

Der Wert kann unter bestimmten Umständen auch die leere Zeichenkette ("") oder NULL sein:

- Wenn Sie in eine ENUM einen ungültigen Wert einfügen (das ist eine Zeichenkette, die es in der Auflistung zugelassener Werte nicht gibt), wird statt dessen die leere Zeichenkette als spezieller Fehlerwert eingefügt. Diese Zeichenkette kann von einer 'normalen' leeren Zeichenkette dadurch unterschieden werden, dass diese Zeichenkette den numerischen Wert 0 hat. Mehr dazu später.
- Wenn ein ENUM als NULL deklariert ist, ist NULL ebenfalls ein zulässiger Wert für die Spalte und der Vorgabewert ist NULL. Wenn ein ENUM als NOT NULL deklariert ist, ist der Vorgabewert das erste Element der Auflistung erlaubter Werte.

Jeder Aufzählungswert hat einen Index:

- Werte der Auflistung zulässiger Elemente in der Spaltenspezifikation fangen mit 1 an.
- Der Indexwert des Fehlerwerts leere Zeichenkette ist 0. Folglich können Sie folgendes SELECT-Statement benutzen, um Zeilen zu finden, denen unzulässige ENUM-Werte zugewiesen wurden:

```
mysql> SELECT * FROM tabelle WHERE enum_spalte=0;
```

- Der Index des NULL-Werts ist NULL.

Wenn beispielsweise eine Spalte als ENUM("eins", "zwei", "drei") festgelegt wurde, kann sie einen der unten dargestellen Werte besitzen. Der Index jedes Werts wird auch dargestellt:

Wert	Index
NULL	NULL
""	0
"eins"	1
"zwei"	2
"drei"	3

Eine Aufzählung kann maximal 65535 Elemente enthalten.

Groß-/Kleinschreibung ist irrelevant, wenn Sie einer ENUM-Spalte Werte zuweisen. Jedoch haben Werte, die später aus der Spalte abgerufen werden, dieselbe Groß-/Kleinschreibung wie die Werte, die für die Festlegung zulässiger Werte bei der Tabellenerzeugung verwendet wurden.

Wenn Sie eine ENUM in einem numerischen Zusammenhang benutzen, wird der Index des Spaltenwerts zurückgegeben. Sie können beispielsweise numerische Werte aus einer ENUM-Spalte wie folgt abrufen:

```
mysql> SELECT enum_spalte+0 FROM tabelle;
```

Wenn Sie eine Zahl in eine ENUM speichern, wird die Zahl als Index behandelt und der gespeicherte Wert ist das Aufzählungselement mit diesem Index. (Das funktioniert jedoch nicht bei LOAD DATA, was alle Eingaben als Zeichenketten behandelt.)

ENUM-Werte werden in der Reihenfolge sortiert, wie die Aufzählungselemente bei der Spaltenspezifizierung eingegeben wurden. (Mit anderen Worten werden ENUM-Werte nach ihren Indexzahlen sortiert.) So wird beispielsweise "a" vor "b" einsortiert bei ENUM("a", "b"), aber "b" wird vor "a" einsortiert bei ENUM("b", "a"). Die leere Zeichenkette wird vor nicht leeren Zeichenketten und NULL-Werte vor allen anderen Aufzählungswerten einsortiert.

Wenn Sie alle möglichen Werte einer ENUM-Spalte erhalten wollen, benutzen Sie: SHOW COLUMNS FROM tabelle LIKE enum_spalte und gehen die ENUM-Definition in der zweiten Spalte durch.

7.2.3.4. Der SET-Typ

Ein SET ist ein Zeichenketten-Objekt, das 0 oder mehr Werte haben kann, wovon jedes aus einer Auflistung zulässiger Werte stammen muss, die bei der Tabellenerzeugung festgelegt wurden. SET-Spaltenwerte, die aus mehrfachen SET-Elementen bestehen, werden angegeben, indem die Elemente durch Kommas (‘,’) getrennt werden. Daraus ergibt sich, dass SET-Elemente selbst keine Kommas enthalten dürfen.

Eine Spalte beispielsweise, die als SET("eins", "zwei") NOT NULL festgelegt wurde, kann folgende Werte haben:

```
""
"eins"
"zwei"
```

```
"eins,zwei"
```

Eine SET kann maximal 64 unterschiedliche Elemente besitzen.

MySQL speichert SET-Werte numerisch, wobei das niedrigste Bit in der Reihenfolge der gespeicherten Werte dem ersten SET-Element entspricht. Wenn Sie einen SET-Wert in einem numerischen Zusammenhang abrufen, hat der abgerufene Werte Bits gesetzt, die den SET-Elementen, aus denen sich der Spaltenwert zusammensetzt, entspricht. Beispielsweise können Sie numerische Werte aus einer SET-Spalte wie folgt abrufen:

```
mysql> SELECT set_spalte+0 FROM tabelle;
```

Wenn in einer SET-Spalte eine Zahl gespeichert wird, legen die Bits, die in der binären Darstellung der Zahl gesetzt sind, die SET-Elemente im Spaltenwert fest. Angenommen, eine Spalte ist als SET("a","b","c","d") festgelegt, dann haben die Elemente folgende Bitwerte:

SET **Element**	**Dezimalwert**	**Binärwert**
a	1	0001
b	2	0010
c	4	0100
d	8	1000

Wenn Sie dieser Spalte einen Wert von 9 zuweisen, ist das binär 1001. Daher werden der erste und der vierte SET-Wert, die Elemente "a" und "d", ausgewählt, und der Ergebniswert ist "a,d".

Bei einem Wert, der mehr als ein SET-Element enthält, spielt es keine Rolle, in welcher Reihenfolge die Elemente aufgelistet sind, wenn Sie den Wert einfügen. Es spielt ebenfalls keine Rolle, wie oft ein gegebenes Element im Wert aufgelistet ist. Wenn der Wert später abgerufen wird, erscheint jedes Element im Wert einmal, wobei die Elemente in der Reihenfolge erscheinen, in der sie bei der Tabellenerzeugung festgelegt wurden. Wenn eine Spalte beispielsweise als SET("a","b","c","d") festgelegt ist, erscheinen "a,d", "d,a" und "d,a,a,d,d" als "a,d", wenn sie abgerufen werden.

SET-Werte werden numerisch sortiert. NULL-Werte werden vor Nicht-NULL-SET-Werten einsortiert.

Normalerweise führt man SELECT auf eine SET-Spalte mit dem LIKE-Operator oder der FIND_IN_SET()-Funktion aus:

```
mysql> SELECT * FROM tabelle WHERE set_spalte LIKE '%wert%';
mysql> SELECT * FROM tabelle WHERE FIND_IN_SET('wert',set_spalte)>0;
```

Aber auch folgendes funktioniert:

```
mysql> SELECT * FROM tabelle WHERE set_spalte = 'wert1,wert2';
mysql> SELECT * FROM tabelle WHERE set_spalte & 1;
```

Das erste dieser Statements sucht nach einer exakten Übereinstimmung, das zweite sucht Werte, die das erste SET-Element enthalten.

Wenn Sie alle möglichen Werte einer SET-Spalte erhalten wollen, benutzen Sie: SHOW COLUMNS FROM tabelle LIKE set_spalte und gehen die SET-Definition in der zweiten Spalte durch.

7.2.4. Den richtigen Typ für eine Spalte auswählen

Um möglichst effizient zu speichern, benutzen Sie in jedem Fall den präzisesten Typ. Wenn zum Beispiel eine Ganzzahl-Spalte für Werte im Bereich zwischen 1 und 99999 benutzt wird, ist MEDIUMINT UNSIGNED der beste Typ.

Akkurate Darstellung monetärer Werte ist ein häufiges Problem. In MySQL sollten Sie den DECIMAL-Typ benutzen. Dieser wird als Zeichenkette gepeichert, weshalb kein Genauigkeitsverlust auftreten sollte. Wenn Genauigkeit nicht allzu wichtig ist, sollte auch der DOUBLE-Typ ausreichen.

Um hohe Präzision zu erzielen, können Sie immer auch in einen Festkommawert umwandeln, der in einer BIGINT gespeichert wird. Das erlaubt Ihnen, alle Berechnungen mit Ganzzahlen durchzuführen und die Ergebnisse nur wenn notwendig in Fließkommawerte zurückzuwandeln.

7.2.5. Spaltentypen anderer Datenbanken benutzen

Um es einfacher zu machen, Code zu verwenden, der für SQL-Implementationen anderer Hersteller geschrieben wurde, ordnet (mappt) MySQL Spaltentypen zu wie in unten stehender Tabelle dargestellt. Diese Mappings machen es leichter,

Tabellendefinitionen anderer Datenbanken nach MySQL zu verschieben:

Typ anderer Hersteller	MySQL-Typ
`BINARY(NUM)`	`CHAR(NUM) BINARY`
`CHAR VARYING(NUM)`	`VARCHAR(NUM)`
`FLOAT4`	`FLOAT`
`FLOAT8`	`DOUBLE`
`INT1`	`TINYINT`
`INT2`	`SMALLINT`
`INT3`	`MEDIUMINT`
`INT4`	`INT`
`INT8`	`BIGINT`
`LONG VARBINARY`	`MEDIUMBLOB`
`LONG VARCHAR`	`MEDIUMTEXT`
`MIDDLEINT`	`MEDIUMINT`
`VARBINARY(NUM)`	`VARCHAR(NUM) BINARY`

Dass Zuordnen (Mapping) von Spaltentypen geschieht bei der Erzeugung der Tabelle. Wenn Sie eine Tabelle mit Typen erzeugen, die von anderen Herstellern benutzt werden, und dann ein `DESCRIBE tabelle`-Statement absetzen, zeigt MySQL die Tabellenstruktur mit den äquivalenten MySQL-Typen an.

7.2.6. Speicherbedarf von Spaltentypen

Der Speicherbedarf jedes Spaltentyps, der von MySQL unterstützt wird, ist unten nach Kategorie sortiert aufgelistet:

Speicherbedarf für numerische Typen

Spaltentyp	Speicherbedarf
`TINYINT`	1 Byte
`SMALLINT`	2 Bytes
`MEDIUMINT`	3 Bytes
`INT`	4 Bytes
`INTEGER`	4 Bytes
`BIGINT`	8 Bytes
`FLOAT(X)`	4, wenn X <= 24, oder 8, wenn 25 <= X <= 53
`FLOAT`	4 Bytes
`DOUBLE`	8 Bytes
`DOUBLE PRECISION`	8 Bytes
`REAL`	8 Bytes
`DECIMAL(M,D)`	`M+2` Bytes, wenn D > 0, `M+1` Bytes, wenn D = 0 (`D`+2, wenn `M < D`)
`NUMERIC(M,D)`	`M+2` Bytes, wenn D > 0, `M+1` Bytes, wenn D = 0 (`D`+2, wenn `M < D`)

Speicherbedarf für Datums- und Zeit-Typen

Spaltentyp	Speicherbedarf
`DATE`	3 Bytes
`DATETIME`	8 Bytes
`TIMESTAMP`	4 Bytes
`TIME`	3 Bytes
`YEAR`	1 Byte

Speicherbedarf für Zeichenketten-Typen

Spaltentyp	Speicherbedarf
`CHAR(M)`	`M` Bytes, `1 <= M <= 255`
`VARCHAR(M)`	`L`+1 Bytes, wobei `L <= M` und `1 <= M <= 255`
`TINYBLOB`, `TINYTEXT`	`L`+1 Bytes, wobei `L` < 2^8
`BLOB`, `TEXT`	`L`+2 Bytes, wobei `L` < 2^16
`MEDIUMBLOB`, `MEDIUMTEXT`	`L`+3 Bytes, wobei `L` < 2^24
`LONGBLOB`, `LONGTEXT`	`L`+4 Bytes, wobei `L` < 2^32
`ENUM('wert1','wert2',...)`	1 oder 2 Bytes, abhängig von der Anzahl der Aufzählungswerte (65535 Werte maximal)
`SET('wert1','wert2',...)`	1, 2, 3, 4 oder 8 Bytes, abhängig von der Anzahl von SET-Elementen (64 Elemente maximal)

`VARCHAR` und die `BLOB`- und `TEXT`-Typen sind Typen variabler Länge, bei denen der Speicherbedarf von der tatsächlichen Länge der Spaltenwerte abhängt (in der vorstehenden Tabelle dargestellt durch `L`) statt von der maximal möglichen Größe des Typs. `VARCHAR(10)` zum Beispiel kann eine Zeichenkette mit einer maximalen Länge von 10 Zeichen enthalten. Der tatsächliche Speicherbedarf ist die Länge der Zeichenkette (`L`) plus 1 Byte, um die Länge zu speichern. Bei der Zeichenkette `'abcd'` ist `L` 4 und der Speicherbedarf 5 Bytes.

Die `BLOB`- und `TEXT`-Typen benötigen 1, 2, 3 oder 4 Bytes, um die Länge des Spaltenwerts zu speichern, abhängig von der maximal möglichen Länge des Typs. See Abschnitt 7.2.3.2, „Die `BLOB`- und `TEXT`-Typen“.

Wenn eine Tabelle irgend welche Spaltentypen variabler Länge enthält, ist das Datensatzformat ebenfalls von variabler Länge. Beachten Sie, dass MySQL bei der Erzeugung einer Tabelle unter bestimmten Umständen eine Spalte eines Typs variabler Länge in einen Typ fester Länge umwandelt, und umgekehrt. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“.

Die Größe eines `ENUM`-Objekts hängt von der Anzahl unterschiedlicher Aufzählungswerte ab. Bei Aufzählungen mit bis zu 255 möglichen Werten wird 1 Byte benutzt, bei Aufzählungen mit bis zu 65535 Werten 2 Bytes. See Abschnitt 7.2.3.3, „Der `ENUM`-Typ“.

Die Größe eines `SET`-Objekts hängt von der Anzahl unterschiedlicher SET-Elemente ab. Wenn die SET-Größe `N` ist, belegt das Objekt `(N+7)/8` Bytes, gerundet auf 1, 2, 3, 4 oder 8 Bytes. Ein `SET` kann maximal 64 Elemente besitzen. See Abschnitt 7.2.3.4, „Der `SET`-Typ“.

7.3. Funktionen für die Benutzung in `SELECT`- und `WHERE`-Klauseln

Ein `select_ausdruck` oder eine `where_definition` in einem SQL-Statement kann aus jedem beliebigen Ausdruck bestehen, der die unten beschriebenen Funktionen benutzt.

Ein Ausdruck, der `NULL` enthält, erzeugt immer einen `NULL`-Wert, wenn es in der Dokumentation für die Operatoren und Funktionen, die im Ausdruck vorkommen, nicht anders beschrieben ist.

HINWEIS: Zwischen Funktionsname und der folgenden Klammer darf kein Leerraum stehen. Das hilft dem MySQL-Parser, zwischen Funktionsaufrufen und Tabellen- oder Spaltenverweisen zu unterscheiden, die denselben Namen haben wie eine Funktion. Leerzeichen um Argumente herum sind dagegen zulässig.

Sie können MySQL zwingen, Leerzeichen nach dem Funktionsnamen zu akzeptieren, indem Sie `mysqld` mit `--ansi` starten oder `CLIENT_IGNORE_SPACE` bei `mysql_connect()`, benutzen, aber in diesem Fall werden alle Funktionsnamen zu reservierten Wörtern. See Abschnitt 2.7.2, „MySQL im ANSI-Modus laufen lassen“.

Der Kürze zuliebe sind die Ausgaben des `mysql`-Programms in gekürzter Form dargestellt. Daher wird

```
mysql> select MOD(29,9);
+-----------+
| mod(29,9) |
+-----------+
|         2 |
+-----------+
1 rows in set (0.00 sec)
```

wie folgt dargestellt:

```
mysql> select MOD(29,9);
        -> 2
```

7.3.1. Nicht typenspezifische Operatoren und Funktionen

7.3.1.1. Klammer

```
( ... )
```

Benutzen Sie Klammern, um die Reihenfolge der Auswertung in einem Ausdruck zu erzwingen. Beispiel:

```
mysql> select 1+2*3;
        -> 7
mysql> select (1+2)*3;
        -> 9
```

7.3.1.2. Vergleichsoperatoren

Vergleichsoperationen ergeben einen Wert von `1` (TRUE), `0` (FALSE) oder `NULL`. Diese Funktionen funktionieren sowohl bei Zahlen als auch bei Zeichenketten. Zeichenketten werden bei Bedarf automatisch in Zahlen und Zahlen in Zeichenketten umgewandelt (wie in Perl oder PHP).

MySQL führt Vergleiche nach folgenden Regeln durch:

- Wenn ein oder beide Argumente `NULL` sind, ist das Ergebnis des Vergleichs `NULL`, ausser beim `<=>` Operator.
- Wenn beide Argumente in einer Vergleichsoperation Zeichenketten sind, werden sie als Zeichenketten verglichen.
- Wenn beide Argumente Ganzzahlen sind, werden sie als Ganzzahlen verglichen.
- Hexadezimale Werte werden als binäre Zeichenketten behandelt, wenn sie nicht mit einer Zahl verglichen werden.
- Wenn eins der Argumente eine `TIMESTAMP`- oder `DATETIME`-Spalte ist und das andere Argument eine Konstante, wird die Konstante in einen Zeitstempel umgewandelt, bevor der Vergleich durchgeführt wird. Das wird gemacht, um ODBC-freundlicher zu sein.
- In allen anderen Fällen werden die Argumente als Fließkommazahlen verglichen.

Vorgabemäßig werden Zeichenketten-Vergleiche unabhängig von der verwendeten Groß-/Kleinschreibung durchgeführt, indem der aktuelle Zeichensatz benutzt wird (vorgabemäßig ISO-8859-1 Latin1, der auch für englisch exzellent funktioniert).

Die unten stehenden Beispiele erläutern die Umwandlung von Zeichenketten in Zahlen für Vergleichsoperationen:

```
mysql> SELECT 1 > '6x';
        -> 0
mysql> SELECT 7 > '6x';
        -> 1
mysql> SELECT 0 > 'x6';
        -> 0
mysql> SELECT 0 = 'x6';
        -> 1
```

- `=`

  Gleich:

  ```
  mysql> select 1 = 0;
          -> 0
  mysql> select '0' = 0;
          -> 1
  mysql> select '0.0' = 0;
          -> 1
  mysql> select '0.01' = 0;
          -> 0
  mysql> select '.01' = 0.01;
          -> 1
  ```

- `<>`, `!=`

  Ungleich:

  ```
  mysql> select '.01' <> '0.01';
          -> 1
  mysql> select .01 <> '0.01';
          -> 0
  ```

```
mysql> select 'zapp' <> 'zappp';
        -> 1
```

- `<=`

  Kleiner oder gleich:

  ```
  mysql> select 0.1 <= 2;
          -> 1
  ```

- `<`

  Kleiner als:

  ```
  mysql> select 2 < 2;
          -> 0
  ```

- `>=`

  Größer oder gleich:

  ```
  mysql> select 2 >= 2;
          -> 1
  ```

- `>`

  Größer als:

  ```
  mysql> select 2 > 2;
          -> 0
  ```

- `<=>`

  Null-sicheres gleich:

  ```
  mysql> select 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
          -> 1 1 0
  ```

- `IS NULL`, `IS NOT NULL`

  Testet, ob eine Wert `NULL` ist oder nicht:

  ```
  mysql> select 1 IS NULL, 0 IS NULL, NULL IS NULL;
          -> 0 0 1
  mysql> select 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
          -> 1 1 0
  ```

- `ausdruck BETWEEN min AND max`

  Wenn `ausdruck` größer oder gleich `min` ist und `ausdruck` kleiner oder gleich `max` ist, gibt `BETWEEN` `1` zurück, andernfalls `0`. Das ist äquivalent zum Ausdruck `(min <= ausdruck AND ausdruck <= max)`, wenn alle Argumente vom selben Typ sind. Das erste Argument (`ausdruck`) legt fest, wie der Vergleich durchgeführt wird:

  - Wenn `ausdruck` eine `TIMESTAMP`-, `DATE`- oder `DATETIME`-Spalte ist, werden `MIN()` und `MAX()` im selben Format formatiert als wären sie Konstanten.
  - Wenn `ausdruck` ein Zeichenketten-Ausdruck ohne Berücksichtigung der Groß-/Kleinschreibung ist, wird ein Zeichenkettenvergleich ohne Berücksichtigung der Groß-/Kleinschreibung durchgeführt.
  - Wenn `ausdruck` ein Zeichenketten-Ausdruck mit Berücksichtigung der Groß-/Kleinschreibung ist, wird ein Zeichenkettenvergleich mit Berücksichtigung der Groß-/Kleinschreibung durchgeführt.
  - Wenn `ausdruck` ist ein Ganzzahl-Ausdruck ist, wird ein Ganzzahlvergleich durchgeführt.

- Ansonsten wird ein Fließkommazahlenvergleich durchgeführt.

```
mysql> select 1 BETWEEN 2 AND 3;
        -> 0
mysql> select 'b' BETWEEN 'a' AND 'c';
        -> 1
mysql> select 2 BETWEEN 2 AND '3';
        -> 1
mysql> select 2 BETWEEN 2 AND 'x-3';
        -> 0
```

- `ausdruck IN (wert,...)`

  Gibt `1` zurück, wenn `ausdruck` einen Wert hat, der in der `IN`-Liste enthalten ist, ansonsten `0`. Wenn alle Werte Konstanten sind, werden alle Werte gemäß dem Typ von `ausdruck` ausgewertet und sortiert. Danach wird ein Element mittels binärer Suche gesucht. Das heißt, dass `IN` sehr schnell ist, wenn die `IN`-Werteliste ausschließlich aus Konstanten besteht. Wenn `ausdruck` ein Zeichenketten-Ausdruck mit Berücksichtigung der Groß-/Kleinschreibung ist, wird der Zeichenkettenvergleich unter Berücksichtigung der Groß-/Kleinschreibung durchgeführt:

```
mysql> select 2 IN (0,3,5,'wefwf');
        -> 0
mysql> select 'wefwf' IN (0,3,5,'wefwf');
        -> 1
```

- `ausdruck NOT IN (wert,...)`

  Dasselbe wie `NOT (ausdruck IN (wert,...))`.

- `ISNULL(ausdruck)`

  Wenn `ausdruck` `NULL` ist, gibt `ISNULL()` `1` zurück, ansonsten `0`:

```
mysql> select ISNULL(1+1);
        -> 0
mysql> select ISNULL(1/0);
        -> 1
```

  Beachten Sie, dass ein Vergleich von `NULL`-Werten mit `=` immer UNWAHR ergibt!

- `COALESCE(liste)`

  Gibt das erste Nicht-`NULL`-Element in der Liste zurück:

```
mysql> select COALESCE(NULL,1);
        -> 1
mysql> select COALESCE(NULL,NULL,NULL);
        -> NULL
```

- `INTERVAL(N,N1,N2,N3,...)`

  Gibt `0` zurück, wenn `N` < `N1`, `1`, wenn `N` < `N2` usw. Alle Argumente werden als Ganzzahlen behandelt. Es ist erforderlich, dass `N1` < `N2` < `N3` < `...` < `Nn` ist, damit diese Funktion korrekt funktioniert. Das liegt daran, dass eine (sehr schnelle) binäre Suche benutzt wird:

```
mysql> select INTERVAL(23, 1, 15, 17, 30, 44, 200);
        -> 3
mysql> select INTERVAL(10, 1, 10, 100, 1000);
        -> 2
mysql> select INTERVAL(22, 23, 30, 44, 200);
        -> 0
```

Wenn Sie eine Zeichenkette, die Groß-/Kleinschreibung nicht berücksichtigt, mit einem der Standard-Operatoren vergleichen (`=`, `<>`..., aber nicht `LIKE`), werden Leerzeichen am Ende ignoriert:

```
mysql> select "a" ="A ";
       -> 1
```

7.3.1.3. Logische Operatoren

Alle logischen Funktionen geben `1` (TRUE), `0` (FALSE) oder `NULL` (unbekannt, was in den meisten Fällen dasselbe wie FALSE ist) zurück:

- `NOT`, `!`

  Logisch NOT. Gibt `1` zurück, wenn das Argument `0` ist, ansonsten `0`. Ausnahme: `NOT NULL` gibt `NULL` zurück:

```
mysql> select NOT 1;
        -> 0
mysql> select NOT NULL;
        -> NULL
mysql> select ! (1+1);
        -> 0
mysql> select ! 1+1;
        -> 1
```

  Das letzte Beispiel gibt `1` zurück, weil der Ausdruck auf dieselbe Art ausgewertet wird wie `(!1)+1`.

- `OR`, `||`

  Logisch OR. Gibt `1` zurück, wenn eins der Argumente nicht `0` und nicht `NULL` ist:

```
mysql> select 1 || 0;
        -> 1
mysql> select 0 || 0;
        -> 0
mysql> select 1 || NULL;
        -> 1
```

- `AND`, `&&`

  Logisch AND. Gibt `0` zurück, wenn eins der Argumente `0` oder `NULL` ist, ansonsten `1`:

```
mysql> select 1 && NULL;
        -> 0
mysql> select 1 && 0;
        -> 0
```

7.3.1.4. Ablaufsteuerungsfunktionen

- `IFNULL(ausdruck1,ausdruck2)`

  Wenn `ausdruck1` nicht `NULL` ist, gibt `IFNULL()` `ausdruck1` zurück, ansonsten `ausdruck2`. `IFNULL()` gibt einen numerischen oder einen Zeichenketten-Wert zurück, je nachdem, in welchem Zusammenhang es benutzt wird:

```
mysql> select IFNULL(1,0);
        -> 1
mysql> select IFNULL(NULL,10);
        -> 10
mysql> select IFNULL(1/0,10);
        -> 10
mysql> select IFNULL(1/0,'ja');
        -> 'ja'
```

- `NULLIF(ausdruck1,ausdruck2)`

  Wenn `ausdruck1 = ausdruck2` wahr ist, gibt die Funktion `NULL` zurück, ansonsten `ausdruck1`. Das ist dasselbe wie `CASE WHEN x = y THEN NULL ELSE x END`:

```
mysql> select NULLIF(1,1);
        -> NULL
mysql> select NULLIF(1,2);
        -> 1
```

  Beachten Sie, dass `ausdruck1` in MySQL zweimal ausgewertet wird, wenn die Argumente gleich sind.

- `IF(ausdruck1,ausdruck2,ausdruck3)`

  Wenn `ausdruck1` TRUE ist (`ausdruck1 <> 0` und `ausdruck1 <> NULL`), gibt `IF()` `ausdruck2` zurück,

ansonsten `ausdruck3`. `IF()` gibt einen numerischen oder einen Zeichenketten-Wert zurück, je nachdem, in welchem Zusammenhang es benutzt wird:

```
mysql> select IF(1>2,2,3);
        -> 3
mysql> select IF(1<2,'ja','nein');
        -> 'ja'
mysql> select IF(strcmp('test','test1'),'nein','ja');
        -> 'nein'
```

`ausdruck1` wird als Ganzzahlwert ausgewertet, woraus folgt, dass Sie das Testen auf Fließkomma- oder Zeichenketten-Werte mit einer Vergleichsoperation durchführen sollten:

```
mysql> select IF(0.1,1,0);
        -> 0
mysql> select IF(0.1<>0,1,0);
        -> 1
```

Im ersten Fall gibt `IF(0.1)` `0` zurück, weil `0.1` in einen Ganzzahlwert umgewandelt wird, wodurch es auf `IF(0)` getestet wird. Das ist vielleicht nicht das, was Sie erwarten. Im zweiten Fall testet der Vergleich den Original-Fließkommawert, um zu sehen, ob er nicht 0 ist. Das Ergebnis des Vergleichs wird als Ganzzahl benutzt.

Der vorgabemäßige Rückgabewert von `IF()` (der eine Rolle spielen kann, wenn er in einer temporären Tabelle gespeichert wird), wird in MySQL-Version 3.23 wie folgt berechnet:

Ausdruck	**Rückgabewert**
ausdruck2 oder ausdruck3 gibt Zeichenkette zurück	Zeichenkette
ausdruck2 oder ausdruck3 gibt Fließkommawert zurück	Fließkommawert
ausdruck2 oder ausdruck3 gibt Ganzzahl zurück	Ganzzahl

- `CASE wert WHEN [vergleichs-wert] THEN ergebnis [WHEN [vergleichs-wert] THEN ergebnis ...] [ELSE ergebnis] END`, `CASE WHEN [bedingung] THEN ergebnis [WHEN [bedingung] THEN ergebnis ...] [ELSE ergebnis] END`

  Die erste Version gibt `ergebnis` zurück, wo `wert=vergleichs-wert`. Die zweite Version gibt das Ergebnis für die erste Bedingung zurück, die WAHR ist. Wenn es keinen übereinstimmenden Ergebniswert gab, wird das Ergebnis nach `ELSE` zurückgegeben. Wenn es keinen `ELSE`-Teil gibt, wird `NULL` zurückgegeben:

```
mysql> SELECT CASE 1 WHEN 1 THEN "eins" WHEN 2 THEN "zwei" ELSE "mehr" END;
        -> "eins"
mysql> SELECT CASE WHEN 1>0 THEN "wahr" ELSE "unwahr" END;
        -> "wahr"
mysql> SELECT CASE BINARY "B" when "a" then 1 when "b" then 2 END;
        -> NULL
```

Der Typ des Rückgabewerts (`INTEGER`, `DOUBLE` oder `STRING`) ist derselbe wie der Typ des ersten zurückgegebenen Werts (der Ausdruck nach dem ersten `THEN`).

7.3.2. Zeichenketten-Funktionen

Funktionen für Zeichenkettenwerte geben `NULL` zurück, wenn die Länge des Ergebnisses größer wäre als der `max_allowed_packet`-Serverparameter. See Abschnitt 6.5.2, „Serverparameter tunen“.

Bei Funktionen, die mit Zeichenkettenpositionen arbeiten, wird die erste Position als 1 gezählt.

- `ASCII(zeichenkette)`

  Gibt den ASCII-Code-Wert des äußersten linken Zeichens der Zeichenkette `zeichenkette` zurück. Gibt `0` zurück, wenn `zeichenkette` die leere Zeichenkette ist. Gibt `NULL` zurück, wenn `zeichenkette` `NULL` ist:

```
mysql> select ASCII('2');
        -> 50
mysql> select ASCII(2);
        -> 50
mysql> select ASCII('dx');
        -> 100
```

Siehe auch `ORD()`-Funktion.

- `ORD(zeichenkette)`

  Wenn das äußerste linke Zeichen der Zeichenkette `zeichenkette` ein Multi-Byte-Zeichen ist, gibt diese Funktion den Code des Multi-Byte-Zeichens zurück, indem der ASCII-Code-Wert des Zeichens in folgendem Format zurückgegeben wird: `((erstes byte ASCII code)*256+(zweites byte ASCII code))[*256+drittes byte ASCII code...]`. Wenn das äußerste linke Zeichen kein Multi-Byte-Zeichen ist, wird derselbe Wert wie bei der `ASCII()`-Funktion zurückgegeben:

```
mysql> select ORD('2');
        -> 50
```

- `CONV(N,von_basis,zu_basis)`

  Wandelt Zahlen zwischen verschiedenen Zahlsystemen um. Gibt eine Zeichenkettendarstellung der Zahl `N` zurück, umgewandelt von Basis `von_basis` zu Basis `zu_basis`. Gibt `NULL` zurück, wenn irgend ein Argument `NULL` ist. Das Argument `N` wird als Ganzzahl interpretiert, kann aber als Ganzzahl oder Zeichenkette angegeben werden. Die kleinste Basis ist `2` und die größte Basis `36`. Wenn `zu_basis` eine negative Zahl ist, wird `N` als vorzeichenbehaftete Zahl betrachtet. Ansonsten wird `N` als vorzeichenlos behandelt. `CONV` arbeitet mit 64-Bit-Genauigkeit:

```
mysql> select CONV("a",16,2);
        -> '1010'
mysql> select CONV("6E",18,8);
        -> '172'
mysql> select CONV(-17,10,-18);
        -> '-H'
mysql> select CONV(10+"10"+'10'+0xa,10,10);
        -> '40'
```

- `BIN(N)`

  Gibt eine Zeichenkettendarstellung des Binärwerts von `N` zurück, wobei `N` eine `BIGINT`-Zahl ist. Das ist äquivalent zu `CONV(N,10,2)`. Gibt `NULL` zurück, wenn `N` `NULL` ist:

```
mysql> select BIN(12);
        -> '1100'
```

- `OCT(N)`

  Gibt eine Zeichenkettendarstellung des Oktalwerts von `N` zurück, wobei `N` eine `BIGINT`-Zahl ist. Das ist äquivalent zu `CONV(N,10,8)`. Gibt `NULL` zurück, wenn `N` `NULL` ist:

```
mysql> select OCT(12);
        -> '14'
```

- `HEX(N)`

  Gibt eine Zeichenkettendarstellung des hexadezimalen Werts von `N` zurück, wobei `N` eine `BIGINT`-Zahl ist. Das ist äquivalent zu `CONV(N,10,16)`. Gibt `NULL` zurück, wenn `N` `NULL` ist:

```
mysql> select HEX(255);
        -> 'FF'
```

- `CHAR(N,...)`

  `CHAR()` interpretiert die Argumente als Ganzzahlen und gibt eine Zeichenkette zurück, die aus den Zeichen besteht, die durch die ASCII-Code-Werte dieser Ganzzahlen gegeben sind. `NULL`-Werte werden übersprungen:

```
mysql> select CHAR(77,121,83,81,'76');
        -> 'MySQL'
mysql> select CHAR(77,77.3,'77.3');
        -> 'MMM'
```

- `CONCAT(zeichenkette1,zeichenkette2,...)`

Gibt die Zeichenkette zurück, die durch die Verkettung der Argumente entsteht. Gibt `NULL` zurück, wenn irgend ein Argument `NULL` ist. Kann mehr als 2 Argumente haben. Ein numerisches Argument wird in die äquivalente Zeichenkettenform umgewandelt:

```
mysql> select CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> select CONCAT('My', NULL, 'QL');
        -> NULL
mysql> select CONCAT(14.3);
        -> '14.3'
```

- `CONCAT_WS(trennzeichen, zeichenkette1, zeichenkette2,...)`

  `CONCAT_WS()` steht für CONCAT mit Trennzeichen und ist eine spezielle Form von `CONCAT()`. Das erste Argument ist das Trennzeichen für die restlichen Argumente. Das Trennzeichen kann eine Zeichenkette sein, so wie die übrigen Argumente. Wenn das Trennzeichen `NULL` ist, ist das Ergebnis `NULL`. Die Funktion überspringt jegliche `NULL`s und leere Zeichenketten nach dem Trennzeichen-Argument. Das Trennzeichen wird zwischen den zu verknüpfenden Zeichenketten hinzugefügt:

```
mysql> select CONCAT_WS(",","Vorname","Zweiter Vorname","Nachname");
       -> 'Vorname,Zweiter Vorname,Nachname'
mysql> select CONCAT_WS(",","Vorname",NULL,"Nachname");
       -> 'Vorname,Nachname'
```

- `LENGTH(zeichenkette)`, `OCTET_LENGTH(zeichenkette)`, `CHAR_LENGTH(zeichenkette)`, `CHARACTER_LENGTH(zeichenkette)`

  Gibt die Länge der Zeichenkette `zeichenkette` an:

```
mysql> select LENGTH('text');
        -> 4
mysql> select OCTET_LENGTH('text');
        -> 4
```

  Beachten Sie, dass bei `CHAR_LENGTH()` Multi-Byte-Zeichen nur einmal gezählt werden.

- `LOCATE(teilzeichenfolge,zeichenkette)`, `POSITION(teilzeichenfolge IN zeichenkette)`

  Gibt die Position des ersten Auftretens der Teilzeichenfolge `teilzeichenfolge` in der Zeichenkette `zeichenkette` an. Gibt `0` zurück, wenn `teilzeichenfolge` nicht in `zeichenkette` enthalten ist:

```
mysql> select LOCATE('bar', 'foobarbar');
        -> 4
mysql> select LOCATE('xbar', 'foobar');
        -> 0
```

  Diese Funktion ist Multi-Byte-sicher.

- `LOCATE(teilzeichenfolge,zeichenkette,position)`

  Gibt die Position des ersten Auftretens der Teilzeichenfolge `teilzeichenfolge` in der Zeichenkette `zeichenkette` ab Position `position` an. Gibt `0` zurück, wenn `teilzeichenfolge` nicht in `zeichenkette` enthalten ist:

```
mysql> select LOCATE('bar', 'foobarbar',5);
        -> 7
```

  Diese Funktion ist Multi-Byte-sicher.

- `INSTR(zeichenkette,teilzeichenfolge)`

  Gibt die Position des ersten Auftretens der Teilzeichenfolge `teilzeichenfolge` in der Zeichenkette `zeichenkette` an. Das ist dasselbe wie `LOCATE()` mit zwei Argumenten, ausser dass die Argumente vertauscht sind:

```
mysql> select INSTR('foobarbar', 'bar');
        -> 4
mysql> select INSTR('xbar', 'foobar');
        -> 0
```

Diese Funktion ist Multi-Byte-sicher.

- `LPAD(zeichenkette,laenge,fuellzeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, links aufgefüllt mit der Zeichenkette `fuellzeichenkette`, bis `zeichenkette` `laenge` Zeichen lang ist. Wenn `zeichenkette` länger als `laenge` ist, wird sie auf `laenge` Zeichen verkürzt.

```
mysql> select LPAD('hi',4,'??');
        -> '??hi'
```

- `RPAD(zeichenkette,laenge,fuellzeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, rechts aufgefüllt mit der Zeichenkette `fuellzeichenkette`, bis `zeichenkette` `laenge` Zeichen lang ist. Wenn `zeichenkette` länger als `laenge` ist, wird sie auf `laenge` Zeichen verkürzt.

```
mysql> select RPAD('hi',5,'?');
        -> 'hi???'
```

- `LEFT(zeichenkette,laenge)`

  Gibt die äußersten linken `laenge` Zeichen der Zeichenkette `zeichenkette` zurück:

```
mysql> select LEFT('foobarbar', 5);
        -> 'fooba'
```

  Diese Funktion ist Multi-Byte-sicher.

- `RIGHT(zeichenkette,laenge)`

  Gibt die äußersten rechten `laenge` Zeichen der Zeichenkette `zeichenkette` zurück:

```
mysql> select RIGHT('foobarbar', 4);
        -> 'rbar'
```

  Diese Funktion ist Multi-Byte-sicher.

- `SUBSTRING(zeichenkette,position,laenge)`, `SUBSTRING(zeichenkette FROM position FOR laenge)`, `MID(zeichenkette,position,laenge)`

  Gibt eine `laenge` Zeichen lange Teilzeichenfolge der Zeichenkette `zeichenkette` ab Position `position` zurück. Die abweichende Form, die `FROM` benutzt, ist ANSI-SQL92-Syntax:

```
mysql> select SUBSTRING('Heinzholger',5,6);
        -> 'zholge'
```

  Diese Funktion ist Multi-Byte-sicher.

- `SUBSTRING(zeichenkette,position)`, `SUBSTRING(zeichenkette FROM position)`

  Gibt eine Teilzeichenfolge der Zeichenkette `zeichenkette` ab Position `position` zurück:

```
mysql> select SUBSTRING('Heinzholger',5);
        -> 'zholger'
mysql> select SUBSTRING('foobarbar' FROM 4);
        -> 'barbar'
```

  Diese Funktion ist Multi-Byte-sicher.

- `SUBSTRING_INDEX(zeichenkette,begrenzer,zaehler)`

Gibt die Teilzeichenfolge von Zeichenkette `zeichenkette` vor `zaehler` Vorkommen des Begrenzers `begrenzer` zurück. Wenn `zaehler` positiv ist, wird alle links vom letzten Begrenzer zurückgegeben (von links gezählt). Wenn `zaehler` negativ ist, wird alles rechts vom letzten Begrenzer (von rechts gezählt) zurückgegeben:

```
mysql> select SUBSTRING_INDEX('www.mysql.com', '.', 2);
        -> 'www.mysql'
mysql> select SUBSTRING_INDEX('www.mysql.com', '.', -2);
        -> 'mysql.com'
```

Diese Funktion ist Multi-Byte-sicher.

- `LTRIM(zeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, bei der führende Leerzeichen entfernt wurden:

  ```
  mysql> select LTRIM('  barbar');
          -> 'barbar'
  ```

- `RTRIM(zeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, bei der Leerzeichen am Ende entfernt wurden:

  ```
  mysql> select RTRIM('barbar   ');
          -> 'barbar'
  ```

  Diese Funktion ist Multi-Byte-sicher.

- `TRIM([[BOTH | LEADING | TRAILING] [entfernzeichenkette] FROM] zeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, bei der alle `entfernzeichenkette`-Präfixe und / oder -Suffixe entfernt wurden. Wenn keiner der Spezifizierer `BOTH`, `LEADING` oder `TRAILING` angegeben wird, wird `BOTH` angenommen. Wenn `entfernzeichenkette` nicht angegeben ist, werden Leerzeichen entfernt:

  ```
  mysql> select TRIM('  bar   ');
          -> 'bar'
  mysql> select TRIM(LEADING 'x' FROM 'xxxbarxxx');
          -> 'barxxx'
  mysql> select TRIM(BOTH 'x' FROM 'xxxbarxxx');
          -> 'bar'
  mysql> select TRIM(TRAILING 'xyz' FROM 'barxxyz');
          -> 'barx'
  ```

  Diese Funktion ist Multi-Byte-sicher.

- `SOUNDEX(zeichenkette)`

  Gibt eine Soundex-Zeichenkette von `zeichenkette` zurück. Zwei Zeichenketten, die fast gleich klingen, sollten identische Soundex-Zeichenketten haben. Eine Standard-Soundex-Zeichenkette ist 4 Zeichen lang, aber die `SOUNDEX()`-Funktion gibt eine beliebig lange Zeichenkette zurück. Sie können `SUBSTRING()` auf das Ergebnis anwenden, um eine Standard-Soundex-Zeichenkette zu erhalten. Alle nicht alphanumerischen Zeichen in der angegebenen Zeichenkette werden ignoriert. Alle internationalen alphabetischen Zeichen ausserhalb des Wertebereichs A bis Z werden als Vokale behandelt:

  ```
  mysql> select SOUNDEX('Hello');
          -> 'H400'
  mysql> select SOUNDEX('Quadratically');
          -> 'Q36324'
  ```

- `SPACE(N)`

  Gibt eine Zeichenkette zurück, die aus `N` Leerzeichen besteht:

  ```
  mysql> select SPACE(6);
          -> '      '
  ```

- `REPLACE(zeichenkette,von_zeichenkette,zu_zeichenkette)`

Gibt die Zeichenkette `zeichenkette` zurück, bei der alle Vorkommen der Zeichenkette `von_zeichenkette` durch die Zeichenkette `zu_zeichenkette` ersetzt wurden:

```
mysql> select REPLACE('www.mysql.com', 'w', 'Ww');
        -> 'WwWwWw.mysql.com'
```

Diese Funktion ist Multi-Byte-sicher.

- `REPEAT(zeichenkette,zaehler)`

  Gibt eine Zeichenkette zurück, die aus der Zeichenkette `zeichenkette` besteht, die `zaehler` mal wiederholt wurde. Wenn `zaehler <= 0` ist, wird eine leere Zeichenkette zurückgegeben. Gibt `NULL` zurück, wenn `zeichenkette` oder `zaehler` `NULL` sind:

```
mysql> select REPEAT('MySQL', 3);
        -> 'MySQLMySQLMySQL'
```

- `REVERSE(zeichenkette)`

  Gibt die Zeichenkette `zeichenkette` in umgedrehter Reihenfolge der Zeichen zurück:

```
mysql> select REVERSE('abc');
        -> 'cba'
```

  Diese Funktion ist Multi-Byte-sicher.

- `INSERT(zeichenkette,position,laenge,neue_zeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, wobei eine Teilzeichenfolge ab Position `position` mit `laenge` Zeichen Länge durch die Zeichenkette `neue_zeichenkette` ersetzt wurde:

```
mysql> select INSERT('Heinzholger', 6, 4, 'DIET');
        -> 'HeinzDIETer'
```

  Diese Funktion ist Multi-Byte-sicher.

- `ELT(N,zeichenkette1,zeichenkette2,zeichenkette3,...)`

  Gibt `zeichenkette1` zurück, wenn `N` = `1` ist, `zeichenkette2`, wenn `N` = `2` ist usw.. Gibt `NULL` zurück, wenn `N` kleiner als `1` oder größer als die Anzahl von Argumenten ist. `ELT()` ist das Komplement von `FIELD()`:

```
mysql> select ELT(1, 'ej', 'Heja', 'hej', 'foo');
        -> 'ej'
mysql> select ELT(4, 'ej', 'Heja', 'hej', 'foo');
        -> 'foo'
```

- `FIELD(zeichenkette,zeichenkette1,zeichenkette2,zeichenkette3,...)`

  Gibt den Index von `zeichenkette` in der Liste `zeichenkette1`, `zeichenkette2`, `zeichenkette3`, `...` zurück. Gibt `0` zurück, wenn `zeichenkette` nicht gefunden wird. `FIELD()` ist das Komplement von `ELT()`:

```
mysql> select FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 2
mysql> select FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 0
```

- `FIND_IN_SET(zeichenkette,zeichenkettenliste)`

  Gibt einen Wert `1` bis `N` zurück, wenn die Zeichenkette `zeichenkette` in der Liste `zeichenkettenliste` ist, die aus `N` Teilzeichenfolgen besteht. Eine Zeichenkettenliste ist eine Zeichenkette, die aus Teilzeichenfolgen zusammen gesetzt ist, die durch ‘`,`’-Zeichen getrennt sind. Wenn das erste Argument eine Zeichenketten-Konstante ist und das zweite eine Spalte des Typs `SET`, wird die `FIND_IN_SET()`-Funktion optimiert, Bit-Arithmetik zu benutzen! Gibt `0` zurück, wenn `zeichenkette` nicht in `zeichenkettenliste` ist oder wenn `zeichenkettenliste` die leere Zeichenkette ist. Gibt `NULL` zurück, wenn eines oder beide Argumente `NULL` sind. Diese Funktion funktioniert nicht korrekt, wenn das erste Argument ein ‘`,`’ enthält:

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
        -> 2
```

- `MAKE_SET(bits,zeichenkette1,zeichenkette2,...)`

  Gibt einen Satz (eine Zeichenkette, die Teilzeichenfolgen enthält, die durch ‘,’ getrennt sind) zurück, der aus Zeichenketten besteht, die das entsprechende Bit in `bits` gesetzt haben. `zeichenkette1` entspricht Bit 0, `zeichenkette2` Bit 1 usw. `NULL`-Zeichenketten in `zeichenkette1`, `zeichenkette2` usw. werden nicht an das Ergebnis angehängt:

```
mysql> SELECT MAKE_SET(1,'a','b','c');
        -> 'a'
mysql> SELECT MAKE_SET(1 | 4,'hallo','liebe','welt');
        -> 'hallo,welt'
mysql> SELECT MAKE_SET(0,'a','b','c');
        -> ''
```

- `EXPORT_SET(bits,an,aus,[trennzeichen,[anzahl_bits]])`

  Gibt eine Zeichenkette zurück, in der Sie für jedes bit, das in 'bit' gesetzt ist, eine 'an'-Zeichenkette erhalten, und für jedes zurückgesetzte Bit eine 'aus'-Zeichenkette. Jede Zeichenkette wird mit 'trennzeichen' getrennt (vorgabemäßig ','), und nur die 'anzahl_bits' (vorgabemäßig 64) von 'bits' wird benutzt:

```
mysql> select EXPORT_SET(5,'Y','N',',',4)
        -> Y,N,Y,N
```

- `LCASE(zeichenkette)`, `LOWER(zeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, bei der alle Zeichen in Kleinschreibung gemäß dem aktuellen Zeichensatz-Mapping (Vorgabe ist ISO-8859-1 Latin1) umgewandelt wurden:

```
mysql> select LCASE('HEINZholger');
        -> 'heinzholger'
```

  Diese Funktion ist Multi-Byte-sicher.

- `UCASE(zeichenkette)`, `UPPER(zeichenkette)`

  Gibt die Zeichenkette `zeichenkette` zurück, bei der alle Zeichen in Großschreibung gemäß dem aktuellen Zeichensatz-Mapping (Vorgabe ist ISO-8859-1 Latin1) umgewandelt wurden:

```
mysql> select UCASE('Hej');
        -> 'HEJ'
```

  Diese Funktion ist Multi-Byte-sicher.

- `LOAD_FILE(datei)`

  Liest die Datei `datei` und gibt den Dateiinhalt als Zeichenkette zurück. Die Datei muss auf dem Server sein, Sie müssen den vollen Pfadnamen zur Datei angeben und Sie müssen die **file**-Berechtigung besitzen. Die Datei muss von allen lesbar sein und kleiner als `max_allowed_packet`.

  Wenn die Datei nicht existiert oder aus den oben genannten Gründen nicht gelesen werden kann, gibt die Funktion `NULL` zurück:

```
mysql> UPDATE tabelle
          SET blob_spalte=LOAD_FILE("/tmp/bild")
          WHERE id=1;
```

Wenn Sie nicht MySQL-Version 3.23 benutzen, müssen Sie das Lesen der Datei innerhalb Ihrer Applikation durchführen und ein `INSERT`-Statement erzeugen, um die Datenbank mit der Dateiinformation zu aktualisieren. Eine Art, das zu tun, finden Sie - wenn Sie die MySQL++-Bibliothek benutzen - unter http://www.mysql.com/documentation/mysql++/mysql++-examples.html.

MySQL konvertiert Zahlen bei Bedarf automatisch in Zeichenketten, und umgekehrt:

```
mysql> SELECT 1+"1";
       -> 2
mysql> SELECT CONCAT(2,' test');
       -> '2 test'
```

Wenn Sie eine Zahl explizit in eine Zeichenkette umwandeln wollen, übergeben Sie sie als Argument an `CONCAT()`.

Wenn in einer Zeichenketten-Funktion eine binäre Zeichenkette als Argument angegeben wird, ist die resultierende Zeichenkette ebenfalls eine binäre Zeichenkette. Eine Zahl, die in eine Zeichenkette umgewandelt wird, wird als binäre Zeichenkette behandelt. Das betrifft nur Vergleichsoperationen.

7.3.2.1. Zeichenketten-Vergleichsfunktionen

Normalerweise wird ein Vergleich unter Berücksichtigung der Groß-/Kleinschreibung durchgeführt, wenn irgend ein Ausdruck in einem Zeichenkettenvergleich abhängig von der verwendeten Groß-/Kleinschreibung ist.

- `ausdruck LIKE muster [ESCAPE 'fluchtzeichen']`

  Mustervergleich, der den einfachen SQL-Vergleich mit regulären Ausdrücken benutzt. Gibt `1` (TRUE) oder `0` (FALSE) zurück. Bei `LIKE` können Sie die folgenden zwei Platzhalterzeichen im Muster benutzen:

`%`	Entspricht einer beliebigen Anzahl von Zeichen, selbst 0 Zeichen
`_`	Entspricht genau einem Zeichen

```
mysql> select 'David!' LIKE 'David_';
       -> 1
mysql> select 'David!' LIKE '%D%v%';
       -> 1
```

Um auf literale Instanzen des Platzhalterzeichens zu testen, stellen Sie dem Zeichen ein Fluchtzeichen (Escape-Zeichen) voran. Wenn Sie das `ESCAPE`-Zeichen nicht angeben, wird ‘`\`’ angenommen:

`\%`	Entspricht einem `%`-Zeichen
`\_`	Entspricht einem `_`-Zeichen

```
mysql> select 'David!' LIKE 'David\_';
       -> 0
mysql> select 'David_' LIKE 'David\_';
       -> 1
```

Um ein anderes Fluchtzeichen (Escape-Zeichen) anzugeben, benutzen Sie die `ESCAPE`-Klausel:

```
mysql> select 'David_' LIKE 'David|_' ESCAPE '|';
       -> 1
```

Die folgenden beiden Statements zeigen, dass Zeichenketten-Vergleiche die Groß-/Kleinschreibung nicht berücksichtigen, solange nicht einer der Operanden eine binäre Zeichenkette ist: case insensitive unless one of the operands ist a binäre Zeichenkette:

```
mysql> select 'abc' LIKE 'ABC';
       -> 1
mysql> SELECT 'abc' LIKE BINARY 'ABC';
       -> 0
```

`LIKE` ist bei numerischen Ausdrücken zulässig! (Das ist eine MySQL-Erweiterung zum ANSI-SQL-`LIKE`.)

```
mysql> select 10 LIKE '1%';
       -> 1
```

HINWEIS: Weil MySQL die C Escape-Syntax in Zeichenketten benutzt (beispielsweise ‘`\n`’), müssen Sie jedes ‘`\`’-Zeichen,

das Sie in `LIKE`-Zeichenketten benutzen, verdoppeln. Um zum Beispiel nach '`\n`' zu suchen, geben Sie '`\\n`' ein. Um nach '`\`' zu suchen, geben Sie '`\\\\`' ein (die Backslashes werden einmal vom Parser entfernt und noch einmal, wenn der Mustervergleich durchgeführt wird, so dass letztlich ein einzelner Backslash übrig bleibt).

- `ausdruck NOT LIKE muster [ESCAPE 'fluchtzeichen']`

  Dasselbe wie `NOT (ausdruck LIKE muster [ESCAPE 'fluchtzeichen'])`.

- `ausdruck REGEXP muster`, `ausdruck RLIKE muster`

  Führt einen Mustervergleich eines Zeichenkettenausdrucks `ausdruck` gegen ein Muster `muster` durch. Das Muster kann ein erweiterter regulärer Ausdruck sein. See Anhang G, *Beschreibung der MySQL-Syntax für reguläre Ausdrücke*. Gibt `1` zurück, wenn `ausdruck` mit `muster` übereinstimmt, ansonsten `0`. `RLIKE` ist ein Synonym für `REGEXP`, was aus Gründen der `mSQL`-Kompatibilität zur Verfügung steht. HINWEIS: Weil MySQL die C-Escape-Syntax in Zeichenketten benutzt (beispielsweise '`\n`'), müssen Sie jeden '`\`', den Sie in Ihren `REGEXP`-Zeichenketten benutzen, verdoppeln. Ab MySQL-Version 3.23.4 berücksichtigt `REGEXP` nicht die verwendete Groß-/Kleinschreibung für normale (nicht binäre) Zeichenketten:

```
mysql> select 'Monty!' REGEXP 'm%y%%';
        -> 0
mysql> select 'Monty!' REGEXP '.*';
        -> 1
mysql> select 'new*\n*line' REGEXP 'new\\*.\\*line';
        -> 1
mysql> select "a" REGEXP "A", "a" REGEXP BINARY "A";
        -> 1  0
mysql> select "a" REGEXP "^[a-d]";
        -> 1
```

- `REGEXP` und `RLIKE` benutzen den aktuellen Zeichensatz (vorgabemäßig ISO-8859-1 Latin1), wenn über den Typ eines Zeichens entschieden wird.

- `ausdruck NOT REGEXP muster`, `ausdruck NOT RLIKE muster`

  Dasselbe wie `NOT (ausdruck REGEXP muster)`.

- `STRCMP(ausdruck1,ausdruck2)`

  `STRCMP()` gibt `0` zurück, wenn die Zeichenketten gleich sind, `-1`, wenn das erste Argument kleiner als das zweite ist (nach der aktuellen Sortierreihenfolge), und ansonsten `1`:

```
mysql> select STRCMP('text', 'text2');
        -> -1
mysql> select STRCMP('text2', 'text');
        -> 1
mysql> select STRCMP('text', 'text');
        -> 0
```

- `MATCH (spalte1,spalte2,...) AGAINST (ausdruck)`

  `MATCH ... AGAINST()` wird für Volltextsuche benutzt und gibt die Relevanz zurück - ein Ähnlichkeitsmaß zwischen dem Text in den Spalten `(spalte1,spalte2,...)` und der Anfrage `ausdruck`. Die Relevanz ist eine positive Fließkommazahl. 0 Relevanz bedeutet keine Ähnlichkeit. Damit `MATCH ... AGAINST()` funktioniert, muss zuerst ein **FULLTEXT**-Index erzeugt werden. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“. `MATCH ... AGAINST()` ist verfügbar ab MySQL-Version 3.23.23. Für Details und Benutzungsbeispiele siehe see Abschnitt 7.8, „MySQL-Volltextsuche“.

7.3.2.2. Groß-/Kleinschreibung

- `BINARY`

  Der `BINARY`-Operator macht die folgende Zeichenkette zu einer binären Zeichenkette. Das ist eine einfache Möglichkeit, einen Spaltenvergleich zwangsweise in Abhängigkeit von der verwendeten Groß-/Kleinschreibung durchzuführen, selbst wenn die Spalte nicht als `BINARY` oder `BLOB` definiert ist:

```
mysql> select "a" = "A";
        -> 1
mysql> select BINARY "a" = "A";
        -> 0
```

  `BINARY` wurde in MySQL-Version 3.23.0 eingeführt.

Beachten Sie, dass MySQL in manchen Fällen nicht in der Lage ist, den Index effizient zu benutzen, wenn Sie eine indizierte Spalte zu `BINARY` machen.

Wenn Sie ein Blob ohne Berücksichtigung der Groß-/Kleinschreibung vergleichen wollen, können Sie den Blob jederzeit in Großschreibung umwandeln, bevor Sie den Vergleich durchführen:

```
SELECT 'A' LIKE UPPER(blob_spalte) FROM tabelle;
```

Wir planen, bald Casting zwischen unterschiedlichen Zeichensätzen einzuführen, um Zeichenketten-Vergleiche noch flexibler zu machen.

7.3.3. Numerische Funktionen

7.3.3.1. Arithmetische Operationen

Es gibt die üblichen arithmetischen Operatoren. Beachten Sie, dass das Ergebnis im Falle von '`-`', '`+`' und '`*`' mit `BIGINT`-Genauigkeit (64-Bit) berechnet wird, wenn beide Argumente Ganzzahlen sind!

- `+`

  Addition:

  ```
  mysql> select 3+5;
          -> 8
  ```

- `-`

  Subtraktion:

  ```
  mysql> select 3-5;
          -> -2
  ```

- `*`

  Multiplication:

  ```
  mysql> select 3*5;
          -> 15
  mysql> select 18014398509481984*18014398509481984.0;
          -> 324518553658426726783156020576256.0
  mysql> select 18014398509481984*18014398509481984;
          -> 0
  ```

  Das Ergebnis des letzten Ausdrucks ist falsch, weil die Ganzzahl-Multiplikation den 64-Bit-Wertebereich von `BIGINT`-Berechnungen überschreitet.

- `/`

  Division:

  ```
  mysql> select 3/5;
          -> 0.60
  ```

  Division durch 0 erzeugt ein `NULL`-Ergebnis:

  ```
  mysql> select 102/(1-1);
          -> NULL
  ```

  Eine Division wird nur dann mit `BIGINT`-Arithmetik berechnet, wenn sie in einem Zusammenhang durchgeführt wird, in dem das Ergebnis in eine Ganzzahl umgewandelt wird!

7.3.3.2. Mathematische Funktionen

Alle mathematischen Funktionen geben im Fehlerfall `NULL` zurück.

- `-`

  Unäres Minus. Ändert das Vorzeichen des Arguments:

```
mysql> select - 2;
        -> -2
```

  Wenn dieser Operator mit einer `BIGINT` benutzt wird, beachten Sie, dass der Rückgabewert eine `BIGINT` ist! Das bedeutet, dass Sie `-` auf Ganzzahlen, die den Wert `-2^63` haben könnten, vermeiden sollten!

- `ABS(X)`

  Gibt den absoluten Wert von `X` zurück:

```
mysql> select ABS(2);
        -> 2
mysql> select ABS(-32);
        -> 32
```

  Diese Funktion kann bei `BIGINT`-Werten sicher benutzt werden.

- `SIGN(X)`

  Gibt das Vorzeichen des Arguments als `-1`, `0` oder `1` zurück, abhängig davon, ob `X` negativ, 0 oder positiv ist:

```
mysql> select SIGN(-32);
        -> -1
mysql> select SIGN(0);
        -> 0
mysql> select SIGN(234);
        -> 1
```

- `MOD(N,M)`
- `%`

  Modulo (wie der `%`-Operator in C). Gibt den Rest von `N` dividiert durch `M` zurück:

```
mysql> select MOD(234, 10);
        -> 4
mysql> select 253% 7;
        -> 1
mysql> select MOD(29,9);
        -> 2
```

  Diese Funktion kann bei `BIGINT`-Werten sicher benutzt werden.

- `FLOOR(X)`

  Gibt den größten Ganzzahl-Wert zurück, der nicht größer als `X` ist:

```
mysql> select FLOOR(1.23);
        -> 1
mysql> select FLOOR(-1.23);
        -> -2
```

  Beachten Sie, dass der Rückgabewert in eine `BIGINT` umgewandelt wird!

- `CEILING(X)`

  Gibt den kleinsten Ganzzahl-Wert zurück, der nicht kleiner als `X` ist:

```
mysql> select CEILING(1.23);
        -> 2
mysql> select CEILING(-1.23);
        -> -1
```

  Beachten Sie, dass der Rückgabewert in eine `BIGINT` umgewandelt wird!

- `ROUND(X)`

  Gibt das Argument `X` zurück, gerundet auf die nächste Ganzzahl:

```
mysql> select ROUND(-1.23);
        -> -1
mysql> select ROUND(-1.58);
        -> -2
mysql> select ROUND(1.58);
        -> 2
```

  Beachten Sie, dass das Verhalten von `ROUND()` abhängig von der C-Bibliothek-Implementation ist, wenn das Argument in der Mitte zwischen zwei Ganzzahlen liegt. Einige runden auf die nächste gerade Zahl, oder immer nach oben, immer nach unten oder immer Richtung 0. Wenn Sie eine bestimmte Art zu runden brauchen, sollten Sie statt dessen wohldefinierte Funktionen wie `TRUNCATE()` oder `FLOOR()` benutzen.

- `ROUND(X,D)`

  Gibt das Argument `X` zurück, gerundet auf eine Zahl mit `D` Dezimalstellen. Wenn `D` `0` ist, hat das Ergebnis keinen Dezimalpunkt oder Bruchteil:

```
mysql> select ROUND(1.298, 1);
        -> 1.3
mysql> select ROUND(1.298, 0);
        -> 1
```

- `EXP(X)`

  Gibt den Wert `e` (die Basis des natürlichen Logarithmus) hoch `X` zurück:

```
mysql> select EXP(2);
        -> 7.389056
mysql> select EXP(-2);
        -> 0.135335
```

- `LOG(X)`

  Gibt den natürlichen Logarithmus von `X` zurück:

```
mysql> select LOG(2);
        -> 0.693147
mysql> select LOG(-2);
        -> NULL
```

  Wenn Sie den Logarithmus einer Zahl `X` zu einer beliebigen Basis `B` errechnen wollen, benutzen Sie die Formel `LOG(X)/LOG(B)`.

- `LOG10(X)`

  Gibt den Logarithmus zur Basis 10 von `X` zurück:

```
mysql> select LOG10(2);
        -> 0.301030
mysql> select LOG10(100);
        -> 2.000000
mysql> select LOG10(-100);
        -> NULL
```

- `POW(X,Y)`, `POWER(X,Y)`

  Gibt den Wert `X` hoch `Y` zurück:

```
mysql> select POW(2,2);
        -> 4.000000
mysql> select POW(2,-2);
        -> 0.250000
```

- `SQRT(X)`

  Gibt die nicht negative Quadratwurzel von `X` zurück:

```
mysql> select SQRT(4);
        -> 2.000000
mysql> select SQRT(20);
        -> 4.472136
```

- `PI()`

  Gibt den Wert PI zurück. Die vorgabemäßig angezeigte Anzahl von Dezimalstellen ist 5, aber MySQL benutzt intern die volle doppelte Genauigkeit für PI.

```
mysql> select PI();
        -> 3.141593
mysql> SELECT PI()+0.000000000000000000;
        -> 3.141592653589793116
```

- `COS(X)`

  Gibt den Cosinus von `X` zurück, wobei `X` in Radianten angegeben wird:

```
mysql> select COS(PI());
        -> -1.000000
```

- `SIN(X)`

  Gibt den Sinus von `X` zurück, wobei `X` in Radianten angegeben wird:

```
mysql> select SIN(PI());
        -> 0.000000
```

- `TAN(X)`

  Gibt den Tangens von `X` zurück, wobei `X` in Radianten angegeben wird:

```
mysql> select TAN(PI()+1);
        -> 1.557408
```

- `ACOS(X)`

  Gibt den Arcuscosinus von `X` zurück, dass heißt den Wert, dessen Cosinus `X` ist. Gibt `NULL` zurück, wenn `X` nicht im Bereich von `-1` bis `1` liegt:

```
mysql> select ACOS(1);
        -> 0.000000
mysql> select ACOS(1.0001);
        -> NULL
mysql> select ACOS(0);
        -> 1.570796
```

- `ASIN(X)`

  Gibt den Arcussinus von `X` zurück, das heißt den Wert, dessen Sinus `X` ist. Gibt `NULL` zurück, wenn `X` nicht im Bereich von `-1` bis `1` liegt:

```
mysql> select ASIN(0.2);
        -> 0.201358
mysql> select ASIN('foo');
        -> 0.000000
```

- `ATAN(X)`

  Gibt den Arcustangens von `X` zurück, das heißt den Wert, dessen Tangens `X` ist:

```
mysql> select ATAN(2);
        -> 1.107149
mysql> select ATAN(-2);
        -> -1.107149
```

- `ATAN2(Y,X)`

  Gibt den Arcustangens der beiden Variablen `X` und `Y` zurück. Das ähnelt der Berechnung des Arcustangens von `Y / X`, ausser dass die Vorzeichen beider Argumente benutzt werden, um den Quadranten des Ergebnisses zu bestimmen:

```
mysql> select ATAN(-2,2);
        -> -0.785398
mysql> select ATAN(PI(),0);
        -> 1.570796
```

- `COT(X)`

  Gibt den Cotangens von `X` zurück:

```
mysql> select COT(12);
        -> -1.57267341
mysql> select COT(0);
        -> NULL
```

- `RAND()`, `RAND(N)`

  Gibt eine Zufallszahl (Fließkommawert) im Bereich von `0` bis `1.0` zurück. Wenn ein Ganzzahl-Argument `N` angegeben wird, wird es als Ausgangswert benutzt:

```
mysql> select RAND();
        -> 0.5925
mysql> select RAND(20);
        -> 0.1811
mysql> select RAND(20);
        -> 0.1811
mysql> select RAND();
        -> 0.2079
mysql> select RAND();
        -> 0.7888
```

  Sie können eine Spalte mit `RAND()`-Werten nicht in einer `ORDER BY`-Klausel verwenden, weil `ORDER BY` die Spalte mehrfach auswerten würde. In MySQL-Version 3.23 können Sie jedoch folgendes tun: `SELECT * FROM tabelle ORDER BY RAND()`

  Das ist nützlich, um eine Zufallsstichprobe aus `SELECT * FROM tabelle1,tabelle2 WHERE a=b AND c<d ORDER BY RAND() LIMIT 1000` zu erhalten.

  Beachten Sie, dass ein `RAND()` in einer `WHERE`-Klausel jedes Mal von Neuem ausgewertet wird, wenn `WHERE` ausgeführt wird.

- `LEAST(X,Y,...)`

  Mit zwei oder mehr Argumenten gibt die Funktion das kleinste Argument (das mit dem niedrigsten Wert) zurück. Die Argumente werden nach folgenden Regeln verglichen:

  - Wenn der Rückgabewert in einem `INTEGER`-Zusammenhang benutzt wird oder alle Argumente Ganzzahl-Werte sind, werden sie als Ganzzahlen verglichen.
  - Wenn der Rückgabewert in einem `REAL`-Zusammenhang benutzt wird oder alle Argumente Realzahlen sind, werden sie als Realzahlen verglichen.
  - Wenn irgend ein Argument eine von der Groß-/Kleinschreibung abhängige Zeichenkette ist, werden die Argumente als Zeichenketten, die von der Groß-/Kleinschreibung abhängen, verglichen.
  - In sonstigen Fällen werden die Argumente als Zeichenketten verglichen, die nicht von der Groß-/Kleinschreibung abhängen:

```
mysql> select LEAST(2,0);
        -> 0
mysql> select LEAST(34.0,3.0,5.0,767.0);
        -> 3.0
mysql> select LEAST("B","A","C");
        -> "A"
```

  In MySQL-Versionen vor Version 3.22.5 können Sie `MIN()` statt `LEAST` benutzen.

- `GREATEST(X,Y,...)`

  Gibt das größte Argument (das mit dem höchsten Wert) zurück. Die Argumente werden nach denselben Regeln wie bei `LEAST` verglichen:

```
mysql> select GREATEST(2,0);
        -> 2
mysql> select GREATEST(34.0,3.0,5.0,767.0);
        -> 767.0
mysql> select GREATEST("B","A","C");
        -> "C"
```

  In MySQL-Versionen vor Version 3.22.5 können Sie `MAX()` statt `GREATEST` benutzen.

- `DEGREES(X)`

  Gibt das Argument `X` zurück, von Radianten zu Grad umgewandelt:

```
mysql> select DEGREES(PI());
        -> 180.000000
```

- `RADIANS(X)`

  Gibt das Argument `X` zurück, von Grad zu Radianten umgewandelt:

```
mysql> select RADIANS(90);
        -> 1.570796
```

- `TRUNCATE(X,D)`

  Gibt die Zahl `X` zurück, auf `D` Dezimalstellen beschnitten. Wenn `D` `0` ist, hat das Ergebnis keinen Dezimalpunkt oder Bruchteil:

```
mysql> select TRUNCATE(1.223,1);
        -> 1.2
mysql> select TRUNCATE(1.999,1);
        -> 1.9
mysql> select TRUNCATE(1.999,0);
        -> 1
```

  Beachten Sie, dass Dezimalzahlen in Computern normalerweise nicht als exakte Zahlen, sondern als Double-Werte gespeichert werden. Daher können verwirrende Ergebnisse wie im folgenden Beispiel auftreten:

```
mysql> select TRUNCATE(10.28*100,0);
       -> 1027
```

  Das Obige passiert, weil 10.28 tatsächlich als etwas wie 10.2799999999999999 gespeichert wird.

7.3.4. Datums- und Zeit-Funktionen

Eine Beschreibung des Wertebereichs aller Typen und der gültigen Formate für Datums- und Zeitwerte finden Sie unter Abschnitt 7.2.2, „Datums- und Zeit-Typen“.

Hier ist ein Beispiel, das Datums-Funktionen benutzt. Die unten stehende Anfrage wählt alle Datensätze mit einem `datum_spalte`-Wert innerhalb der letzten 30 Tage aus:

```
mysql> SELECT etwas FROM tabelle
          WHERE TO_DAYS(NOW()) - TO_DAYS(datum_spalte) <= 30;
```

- `DAYOFWEEK(datum)`

  Gibt den Wochentag-Index zurück.

  Für `datum` gilt: `1` = Sonntag, `2` = Montag, ... `7` = Samstag). Diese Index-Werte entsprechen dem ODBC-Standard:

```
mysql> select DAYOFWEEK('1998-02-03');
        -> 3
```

- `WEEKDAY(datum)`

  Gibt den Wochentag-Index für `datum` zurück (`0` = Montag, `1` = Dienstag, ... `6` = Sonntag):

```
mysql> select WEEKDAY('1997-10-04 22:23:00');
        -> 5
mysql> select WEEKDAY('1997-11-05');
        -> 2
```

- `DAYOFMONTH(datum)`

  Gibt den Tag des Monats für `datum` im Bereich `1` bis `31` zurück:

```
mysql> select DAYOFMONTH('1998-02-03');
        -> 3
```

- `DAYOFYEAR(datum)`

  Gibt den Tag des Jahres für `datum` im Bereich `1` bis `366` zurück:

```
mysql> select DAYOFYEAR('1998-02-03');
        -> 34
```

- `MONTH(datum)`

  Gibt den Monat für `datum` im Bereich `1` bis `12` zurück:

```
mysql> select MONTH('1998-02-03');
        -> 2
```

- `DAYNAME(datum)`

  Gibt den Namen des Wochentags für `datum` zurück (auf englisch):

```
mysql> select DAYNAME("1998-02-05");
        -> 'Thursday'
```

- `MONTHNAME(datum)`

  Gibt den Namen des Monats für `datum` zurück (auf englisch):

```
mysql> select MONTHNAME("1998-02-05");
        -> 'February'
```

- `QUARTER(datum)`

  Gibt das Quartal des Jahres für `datum` im Bereich `1` bis `4` zurück:

```
mysql> select QUARTER('98-04-01');
        -> 2
```

- `WEEK(datum)`, `WEEK(datum,erste)`

  Mit einem einzelnen Argument gibt diese Funktion die Woche für `datum` im Bereich `0` bis `53` zurück (ja, es kann Anfänge der Woche 53 geben), für Orte, in denen Sonntag der erste Wochentag ist. In der Form mit zwei Argumenten gestattet `WEEK()` es, festzulegen, ob die Woche am Sonntag oder am Montag beginnt. Die Woche beginnt am Sonntag, wenn das zweite Argument `0` ist, und am Montag, wenn das zweite Argument `1` ist:

```
mysql> select WEEK('1998-02-20');
        -> 7
mysql> select WEEK('1998-02-20',0);
        -> 7
mysql> select WEEK('1998-02-20',1);
        -> 8
mysql> select WEEK('1998-12-31',1);
        -> 53
```

- `YEAR(datum)`

  Gibt das Jahr für `datum` im Bereich `1000` bis `9999` zurück:

```
mysql> select YEAR('98-02-03');
        -> 1998
```

- `YEARWEEK(datum)`, `YEARWEEK(datum,erste)`

  Gibt Jahr und Woche für ein Datum zurück. Das zweite Argument funktioniert genau wie das zweite Argument von `WEEK()`.

Beachten Sie, dass das Jahr sich in der ersten und letzten Woche des Jahres vom Jahr im Datums-Argument unterscheiden kann:

```
mysql> select YEARWEEK('1987-01-01');
       -> 198653
```

- `HOUR(zeit)`

  Gibt die Stunde für `zeit` im Bereich `0` bis `23` zurück:

  ```
  mysql> select HOUR('10:05:03');
         -> 10
  ```

- `MINUTE(zeit)`

  Gibt die Minute für `zeit` im Bereich `0` bis `59` zurück:

  ```
  mysql> select MINUTE('98-02-03 10:05:03');
         -> 5
  ```

- `SECOND(zeit)`

  Gibt die Sekunde für `zeit` im Bereich `0` bis `59` zurück:

  ```
  mysql> select SECOND('10:05:03');
         -> 3
  ```

- `PERIOD_ADD(P,N)`

  Zählt `N` Monate zur Periode `P` hinzu (im Format `YYMM` oder `YYYYMM`). Gibt einen Wert im Format `YYYYMM` zurück.

  Beachten Sie, dass das Perioden-Argument `P` *kein* Datums-Wert ist:

  ```
  mysql> select PERIOD_ADD(9801,2);
         -> 199803
  ```

- `PERIOD_DIFF(P1,P2)`

  Gibt die Anzahl von Monaten zwischen den Perioden `P1` und `P2` zurück. `P1` und `P2` sollten im Format `YYMM` oder `YYYYMM` sein.

  Beachten Sie, dass die Perioden-Argumente `P1` und `P2` *keine* Datumswerte sind:

  ```
  mysql> select PERIOD_DIFF(9802,199703);
         -> 11
  ```

- `DATE_ADD(datum,INTERVAL ausdruck typ)`, `DATE_SUB(datum,INTERVAL ausdruck typ)`, `ADDDATE(datum,INTERVAL ausdruck typ)`, `SUBDATE(datum,INTERVAL ausdruck typ)`

  Diese Funktionen führen Datumsberechnungen durch. Sie wurden in MySQL-Version 3.22 eingeführt. `ADDDATE()` und `SUBDATE()` sind Synonyme für `DATE_ADD()` und `DATE_SUB()`.

  In MySQL-Version 3.23 können Sie `+` und `-` anstelle von `DATE_ADD()` und `DATE_SUB()` benutzen, wenn der Ausdruck auf der rechten Seite eine DATE oder DATETIME-Spalte ist (siehe Beispiel).

  `datum` ist ein `DATETIME`- oder `DATE`-Wert, der das Anfangsdatum festlegt. `ausdruck` ist ein Ausdruck, der den Intervallwert festlegt, der zum Anfangsdatum hinzugezählt oder von diesem abgezogen wird. `ausdruck` ist eine Zeichenkette; sie kann mit einem '`-`' für negative Intervalle beginnen. `typ` ist ein Schlüsselwort, das angibt, wie der Ausdruck interpretiert werden soll.

  Die verwandte Funktion `EXTRACT(typ FROM datum)` gibt das 'typ'-Intervall des Datums zurück.

  Folgende Tabelle zeigt, in welchem Zusammenhang die `typ`- und `ausdruck`-Argumente stehen:

`typ` **wert**	**erwartet** `ausdruck` **format**
`SECOND`	`Sekunden`
`MINUTE`	`Minuten`
`HOUR`	`Stunden`
`DAY`	`Tage`
`MONTH`	`Monate`
`YEAR`	`Jahre`
`MINUTE_SECOND`	`"Minuten:Sekunden"`
`HOUR_MINUTE`	`"Stunden:Minuten"`
`DAY_HOUR`	`"Tage Stunden"`
`YEAR_MONTH`	`"Jahre-Monate"`
`HOUR_SECOND`	`"Stunden:Minuten:Sekunden"`
`DAY_MINUTE`	`"Tage Stunden:Minuten"`
`DAY_SECOND`	`"Tage Stunden:Minuten:Sekunden"`

MySQL erlaubt beliebige Satzzeichen-Begrenzer im `ausdruck`-Format. Die in der Tabelle gezeigten Begrenzer sind Vorschläge. Wenn das `datum`-Argument ein `DATE`-Wert ist und Ihre Berechnungen nur `YEAR`, `MONTH` und `DAY`-Anteile beinhalten (also keine Zeit-Anteile), ist das Ergebnis ein `DATE`-Wert. Ansonsten ist das Ergebnis ein `DATETIME`-Wert:

```
mysql> SELECT "1997-12-31 23:59:59" + INTERVAL 1 SECOND;
        -> 1998-01-01 00:00:00
mysql> SELECT INTERVAL 1 DAY + "1997-12-31";
        -> 1998-01-01
mysql> SELECT "1998-01-01" - INTERVAL 1 SECOND;
       -> 1997-12-31 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
                       INTERVAL 1 SECOND);
        -> 1998-01-01 00:00:00
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
                       INTERVAL 1 DAY);
        -> 1998-01-01 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
                       INTERVAL "1:1" MINUTE_SECOND);
        -> 1998-01-01 00:01:00
mysql> SELECT DATE_SUB("1998-01-01 00:00:00",
                       INTERVAL "1 1:1:1" DAY_SECOND);
        -> 1997-12-30 22:58:59
mysql> SELECT DATE_ADD("1998-01-01 00:00:00",
                       INTERVAL "-1 10" DAY_HOUR);
        -> 1997-12-30 14:00:00
mysql> SELECT DATE_SUB("1998-01-02", INTERVAL 31 DAY);
        -> 1997-12-02
```

Wenn Sie einen Intervallwert angeben, der zu kurz ist (nicht alle Intervall-Anteile beinhaltet, die vom `typ`-Schlüsselwort erwartet werden), nimmt MySQL an, dass Sie den äußersten linken Teil des Intervallwerts ausgelassen haben. Wenn Sie beispielsweise einen `typ` `DAY_SECOND` angeben, wird vom Wert von `ausdruck` erwartet, dass dieser Tages-, Stunden-, Minuten- und Sekunden-Anteile enthält. Wenn Sie einen Wert wie `"1:10"` angeben, nimmt MySQL an, dass die Tages- und Stunden-Anteile fehlen und der Wert Minuten und Sekunden darstellt. Mit anderen Worten wird `"1:10" DAY_SECOND` so interpretiert, dass es äquivalent zu `"1:10" MINUTE_SECOND` ist. Das ist analog zur Weise, wie MySQL `TIME`-Werte interpretiert, die eher vergangene Zeit als Tageszeit darstellen.

Beachten Sie, dass ein Datumswert automatisch in einen DATETIME-Wert umgewandelt wird, wenn Sie einen DATE-Wert zu etwas hinzuzählen oder von etwas abziehen, das einen Zeit-Anteil hat:

```
mysql> select date_add("1999-01-01", interval 1 day);
       -> 1999-01-02
mysql> select date_add("1999-01-01", interval 1 hour);
       -> 1999-01-01 01:00:00
```

Wenn Sie wirklich falsche Datumsangaben benutzen, ist das Ergebnis `NULL`. Wenn Sie `MONTH`, `YEAR_MONTH` oder `YEAR` hinzuzählen und das Datumsergebnis einen Tag hat, der größer ist als der höchste Tag für den neuen Monat, wird der Tag auf den höchsten Tag des neuen Monats angepasst:

```
mysql> select DATE_ADD('1998-01-30', Interval 1 month);
        -> 1998-02-28
```

Beachten Sie, dass das Wort `INTERVAL` und das `typ`-Schlüsselwort in den vorstehenden Beispielen nicht von der verwendeten Groß-/Kleinschreibung abhängen.

- `EXTRACT(typ FROM datum)`

  Die `EXTRACT()`-Funktion benutzt dieselbe Art von Intervalltyp-Spezifikatoren wie `DATE_ADD()` oder `DATE_SUB()`, extrahiert aber Anteile aus dem Datum, statt Datumsberechnungen durchzuführen:

```
mysql> SELECT EXTRACT(YEAR FROM "1999-07-02");
       -> 1999
mysql> SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03");
       -> 199907
mysql> SELECT EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03");
       -> 20102
```

- `TO_DAYS(datum)`

  Gibt für ein Datum `datum` eine Tagesanzahl zurück (die Anzahl von Tagen seit dem Jahr 0):

```
mysql> select TO_DAYS(950501);
        -> 728779
mysql> select TO_DAYS('1997-10-07');
        -> 729669
```

  `TO_DAYS()` ist nicht für die Benutzung mit Werten vor der Einführung des Gregorianischen Kalenders (1582) vorgesehen, weil es nicht die Tage berücksichtigt, die verloren gingen, als der Kalender geändert wurde.

- `FROM_DAYS(N)`

  Gibt für eine Tagesanzahl `N` einen `DATE`-Wert zurück:

```
mysql> select FROM_DAYS(729669);
        -> '1997-10-07'
```

  `FROM_DAYS()` ist nicht für die Benutzung mit Werten vor der Einführung des Gregorianischen Kalenders (1582) vorgesehen, weil es nicht die Tage berücksichtigt, die verloren gingen, als der Kalender geändert wurde.

- `DATE_FORMAT(datum,format)`

  Formatiert den `datum`-Wert gemäß der `format`-Zeichenkette. Folgende Spezifikatoren können in der `format`-Zeichenkette benutzt werden:

`%M`	Monatsname auf englisch (`January` bis `December`)
`%W`	Name des Wochentags auf englisch (`Sunday` bis `Saturday`)
`%D`	Tag des Monats mit englischem Suffix (`1st`, `2nd`, `3rd` usw.)
`%Y`	Jahr, numerisch, 4 Ziffern
`%y`	Jahr, numerisch, 2 Ziffern
`%X`	Jahr der Woche, wobei Sonntag der erste Tag der Woche ist, numerisch, 4 Ziffern, benutzt mit '%V'
`%x`	Jahr der Woche, wobei Montag der erste Tag der Woche ist, numerisch, 4 Ziffern, benutzt mit '%v'
`%a`	Abgekürzter Name des Wochentags auf englisch (`Sun`..`Sat`)
`%d`	Tag des Monats, numerisch (`00` bis `31`)
`%e`	Tag des Monats, numerisch (`0` bis `31`)
`%m`	Monat, numerisch (`01` bis `12`)
`%c`	Monat, numerisch (`1` bis `12`)
`%b`	Abgekürzter Monatsname auf englisch (`Jan` bis `Dec`)
`%j`	Tag des Jahrs (`001` bis `366`)
`%H`	Stunde (`00` bis `23`)
`%k`	Stunde (`0` bis `23`)
`%h`	Stunde (`01` bis `12`)
`%I`	Stunde (`01` bis `12`)

%l	Stunde (1 bis 12)
%i	Minuten, numerisch (00 bis 59)
%r	Uhrzeit, 12-Stunden-Format (hh:mm:ss [AP]M)
%T	Uhrzeit, 24-Stunden-Format (hh:mm:ss)
%S	Sekunden (00 bis 59)
%s	Sekunden (00 bis 59)
%p	AM oder PM
%w	Wochentag (0=Sonntag bis 6=Samstag)
%U	Woche (0 bis 53), wobei Sonntag der erste Tag der Woche ist
%u	Woche (0 bis 53), wobei Montag der erste Tag der Woche ist
%V	Woche (1 bis 53), wobei Sonntag der erste Tag der Woche ist. Benutzt mit '%X'
%v	Woche (1 bis 53), wobei Montag der erste Tag der Woche ist. Benutzt mit '%x'
%%	Ein Literal '%'.

Alle anderen Zeichen werden einfach ohne Interpretation ins Ergebnis kopiert:

```
mysql> select DATE_FORMAT('1997-10-04 22:23:00', '%W%M%Y');
        -> 'Saturday October 1997'
mysql> select DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');
        -> '22:23:00'
mysql> select DATE_FORMAT('1997-10-04 22:23:00',
                          '%D%y%a%d%m%b%j');
        -> '4th 97 Sat 04 10 Oct 277'
mysql> select DATE_FORMAT('1997-10-04 22:23:00',
                          '%H%k%I%r%T%S%w');
        -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> select DATE_FORMAT('1999-01-01', '%X%V');
        -> '1998 52'
```

Ab MySQL-Version 3.23 ist das '%'-Zeichen vor Format-Spezifikator-Zeichen erforderlich. In früheren Versionen von MySQL war '%' optional.

- `TIME_FORMAT(zeit,format)`

  Dieses wird benutzt wie die obige `DATE_FORMAT()`-Funktion, aber die `format`-Zeichenkette darf nur die Spezifikatoren enthalten, die Stunden, Minuten und Sekunden handhaben. Andere Spezifikatoren erzeugen einen `NULL`-Wert oder `0`.

- `CURDATE()`, `CURRENT_DATE`

  Gibt das Datum von heute im `'YYYY-MM-DD'`- oder `YYYYMMDD`-format zurück, abhängig davon, ob die Funktion in einem Zeichenketten- oder in einem numerischen Zusammenhang benutzt wird:

```
mysql> select CURDATE();
        -> '1997-12-15'
mysql> select CURDATE() + 0;
        -> 19971215
```

- `CURTIME()`, `CURRENT_TIME`

  Gibt die aktuelle Zeit als einen Wert im `'HH:MM:SS'`- oder `HHMMSS`-format zurück, abhängig davon, ob die Funktion in einem Zeichenketten- oder in einem numerischen Zusammenhang benutzt wird:

```
mysql> select CURTIME();
        -> '23:50:26'
mysql> select CURTIME() + 0;
        -> 235026
```

- `NOW()`, `SYSDATE()`, `CURRENT_TIMESTAMP`

  Gibt das aktuelle Datum und die aktuelle Zeit als einen Wert im `'YYYY-MM-DD HH:MM:SS'`- oder `YYYYMMDDHHMMSS`-Format zurück, abhängig davon, ob die Funktion in einem Zeichenketten- oder in einem numerischen Zusammenhang benutzt

wird:

```
mysql> select NOW();
        -> '1997-12-15 23:50:26'
mysql> select NOW() + 0;
        -> 19971215235026
```

- `UNIX_TIMESTAMP()`, `UNIX_TIMESTAMP(datum)`

  Ohne Argument aufgerufen gibt die Funktion einen Unix-Zeitstempel zurück (Sekunden seit `'1970-01-01 00:00:00'` GMT). Wenn `UNIX_TIMESTAMP()` mit einem `datum`-Argument aufgerufen wird, gibt sie den Wert des Arguments als Sekunden seit `'1970-01-01 00:00:00'` GMT zurück. `datum` kann eine `DATE`-Zeichenkette, eine `DATETIME`-Zeichenkette, ein `TIMESTAMP` oder eine Zahl im Format `YYMMDD` oder `YYYYMMDD` in lokaler Zeit sein:

```
mysql> select UNIX_TIMESTAMP();
        -> 882226357
mysql> select UNIX_TIMESTAMP('1997-10-04 22:23:00');
        -> 875996580
```

  Wenn `UNIX_TIMESTAMP` auf einer `TIMESTAMP`-Spalte benutzt wird, erhält die Funktion den Wert direkt, ohne implizite ``zeichenkette-zu-unix-zeitstempel''-Umwandlung. Wenn Sie `UNIX_TIMESTAMP()` einen falschen Wert oder einen Wert ausserhalb des Wertebereichs angeben, gibt sie 0 zurück.

- `FROM_UNIXTIME(unix_zeitstempel)`

  Gibt das `unix_timestamp`-Argument als Wert im `'YYYY-MM-DD HH:MM:SS'`- oder `YYYYMMDDHHMMSS`-Format zurück, abhängig davon, ob die Funktion in einem Zeichenketten- oder in einem numerischen Zusammenhang benutzt wird:

```
mysql> select FROM_UNIXTIME(875996580);
        -> '1997-10-04 22:23:00'
mysql> select FROM_UNIXTIME(875996580) + 0;
        -> 19971004222300
```

- `FROM_UNIXTIME(unix_zeitstempel,format)`

  Gibt das `unix_timestamp`-Argument als Wert zurück, der wie mit der `format`-Zeichenkette angegeben formatiert ist. `format` kann dieselben Spezifikatoren wie die `DATE_FORMAT()`-Funktion enthalten:

```
mysql> select FROM_UNIXTIME(UNIX_TIMESTAMP(),
                            '%Y%D%M%h:%i:%s%x');
        -> '1997 23rd December 03:43:30 x'
```

- `SEC_TO_TIME(sekunden)`

  Gibt das `sekunden`-Argument, umgewandelt in Stunden, Minuten und Sekunden, als Wert im `'HH:MM:SS'`- oder `HHMMSS`-Format zurück, abhängig davon, ob die Funktion in einem Zeichenketten- oder in einem numerischen Zusammenhang benutzt wird:

```
mysql> select SEC_TO_TIME(2378);
        -> '00:39:38'
mysql> select SEC_TO_TIME(2378) + 0;
        -> 3938
```

- `TIME_TO_SEC(zeit)`

  Gibt das `zeit`-Argument, umgewandelt in Sekunden, zurück:

```
mysql> select TIME_TO_SEC('22:23:00');
        -> 80580
mysql> select TIME_TO_SEC('00:39:38');
        -> 2378
```

7.3.5. Weitere Funktionen

7.3.5.1. Bit-Funktionen

MySQL benutzt `BIGINT`-Berechnungen (64-Bit) für Bit-Operationen, so dass diese Operatoren einen maximalen Wertebereich von 64 Bits haben.

- `|`

  Bitweises OR:

```
mysql> select 29 | 15;
        -> 31
```

- `&`

  Bitweises AND:

```
mysql> select 29 & 15;
        -> 13
```

- `<<`

  Verschiebt eine `BIGINT`-Zahl nach links:

```
mysql> select 1 << 2;
        -> 4
```

- `>>`

  Verschiebt eine `BIGINT`-Zahl nach rechts:

```
mysql> select 4 >> 2;
        -> 1
```

- `~`

  Invertiert alle Bits:

```
mysql> select 5 & ~1;
        -> 4
```

- `BIT_COUNT(N)`

  Gibt die Anzahl von Bits, die im Argument `N` gesetzt sind, zurück:

```
mysql> select BIT_COUNT(29);
        -> 4
```

7.3.5.2. Verschiedene Funktionen

- `DATABASE()`

  Gibt den aktuellen Datenbanknamen zurück:

```
mysql> select DATABASE();
        -> 'test'
```

  Wenn es keine aktuelle Datenbank gibt, gibt `DATABASE()` die leere Zeichenkette zurück.

- `USER()`, `SYSTEM_USER()`, `SESSION_USER()`

  Gibt den aktuellen MySQL-Benutzernamen zurück:

```
mysql> select USER();
        -> 'heinzholger@localhost'
```

Ab MySQL-Version 3.22.11 beinhaltet dieser Wert den Client-Hostnamen sowie den Benutzernamen. Sie können nur den Benutzernamen-Anteil wie folgt extrahieren (was funktioniert, ob der Wert nun einen Hostnamen-Anteil hat oder nicht):

```
mysql> select substring_index(USER(),"@",1);
        -> 'heinzholger'
```

- `PASSWORD(zeichenkette)`

  Berechnet eine Passwort-Zeichenkette aus dem Klartext-Passwort `zeichenkette`. Diese Funktion wird benutzt, um MySQL-Passwörter zum Speichern in der `Password`-Spalte der `user`-Berechtigungstabelle zu verschlüsseln:

  ```
  mysql> select PASSWORD('schlechtespasswort');
          -> '1ccbb34b4e2b2f95'
  ```

  Die `PASSWORD()`-Verschlüsselung ist nicht umkehrbar.

  `PASSWORD()` führt keine Passwort-Verschlüsselung in der Art durch, wie Unix-Passwörter verschlüsselt werden. Sie sollten nicht annehmen, dass Ihr Unix-Passwort und Ihr MySQL-Passwort dasselbe sind. `PASSWORD()` ergibt denselben verschlüsselten Wert, wie er in der Unix-Passwortdatei gespeichert ist. Siehe `ENCRYPT()`.

- `ENCRYPT(zeichenkette[,salt])`

  Verschlüsselt `zeichenkette` unter Benutzung des Unix-`crypt()`-Systemaufrufs. Das `salt`-Argument sollte eine Zeichenkette mit zwei Zeichen sein (ab MySQL-Version 3.22.16 darf `salt` länger als zwei Zeichen sein):

  ```
  mysql> select ENCRYPT("hello");
          -> 'VxuFAJXVARROc'
  ```

  Wenn `crypt()` auf Ihrem System nicht verfügbar ist, gibt `ENCRYPT()` immer `NULL` zurück.

  `ENCRYPT()` ignoriert alle ausser den ersten 8 Zeichen von `zeichenkette`, zumindest auf einigen Systemen. Das wird durch den zugrunde liegenden `crypt()`-Systemaufruf festgelegt.

- `ENCODE(zeichenkette,passwort_zeichenkette)`

  Verschlüsselt `zeichenkette`, indem `passwort_zeichenkette` als Passwort benutzt wird. Um das Ergebnis zu entschlüsseln, benutzen Sie `DECODE()`.

  Das Ergebnis ist eine binäre Zeichenkette derselben Länge wie `zeichenkette`. Wenn Sie sie in einer Spalte speichern wollen, benutzen Sie eine `BLOB`-Spalte.

- `DECODE(crypt_zeichenkette,passwort_zeichenkette)`

  Entschlüsselt die verschlüsselte Zeichenkette `crypt_zeichenkette`, indem `passwort_zeichenkette` als Passwort benutzt wird. `crypt_zeichenkette` sollte eine Zeichenkette sein, die von `ENCODE()` zurückgegeben wird.

- `MD5(zeichenkette)`

  Berechnet eine MD5-Prüfsumme für die Zeichenkette. Der Wert wird als eine 32 Stellen lange hexadezimale Zahl zurückgegeben, die zum Beispiel als Hash-Schlüssel benutzt werden kann:

  ```
  mysql> select MD5("testing");
          -> 'ae2b1fca515949e5d54fb22b8ed95575'
  ```

  Das ist ein "RSA Data Sicherheit, Inc. MD5 Message-Digest Algorithm".

- `LAST_INSERT_ID([ausdruck])`

  Gibt den letzten automatisch erzeugten Wert zurück, der in eine `AUTO_INCREMENT`-Spalte eingefügt wurde. See Abschnitt 9.4.3.30, „`mysql_insert_id()`“.

  ```
  mysql> select LAST_INSERT_ID();
          -> 195
  ```

  Die letzte ID, die erzeugt wurde, wird im Server für jede Verbindung separat gespeichert. Sie wird nicht durch andere Clients

geändert. Sie wird nicht einmal geändert, wenn Sie eine andere `AUTO_INCREMENT`-Spalte mit einem nicht 'magischen' Wert aktualisieren (also einem Wert, der nicht `NULL` und nicht `0` ist).

Wenn Sie viele Zeilen zugleich mit einem Insert-Statement einfügen, gibt `LAST_INSERT_ID()` den Wert für die erste eingefügte Zeile zurück. Der Grund dafür liegt darin, dass es Ihnen dadurch ermöglicht wird, dasselbe `INSERT`-Statement auf einfache Weise auf einem anderen Server zu reproduzieren.

Wenn `ausdruck` als Argument zu `LAST_INSERT_ID()` angegeben wird, wird der Wert des Arguments von der Funktion zurückgegeben, als nächster Wert gesetzt, der von `LAST_INSERT_ID()` zurückgegeben wird und als nächster auto_increment-Wert benutzt. Damit können Sie Zahlenfolgen emulieren:

Erzeugen Sie zuerst die Tabelle:

```
mysql> create table sequenz (id int not null);
mysql> insert into sequenz values (0);
```

Danach kann die Tabelle benutzt werden, um wie folgt Zahlenfolgen zu erzeugen:

```
mysql> update sequenz set id=LAST_INSERT_ID(id+1);
```

Sie können Zahlenfolgen erzeugen, ohne `LAST_INSERT_ID()` aufzurufen, aber der Nutzen, die Funktion auf diese Art zu benutzen, liegt darin, dass der ID-Wert im Server als letzter automatisch erzeugter Wert gehalten wird. Sie können die neue ID auf dieselbe Art abrufen, wie Sie jeden anderen normalen `AUTO_INCREMENT`-Wert in MySQL lesen würden. `LAST_INSERT_ID()` (ohne Argument) zum Beispiel gibt die neue ID zurück. Die C-API-Funktion `mysql_insert_id()` kann ebenfalls benutzt werden, um den Wert zu erhalten.

Beachten Sie, dass Sie diese Funktion nicht benutzen können, um den Wert von `LAST_INSERT_ID(ausdruck)` abzurufen, nachdem Sie andere SQL-Statements wie `SELECT` oder `SET` ausgeführt haben, weil `mysql_insert_id()` nur nach `INSERT`- und `UPDATE`-Statements aktualisiert wird.

- `FORMAT(X,D)`

  Formatiert die Zahl `X` in ein Format wie `'#,###,###.##'`, gerundet auf `D` Dezimalstellen. Wenn `D` `0` ist, hat das Ergebnis keinen Dezimalpunkt oder Bruchteil:

  ```
  mysql> select FORMAT(12332.123456, 4);
         -> '12,332.1235'
  mysql> select FORMAT(12332.1,4);
         -> '12,332.1000'
  mysql> select FORMAT(12332.2,0);
         -> '12,332'
  ```

- `VERSION()`

  Gibt eine Zeichenkette zurück, die die MySQL-Serverversion anzeigt:

  ```
  mysql> select VERSION();
         -> '3.23.13-log'
  ```

  Wenn Ihre Versionsnummer mit `-log` endet, bedeutet das, dass Loggen angeschaltet ist.

- `CONNECTION_ID()`

  Gibt die Verbindungskennnummer (`Thread_id`) für die Verbindung zurück. Jede Verbindung hat ihre eigene eindeutige Kennnummer:

  ```
  mysql> select CONNECTION_ID();
         -> 1
  ```

- `GET_LOCK(zeichenkette,zeitueberschreitung)`

  Versucht, eine Sperre mit dem Namen, der durch die Zeichenkette `zeichenkette` angegeben wird, zu erlangen, mit einem Timeout von `zeitueberschreitung` Sekunden. Gibt `1` zurück, wenn die Sperre erfolgreich erlangt wurde, und `0`, wenn der Versuch wegen Zeitüberschreitung abgebrochen wurde, oder `NULL`, wenn ein Fehler auftrat (wenn zum Beispiel kein

Arbeitsspeicher mehr frei ist oder der Thread mit `mysqladmin kill` gekillt wurde). Eine Sperre wird aufgehoben, wenn Sie `RELEASE_LOCK()` ausführen, einen neuen `GET_LOCK()` ausführen oder der Thread beendet wird. Diese Funktion kann benutzt werden, um Applikations-Sperren zu implementieren oder um Datensatz-Sperren zu simulieren. Sie blockiert Anfragen von anderen Clients nach Sperren mit demselben Namen; Clients, die sich auf einen angegebenen Namen für die Sperr-Zeichenkette einigen, können die Zeichenkette benutzen, um kooperatives beratendes Sperren (advisory locking) auszuführen:

```
mysql> select GET_LOCK("lock1",10);
        -> 1
mysql> select GET_LOCK("lock2",10);
        -> 1
mysql> select RELEASE_LOCK("lock2");
        -> 1
mysql> select RELEASE_LOCK("lock1");
        -> NULL
```

Beachten Sie, dass der zweite `RELEASE_LOCK()`-Aufruf `NULL` zurückgibt, weil die Sperre `"lock1"` automatisch durch den zweiten `GET_LOCK()`-Aufruf aufgehoben wurde.

- `RELEASE_LOCK(zeichenkette)`

  Hebt die Sperre auf, die durch die Zeichenkette `zeichenkette` benannt ist, die mit `GET_LOCK()` erlangt wurde. Gibt `1` zurück, wenn die Sperre aufgehoben wurde, und `0`, wenn die Sperre nicht durch diesen Thread gemacht wurde (in diesem Fall wird die Sperre nicht aufgehoben), oder `NULL`, wenn die benannte Sperre nicht existiert. Die Sperre existiert nicht, wenn sie nie durch einen Aufruf von `GET_LOCK()` erlangt wurde oder wenn sie bereits aufgehoben wurde.

- `BENCHMARK(zaehler,ausdruck)`

  Die `BENCHMARK()`-Funktion den Ausdruck `ausdruck` wiederholt `zaehler` mal aus. Sie kann benutzt werden, um die Zeit zu ermitteln, die MySQL benötigt, um den Ausdruck zu verarbeiten. Der Ergebniswert ist immer `0`. Die Funktion ist für die Benutzung im `mysql`-Client gedacht, der die Ausführungszeiten von Anfragen zum Beispiel wie folgt darstellt:

```
mysql> select BENCHMARK(1000000,encode("hello","goodbye"));
+----------------------------------------------+
| BENCHMARK(1000000,encode("hello","goodbye")) |
+----------------------------------------------+
|                                            0 |
+----------------------------------------------+
1 row in set (4.74 sec)
```

  Die berichtete Zeit ist die am Client-Ende verstrichene Zeit, nicht die Prozessorzeit am Server-Ende. Es ist ratsam, `BENCHMARK()` mehrere Male auszuführen und das Ergebnis unter Berücksichtigung der Last, unter der die Server-Maschine fährt, zu interpretieren.

- `INET_NTOA(ausdruck)`

  Gibt die Netzwerk-Adresse (4 oder 8 Bytes) für den numerischen Ausdruck zurück:

```
mysql> select INET_NTOA(3520061480);
       ->  "209.207.224.40"
```

- `INET_ATON(ausdruck)`

  Gibt eine Ganzzahl zurück, die den numerischen Wert einer Netzwerk-Adresse darstellt. Adressen können 4-Byte- oder 8-Byte-Adressen sein:

```
mysql> select INET_ATON("209.207.224.40");
       ->  3520061480
```

  Die erzeugte Zahl ist immer in Netzwerk-Byte-Reihenfolge; die obige Zahl wird zum Beispiel errechnet als `209*255^3 + 207*255^2 + 224*255 +40`.

- `MASTER_POS_WAIT(log_name, log_position)`

  Blockiert, bis der Slave während der Replikation die festgelegte Position in der Master-Log-Datei erreicht. Wenn die Master-Information nicht initialisiert wird, wird NULL zurückgegeben. Wenn der Slave nicht läuft, blockiert die Funktion und wartet, bis er gestartet wurde, und geht dann hinter die angegebene Position. Wenn der Slave bereits hinter der angegebenen Position ist, kehrt die Funktion sofort zurück. Der Rückgabewert ist die Anzahl von Log-Events, die sie warten muss, um bis zur angegebenen Position zu kommen, oder NULL in Fehlerfällen. Nützlich für die Steuerung der Master-Slave-Synchronisation, aber ursprünglich geschrieben, um das Testen der Replikation zu erleichtern.

7.3.6. Funktionen zur Benutzung bei GROUP BY-Klauseln

Wenn Sie in einem Statement eine Gruppierungsfunktion benutzen, die keine GROUP BY-Klausel enthält, ist das gleichbedeutend mit der Gruppierung aller Zeilen.

- COUNT(ausdruck)

  Gibt die Anzahl der Zeilen mit Nicht-NULL-Werten zurück, die durch ein SELECT-Statement abgerufen werden:

```
mysql> select student.student_name,COUNT(*)
           from student,kurs
           where student.student_id=kurs.student_id
           GROUP BY student_name;
```

  COUNT(*) ist insofern anders, als es die Anzahl der abgerufenen Zeilen zurückgibt, egal ob sie NULL-Werte enthalten oder nicht.

  COUNT(*) ist darauf optimiert, das Ergebnis sehr schnell zurückzugeben, wenn es mittels eines SELECT von einer Tabelle abruft, wenn keine weiteren Spalten abgerufen werden und es keine WHERE-Klausel gibt. Beispiel:

```
mysql> select COUNT(*) from student;
```

- COUNT(DISTINCT ausdruck,[ausdruck...])

  Gibt die Anzahl unterschiedlicher Nich-NULL-Werte zurück:

```
mysql> select COUNT(DISTINCT ergebnisse) from student;
```

  Bei MySQL erhalten Sie die Anzahl unterschiedlicher Ausdruckskombinationen, die nicht NULL enthalten, indem Sie eine Liste von Ausdrücken angeben. In ANSI-SQL müssten Sie eine Verkettung aller Ausdrücke innerhalb von CODE(DISTINCT ..) angeben.

- AVG(ausdruck)

  Gibt den Durchschnittswert von ausdruck zurück:

```
mysql> select student_name, AVG(test_ergebnis)
           from student
           GROUP BY student_name;
```

- MIN(ausdruck), MAX(ausdruck)

  Gibt den kleinsten oder größten Wert von ausdruck zurück. MIN() und MAX() können Zeichenketten-Argumente aufnehmen und geben in solchen Fällen den kleinsten oder größten Zeichenketten- Wert zurück. See Abschnitt 6.4.3, „Wie MySQL Indexe benutzt“.

```
mysql> select student_name, MIN(test_ergebnis), MAX(test_ergebnis)
           from student
           GROUP BY student_name;
```

- SUM(ausdruck)

  Gibt die Summe von ausdruck zurück. Beachten Sie, dass der Rückgabewert NULL ist, wenn die Ergebnismenge keine Zeilen hat!

- STD(ausdruck), STDDEV(ausdruck)

  Gibt die Standardabweichung von ausdruck zurück. Das ist eine Erweiterung zu ANSI-SQL. Die STDDEV()-Form dieser Funktion wird aus Gründen der Oracle-Kompatibilität zur Verfügung gestellt.

- BIT_OR(ausdruck)

Gibt das bitweise `OR` aller Bits in `ausdruck` zurück. Die Berechnung wird mit 64-Bit-(`BIGINT`)-Genauigkeit durchgeführt.

- `BIT_AND(ausdruck)`

  Gibt das bitweise `AND` aller Bits in `ausdruck` zurück. Die Berechnung wird mit 64-Bit-(`BIGINT`)-Genauigkeit durchgeführt.

MySQL hat die Benutzung von `GROUP BY` erweitert. Sie können Spalten oder Berechnungen im `SELECT`-Ausdruck angeben, die nicht im `GROUP BY`-Teil erscheinen. Das steht für *jeden möglichen Wert für diese Gruppe*. Das können Sie benutzen, um bessere Performance zu erzielen, indem Sie Sortieren und Gruppieren unnötiger Bestandteile vermeiden. Zum Beispiel müssen Sie in folgender Anfrage nicht nach `kunde.name` gruppieren:

```
mysql> select bestellung.kunde_id,kunde.name,max(zahlungen)
       from bestellung,kunde
       where bestellung.kunde_id = kunde.kunde_id
       GROUP BY bestellung.kunde_id;
```

In ANSI-SQL müssten Sie der `GROUP BY`-Klausel `kunde.name` hinzufügen. In MySQL ist der Name überflüßig, solange Sie nicht im ANSI-Modus fahren.

Benutzen Sie dieses Feature nicht, wenn die Spalten, die Sie im `GROUP BY`-Teil auslassen, in der Gruppe nicht eindeutig sind! Sonst erhalten Sie unvorhersagbare Ergebnisse.

In einigen Fällen können Sie `MIN()` und `MAX()` benutzen, um einen bestimmten Spaltenwert zu erhalten, selbst wenn er nicht eindeutig ist. Folgendes gibt den Wert von `spalte` aus der Zeile zurück, die den kleinsten Wert in der `sortierung`-Spalte enthält:

```
substr(MIN(concat(rpad(sortierung,6,' '),spalte)),7)
```

7.4. Datenmanipulation: `SELECT`, `INSERT`, `UPDATE`, `DELETE`

7.4.1. `SELECT`-Syntax

```
SELECT [STRAIGHT_JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
       [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
    select_ausdruck,...
    [INTO {OUTFILE | DUMPFILE} 'datei' export_optionen]
    [FROM tabellenreferenz
        [WHERE where_definition]
        [GROUP BY {positive_ganzzahl | spalten_name | formel} [ASC | DESC], ...]
        [HAVING where_definition]
        [ORDER BY {positive_ganzzahl | spalten_name | formel} [ASC | DESC] ,...]
        [LIMIT [offset,] zeilen]
        [PROCEDURE prozedur_name]
        [FOR UPDATE | LOCK IN SHARE MODE]]
```

`SELECT` wird benutzt, um ausgewählte Zeilen aus einer oder mehreren Tabellen abzurufen. `select_ausdruck` gibt die Spalten an, die Sie abrufen wollen. `SELECT` kann auch benutzt werden, um Zeilen ohne Bezug zu irgend einer Tabelle abzurufen. Beispiel:

```
mysql> SELECT 1 + 1;
         -> 2
```

Alle benutzten Schlüsselwörter müssen genau in der oben angegebenen Reihenfolge genannt werden. Beispielsweise muss eine `HAVING`-Klausel nach jeglicher `GROUP BY`-Klausel und vor jeglicher `ORDER BY`-Klausel kommen.

- Einem `SELECT`-Ausdruck kann mit `AS` ein Alias zugewiesen werden. Der Alias wird als Spaltenname verwendet und kann bei `ORDER BY`- oder `HAVING`-Klauseln benutzt werden. Beispiel:

  ```
  mysql> select concat(nachname,', ',vorname) AS voller_name
      from tabelle ORDER BY voller_name;
  ```

- The `FROM tabellenreferenz`-Klausel gibt die Tabellen an, aus denen Zeilen abgerufen werden sollen. Wenn Sie mehr als eine Tabelle aufführen, führen Sie einen Join durch. Informationen über die Join-Syntax finden Sie unter Abschnitt 7.4.1.1, „`JOIN`-Syntax“.

- Sie können auf eine Spalte als `spalten_name` verweisen, als `tabelle.spalten_name` oder als `datenbank.tabelle.spalten_name`. Sie müssen das `tabelle`- oder `datenbank.tabelle`-Präfix für einen Spaltenverweis in einem `SELECT`-Statement nicht angeben, es sei denn, der Verweis wäre ansonsten mehrdeutig. Sie Abschnitt 7.1.2, „Datenbank-, Tabellen-, Index-, Spalten- und Alias-Namen“; hier finden sich Beispiele von Mehrdeutigkeit, die

erfordern, dass Sie ausführlichere Spaltenverweis-Formen benutzen.

- Einem Tabellenverweis kann mit `tabelle [AS] alias_name` ein Tabellen-Alias zugewiesen werden:

```
mysql> select t1.name, t2.gehalt from angestellte AS t1, info AS t2
           where t1.name = t2.name;
mysql> select t1.name, t2.gehalt from angestellte t1, info t2
           where t1.name = t2.name;
```

- Auf Spalten, die für die Ausgabe ausgewählt wurden, kann in `ORDER BY`- und `GROUP BY`-Klauseln mit Spaltennamen, Spalten-Aliasen oder Spaltenpositionen verwiesen werden. Spaltenpositionen fangen mit 1 an:

```
mysql> select hochschule, region, seed von tournament
           ORDER BY region, seed;
mysql> select hochschule, region AS r, seed AS s from turnier
           ORDER BY r, s;
mysql> select hochschule, region, seed from turnier
           ORDER BY 2, 3;
```

  Um in absteigender Reihenfolge zu sortieren, fügen Sie dem Namen der Spalte das `DESC`-Schlüsselwort in the `ORDER BY`-Klausel hinzu (descending, absteigend), nach der Sie sortieren. Die Vorgabe ist aufsteigende Reihenfolge. Das können Sie auch explizit angeben, indem Sie das `ASC`-Schlüsselwort verwenden.

- In der `WHERE`-Klausel können Sie beliebige Funktionen verwenden, die MySQL unterstützt. See Abschnitt 7.3, „Funktionen für die Benutzung in SELECT- und WHERE-Klauseln“.

- Die `HAVING`-Klausel kann auf jede Spalte oder jeden Alias verweisen, die bzw. der im `select_ausdruck` genannt wurde. Die Klausel wird zuletzt angewandt, direkt bevor Ergebnisse an den Client geschickt werden, ohne jede Optimierung. Benutzen Sie kein `HAVING` für Dinge, die in der `WHERE`-Klausel stehen sollten. Schreiben Sie beispielsweise nicht folgendes:

```
mysql> select spalten_name from tabelle HAVING spalten_name > 0;
```

  Sondern statt dessen:

```
mysql> select spalten_name from tabelle WHERE spalten_name > 0;
```

  Ab MySQL-Version 3.22.5 können Sie Anfragen auch wie folgt schreiben:

```
mysql> select user,max(gehalt) from benutzer
           group by benutzer HAVING max(gehalt)>10;
```

  In älteren MySQL-Versionen schreiben Sie statt dessen:

```
mysql> select benutzer,max(gehalt) AS summe from benutzer
           group by benutzer HAVING summe>10;
```

- `SQL_SMALL_RESULT`, `SQL_BIG_RESULT`, `SQL_BUFFER_RESULT`, `STRAIGHT_JOIN` und `HIGH_PRIORITY` sind MySQL Erweiterungen zu ANSI-SQL92.

- `HIGH_PRIORITY` gibt dem `SELECT` höhere Priorität als einem Statement, das eine Tabelle aktualisiert. Sie sollten das nur für Anfragen benutzen, die sehr schnell sind und sofort durchgeführt werden müssen. Eine `SELECT HIGH_PRIORITY`-Anfrage läuft, wenn die Tabelle eine Lese-Sperre hat, selbst wenn es ein Update-Statement gibt, das darauf wartet, dass die Tabelle freigegeben wird.

- `SQL_BIG_RESULT` kann bei `GROUP BY` oder `DISTINCT` benutzt werden, um dem Optimierer mitzuteilen, dass das Ergebnis sehr viele Zeilen haben wird. In diesem Fall benutzt MySQL bei Bedarf direkt Festplatten-basierende temporäre Tabellen. Ausserdem bevorzugt MySQL in diesem Fall Sortieren vor dem Anlegen einer temporären Tabelle mit einem Schlüssel auf den `GROUP BY`-Elementen.

- Wenn Sie `GROUP BY` benutzen, werden die Ausgabe-Zeilen gemäß dem `GROUP BY` sortiert, als hätten Sie ein `ORDER BY` für alle Felder im `GROUP BY` angegeben. MySQL hat `GROUP BY` erweitert, so dass Sie dafür auch `ASC` und `DESC` angeben können:

```
SELECT a,COUNT(b) FROM tabelle GROUP BY a DESC
```

- MySQL hat die Benutzung von `GROUP BY` erweitert, um es Ihnen zu gestatten, auch Felder auszuwählen, die nicht in der `GROUP BY`-Klausel erwähnt wurden. Wenn Sie nicht die Ergebnisse erhalten, die Sie von Ihrer Anfrage erwarten, lesen Sie bitte die `GROUP BY`-Beschreibung.

- `SQL_BUFFER_RESULT` erzwingt, dass das Ergebnis in eine temporäre Tabelle geschrieben wird. Das hilft MySQL, frühzeitig

Tabellensperren aufzuheben, und hilft in Fällen, in denen es lange dauert, das Ergebnis an den Client zu senden.

- `SQL_SMALL_RESULT`, eine MySQL-spezifische Option, kann bei `GROUP BY` oder `DISTINCT` benutzt werden, um dem Optimierer mitzuteilen, dass der Ergebnissatz klein sein wird. In diesem Fall benutzt MySQL schnelle temporäre Tabellen, um die Ergebnistabelle zu speichern, anstatt Sortieren zu benutzen. In MySQL-Version 3.23 sollte das normalerweise nicht benötigt werden.

- `STRAIGHT_JOIN` zwingt den Optimierer, Tabellen in der Reihenfolge zu verknüpfen, in der sie in der `FROM`-Klausel aufgelistet sind. Sie können das benutzen, um die Geschwindigkeit einer Anfrage zu erhöhen, wenn der Optimierer Tabellen in nicht optimaler Reihenfolge verknüpft. See Abschnitt 6.2.1, „`EXPLAIN`-Syntax (Informationen über ein `SELECT` erhalten)“.

- Die `LIMIT`-Klausel wird benutzt, um die Anzahl von Zeilen, die vom `SELECT`-Statement zurückgegeben werden, zu beschränken. `LIMIT` erwartet ein oder zwei numerische Argumente.

  Wenn zwei Argumente angegeben sind, legt das erste den Offset der ersten Zeile fest, die zurückgegeben wird, und das zweite gibt die maximale Anzahl von Zeilen an, die zurückgegeben werden. Der Offset der anfänglichen Zeile ist 0 (nicht 1):

  ```
  mysql> select * from tabelle LIMIT 5,10;  # Zeilen 6 bis 15 zurückgeben
  ```

  Wenn ein Argument angegeben wird, stellt es die maximale Anzahl von Zeilen dar, die zurückgegeben werden:

  ```
  mysql> select * from tabelle LIMIT 5;     # Die ersten 5 Zeilen zurückgeben
  ```

  Mit anderen Worten ist `LIMIT n` äquivalent zu `LIMIT 0,n`.

- Die `SELECT ... INTO OUTFILE 'datei'`-Form von `SELECT` schreibt die ausgewählten Zeilen in eine Datei. Die Datei wird auf dem Server-Host erzeugt und darf nicht bereits bestehen (das verhindert unter anderem, dass Datenbanktabellen und Dateien wie `/etc/passwd` zerstört werden). Sie benötigen die **file**-Berechtigung auf dem Server-Host, um diese Form von `SELECT` auszuführen.

  `SELECT ... INTO OUTFILE` ist hauptsächlich dafür vorgesehen, um eine Tabelle auf der Server-Maschine schnell zu dumpen. Wenn Sie die resultierende Datei auf einem anderen Host als dem Server-Host haben wollen, können Sie `SELECT ... INTO OUTFILE` nicht benutzen. In diesem Fall sollten Sie statt dessen ein Client-Programm wie `mysqldump --tab` oder `mysql -e "SELECT ..." > outfile` benutzen, um die Datei zu erzeugen.

  `SELECT ... INTO OUTFILE` ist das Komplement von `LOAD DATA INFILE`; die Syntax für den `export_optionen`-Teil des Statements besteht aus denselben `FIELDS`- und `LINES`-Klauseln, die beim `LOAD DATA INFILE`-Statement benutzt werden. See Abschnitt 7.4.9, „`LOAD DATA INFILE`-Syntax“.

  In der resultierenden Textdatei werden nur folgende Zeichen durch das `ESCAPED BY`-Zeichen escapet:

  - Das `ESCAPED BY`-Zeichen
  - Das erste Zeichen in `FIELDS TERMINATED BY`
  - Das erste Zeichen in `LINES TERMINATED BY`

  Zusätzlich wird `ASCII 0` in `ESCAPED BY`, gefolgt von 0 (`ASCII 48`), umgewandelt.

  Der Grund hierfür ist, dass Sie jegliche `FIELDS TERMINATED BY`-, `ESCAPED BY`- oder `LINES TERMINATED BY`-Zeichen escapen MÜSSEN, um die Datei zuverlässig wieder einlesen zu können. `ASCII 0` wird escapet, um das Lesen mit einigen Pagern zu erleichtern.

  Weil sich die resultierende Datei nicht nach der SQL-Syntax richten muss, muss nicht weiter escapet werden.

  Im Folgenden ein Beispiel, wie man eine Datei in einem Format erhält, das von vielen alten Programmen benutzt wird:

  ```
  SELECT a,b,a+b INTO OUTFILE "/tmp/resultat.text"
  FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
  LINES TERMINATED BY "\n"
  FROM tabelle;
  ```

- Wenn Sie `INTO DUMPFILE` anstelle von `INTO OUTFILE` benutzen, schreibt MySQL nur eine Zeile in die Datei, ohne jede Spalten- oder Zeilen-Begrenzer und ohne jedes Escapen. Das ist nützlich, wenn Sie ein Blob in eine Datei speichern wollen.

- Beachten Sie, dass jede Datei, die von `INTO OUTFILE` und `INTO DUMPFILE` erzeugt wird, für alle Benutzer lesbar ist! Der Grund liegt darin, dass der MySQL-Server keine Datei erzeugen kann, die jemandem anderen gehört als dem Benutzer, unter dem er läuft (Sie sollten `mysqld` nie als Root laufen lassen), daher muss die Datei für jedermann lesbar sein, damit Sie die Zeilen abrufen können.

- Wenn Sie `FOR UPDATE` bei einem Tabellen-Handler mit Seiten-/Zeilen-Sperren benutzen, werden die untersuchten Zeilen

schreib-gesperrt.

7.4.1.1. JOIN-Syntax

MySQL unterstützt folgende JOIN-Syntaxen für SELECT-Statements:

```
tabellen_verweis, tabellen_verweis
tabellen_verweis [CROSS] JOIN tabellen_verweis
tabellen_verweis INNER JOIN tabellen_verweis join_bedingung
tabellen_verweis STRAIGHT_JOIN tabellen_verweis
tabellen_verweis LEFT [OUTER] JOIN tabellen_verweis join_bedingung
tabellen_verweis LEFT [OUTER] JOIN tabellen_verweis
tabellen_verweis NATURAL [LEFT [OUTER]] JOIN tabellen_verweis
{ oder tabellen_verweis LEFT OUTER JOIN tabellen_verweis ON bedingungs_ausdruck }
tabellen_verweis RIGHT [OUTER] JOIN tabellen_verweis join_bedingung
tabellen_verweis RIGHT [OUTER] JOIN tabellen_verweis
tabellen_verweis NATURAL [RIGHT [OUTER]] JOIN tabellen_verweis
```

Wobei tabellen_verweis definiert ist als:

```
tabelle [[AS] alias] [USE INDEX (schluessel_liste)] [IGNORE INDEX (schluessel_liste)]
```

Und join_bedingung definiert ist als:

```
ON bedingungs_ausdruck |
USING (spalten_liste)
```

Sie sollten nie irgend welche Bedingungen im ON-Teil haben, die dazu benutzt werden, um die Zeilen, die im Ergebnissatz auftauchen, zu beschränken. Wenn Sie so etwas tun wollen, müssen Sie das in der WHERE-Klausel tun.

Beachten Sie, dass vor Version 3.23.17 INNER JOIN keine join_bedingung aufnahm!

Die letzte oben dargestellte LEFT OUTER JOIN-Syntax gibt es nur aus Gründen der Kompatibilität mit ODBC:

- Einem Tabellenverweis kann mit tabelle AS alias_name oder tabelle alias_name ein Alias zugewiesen werden:

  ```
  mysql> select t1.name, t2.gehalt from angestellte AS t1, info AS t2
            where t1.name = t2.name;
  ```

- Der ON-Bedingungscode ist jeglicher Bedingungscode der Form, wie er auch in einer WHERE-Klausel benutzt werden kann.

- Wenn es für die rechte Tabelle keinen übereinstimmenden Datensatz im ON- oder USING-Teil eines LEFT JOIN gibt, wird für die rechte Tabelle eine Zeile benutzt, in der alle Spalten auf NULL gesetzt sind. Das können Sie benutzen, um Datensätze in einer Tabelle herauszusuchen, die in einer anderen Tabelle kein Gegenstück haben:

  ```
  mysql> select tabelle1.* from tabelle1
            LEFT JOIN tabelle2 ON tabelle1.id=tabelle2.id
            where tabelle2.id is NULL;
  ```

  Dieses Beispiel findet alle Zeilen in tabelle1 mit einem id-Wert, der in tabelle2 nicht vorhanden ist (also alle Zeilen in tabelle1 ohne entsprechende Zeile in tabelle2). Hierbei wird natürlich angenommen, dass tabelle2.id als NOT NULL deklariert ist. See Abschnitt 6.2.6, „Wie MySQL LEFT JOIN optimiert“.

- Die USING-(spalten_liste)-Klausel nennt eine Auflistung von Spalten, die in beiden Tabellen existieren müssen. Eine USING-Klausel wie:

  ```
  A LEFT JOIN B USING (C1,C2,C3,...)
  ```

  Ist definiert als semantisch identisch mit einem ON-Ausdruck wie diesem:

  ```
  A.C1=B.C1 AND A.C2=B.C2 AND A.C3=B.C3,...
  ```

- Der NATURAL [LEFT] JOIN zweier Tabellen ist definiert als semantisch identisch äquivalent zu einem INNER JOIN oder einem LEFT JOIN mit einer USING-Klausel, die alle Spalten nennt, die in beiden Tabellen existieren.

- RIGHT JOIN funktioniert analog wie LEFT JOIN. Um Code zwischen Datenbanken portabel zu halten, wird empfohlen, LEFT JOIN anstelle von RIGHT JOIN zu benutzen.

- STRAIGHT_JOIN ist identisch mit JOIN, ausser dass die linke Tabelle immer vor der rechten Tabelle gelesen wird. Das kann

in den (wenigen) Fällen benutzt werden, wo der Optimierer die Tabellen in die falsche Reihenfolge bringt.

- Ab MySQL-Version 3.23.12 können Sie Hinweise darüber geben, welchen Index MySQL benutzen sollte, wenn Informationen aus einer Tabelle abgerufen werden. Das ist nützlich, wenn `EXPLAIN` zeigt, dass MySQL den falschen Index benutzt. Indem Sie `USE INDEX (schluessel_liste)` angeben, können Sie MySQL anweisen, nur einen der angegebenen Indexe zu benutzen, um Zeilen in der Tabelle zu finden. Die alternative Syntax `IGNORE INDEX (schluessel_liste)` kann benutzt werden, um MySQL anzuweisen, einen bestimmten Index nicht zu benutzen.

Einige Beispiele:

```
mysql> select * from tabelle1,tabelle2 where tabelle1.id=tabelle2.id;
mysql> select * from tabelle1 LEFT JOIN tabelle2 ON tabelle1.id=tabelle2.id;
mysql> select * from tabelle1 LEFT JOIN tabelle2 USING (id);
mysql> select * from tabelle1 LEFT JOIN tabelle2 ON tabelle1.id=tabelle2.id
           LEFT JOIN table3 ON tabelle2.id=table3.id;
mysql> select * from tabelle1 USE INDEX (schluessel1,schluessel2) WHERE schluessel1=1 und schluessel2=2 AND
       schluessel3=3;
mysql> select * from tabelle1 IGNORE INDEX (schluessel3) WHERE schluessel1=1 und schluessel2=2 AND
       schluessel3=3;
```

See Abschnitt 6.2.6, „Wie MySQL `LEFT JOIN` optimiert“.

7.4.1.2. `UNION`-Syntax

```
SELECT ....
UNION [ALL]
SELECT ....
  [UNION
   SELECT ...]
```

`UNION` ist implementiert in MySQL 4.0.0.

`UNION` wird benutzt, um das Ergebnis vieler `SELECT`-Statements in einem Ergebnissatz zu kombinieren.

Die `SELECT`-Befehle sind normale SELECT-Befehle, aber mit folgenden Einschränkungen:

- Nur der letzte `SELECT`-Befehl darf `INTO OUTFILE` enthalten.
- Nur der letzte `SELECT`-Befehl darf `ORDER BY` enthalten.

Wenn Sie das Schlüsselwort `ALL` für `UNION` nicht benutzen, sind alle zurückgegebenen Zeilen eindeutig (unique), als hätten Sie ein `DISTINCT` für den gesamten Ergebnissatz gemacht. Wenn Sie `ALL` angeben, erhalten Sie alle übereinstimmenden Zeilen von allen benutzten `SELECT`-Statements.

7.4.2. `INSERT`-Syntax

```
    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tabelle [(spalten_name,...)]
        VALUES (ausdruck,...),(...),...
oder  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tabelle [(spalten_name,...)]
        SELECT ...
oder  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tabelle
        SET spalten_name=ausdruck, spalten_name=ausdruck, ...
```

`INSERT` fügt neue Zeilen in eine bestehende Tabelle ein. Die `INSERT ... VALUES`-Form des Statements fügt Zeilen basierend auf explizit angegebenen Werten ein. Die `INSERT ... SELECT`-Form fügt Zeilen ein, die aus einer oder mehreren anderen Tabellen ausgewählt wurden. Die `INSERT ... VALUES`-Form mit mehrfachen Wertelisten wird ab MySQL-Version 3.22.5 unterstützt. Die `spalten_name=expression`-Syntax wird ab MySQL-Version 3.22.10 unterstützt.

`tabelle` ist die Tabelle, in die Zeilen eingefügt werden sollen. Die Spaltennamenliste oder die `SET`-Klausel geben an, für welche Spalten das Statement Werte angibt:

- Wenn Sie keine Spaltenliste für `INSERT ... VALUES` oder `INSERT ... SELECT` angeben, müssen für alle Spalten Werte in der `VALUES()`-Liste oder vom `SELECT` bereit stehen. Wenn Sie die Reihenfolge der Tabellenspalten nicht kennen, benutzen Sie `DESCRIBE tabelle`, um sie herauszufinden.
- Jede Spalte, die nicht explizit in einer Werteliste angegeben wird, wird auf ihren Vorgabewert gesetzt. Wenn Sie beispielsweise eine Spaltenliste angeben, die nicht alle Tabellenspalten nennt, werden unbenannte Spalten auf ihre Vorgabewerte gesetzt. Die

Zuweisung von Vorgabewerten ist in Abschnitt 7.5.3, „CREATE TABLE-Syntax“ beschrieben.

- Ein `ausdruck` kann sich auf jede Spalte beziehen, die vorher in einer Werteliste angegeben wurde. Beispielsweise können Sie folgendes eingeben:

```
mysql> INSERT INTO tabelle (spalte1,spalte2) VALUES(15,spalte1*2);
```

Aber nicht das hier:

```
mysql> INSERT INTO tabelle (spalte1,spalte2) VALUES(spalte2*2,15);
```

- Wenn Sie das Schlüsselwort `LOW_PRIORITY` angeben, wird die Ausführung von `INSERT` verzögert, bis kein anderer Client mehr aus der Tabelle liest. In diesem Fall muss der Client warten, bis das INSERT-Statement fertig ist, was lange Zeit dauern kann, wenn die Tabelle stark benutzt wird. Das ist im Gegensatz zu `INSERT DELAYED`, was den Client sofort weitermachen läßt. See Abschnitt 7.4.4, „`INSERT DELAYED`-Syntax“. Beachten Sie, dass `LOW_PRIORITY` normalerweise nicht bei `MyISAM`-Tabellen benutzt werden sollte, weil dadurch gleichzeitige Einfügeoperationen verhindert werden. See Abschnitt 8.1, „MyISAM-Tabellen“.

- Wenn Sie das Schlüsselwort `IGNORE` in einem `INSERT` mit vielen Wertezeilen angeben, werden alle Zeilen, die einen bestehenden `PRIMARY`- oder `UNIQUE`-Schlüssel duplizieren würden, ignoriert und nicht eingefügt. Wenn Sie `IGNORE` nicht angeben, wird die Einfügeoperation abgebrochen, wenn es eine Zeile gibt, die einen bestehenden Schlüsselwert duplizieren würde. Mit der C-API-Funktion `mysql_info()` können Sie feststellen, wie viele Zeilen in die Tabelle eingefügt wurden.

- Wenn MySQL mit der `DONT_USE_DEFAULT_FIELDS`-Option konfiguriert wurde, erzeugen `INSERT`-Statements einen Fehler, wenn Sie nicht explizit Werte für alle Spalten angeben, die einen Nicht-`NULL`-Wert erfordern. See Abschnitt 3.3.3, „Typische `configure`-Optionen“.

- Den Wert, der für eine `AUTO_INCREMENT`-Spalte benutzt wurde, finden Sie mit der `mysql_insert_id`-Funktion heraus. See Abschnitt 9.4.3.30, „`mysql_insert_id()`“.

Wenn Sie ein `INSERT ... SELECT`- oder ein `INSERT ... VALUES`-Statement mit mehrfachen Wertlisten benutzen, können Sie die C-API-Funktion `mysql_info()` benutzen, um Informationen über die Anfrage zu erhalten. Das Format der Informationszeichenkette ist unten dargestellt:

```
Records: 100 Duplicates: 0 Warnings: 0
```

`Duplicates` zeigt die Anzahl von Zeilen, die nicht eingefügt werden konnten, weil sie einen bestehenden eindeutigen Indexwert dupliziert hätten. `Warnings` zeigen die Anzahl von Versuchen, Spaltenwerte einzufügen, die in irgend einer Weise problematisch waren. Warnungen erfolgen unter folgenden Umständen:

- Wenn `NULL` in eine Spalte eingefügt wird, die als `NOT NULL` deklariert ist. Die Spalte wird auf ihren Vorgabewert gesetzt.

- Wenn eine numerische Spalte auf einen Wert ausserhalb des Wertebereichs der Spalte gesetzt wird. Der Wert wird auf den entsprechenden Endpunkt des Bereichs abgeschnitten.

- Wenn eine numerische Spalte auf einen Wert wie `'10.34 a'` gesetzt wird. Die unsinnigen Zeichen am Ende werden entfernt und der verbleibende numerische Anteil eingefügt. Wenn der Wert als Zahl überhaupt keinen Sinn ergibt, wird die Spalte auf `0` gesetzt.

- Wenn eine Zeichenkette in eine `CHAR`-, `VARCHAR`-, `TEXT`- oder `BLOB`-Spalte eingefügt wird, die die maximale Länge der Spalte überschreitet. Der Wert wird auf die maximale Spaltenlänge beschnitten.

- Wenn ein Wert in eine DATE- oder TIME-Spalte eingefügt wird, der für den Spaltentyp nicht zulässig ist. Die Spalte wird auf den entsprechenden 0-Wert für diesen Typ gesetzt.

7.4.3. `HANDLER`-Syntax

```
HANDLER tabelle OPEN [ AS alias ]
HANDLER tabelle READ index { = | >= | <= | < } (wert1, wert2, ... )  [ WHERE ... ] [LIMIT ... ]
HANDLER tabelle READ index { FIRST | NEXT | PREV | LAST } [ WHERE ... ] [LIMIT ... ]
HANDLER tabelle READ { FIRST | NEXT }  [ WHERE ... ] [LIMIT ... ]
HANDLER tabelle CLOSE
```

Das `HANDLER`-Statement ermöglicht direkten Zugriff auf die MySQL-Tabellenschnittstelle unter Umgehung des SQL-Optimierers. Daher ist es schneller als SELECT.

Die erste Form des `HANDLER`-Statements öffnet eine Tabelle und macht sie über die folgenden `HANDLER ... READ`-Routinen

zugänglich. Dieses Tabellenobjekt wird nicht mit anderen Threads geteilt und wird nicht geschlossen, bis der Thread `HANDLER tabelle CLOSE` aufruft oder stirbt.

Die zweite Form holt eine (oder mehrere, festgelegt durch die `LIMIT`-Klausel) Zeile, bei der der angegebene Index mit der Bedingung übereinstimmt und die `WHERE`-Bedingung erfüllt ist. Wenn der Index aus mehreren Teilen besteht (also mehrere Spalten überspannt), werden die Werte in einer Komma-getrennten Liste angegeben, wobei es möglich ist, nur Werte für einige erste Spalten anzugeben.

Die dritte Form holt eine (oder mehrere, festgelegt durch die `LIMIT`-Klausel) Zeile in Index-Reihenfolge aus der Tabelle, bei der die `WHERE`-Bedingung erfüllt ist.

Die vierte Form (ohne Index-Angabe) holt eine (oder mehrere, festgelegt durch die `LIMIT`-Klausel) Zeile in natürlicher Zeilenreihenfolge aus der Tabelle (wie in der Daten-Datei gespeichert), bei der die `WHERE`-Bedingung erfüllt ist. Das ist schneller als `HANDLER tabelle READ index`, wenn ein kompletter Tabellen-Scan erwünscht ist.

Die letzte Form schließt eine mit `HANDLER ... OPEN` geöffnete Tabelle.

`HANDLER` ist in gewisser Hinsicht ein Statement auf niedriger Ebene (Low-Level), dass zum Beispiel keine Konsistenz gewährleistet. Das heißt, `HANDLER ... OPEN` nimmt **KEINEN** Schnappschuss der Tabelle auf und sperrt die Tabelle **NICHT**. Das bedeutet, dass nach `HANDLER ... OPEN` Tabellendaten verändert werden können (durch diesen oder einen anderen Thread) und dass diese Veränderungen nur teilweise in `HANDLER ... NEXT`- oder `HANDLER ... PREV`-Scans erscheinen.

7.4.3.1. `INSERT ... SELECT`-Syntax

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] tabelle [(spalten_liste)] SELECT ...
```

Mit dem `INSERT ... SELECT`-Statement können Sie schnell viele Zeilen aus einer oder mehreren anderen Tabellen einfügen.

```
INSERT INTO temporaere_tabelle2 (fldID) SELECT temporaere_tabelle1.fldOrder_ID FROM temporaere_tabelle1 WHERE
temporaere_tabelle1.fldOrder_ID > 100;
```

Folgende Bedingungen gelten für ein `INSERT ... SELECT`-Statement:

- Die Ziel-Tabelle des `INSERT`-Statements darf nicht in der `FROM`-Klausel des `SELECT`-Teils der Anfrage erscheinen, weil es in ANSI-SQL verboten ist, aus derselben Tabelle auszuwählen (`SELECT`), in die eingefügt wird. (Das Problem liegt darin, dass das `SELECT` möglicherweise Datensätze finden würde, die früher während desselben Laufs eingefügt wurden. Wenn man Sub-Select-Klauseln verwendet, könnte die Situation schnell sehr verwirrend werden!)
- `AUTO_INCREMENT`-Spalten funktionieren wie gehabt.
- Sie können die C-API-Funktion `mysql_info()` benutzen, um Informationen über die Anfrage zu erhalten. See Abschnitt 7.4.3, „HANDLER-Syntax“.
- Um sicherzustellen, dass die Update-Log-Datei/Binär-Log-Datei benutzt werden kann, um die Original-Tabellenlänge neu zu erzeugen, läßt MySQL während `INSERT .... SELECT` keine gleichzeitigen Einfügeoperationen zu.

Sie können natürlich `REPLACE` anstelle von `INSERT` benutzen, um alte Zeilen zu überschreiben.

7.4.4. `INSERT DELAYED`-Syntax

```
INSERT DELAYED ...
```

Die `DELAYED`-Option für das `INSERT`-Statement ist eine MySQL-spezifische Option, die sehr nützlich ist, wenn Sie Clients haben, die nicht warten können, bis das `INSERT` fertig ist. Die ist ein häufiges Problem, wenn Sie MySQL zum Loggen benutzen und gelegentlich `SELECT`- und `UPDATE`-Statements laufen lassen, die lange Zeit benötigen. `DELAYED` wurde in MySQL-Version 3.22.15 eingeführt. Es ist eine MySQL Erweiterung zu ANSI-SQL92.

`INSERT DELAYED` funktioniert nur bei `ISAM`- und `MyISAM`-Tabellen. Beachten Sie: Weil `MyISAM`-Tabellen gleichzeitige `SELECT` und `INSERT` unterstützen, wenn es keine freien Blöcke mitten in der Daten-Datei gibt, müssen Sie `INSERT DELAYED` bei `MyISAM` nur sehr selten benutzen. See Abschnitt 8.1, „MyISAM-Tabellen“.

Wenn Sie `INSERT DELAYED` benutzen, erhält der Client sofort ein Okay, und die Zeile wird eingefügt, wenn die Tabelle nicht mehr durch einen anderen Thread in Benutzung ist.

Ein weiterer großer Vorteil von `INSERT DELAYED` ist, dass Einfügeoperationen vieler Clients gebündelt und in einem Block geschrieben werden. Das ist viel schneller als viele separate Inserts durchzuführen.

Beachten Sie, dass momentan die Zeilen in der Warteschlange solange nur im Arbeitsspeicher gehalten werden, bis sie in die

Tabelle eingefügt sind. Das heißt, wenn Sie `mysqld` auf die harte Tour killen (`kill -9`) oder wenn `mysqld` unerwartet stirbt, sind Zeilen in der Warteschlange, die noch nicht auf Festplatte geschrieben wurden, verloren!

Im Folgenden ist detailliert beschrieben, was geschieht, wenn Sie die `DELAYED`-Option für `INSERT` oder `REPLACE` benutzen. In dieser Beschreibung ist der ``Thread'' der Thread, der einen `INSERT DELAYED`-Befehl empfängt. ``Handler'' ist der Thread, der alle `INSERT DELAYED`-Statements für ein bestimmte Tabelle handhabt.

- When ein Thread ein `DELAYED`-Statement für eine Tabelle ausführt, wird ein Handler-Thread erzeugt, um alle `DELAYED`-Statements für die Tabelle auszuführen, wenn ein solcher Handler nicht schon existiert.
- Der Thread prüft, ob der Handler bereit eine `DELAYED`-Sperre erhalten hat oder nicht. Wenn nicht, weist es den Handler-Thread an, das zu tun. Die `DELAYED`-Sperre kann selbst dann erlangt werden, wenn ein anderer Thread eine `READ`- oder `WRITE`-Sperre auf der Tabelle hat. Der Handler wartet jedoch auf alle `ALTER TABLE`-Sperren oder `FLUSH TABLES`, um sicherzustellen, dass die Tabellenstruktur aktuell ist.
- Der Thread führt das `INSERT`-Statement aus, aber statt die Zeile in die Tabelle zu schreiben stellt er eine Kopie der endgültigen Zeile in eine Warteschlange, die vom Handler-Thread verwaltet wird. Alle Syntaxfehler werden vom Thread erkannt und dem Client-Programm mitgeteilt.
- Der Client kann die Anzahl von Duplikaten oder den `AUTO_INCREMENT`-Wert für die resultierende Zeile nicht mitteilen. Er kann Sie vom Server nicht erhalten, weil das `INSERT` zurückkehrt, bevor die Einfügeoperation fertig ist. Wenn Sie die C-API benutzen, gibt die `mysql_info()`-Funktion aus demselben Grund nichts Sinnvolles zurück.
- Die Update-Log-Datei wird vom Handler-Thread aktualisiert, wenn die Zeile in die Tabelle eingefügt wird. Im Falle des Einfügens mehrerer Zeilen wird die Update-Log-Datei aktualisiert, wenn die erste Zeile eingefügt wird.
- Nachdem alle `delayed_insert_limit` Zeilen geschrieben wurden, prüft der Handler, ob noch irgend welche `SELECT`-Statements anhängig sind oder nicht. Falls ja, gestattet er diesen, ausgeführt zu werden, bevor weiter gemacht wird.
- Wenn der Handler keine Zeilen mehr in seiner Warteschlange hat, wird die Tabellensperre aufgehoben. Wenn innerhalb von `delayed_insert_timeout` Sekunden keine neuen `INSERT DELAYED`-Befehle mehr empfangen werden, beendet sich der Handler.
- Wenn mehr als `delayed_queue_size` Zeilen bereits in einer bestimmten Handler-Warteschlange anhängig sind, wartet der Thread, der nach `INSERT DELAYED` anfragt, bis es wieder Platz in der Warteschlange gibt. Damit wird sichergestellt, dass der `mysqld`-Server nicht den gesamten Arbeitsspeicher für die DELAYED-Warteschlange verbraucht.
- Der Handler-Thread zeigt sich in der MySQL-Prozessliste mit `delayed_insert` in der `Command`-Spalte. Er wird gekillt, wenn Sie einen `FLUSH TABLES`-Befehl ausführen oder ihn mit `KILL Thread_id` killen. Er wird jedoch zuerst alle Zeilen in der Warteschlange in die Tabelle schreiben, bevor er sich beendet. Während dieser Zeit akzeptiert er keine neuen `INSERT`-Befehle von anderen Threads mehr. Wenn Sie danach einen `INSERT DELAYED`-Befehl ausführen, wird ein neuer Handler-Thread erzeugt.
- Beachten Sie, dass oben Gesagtes bedeutet, dass `INSERT DELAYED`-Befehle höhere Priorität haben als normale `INSERT`-Befehle, wenn es einen `INSERT DELAYED`-Handler gibt, der bereits läuft! Andere Aktualisierungsbefehle müssen warten, bis die `INSERT DELAYED`-Warteschlange leer ist, jemand den Handler-Thread killt (mit `KILL Thread_id`) oder jemand `FLUSH TABLES` ausführt.
- Die folgenden Status-Variablen stellen Informationen über `INSERT DELAYED`-Befehle bereits:

Variable	Bedeutung
`Delayed_insert_thread`	Nummer des Handler-Threads
`Delayed_writes`	Anzahl der Zeilen, die mit `INSERT DELAYED` geschrieben wurden
`Not_flushed_delayed_rows`	Anzahl der Zeilen, die darauf warten, geschrieben zu werden

  Sie können diese Variablen betrachten, wenn Sie ein `SHOW STATUS`-Statement oder einen `mysqladmin extended-status`-Befehl ausführen.

Beachten Sie, dass `INSERT DELAYED` langsamer ist als ein normales INSERT, wenn die Tabelle nicht in Benutzung ist. Ausserdem gibt es einen zusätzlichen Overhead für den Server, um einen separaten Thread für jede Tabelle zu handhaben, für die Sie `INSERT DELAYED` benutzen. Das heißt, Sie sollten `INSERT DELAYED` nur benutzen, wenn Sie es wirklich benötigen!

7.4.5. `UPDATE`-Syntax

```
UPDATE [LOW_PRIORITY] [IGNORE] tabelle
    SET spalten_name1=ausdruck1, [spalten_name2=ausdruck2, ...]
```

```
    [WHERE where_definition]
    [LIMIT #]
```

`UPDATE` aktualisiert Spalten in bestehenden Tabellenzeilen mit neuen Werten. Die `SET`-Klausel gibt an, welche Spalten geändert werden sollen und welche Werte ihnen zugewiesen werden. Die `WHERE`-Klausel legt - falls angegeben - fest, welche Zeilen aktualisiert werden sollen. Ansonsten werden alle Zeile aktualisiert. Wenn die `ORDER BY`-Klausel angegeben ist, werden die Zeilen in der angegebenen Reihenfolge aktualisiert.

Wenn Sie das Schlüsselwort `LOW_PRIORITY` angeben, wird die Ausführung von `UPDATE` verzögert, bis keine anderen Clients mehr aus der Tabelle lesen.

Wenn Sie das Schlüsselwort `IGNORE` angeben, bricht das UPDATE-Statement nicht ab, selbst wenn während der Aktualisierung Fehler wegen doppelter Schlüsseleinträge auftreten. Zeilen, die Konflikte verursachen würden, werden nicht aktualisiert.

Wenn Sie auf eine Spalte von `tabelle` in einem Ausdruck zugreifen, benutzt `UPDATE` den momentanen Wert der Spalte. Folgendes Statement zum Beispiel setzt die `age`-Spalte auf ihren momentanen Wert plus 1:

```
mysql> UPDATE personen SET age=age+1;
```

`UPDATE`-Zuweisungen werden von links nach rechts ausgewertet. Folgendes Statement zum Beispiel verdoppelt die `age`-Spalte und inkrementiert sie danach:

```
mysql> UPDATE personen SET age=age*2, age=age+1;
```

Wenn Sie eine Spalte auf einen Wert setzen, den sie momentan besitzt, erkennt MySQL dies und aktualisiert sie nicht.

`UPDATE` gibt die Anzahl von Zeilen zurück, die tatsächlich geändert wurden. Ab MySQL-Version 3.22 gibt die C-API-Funktion `mysql_info()` die Anzahl von Zeilen zurück, die übereinstimmten und aktualisiert wurden, und die Anzahl von Warnungen, die während `UPDATE` geschahen.

In MySQL-Version 3.23 können Sie `LIMIT #` benutzen, um sicherzustellen, dass nur eine angegebene Anzahl von Zeilen geändert wird.

7.4.6. `DELETE`-Syntax

```
DELETE [LOW_PRIORITY | QUICK] FROM tabelle
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT zeilen]

oder

DELETE [LOW_PRIORITY | QUICK] tabelle[.*] [tabelle[.*] ...] FROM
tabellenverweis [WHERE where_definition]
```

`DELETE` löscht Zeilen aus `tabelle`, die mit der in `where_definition` angegebenen Bedingung übereinstimmen, und gibt die Anzahl der gelöschten Datensätze zurück.

Wenn Sie `DELETE` ohne `WHERE`-Klausel angeben, werden alle Zeilen gelöscht. Wenn Sie das im `AUTOCOMMIT`-Modus machen, funktioniert es wie `TRUNCATE`. See Abschnitt 7.4.7, „`TRUNCATE`-Syntax“. In MySQL 3.23 gibt `DELETE` ohne eine `WHERE`-Klausel als Anzahl von betroffenen Datensätzen 0 zurück.

Wenn Sie wissen wollen, wie viele Datensätze tatsächlich gelöscht wurden, wenn Sie alle Zeilen löschen, und eine Geschwindigkeitseinbusse in Kauf nehmen, können Sie ein `DELETE`-Statement folgender Form eingeben:

```
mysql> DELETE FROM tabelle WHERE 1>0;
```

Beachten Sie, dass das VIEL langsamer als `DELETE FROM tabelle` ohne `WHERE`-Klausel ist, weil es Zeilen eine nach der anderen löscht.

Wenn Sie das Schlüsselwort `LOW_PRIORITY` angeben, wird die Ausführung von `DELETE` verzögert, bis kein anderer Client mehr aus der Tabelle liest.

Wenn Sie das Wort `QUICK` angeben, fasst der Tabellen-Handler während des Löschvorgangs keine Index-Blätter (Index Leafs) zusammen, was bestimmte Arten von Löschvorgängen beschleunigen kann.

In MyISAM-Tabellen werden gelöschte Datensätze in einer verknüpften Liste verwaltet und nachfolgende `INSERT`-Operationen benutzen alte Datensatzpositionen neu. Um unbenutzten Platz freizugeben und Dateigrößen zu verringern, benutzen Sie das `OPTIMIZE TABLE`-Statement oder das `myisamchk`-Dienstprogramm, um die Tabellen neu zu organisieren. `OPTIMIZE TABLE` ist einfacher, aber `myisamchk` ist schneller. Siehe Abschnitt 5.5.1, „`OPTIMIZE TABLE`-Syntax“ und Abschnitt 5.4.6.10,

„Tabellenoptimierung“.

Das Multi-Tabellen-Löschformat wird ab MySQL 4.0.0 unterstützt.

Die Idee ist, dass nur übereinstimmende Zeilen aus den Tabellen, die VOR der `FROM`-Klausel stehen, gelöscht werden. Die Auswirkung ist, dass Sie Zeilen aus vielen Tabellen zugleich löschen können, sowie dass zusätzliche Tabellen zum Suchen benutzt werden.

Das `.*`-Zeichen nach den Tabellennamen ist nur aus Gründen der Kompatibilität mit `Access` vorhanden:

```
DELETE t1,t2 FROM t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
```

In diesem Fall werden übereinstimmende Zeilen nur aus den Tabellen `t1` und `t2` gelöscht.

`ORDER BY` und Benutzung mehrfacher Tabellen bei DELETE wird in MySQL 4.0 unterstützt.

Wenn eine `ORDER BY`-Klausel benutzt wird, werden die Zeilen in dieser Reihenfolge gelöscht. Das ist nur in Verbindung mit `LIMIT` wirklich sinnvoll. Beispiel:

```
DELETE FROM logdatei
WHERE user = 'jcole'
ORDER BY zeitstempel
LIMIT 1
```

Das löscht den ältesten Eintrag (von `zeitstempel`), wo die Zeile mit der `WHERE`-Klausel übereinstimmt.

Die MySQL-spezifische `LIMIT rows`-Option für `DELETE` weist den Server an, welche maximale Anzahl von Zeilen gelöscht wird, bevor die Kontrolle an den Client zurück gegeben wird. Das kann benutzt werden um sicherzustellen, dass ein bestimmter `DELETE`-Befehl nicht zu viel Zeit beansprucht. Sie können den `DELETE`-Befehl einfach wiederholen, bis die Anzahl betroffener Zeilen kleiner ist als der `LIMIT`-Wert.

7.4.7. `TRUNCATE`-Syntax

```
TRUNCATE TABLE tabelle
```

In Version 3.23 wird `TRUNCATE TABLE` auf `COMMIT ; DELETE FROM tabelle` gemappt. See Abschnitt 7.4.6, „`DELETE`-Syntax“.

Die Unterschiede zwischen `TRUNCATE TABLE` und `DELETE FROM ..` sind:

- TRUNCATE führt ein Löschen und Neuerzeugen der Tabelle durch, was viel schneller sit, als Zeilen eine nach der anderen zu löschen.
- Nicht transaktionssicher. Sie erhalten einen Fehler, wenn Sie eine aktive Transaktion haben oder eine aktive Tabellensperre.
- Gibt die Anzahl gelöschter Zeilen nicht zurück.
- Solange die Tabellendefinitionsdatei `tabelle.frm` gültig ist, kann die Tabelle auf diese Weise neu erzeugt werden, selbst wenn die Daten- oder Index-Dateien beschädigt wurden.

`TRUNCATE` ist eine Oracle-SQL-Erweiterung.

7.4.8. `REPLACE`-Syntax

```
    REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] tabelle [(spalten_name,...)]
        VALUES (ausdruck,...),(...),...
or  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] tabelle [(spalten_name,...)]
        SELECT ...
or  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] tabelle
        SET spalten_name=ausdruck, spalten_name=ausdruck,...
```

`REPLACE` funktioniert genau wie `INSERT`, ausser dass der alte Datensatz gelöscht wird, bevor ein neuer eingefügt wird, wenn ein alter Datensatz in der Tabelle denselben Wert wie der neue auf einem eindeutigen Index hat. See Abschnitt 7.4.3, „`HANDLER`-Syntax“.

Mit anderen Worten können Sie auf die Werte einer alten Zeile nicht mit einem `REPLACE`-Statement zugreifen. In einigen alten MySQL-Versionen sah es so aus, als könnten Sie das tun, aber das war ein Bug und wurde korrigiert.

Wenn man einen REPLACE-Befehl benutzt, gibt mysql_affected_rows() 2 zurück, wenn die neue Zeile eine alte ersetzte. Das liegt daran, dass in diesem Fall eine Zeile eingefügt wurde und dann das Duplikat gelöscht wurde.

Das macht es einfach zu überprüfen, ob REPLACE eine Zeile hinzugefügt oder eine ersetzt hat.

7.4.9. LOAD DATA INFILE-Syntax

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'datei.txt'
    [REPLACE | IGNORE]
    INTO TABLE tabelle
    [FIELDS
        [TERMINATED BY '\t']
        [[OPTIONALLY] ENCLOSED BY '']
        [ESCAPED BY '\\' ]
    ]
    [LINES TERMINATED BY '\n']
    [IGNORE Anzahl LINES]
    [(spalten_name,...)]
```

Das LOAD DATA INFILE-Statement liest Zeilen aus einer Textdatei in eine Tabelle mit sehr hoher Geschwindigkeit. Wenn das LOCAL-Schlüsselwort angegeben wird, wird die Datei vom Client-Host gelesen. Wenn LOCAL nicht angegeben wird, muss die Datei auf dem Server liegen. (LOCAL ist verfügbar ab MySQL-Version 3.22.6.)

Aus Sicherheitsgründen müssen Dateien, die als auf dem Server liegende Textdateien eingelesen werden, entweder im Datenbank-Verzeichnis liegen oder von allen lesbar sein. Darüber hinaus brauchen Sie, wenn Sie LOAD DATA INFILE mit Server-Dateien benutzen, die **file**-Berechtigung auf dem Server-Host. See Abschnitt 5.2.5, „Wie das Berechtigungssystem funktioniert“.

Wenn Sie das Schlüsselwort LOW_PRIORITY angeben, wird das LOAD DATA-Statement verzögert, bis keine anderen Clients mehr aus der Tabelle lesen.

Wenn Sie das Schlüsselwort CONCURRENT bei einer MyISAM-Tabelle angeben, können andere Threads Daten aus der Tabelle abrufen, während LOAD DATA ausgeführt wird. Die Benutzung dieser Option beeinflusst natürlich die Performance von LOAD DATA ein bisschen, selbst wenn kein anderer Thread die Tabelle zur gleichen Zeit benutzt.

LOCAL ist etwas langsamer, als wenn der Server direkt auf die Dateien zugreifen kann, weil die Inhalte der Datei vom Client-Host auf den Server-Host übertragen werden müssen. Auf der anderen Seite benötigen Sie keine **file**-Berechtigung, um lokale Dateien zu laden.

Wenn Sie MySQL vor Version 3.23.24 benutzen, können Sie nicht aus einer FIFO lesen, wenn Sie LOAD DATA INFILE benutzen. Wenn Sie aus einer FIFO lesen müssen (zum Beispiel aus der Ausgabe von gunzip), benutzen Sie statt dessen LOAD DATA LOCAL INFILE.

Sie können Daten-Dateien auch mit dem mysqlimport-Dienstprogramm laden; es arbeitet, indem es einen LOAD DATA INFILE-Befehl an den Server schickt. Die --local-Option veranlasst mysqlimport, Daten-Dateien vom Client-Host zu lesen. Sie können die --compress-Option angeben, um bessere Performance über langsame Netzwerke zu erzielen, wenn der Client und der Server das komprimierte Protokoll unterstützen.

Bei der Suche nach Dateien auf dem Server-Host geht der Server nach folgenden Regeln vor:

- Wenn ein absoluter Pfadname angegeben wird, benutzt der Server den Pfadnamen so, wie er ist.
- Wenn ein relativer Pfadname mit einer oder mehreren führenden Bestandteilen angegeben wird, sucht der Server die Datei relativ zum Daten-Verzeichnis des Servers.
- Wenn ein Dateiname ohne führende Bestandteile angegeben wird, sucht der Server die Datei im Datenbank-Verzeichnis der aktuellen Datenbank.

Beachten Sie, dass diese Regeln bedeuten, dass eine Datei, die als ./meinedatei.txt angegeben wird, aus dem Daten-Verzeichnis des Servers gelesen wird, wohingegen eine Datei, die als meinedatei.txt angegeben wird, aus dem Datenbank-Verzeichnis der aktuellen Datenbank gelesen wird. Das folgende LOAD DATA-Statement beispielsweise liest die Datei daten.txt aus dem Datenbank-Verzeichnis von datenbank1, weil datenbank1 die aktuelle Datenbank ist, obwohl das Statement die Datei explizit in eine Tabelle in der datenbank2-Datenbank lädt:

```
mysql> USE datenbank1;
mysql> LOAD DATA INFILE "daten.txt" INTO TABLE datenbank2.meine_tabelle;
```

Die REPLACE- und IGNORE-Schlüsselwörter steuern die Handhabung von Eingabe-Datensätzen, die bestehende Datensätze auf eindeutigen Schlüsselwerten duplizieren. Wenn Sie REPLACE angeben, ersetzen neue Zeilen bestehende Zeilen, die denselben eindeutigen Schlüsselwert besitzen. Wenn Sie IGNORE angeben, werden Eingabe-Zeilen, die eine bestehende Zeile auf einem Schlüsselwert duplizieren, übersprungen. Wenn Sie keine der beiden Optionen angeben, tritt ein Fehler auf, wenn ein doppelter

Schlüsselwert gefunden wird, und der Rest der Textdatei wird ignoriert.

Wenn Sie Daten aus einer lokalen Datei mit dem `LOCAL`-Schlüsselwort laden, hat der Server keine Möglichkeit, die Übertragung der Datei mitten in einer Operation zu beenden. Daher ist das vorgabemäßige Verhalten dasselbe, als wenn `IGNORE` angegeben wäre.

Wenn Sie `LOAD DATA INFILE` auf einer leeren `MyISAM`-Tabelle benutzen, werden alle nicht eindeutigen Indexe in einem separaten Stapel erzeugt (wie bei `REPAIR`). Das macht `LOAD DATA INFILE` normalerweise viel schneller, wenn Sie viele Indexe haben.

`LOAD DATA INFILE` ist das Komplement von `SELECT ... INTO OUTFILE`. See Abschnitt 7.4.1, „`SELECT`-Syntax“. Um Daten aus einer Datenbank in eine Datei zu schreiben, benutzen Sie `SELECT ... INTO OUTFILE`. Um die Datei zurück in die Datenbank zu lesen, benutzen Sie `LOAD DATA INFILE`. Die Syntax der `FIELDS`- und `LINES`-Klauseln ist für beide Befehle dieselbe. Beide Klauseln sind optional, aber `FIELDS` muss `LINES` vorangehen, wenn beide angegeben werden.

Wenn Sie eine `FIELDS`-Klausel angeben, ist jede ihrer Unterklauseln (`TERMINATED BY`, `[OPTIONALLY] ENCLOSED BY` und `ESCAPED BY`) ebenfalls optional, ausser dass Sie zumindest eine von ihnen angeben müssen.

Wenn Sie keine `FIELDS`-Klausel benutzen, sind die Vorgabewerte dieselben, als wenn Sie folgendes geschrieben hätten:

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
```

Wenn Sie keine `LINES`-Klausel angeben, sind die Vorgabewerte dieselben, als wenn Sie folgendes geschrieben hätten:

```
LINES TERMINATED BY '\n'
```

Mit anderen Worten veranlassen die Vorgabewerte `LOAD DATA INFILE`, beim Lesen von Eingaben wie folgt zu arbeiten:

- Zeilenbegrenzungen werden an Neue-Zeile-Zeichen gesucht (\n).
- Zeilen werden an Tabulatoren (\t) in Felder aufgeteilt.
- Es wird nicht davon ausgegangen, dass Felder in Anführungszeichen eingeschlossen sind.
- Tabulatoren, Neue-Zeile-Zeichen oder ‘`\`’, denen ein ‘`\`’-Zeichen voran gestellt ist, werden als Literale interpretiert, die Teil des Feldwerts sind.

Im Vergleich dazu veranlassen die Vorgabewerte von `SELECT ... INTO OUTFILE` dieses, wie folgt zu arbeiten:

- Zwischen Felder werden Tabulatoren (\t) geschrieben.
- Felder werden nicht in Anführungsstriche geschrieben.
- ‘`\`’ wird benutzt, um Tabulator, Neue-Zeile-Zeichen oder ‘`\`’ innerhalb von Feldwerten zu escapen.
- Am Ende von Zeilen werden Neue-Zeile-Zeichen (\n) geschrieben.

Beachten Sie, dass Sie `FIELDS ESCAPED BY '\\'` (mit zwei Backslashes) schreiben müssen, damit der Wert als ein einzelner Backslash gelesen wird.

Die `IGNORE anzahl LINES`-Option kann benutzt werden, um eine Kopfzeile aus Spaltennamen am Anfang der Datei zu ignorieren:

```
mysql> LOAD DATA INFILE "/tmp/datei.txt" into Tabelle test IGNORE 1 LINES;
```

Wenn Sie `SELECT ... INTO OUTFILE` zusammen mit `LOAD DATA INFILE` benutzen, um Daten aus einer Datenbank in eine Datei zu schreiben und dann die Datei später zurück in die Datenbank zu lesen, müssen die Optionen für die Behandlung von Feldern und Zeilen für beide Befehle übereinstimmen. Ansonsten interpretiert `LOAD DATA INFILE` die Inhalte der Datei nicht korrekt. Angenommen, Sie benutzen `SELECT ... INTO OUTFILE`, um eine Datei zu schreiben, deren Feldern durch Kommas begrenzt sind:

```
mysql> SELECT * INTO OUTFILE 'daten.txt'
           FIELDS TERMINATED BY ','
           FROM ...;
```

Um die Komma-begrenzte Datei wieder einzulesen, lautet das korrekte Statement:

```
mysql> LOAD DATA INFILE 'daten.txt' INTO TABLE tabelle2
```

```
          FIELDS TERMINATED BY ',';
```

Wenn Sie statt dessen versuchen, die Datei mit dem unten stehenden Statement einzulesen, funktioniert das nicht, weil es `LOAD DATA INFILE` anweist, nach Tabulatoren zwischen Feldern zu suchen:

```
mysql> LOAD DATA INFILE 'daten.txt' INTO TABLE tabelle2
          FIELDS TERMINATED BY '\t';
```

Das wahrscheinliche Ergebnis ist, dass jede Eingabezeile als ein einzelnes Feld interpretiert wird.

`LOAD DATA INFILE` kann auch benutzt werden, um Dateien aus externen Quellen einzulesen. Eine Datei im dBASE-Format zum Beispiel hat Felder, die durch Kommas getrennt und in Anführungszeichens eingeschlossen sind. Wenn Zeilen in der Datei von Neue-Zeile-Zeichen begrenzt sind, zeigt der unten stehende Befehl die Feld- und Zeilen-Handhabungsoptionen, die für das Laden der Datei benutzt werden:

```
mysql> LOAD DATA INFILE 'daten.txt' INTO TABLE tabelle
          FIELDS TERMINATED BY ',' ENCLOSED BY '"'
          LINES TERMINATED BY '\n';
```

Jede der Feld- oder Zeilen-Handhabungsoptionen kann eine leere Zeichenkette angeben (`''`). Wenn nicht leer, müssen die `FIELDS [OPTIONALLY] ENCLOSED BY`- und `FIELDS ESCAPED BY`-Werte ein einzelnes Zeichen sein. Die `FIELDS TERMINATED BY`- und `LINES TERMINATED BY`-Werte können aus mehr als einem Zeichen bestehen. Um zum Beispiel Zeilen zu schreiben, die durch Wagenrücklauf-Neue-Zeile-Paare getrennt sind, oder um eine Datei einzulesen, die solche Zeilen enthält, geben Sie eine `LINES TERMINATED BY '\r\n'`-Klausel an.

Um beispielsweise eine Datei mit Witzen einzulesen, die durch `%%` getrennt sind, können Sie folgendes eingeben:

```
create table witze (a int not null auto_increment primary key, witz text not null);
load data infile "/tmp/witze.txt" into table witze fields terminated by "" lines terminated by "\n%%\n" (witz);
```

`FIELDS [OPTIONALLY] ENCLOSED BY` steuert die Art von Anführungszeichen von Feldern. Wenn Sie bei der Ausgabe (`SELECT ... INTO OUTFILE`) das Wort `OPTIONALLY` auslassen, sind alle Felder vom `ENCLOSED BY`-Zeichen eingeschlossen. Ein Beispiel einer solchen Ausgabe (mit Kommas als Feldbegrenzern) ist unten dargestellt:

```
"1","eine Zeichenkette","100.20"
"2","eine Zeichenkette, die ein Komma (,) enthält","102.20"
"3","eine Zeichenkette, die ein \" Anführungszeichen enthält","102.20"
"4","eine Zeichenkette, die ein \", Anführungszeichen und Komma (,) enthält","102.20"
```

Wenn Sie `OPTIONALLY` angeben, wird das `ENCLOSED BY`-Zeichen nur benutzt, um `CHAR`- und `VARCHAR`-Felder zu umschließen:

```
1,"eine Zeichenkette",100.20
2,"eine Zeichenkette mit einem , Komma",102.20
3,"eine Zeichenkette mit einem \" Anführungszeichen",102.20
4,"eine Zeichenkette mit \", Anführungszeichen und Komma",102.20
```

Beachten Sie, dass `ENCLOSED BY`-Zeichen innerhalb eines Feldwerts escapet werden, indem ihnen das `ESCAPED BY`-Zeichen vorangestellt wird. Beachten Sie auch, dass es bei der Angabe eines leeren empty `ESCAPED BY`-Werts möglich ist, Ausgaben zu erzeugen, die nicht korrekt von `LOAD DATA INFILE` eingelesen werden können. Die oben dargestellte Ausgabe zum Beispiel würde wie im Folgenden gezeigt erscheinen, wenn das Fluchtzeichen (Escape-Zeichen) leer ist. Beachten Sie, dass das zweite Feld der vierten Zeile nach dem Anführungszeichen ein Komma enthält, was (irrtümlich) als Feldbegrenzer interpretiert wird:

```
1,"eine Zeichenkette",100.20
2,"eine Zeichenkette mit einem , Komma",102.20
3,"eine Zeichenkette mit einem " Anführungszeichen",102.20
4,"eine Zeichenkette mit ", Anführungszeichen und Komma",102.20
```

Für die Eingabe wird das `ENCLOSED BY`-Zeichen - falls vorhanden - vom Ende von Feldwerten entfernt. (Das gilt, egal ob `OPTIONALLY` angegeben ist oder nicht; `OPTIONALLY` hat keine Auswirkung auf die Interpretation der Eingabe.) `ENCLOSED BY`-Zeichen, denen das `ESCAPED BY`-Zeichen vorangestellt ist, werden als Teil des aktuellen Feldwerts interpretiert. Zusätzlich werden verdoppelte `ENCLOSED BY`-Zeichen innerhalb von Feldern als ein einzelnes `ENCLOSED BY`-Zeichen interpretiert, falls das Feld selbst mit diesem Zeichen anfängt. Wenn beispielsweise `ENCLOSED BY '"'` angegeben wird, werden Anführungszeichen wie folgt behandelt:

```
"Der ""BIG"" Boss"  -> Der "BIG" Boss
Der "BIG" Boss      -> Der "BIG" Boss
Der ""BIG"" Boss    -> Der ""BIG"" Boss
```

`FIELDS ESCAPED BY` steuert, wie Sonderzeichen geschrieben oder gelesen werden. Wenn das `FIELDS ESCAPED BY`-Zeichen nicht leer ist, wird es benutzt, um es bei der Ausgabe folgenden Zeichen voranzustellen:

- Dem `FIELDS ESCAPED BY`-Zeichen
- Dem `FIELDS [OPTIONALLY] ENCLOSED BY`-Zeichen
- Dem ersten Zeichen von `FIELDS TERMINATED BY`- und `LINES TERMINATED BY`-Werten
- ASCII `0` (was tatsächlich nach dem Fluchtzeichen (Escape-Zeichen) als ASCII `'0'` geschrieben wird, nicht ein Byte mit Wert 0)

Wenn das `FIELDS ESCAPED BY`-Zeichen leer ist, werden keine Zeichen escapet. Es ist wahrscheinlich keine gute Idee, ein leeres Fluchtzeichen (Escape-Zeichen) anzugeben, insbesondere, wenn Feldwerte in Ihren Daten irgend welche der Zeichen enthalten, die gerade aufgelistet wurden.

Für die Eingabe werden, falls das `FIELDS ESCAPED BY`-Zeichen nicht leer ist, Vorkommen dieses Zeichens entfernt, und die folgenden Zeichen werden buchstäblich als Teil des Feldwerts genommen. Die Ausnahmen sind ein escapetes '`0`' oder '`N`' (beispielsweise `\0` oder `\N`, wenn das Fluchtzeichen (Escape-Zeichen) '`\`' ist). Diese Folgen werden als ASCII-`0` interpretiert (ein Byte mit Wert 0) und `NULL`. Siehe unten zu den Regeln der `NULL`-Handhabung.

Weitere Informationen über die '`\`'-Escape-Syntax finden Sie unter Abschnitt 7.1.1, „Literale: Wie Zeichenketten und Zahlen geschrieben werden“.

In bestimmten Fällen beeinflussen sich die Handhabungsoptionen für Felder und Zeilen gegenseitig:

- Wenn `LINES TERMINATED BY` eine leere Zeichenkette ist und `FIELDS TERMINATED BY` nicht leer ist, werden Zeile auch durch `FIELDS TERMINATED BY` begrenzt.
- Wenn die `FIELDS TERMINATED BY`- und `FIELDS ENCLOSED BY`-Werte beide leer sind (`''`), wird ein Festzeilen- (nicht begrenztes) Format benutzt. Beim Festzeilenformat werden keine Begrenzer zwischen Feldern benutzt. Statt dessen werden Spaltenwerte geschrieben und gelesen, indem die Anzeigebreite der Spalten benutzt wird. Wenn eine Spalte zum Beispiel als `INT(7)` deklariert ist, werden Werte für die Spalte mit 7-Zeichen-Feldern geschrieben. Bei der Eingabe werden Werte für die Spalte mit 7-Zeichen-Feldern bezogen. Festzeilenformate beeinflussen auch die Handhabung von `NULL`-Werten (siehe unten). Beachten Sie, dass Festgrößenformate nicht funktionieren, wenn Sie einen Multi-Byte-Zeichensatz benutzen.

Die Handhabung von `NULL`-Werten variiert in Abhängigkeit von den `FIELDS`- und `LINES`-Optionen, die Sie benutzen:

- Bei den vorgabemäßigen `FIELDS`- und `LINES`-Werten wird `NULL` für die Ausgabe als `\N` geschrieben und `\N` als `NULL` für die Eingabe gelesen (unter der Annahme, dass das `ESCAPED BY`-Zeichen '`\`' ist).
- Wenn `FIELDS ENCLOSED BY` nicht leer ist, wird ein Feld, das das Literalwort `NULL` als seinen Wert enthält, als `NULL`-Wert gelesen (das weicht ab vom Wort `NULL`, begrenzt durch `FIELDS ENCLOSED BY`-Zeichen, was als die Zeichenkette `'NULL'` gelesen wird).
- Wenn `FIELDS ESCAPED BY` leer ist, wird `NULL` als das Wort `NULL` gelesen.
- Beim Festzeilenformat (was auftritt, wenn sowohl `FIELDS TERMINATED BY` als auch `FIELDS ENCLOSED BY` leer sind), wird `NULL` als leere Zeichenkette geschrieben. Beachten Sie, dass das dazu führt, dass `NULL`-Werte und leere Zeichenketten in der Tabelle nicht mehr unterscheidbar sind, wenn in die Datei geschrieben wird, weil sie beide als leere Zeichenketten geschrieben werden. Wenn Sie in der Lage sein müssen, diese zu unterscheiden, wenn Sie die Datei wieder einlesen, sollten Sie kein Festzeilenformat benutzen.

Einige Fälle werden von `LOAD DATA INFILE` nicht unterstützt:

- Festgrößenzeilen (`FIELDS TERMINATED BY` und `FIELDS ENCLOSED BY` sind beide leer) und `BLOB`- oder `TEXT`-Spalten.
- Wenn Sie ein Trennzeichen angeben, das dasselbe wie ein anderes ist oder einem anderen vorangestellt ist. `LOAD DATA INFILE` kann in diesem Fall die Eingabe nicht korrekt interpretieren. Folgende `FIELDS`-Klausel zum Beispiel würde Probleme bereiten:

```
FIELDS TERMINATED BY '"' ENCLOSED BY '"'
```

- Wenn `FIELDS ESCAPED BY` leer ist, führt ein Feldwert, der ein Vorkommen von `FIELDS ENCLOSED BY` oder `LINES TERMINATED BY` gefolgt vom `FIELDS TERMINATED BY`-Wert enthält, dazu, dass `LOAD DATA INFILE` mit dem Einlesen eines Feldes oder einer Zeile zu früh aufhört. Das passiert, weil `LOAD DATA INFILE` nicht korrekt festlegen kann, wo der Feld- oder Zeilenwert endet.

Das folgende Beispiel lädt alle Spalten der `personen`-Tabelle:

```
mysql> LOAD DATA INFILE 'personen.txt' INTO TABLE personen;
```

Es ist keine Felderliste angegeben, daher erwartet `LOAD DATA INFILE`, dass die Eingabefelder ein Feld für jede Tabellenspalte enthalten. Die Vorgabewerte für `FIELDS` und `LINES`-Werte werden benutzt.

Wenn Sie Daten nur in einige Tabellenspalten einladen wollen, geben Sie eine Felderliste an:

```
mysql> LOAD DATA INFILE 'personen.txt'
           INTO TABLE personen (spalte1,spalte2,...);
```

Eine Felderliste müssen Sie ausserdem angeben, wenn die Reihenfolge der Felder in der Eingabedatei von der Reihenfolge der Tabellenspalten abweicht. Ansonsten kann MySQL nicht feststellen, wie er Eingabefelder Tabellenspalten zuordnen soll.

Wenn eine Zeile zu wenige Felder hat, werden die Spalten, für die es kein Eingabefeld gibt, auf ihre Vorgabewerte gesetzt. Die Zuweisung von Vorgabewerten ist unter Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“ beschrieben.

Ein leerer Feldwert wird anders interpretiert als ein fehlender Feldwert:

- Bei Zeichenketten-Typen wird die Spalte auf die leere Zeichenkette gesetzt.
- Bei numerischen Typen wird die Spalte auf `0` gesetzt.
- Bei Datums- und Zeit-Typen wird die Spalte auf den entsprechenden ``0"-Wert für den Typ gesetzt. See Abschnitt 7.2.2, „Datums- und Zeit-Typen“.

Beachten Sie, dass das dieselben Werte sind, die sich ergeben, wenn Sie einer Zeichenkette explizit eine leere Zeichenkette zuweisen oder solches für einen DATE- oder TIME-Type in einem `INSERT`- oder `UPDATE`-Statement tun.

`TIMESTAMP`-Spalten werden nur dann auf das aktuelle Datum und die aktuelle Zeit gesetzt, wenn es einen `NULL`-Wert für die Spalte gibt oder (nur für die erste `TIMESTAMP`-Spalte) die `TIMESTAMP`-Spalte in der Felderliste ausgelassen ist, wenn eine Felderliste angegeben wird.

Wenn eine Eingabezeile zu viele Felder hat, werden die zusätzlichen Felder ignoriert und die Anzahl von Warnungen herauf gezählt.

`LOAD DATA INFILE` betrachtet alle Eingaben als Zeichenketten, daher können Sie für `ENUM` oder `SET`-Spalten keine numerischen Werte benutzen, wie Sie das bei `INSERT`-Statements tun können. Alle `ENUM`- und `SET`-Werte müssen als Zeichenketten angegeben werden!

Wenn Sie die C-API benutzen, können Sie Informationen über die Anfrage durch den Aufruf der API-Funktion `mysql_info()` erhalten, wenn die `LOAD DATA INFILE`-Anfrage beendet ist. Das Format der Informationszeichenkette sieht wie folgt aus:

```
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

Warnungen erfolgen unter denselben Umständen, als wenn Werte über das `INSERT`-Statement (see Abschnitt 7.4.3, „`HANDLER`-Syntax“) eingefügt werden, ausser dass `LOAD DATA INFILE` zusätzlich Warnungen erzeugt, wenn es zu wenige oder zu viele Felder in der Eingabezeile gibt. Die Warnungen werden nirgendwo gespeichert; die Anzahl von Warnungen kann daher nur als Anhaltspunkt dafür benutzt werden, ob alles gut ging. Wenn Sie Warnungen erhalten und genau wissen wollen, warum Sie diese erhalten, besteht eine Möglichkeit dafür darin, `SELECT ... INTO OUTFILE` in eine andere Datei zu benutzen und diese mit der Original-Eingabedatei zu vergleichen.

Wenn Sie wollen, dass `LOAD DATA` aus einer Pipe liest, können Sie folgenden Trick benutzen:

```
mkfifo /mysql/db/x/x
chmod 666 /mysql/db/x/x
cat < /dev/tcp/10.1.1.12/4711 > /nt/mysql/db/x/x
mysql -e "LOAD DATA INFILE 'x' INTO TABLE x" x
```

Wenn Sie eine MySQL-Version vor 3.23.25 benutzen, können Sie das nur mit `LOAD DATA LOCAL INFILE` durchführen.

Weitere Informationen über die Effizienz von `INSERT` versus `LOAD DATA INFILE` und Möglichkeiten, die Geschwindigkeit zu steigern, finden Sie unter `LOAD DATA INFILE`, See Abschnitt 6.2.8, „Geschwindigkeit von `INSERT`-Anfragen“.

7.5. Datendefinition: `CREATE`, `DROP`, `ALTER`

7.5.1. CREATE DATABASE-Syntax

```
CREATE DATABASE [IF NOT EXISTS] datenbank
```

CREATE DATABASE erzeugt eine Datenbank mit dem angegebenen Namen. Die Regeln für erlaubte Datenbanknamen finden Sie unter Abschnitt 7.1.2, „Datenbank-, Tabellen-, Index-, Spalten- und Alias-Namen“. Ein Fehler tritt auf, wenn die Datenbank bereits existiert und Sie IF NOT EXISTS nicht angeben.

Datenbanken sind in MySQL als Verzeichnisse implementiert, die Dateien enthalten, die den Tabellen in der Datenbank entsprechen. Weil es keine Tabellen in einer Datenbank gibt, wenn diese erstmalig erzeugt wird, erzeugt das CREATE DATABASE-Statement nur ein Verzeichnis unter dem MySQL-Daten-Verzeichnis.

Sie können auch mit mysqladmin Datenbanken erzeugen. See Abschnitt 5.8, „Clientseitige Skripte und Hilfsprogramme von MySQL“.

7.5.2. DROP DATABASE-Syntax

```
DROP DATABASE [IF EXISTS] datenbank
```

DROP DATABASE löscht alle Tabellen in der Datenbank und löscht die Datenbank. Wenn Sie ein DROP DATABASE auf eine symbolisch verknüpfte Datenbank ausführen, werden sowohl der Link als auch die Original-Datenbank gelöscht. **Seien Sie mit diesem Befehl sehr vorsichtig!**

DROP DATABASE gibt die Anzahl von Dateien zurück, die aus dem Datenbank-Verzeichnis entfernt wurden. Normalerweise ist das dreimal die Anzahl der Tabellen, weil normalerweise jede Tabelle einer .MYD-Datei, einer .MYI-Datei und einer .frm-Datei entspricht.

Der DROP DATABASE-Befehl entfernt aus dem angegebenen Datenbank-Verzeichnis alle Dateien mit folgenden Erweiterungen:

.BAK	.DAT	.HSH	.ISD
.ISM	.ISM	.MRG	.MYD
.MYI	.db	.frm	

Alle Unterverzeichnisse, die aus 2 Ziffern bestehen (RAID-Verzeichnisse), werden ebenfalls gelöscht.

Ab MySQL-Version 3.22 können Sie die Schlüsselwörter IF EXISTS benutzen, um eine Fehlermeldung zu vermeiden, die erscheint, wenn die Datenbank nicht existiert.

Sie können Datenbanken auch mit mysqladmin löschen. See Abschnitt 5.8, „Clientseitige Skripte und Hilfsprogramme von MySQL“.

7.5.3. CREATE TABLE-Syntax

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tabelle [(create_definition,...)]
[tabellen_optionen] [select_statement]

create_definition:
  spalten_name typ [NOT NULL | NULL] [DEFAULT vorgabe_wert] [AUTO_INCREMENT]
            [PRIMARY KEY] [referenz_definition]
  oder    PRIMARY KEY (index_spalten_name,...)
  oder    KEY [index_name] (index_spalten_name,...)
  oder    INDEX [index_name] (index_spalten_name,...)
  oder    UNIQUE [INDEX] [index_name] (index_spalten_name,...)
  oder    FULLTEXT [INDEX] [index_name] (index_spalten_name,...)
  oder    [CONSTRAINT symbol] FOREIGN KEY index_name (index_spalten_name,...)
            [referenz_definition]
  oder    CHECK (ausdruck)

typ:
        TINYINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    SMALLINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    MEDIUMINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    INT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    INTEGER[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    BIGINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    REAL[(laenge,dezimalstellen)] [UNSIGNED] [ZEROFILL]
  oder    DOUBLE[(laenge,dezimalstellen)] [UNSIGNED] [ZEROFILL]
  oder    FLOAT[(laenge,dezimalstellen)] [UNSIGNED] [ZEROFILL]
  oder    DECIMAL(laenge,dezimalstellen) [UNSIGNED] [ZEROFILL]
  oder    NUMERIC(laenge,dezimalstellen) [UNSIGNED] [ZEROFILL]
  oder    CHAR(laenge) [BINARY]
  oder    VARCHAR(laenge) [BINARY]
  oder    DATE
  oder    TIME
  oder    TIMESTAMP
  oder    DATETIME
```

```
  oder    TINYBLOB
  oder    BLOB
  oder    MEDIUMBLOB
  oder    LONGBLOB
  oder    TINYTEXT
  oder    TEXT
  oder    MEDIUMTEXT
  oder    LONGTEXT
  oder    ENUM(wert1,wert2,wert3,...)
  oder    SET(wert1,wert2,wert3,...)

index_spalten_name:
        spalten_name [(laenge)]

referenz_definition:
        REFERENCES tabelle [(index_spalten_name,...)]
                   [MATCH FULL | MATCH PARTIAL]
                   [ON DELETE referenz_option]
                   [ON UPDATE referenz_option]

referenz_option:
        RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

tabellen_optionen:
        TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM }
or AUTO_INCREMENT = #
or AVG_ROW_LENGTH = #
or CHECKSUM = {0 | 1}
or COMMENT = "string"
or MAX_ROWS = #
or MIN_ROWS = #
or PACK_KEYS = {0 | 1 | DEFAULT}
or PASSWORD = "string"
or DELAY_KEY_WRITE = {0 | 1}
or      ROW_FORMAT= { default | dynamic | fixed | compressed }
or RAID_TYPE= {1 | STRIPED | RAID0 } RAID_CHUNKS=#  RAID_CHUNKSIZE=#
or UNION = (tabelle,[tabelle...])
or INSERT_METHOD= {NO | FIRST | LAST }
or      DATA directory="verzeichnis"
or      INDEX directory="verzeichnis"

select_statement:
        [IGNORE | REPLACE] SELECT ...  (jedes zulässige SELECT-Statement)
```

`CREATE TABLE` erzeugt eine Tabelle mit dem angegebenen Namen in der aktuellen Datenbank. Die Regeln für erlaubte Tabellennamen finden Sie unter Abschnitt 7.1.2, „Datenbank-, Tabellen-, Index-, Spalten- und Alias-Namen“. Ein Fehler tritt auf, wenn es keine aktuelle Datenbank gibt oder wenn die Tabelle bereits existiert.

Ab MySQL-Version 3.22 kann der Tabellenname als `datenbank.tabelle` angegeben werden. Das funktioniert unabhängig davon, ob es eine aktuelle Datenbank gibt oder nicht.

In MySQL-Version 3.23 können Sie das `TEMPORARY`-Schlüsselwort benutzen, wenn Sie eine Tabelle erzeugen. Eine temporäre Tabelle wird automatisch gelöscht, wenn eine Verbindung stirbt und der Name sich auf die Verbindung bezieht. Das bedeutet, dass zwei verschiedene Verbindungen beide denselben temporären Tabellenname benutzen können, oder miteinander oder einer bestehenden Tabelle gleichen Namens in Konflikt zu geraten. (Die bestehende Tabelle ist versteckt, bis die temporäre Tabelle gelöscht wird.)

Ab MySQL-Version 3.23 können Sie die Schlüsselwörter `IF NOT EXISTS` benutzen, so dass kein Fehler auftritt, wenn die Tabelle bereits besteht. Beachten Sie, dass keine Überprüfung erfolgt, dass die Tabellenstrukturen identisch sind.

Jede Tabelle `tabelle` wird durch einige Dateien im Datenbank-Verzeichnis dargestellt. Im Falle von MyISAM-Tabellen erhalten Sie:

Datei	Zweck
`tabelle.frm`	Tabellendefinitionsdatei (form)
`tabelle.MYD`	Daten-Datei
`tabelle.MYI`	Index-Datei

Weitere Information über die Eigenschaften der verschiedenen Spaltentypen finden Sie unter Abschnitt 7.2, „Spaltentypen“:

- Wenn weder `NULL` noch `NOT NULL` angegeben ist, wird die Spalte behandelt, als wenn `NULL` angegeben worden wäre.
- Eine Ganzzahl-Spalte kann das zusätzliche Attribut `AUTO_INCREMENT` haben. Wenn Sie einen Wert von `NULL` (empfohlen) oder `0` in eine `AUTO_INCREMENT`-Spalte einfügen, wird die Spalte auf `wert+1` gesetzt, wobei `wert` der größte momentan in der Tabelle vorhandene Spaltenwert ist. `AUTO_INCREMENT`-Folgen fangen mit `1` an. See Abschnitt 9.4.3.30, „`mysql_insert_id()`“.

  Wenn Sie eine Zeile löschen, die den höchsten Wert einer `AUTO_INCREMENT`-Spalte enthält, wird der Wert bei einer `ISAM`-

oder BDB-Tabelle wieder verwendet, nicht aber bei einer MyISAM- oder InnoDB-Tabelle. Wenn Sie alle Zeilen in der Tabelle mit DELETE FROM tabelle (ohne ein WHERE) im AUTOCOMMIT-Modus löschen, fängt die Folge bei allen Tabellentypen von Neuem an.

HINWEIS: Es darf nur eine AUTO_INCREMENT-Spalte pro Tabelle geben und diese muss indiziert sein. MySQL-Version 3.23 funktioniert darüber hinaus nur korrekt, wenn die AUTO_INCREMENT-Spalte nur positive Werte hat. Das Einfügen einer negativen Zahl wird als Einfügen einer sehr großen positiven Zahl betrachtet. Damit werden Genauigkeitsprobleme vermieden, wenn Zahlen vom positiven zum negativen Bereich übergehen. Ausserdem wird sichergestellt, dass man nicht versehentlich eine AUTO_INCREMENT-Spalte erhält, die 0 enthält.

Um MySQL kompatibel mit einigen ODBC-Applikationen zu machen, können Sie die letzte eingefügte Zeile mit folgender Anfrage finden:

```
SELECT * FROM tabelle WHERE auto_spalte IS NULL
```

- NULL-Werte werden bei TIMESTAMP-Spalten anders als bei anderen Spaltentypen gehandhabt. Sie können NULL nicht wortgetreu in einer TIMESTAMP-Spalte speichern: Wenn Sie die Spalte auf NULL setzen, wird sie auf das aktuelle Datum und die aktuelle Zeit gesetzt. Weil TIMESTAMP-Spalten sich auf diese Art verhalten, treffen die NULL- und NOT NULL-Attribute nicht auf normale Art zu und werden ignoriert, wenn Sie sie angeben.

  Auf der anderen Seite berichtet der Server, um es für MySQL-Clients leichter zu machen, TIMESTAMP-Spalten zu benutzen, dass solchen Spalten NULL-Werte zugewiesen werden können (was stimmt), obwohl TIMESTAMP nie wirklich einen NULL-Wert enthalten wird. Sie können das sehen, wenn Sie DESCRIBE tabelle eingeben, um eine Beschreibung Ihrer Tabelle zu erhalten.

  Beachten Sie, dass das Setzen einer TIMESTAMP-Spalte auf 0 nicht dasselbe ist wie das Setzen auf NULL, weil 0 ein gültiger TIMESTAMP-Wert ist.

- Wenn kein DEFAULT-Wert für eine Spalte angegeben wird, weist MySQL automatisch einen zu.

  Wenn die Spalte NULL als Wert aufnehmen darf, ist der Vorgabewert NULL.

  Wenn die Spalte als NOT NULL deklariert ist, hängt der Vorgabewert vom Spaltentyp ab:

  - Bei numerischen Typen ausser denen, die mit dem AUTO_INCREMENT-Attribut deklariert wurden, ist der Vorgabewert 0. Bei einer AUTO_INCREMENT-Spalte ist der Vorgabewert der nächste Wert in der Folge.
  - Bei Datums- und Zeit-Typen ausser TIMESTAMP ist der Vorgabewert der entsprechende 0-Wert für den Typ. Bei der ersten TIMESTAMP-Spalte einer Tabelle ist der Vorgabewert das aktuelle Datum und die aktuelle Zeit. See Abschnitt 7.2.2, „Datums- und Zeit-Typen“.
  - Bei Zeichenketten-Typen ausser ENUM ist der Vorgabewert die leere Zeichenkette. Bei ENUM ist der Vorgabewert der erste Aufzählungswert.

  Vorgabewerte müssen Konstanten sein. Das heißt zum Beispiel, dass Sie den Vorgabewert einer DATE-Spalte nicht als Wert einer Funktion wie NOW() oder CURRENT_DATE setzen können.

- KEY ist ein Synonym für INDEX.

- In MySQL darf ein UNIQUE-Schlüssel nur unterschiedliche Werte haben. Ein Fehler tritt auf, wenn Sie versuchen, eine neue Zeile hinzuzufügen, der Schlüsselwert dem einer bestehenden Zeile entspricht.

- Ein PRIMARY KEY ist ein eindeutiger KEY mit der zusätzlichen Beschränkung, dass alle Schlüsselspalten als NOT NULL deklariert sein müssen. In MySQL wird der Schlüssel PRIMARY genannt. Eine Tabelle darf nur einen PRIMARY KEY haben. Wenn Sie keinen PRIMARY KEY haben und irgend welche Applikationen nach einem PRIMARY KEY in Ihrer Tabelle verlangen, gibt MySQL den ersten UNIQUE-Schlüssel, der keinerlei NULL-Spalten hat, als PRIMARY KEY zurück.

- Ein PRIMARY KEY kann ein mehrspaltiger Index sein. Sie können jedoch keinen mehrspaltiger Index mit dem PRIMARY KEY-Schlüsselattribut in einer Spaltenspezifikation erzeugen. Wenn Sie das tun, wird nur die erste Spalte als PRIMARY gekennzeichnet. Sie müssen die PRIMARY KEY(index_spalten_name, ...)-Syntax benutzen.

- Wenn der PRIMARY- oder UNIQUE-Schlüssel aus nur einer Spalte besteht und diese vom Typ Ganzzahl ist, können Sie auf sie auch als _rowid verweisen (neu ab Version 3.23.11).

- Wenn Sie einem Index keinen Namen zuweisen, wird dem Index derselbe Name zugewiesen wie der erste index_spalten_name, mit einem optionalen Suffix (_2, _3, ...), um ihn eindeutig zu machen. Sie können die Indexnamen für eine Tabelle mit SHOW INDEX FROM tabelle anzeigen. See Abschnitt 5.5.5, „SHOW-Syntax“.

- Nur der MyISAM-Tabellentyp unterstützt Indexe auf Spalten, die NULL-Werte enthalten können. In anderen Fällen müssen Sie solche Spalten als NOT NULL deklarieren, sonst tritt ein Fehler auf.

- Mit der `spalten_name(laenge)`-Syntax können Sie einen Index festlegen, der nur einen Teil einer `CHAR`- oder `VARCHAR`-Spalte enthält. Das kann die Index-Datei viel kleiner machen. See Abschnitt 6.4.4, „Spalten-Indexe“.

- Nur der `MyISAM`-Tabellentyp unterstützt Indexierung auf `BLOB`- und `TEXT`-Spalten. Wenn Sie einen Index auf eine `BLOB`- oder `TEXT`-Spalte setzen, MÜSSEN Sie immer die Länge des Indexes angeben:

```
CREATE TABLE test (blob_spalte BLOB, index(blob_spalte(10)));
```

- Wenn Sie `ORDER BY` oder `GROUP BY` bei einer `TEXT`- oder `BLOB`-Spalte benutzen, werden nur die ersten `max_sort_length` Bytes benutzt. See Abschnitt 7.2.3.2, „Die `BLOB`- und `TEXT`-Typen“.

- Ab MySQL-Version 3.23.23 können Sie auch spezielle **FULLTEXT**-Indexe erzeugen, Diese werden für Volltextsuche benutzt. Nur der `MyISAM`-Tabellentyp unterstützt `FULLTEXT`-Indexe. Sie können auf `VARCHAR`- und `TEXT`-Spalten erzeugt werden. Die Indexierung erfolgt immer über die gesamte Spalte, teilweise Indexierung wird nicht unterstützt. Siehe Abschnitt 7.8, „MySQL-Volltextsuche“ für Details zur Funktionsweise.

- Die `FOREIGN KEY`-, `CHECK`- und `REFERENCES`-Klauseln tun momentan noch nichts. Die Syntax wird nur aus Gründen der Kompatibilität bereit gestellt, um das Portieren von Code von anderen SQL-Servern zu erleichtern und um Applikationen laufen zu lassen, die Tabellen mit Referenzen erzeugen.

  See Abschnitt 2.7.4.5, „Fremdschlüssel“.

- Jede `NULL`-Spalte benötigt ein zusätzliches Bit, gerundet auf das nächste Byte.

- Die maximale Datensatzlänge in Bytes kann wie folgt berechnet werden:

```
Zeilenlänge = 1
              + (Summe Spaltenlängen)
              + (Anzahl von NULL-Spalten + 7)/8
              + (Anzahl von Spalten variabler Länge)
```

- Die `tabellen_optionen`- und `SELECT`-Optionen sind implementiert ab MySQL-Version 3.23.

  Die unterschiedlichen Tabellentypen sind:

BDB oder Berkeley_db	Transaktionssichere Tabellen mit Seitensperren (Page Locking). See Abschnitt 8.6, „BDB- oder Berkeley_db-Tabellen“.
HEAP	Die Daten dieser Tabelle werden nur im Arbeitsspeicher gehalten. See Abschnitt 8.4, „HEAP-Tabellen“.
ISAM	Der Original-Tabellen-Handler. See Abschnitt 8.3, „ISAM-Tabellen“.
InnoDB	Transaktionssichere Tabellen mit Zeilensperren. See Abschnitt 8.5, „InnoDB-Tabellen“.
MERGE	Eine Sammlung von MyISAM-Tabellen, die als eine Tabelle benutzt werden. See Abschnitt 8.2, „MERGE-Tabellen“.
MRG_MERGE	Ein Alias für MERGE-Tabellen.
MyISAM	Der neue binäre portable Tabellen-Handler, der ISAM ersetzt. See Abschnitt 8.1, „MyISAM-Tabellen“.

  See Kapitel 8, *MySQL-Tabellentypen*.

  Wenn ein Tabellentyp angegeben wird und dieser besondere Typ nicht verfügbar ist, wählt MySQL den Tabellentyp, der dem angegebenen am nächsten kommt. Wenn beispielsweise `TYPE=BDB` angegeben wird und die Distribution von MySQL keine `BDB`-Tabellen unterstützt, wird die Tabelle statt dessen als `MyISAM` erzeugt.

  Die anderen Tabellenoptionen werden benutzt, um das Verhalten der Tabelle zu optimieren. In den meisten Fällen müssen Sie keine davon angeben. Die Optionen funktionieren bei allen Tabellentypen, falls nicht anders angegeben:

`AUTO_INCREMENT`	Der nächste auto_increment-Wert, den Sie für Ihre Tabelle setzen wollen (MyISAM).
`AVG_ROW_LENGTH`	Näherungsweise die durchschnittliche Zeilenlänge Ihrer Tabelle. Diese Option müssen Sie nur für große Tabellen mit unterschiedlich großen Datensätzen setzen.
`CHECKSUM`	Setzen Sie diesen Wert auf 1, wenn Sie wollen, dass MySQL eine Prüfsumme für alle Zeilen unterhält (macht die Tabelle ein bisschen langsamer bei der Aktualisierung, aber macht es einfacher, beschädigte Tabellen zu finden) (MyISAM).
`COMMENT`	Ein 60-Zeichen-Kommentar für Ihre Tabelle.
`MAX_ROWS`	Maximale Anzahl von Zeilen, die Sie in Ihrer Tabelle zu speichern planen.
`MIN_ROWS`	Minimale Anzahl von Zeilen, die Sie in Ihrer Tabelle zu speichern planen.
`PACK_KEYS`	Setzen Sie diesen Wert auf 1, wenn Sie einen kleineren Index erhalten wollen. Das macht

	Aktualisierungen üblicherweise langsamer und liest schneller (MyISAM, ISAM). Setzen auf 0 schaltet die Komprimierung von Schlüsseln ab. Setzen auf DEFAULT (MySQL 4.0) weist die Tabellen-Handler an, nur lange CHAR- / VARCHAR-Spalten zu packen.
PASSWORD	Verschlüsselt die .frm-Datei mit einem Passwort. Diese Option tut nichts in der Standard-MySQL-Version.
DELAY_KEY_WRITE	Setzen Sie diesen Wert auf 1, wenn Sie Schlüssel-Tabellen-Aktualisierungen verzögern wollen, bis die Tabelle geschlossen wird (MyISAM).
ROW_FORMAT	Definiert, wie die Zeilen gespeichert werden sollen. Momentan funktioniert diese Option nur bei MyISAM-Tabellen, die die DYNAMIC- und FIXED-Zeilenformate unterstützen. See Abschnitt 8.1.2, „MyISAM-Tabellenformate“.

Wenn Sie eine MyISAM-Tabelle benutzen, verwendet MySQL das Produkt aus max_rows * avg_row_length um zu entscheiden, wie Groß die resultierende Tabelle sein wird. Wenn Sie keine der obigen Optionen angeben, ist die maximale Größe für eine Tabelle 4 GB (oder 2 GB, wenn Ihr Betriebssystem nur 2 GB-Tabellen unterstützt). Das geschieht, um Zeigergrößen gering zu halten und um den Index kleiner und schneller zu machen, wenn Sie nicht wirklich große Dateien benötigen.

Wenn Sie PACK_KEYS nicht benutzen, ist die Vorgabe, nur Zeichenketten zu komprimieren, nicht Zahlen. Wenn Sie PACK_KEYS=1 benutzen, werden auch Zahlen komprimiert.

Wenn Sie binäre Zahlschlüssel komprimieren, benutzt MySQL die Präfix-Komprimierung. Das bedeutet, dass Sie nur dann einen Nutzen daraus ziehen, wenn Sie Zahlen haben, die sich oft wiederholen. Präfix-Kompression bedeutet, das jeder Schlüssel ein zusätzliches Byte benötigt, um darzustellen, wie viele Bytes des vorherigen Schlüssels für den nächsten Schlüssel dieselben sind (beachten Sie, dass der Zeiger auf die Zeile in der Reihenfolge 'hohes Byte zuerst' direkt nach dem Schlüssel gespeichert wird, um die Kompression zu verbessern). Das heißt, wenn Sie viele gleiche Schlüssel auf zwei Zeilen hintereinander haben, werden alle folgenden 'gleichen' Schlüssel üblicherweise nur 2 Bytes in Anspruch nehmen (inklusive dem Zeiger auf die Zeile). Vergleichen Sie das mit dem Normalfall, bei dem die folgenden Schlüssel speicher_platz_fuer_schlüssel + zeiger_groesse beanspruchen (üblicherweise 4). Auf der anderen Seite verlieren Sie 1 Byte pro Schlüssel, wenn alle Schlüssel völlig unterschiedlich sind, falls der Schlüssel kein Schlüssel ist, der NULL-Werte haben kann (in diesem Fall wird die komprimierte Schlüssellänge, die im selben Byte gespeichert ist, benutzt, um zu kennzeichnen, ob ein Schlüssel NULL ist).

- Wenn Sie ein SELECT nach dem CREATE-Statement angeben, erzeugt MySQL neue Felder für alle Elemente im SELECT. Beispiel:

```
mysql> CREATE TABLE test (a int not null auto_increment,
           primary key (a), key(b))
           TYPE=MyISAM SELECT b,c from test2;
```

Das erzeugt eine MyISAM-Tabelle mit drei Spalten a, b und c. Beachten Sie, dass die Spalten des SELECT-Statements an die rechte Seite der Tabelle angehängt werden, nicht überlappend. Nehmen wir folgendes Beispiel:

```
mysql> select * from foo;
+---+
| n |
+---+
| 1 |
+---+

mysql> create table bar (m int) select n from foo;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> select * from bar;
+------+---+
| m    | n |
+------+---+
| NULL | 1 |
+------+---+
1 row in set (0.00 sec)
```

Für jede Zeile in Tabelle foo wird eine Zeile in bar mit den Werten von foo und Vorgabewerten für die neuen Spalten eingefügt.

CREATE TABLE ... SELECT erzeugt nicht automatisch irgend welche Indexe. Das wird absichtlich gemacht, um den Befehl so flexibel wie möglich zu machen. Wenn Sie Indexe in der erzeugten Tabelle haben wollen, geben Sie diese vor dem SELECT-Statement an:

```
mysql> create table bar (unique (n)) select n von foo;
```

Wenn Fehler beim Kopieren der Daten in die Tabelle auftreten, wird diese automatisch gelöscht.

Um sicherzustellen, dass die Update-Log-Datei/Binär-Log-Datei benutzt werden kann, um die Original-Tabellen neu zu erzeugen, läßt MySQL keine gleichzeitigen Einfügeoperationen während `CREATE TABLE .... SELECT` zu.

- Die `RAID_TYPE`-Option hilft, die 2 GB- / 4 GB-Grenze für die MyISAM-Daten-Datei zu durchbrechen (nicht für die Index-Datei), auf Betriebssystemen, die keine großen Dateien unterstützen. Sie erzielen mehr Geschwindigkeit vom I/O-Flaschenhals, wenn Sie die `RAID`-Verzeichnisse auf unterschiedliche physikalische Platten legen. `RAID_TYPE` funktioniert auf jedem Betriebssystem, solange Sie MySQL mit `--with-raid` konfiguriert haben. Momentan ist der einzige zulässige `RAID_TYPE` `STRIPED` (`1` und `RAID0` sind Aliase dafür).

  Wenn Sie `RAID_TYPE=STRIPED` bei einer `MyISAM`-Tabelle angeben, erzeugt `MyISAM` `RAID_CHUNKS`-Unterverzeichnisse namens 00, 01, 02 im Datenbank-Verzeichnis. In jedem dieser Verzeichnisse erzeugt `MyISAM` eine `tabelle.MYD`. Wenn Sie Daten in die Daten-Datei schreiben, mappt der `RAID`-Handler die ersten `RAID_CHUNKSIZE` * 1024 Bytes auf die erste Datei, die nächsten `RAID_CHUNKSIZE` * 1024 Bytes auf die nächste Datei usw.

- `UNION` wird benutzt, wenn Sie eine Sammlung identischer Tabelle als eine benutzen wollen. Das funktioniert nur bei MERGE-Tabellen. See Abschnitt 8.2, „MERGE-Tabellen“.

  Momentan benötigen Sie `SELECT`-, `UPDATE`- und-`DELETE`-Berechtigungen auf die Tabellen, die Sie auf eine `MERGE`-Tabelle mappen. Alle gemappten Tabellen müssen sich in derselben Datenbank wie die `MERGE`-Tabelle befinden.

- Wenn Sie Daten in eine `MERGE`-Tabelle einfügen wollen, müssen Sie mit `INSERT_METHOD` angeben, in welche Tabelle die Zeile eingefügt werden soll. See Abschnitt 8.2, „MERGE-Tabellen“.

- In der erzeugten Tabelle wird der `PRIMARY`-Schlüssel zuerst platziert, gefolgt von allen `UNIQUE`-Schlüsseln und danach von den normalen Schlüsseln. Das hilft dem MySQL-Optimierer zu priorisieren, welcher Schlüssel benutzt werden soll, und auch, Duplikate von `UNIQUE`-Schlüsseln zu entdecken.

- Wenn Sie `DATA directory="verzeichnis"` oder `INDEX directory="verzeichnis"` benutzen, können Sie angeben, wohin die Tabellen-Handler ihre Tabellen- und Index-Dateien legen sollen. Das funktioniert nur bei `MyISAM`-Tabellen in `MySQL` 4.0, wenn Sie die `--skip-symlink`-Option nicht benutzen. See Abschnitt 6.6.1.2, „Benutzung symbolischer Links für Tabellen“.

7.5.3.1. Stille Spaltentyp-Änderungen

In einigen Fällen ändert MySQL lautlos eine Spaltenspezifikation von der, die in einem `CREATE TABLE`-Statement angegeben wurde. (Das kann auch bei `ALTER TABLE` passieren.):

- `VARCHAR`-Spalten mit einer Länge kleiner 4 werden in `CHAR` geändert.

- Wenn irgend eine Spalte in einer Tabelle eine variable Länge hat, hat im Ergebnis jede Zeile eine variable Länge. Wenn daher eine Tabelle irgend welche Spalten variabler Länge enthält (`VARCHAR`, `TEXT` oder `BLOB`), werden alle `CHAR`-Spalten, die länger als drei Zeichen sind, in `VARCHAR`-Spalten umgewandelt. Das beeinflusst die Benutzung dieser Spalten in keiner Weise, denn in MySQL ist `VARCHAR` nur eine andere Art, Zeichen zu speichern. MySQL führt diese Umwandlung durch, weil sie Platz spart und Tabellenoperationen schneller macht. See Kapitel 8, *MySQL-Tabellentypen*.

- `TIMESTAMP`-Anzeigebreiten müssen geradzahlig und im Bereich von 2 bis 14 sein. Wenn Sie eine Anzeigebreite von 0 oder größer als 14 angeben, wird die Größe auf 14 gesetzt. Ungerade Werte im Bereich von 1 bis 13 werden auf den nächst höheren geraden Wert gesetzt.

- Sie können keinen echten `NULL`-Wert in einer `TIMESTAMP`-Spalte speichern. Wenn Sie sie auf `NULL` setzen, wird sie auf das aktuelle Datum und die aktuelle Zeit gesetzt. Weil sich `TIMESTAMP`-Spalten so verhalten, treffen die Attribute `NULL` und `NOT NULL` nicht auf normale Weise zu und werden ignoriert, wenn Sie sie angeben. `DESCRIBE tabelle` zeigt dagegen immer an, dass einer `TIMESTAMP`-Spalte `NULL`-Werte zugewiesen werden können.

- MySQL mappt bestimmte Spaltentypen, die von anderen SQL-Datenbank-Herstellern benutzt werden, auf MySQL-Typen. See Abschnitt 7.2.5, „Spaltentypen anderer Datenbanken benutzen“.

Wenn Sie sehen wollen, ob MySQL einen anderen Spaltentyp als den, den Sie angegeben haben, benutzt hat, geben Sie nach dem Erzeugen oder Ändern Ihrer Tabelle ein `DESCRIBE tabelle`-Statement ein.

Bestimmte andere Spaltentyp-Änderungen können auftreten, wenn Sie eine Tabelle mit `myisampack` komprimieren. See Abschnitt 8.1.2.3, „Kennzeichen komprimierter Tabellen“.

7.5.4. `ALTER TABLE`-Syntax

```
ALTER [IGNORE] TABLE tabelle aenderungs_angabe [, aenderungs_angabe ...]
```

```
aenderungs_angabe:
        ADD [COLUMN] create_definition [FIRST | AFTER spalten_name]
  oder    ADD [COLUMN] (create_definition, create_definition,...)
  oder    ADD INDEX [index_name] (index_spalten_name,...)
  oder    ADD PRIMARY KEY (index_spalten_name,...)
  oder    ADD UNIQUE [index_name] (index_spalten_name,...)
  oder    ADD FULLTEXT [index_name] (index_spalten_name,...)
  or ADD [CONSTRAINT symbol] FOREIGN KEY index_name (index_spalten_name,...)
             [referenz_definition]
  oder    ALTER [COLUMN] spalten_name {SET DEFAULT literal | DROP DEFAULT}
  oder    CHANGE [COLUMN] alter_spalten_name create_definition
  oder    MODIFY [COLUMN] create_definition
  oder    DROP [COLUMN] spalten_name
  oder    DROP PRIMARY KEY
  oder    DROP INDEX index_name
  oder    DISABLE KEYS
  oder    ENABLE KEYS
  oder    RENAME [TO] neue_tabelle
  oder    ORDER BY spalte
  oder    tabellen_optionen
```

Mit `ALTER TABLE` können Sie die Struktur einer bestehenden Tabelle ändern. Sie können beispielsweise Spalten hinzufügen oder löschen, Indexe erzeugen oder löschen, den Typ bestehender Spalten ändern oder Spalten oder die Tabelle selbst umbenennen. Sie können auch den Kommentar für die Tabelle und den Typ der Tabelle ändern. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.

Wenn Sie `ALTER TABLE` benutzen, um eine Spaltenspezifikation zu ändern, und `DESCRIBE tabelle` anzeigt, dass die Spalte nicht geändert wurde, ist es möglich, dass MySQL Ihre Änderungen aus einem der Gründe ignoriert hat, die in Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“ beschrieben sind. Wenn Sie beispielsweise versuchen, eine `VARCHAR`-Spalte zu `CHAR` zu ändern, benutzt MySQL dennoch `VARCHAR`, wenn die Tabelle weitere Spalten variabler Länge enthält.

`ALTER TABLE` funktioniert mittels Anlegen einer temporären Kopie der Original-Tabelle. Die Änderungen werden an der Kopie durchgeführt, dann wird die Original-Tabelle gelöscht und die neue umbenannt. Das wird so durchgeführt, dass alle Aktualisierungen automatisch ohne irgend welche fehlgeschlagenen Aktualisierungen an die neue Tabelle weitergeleitet werden. Während `ALTER TABLE` ausgeführt wird, ist die alte Tabelle durch andere Clients lesbar. Aktualisierungen und Schreibvorgänge in die Tabelle werden angehalten, bis die neue Tabelle bereit ist.

Beachten Sie, dass MySQL immer eine temporäre Tabelle anlegt, wenn Sie für `ALTER TABLE` irgend eine Option ausser `RENAME` angeben, selbst wenn die Daten eigentlich nicht kopiert werden müssten (zum Beispiel, wenn Sie einen Spaltennamen ändern). Wir planen, dass zu beheben, aber da man `ALTER TABLE` normalerweise nicht ausführen muss, ist das auf unserer TODO-Liste nicht sehr hoch angesetzt.

- Um `ALTER TABLE` ausführen zu können, benötigen Sie **ALTER**-, **INSERT**- und **CREATE**-Berechtigungen für die Tabelle.
- `IGNORE` ist eine MySQL-Erweiterung zu ANSI-SQL92. Es steuert, wie `ALTER TABLE` funktioniert, wenn es in der neuen Tabelle Duplikate auf eindeutigen Schlüsseln gibt. Wenn `IGNORE` nicht angegeben wird, wird das Kopieren abgebrochen und zurückgesetzt. Wenn `IGNORE` angegeben wird, wird bei Zeilen mit Duplikaten auf einem eindeutigen Schlüssel nur die erste Zeile benutzt, die anderen werden gelöscht.
- Sie können mehrfache `ADD`-, `ALTER`-, `DROP`- und `CHANGE`-Klauseln in einem einzigen `ALTER TABLE`-Statement angeben. Das ist eine MySQL-Erweiterung zu ANSI-SQL92, welches nur eine Klausel pro `ALTER TABLE`-Statement zuläßt.
- `CHANGE spalten_name`, `DROP spalten_name` und `DROP INDEX` sind MySQL-Erweiterungen zu ANSI-SQL92.
- `MODIFY` ist eine Oracle-Erweiterung zu `ALTER TABLE`.
- Das optionale Wort `COLUMN` kann weggelassen werden.
- Wenn Sie `ALTER TABLE tabelle RENAME TO neuer_name` ohne weitere Optionen benutzen, benennt MySQL einfach die Dateien um, die der Tabelle `tabelle` entsprechen. Es besteht keine Notwendigkeit, die temporäre Tabelle zu erzeugen. See Abschnitt 7.5.5, „`RENAME TABLE`-Syntax“.
- Ab **MySQL 4.0** kann das obige Feature explizit aktiviert werden. `ALTER TABLE ... DISABLE KEYS` veranlasst MySQL, mit dem Aktualisieren nicht eindeutiger Indexe für die `MyISAM`-Tabelle aufzuhören. Dann sollte `ALTER TABLE ... ENABLE KEYS` benutzt werden, um fehlende Indexe wieder zu erzeugen. Weil MySQL das mit Algorithmen durchführt, die viel schneller sind als das Einfügen von Schlüsseln nacheinander, kann das Abschalten von Schlüsseln bei Masseneinfügeoperationen erheblich Geschwindigkeitsvorteile bringen.
- `create_definition`-Klauseln benutzen dieselbe Syntax für `ADD` und `CHANGE` wie bei `CREATE TABLE`. Beachten Sie, dass diese Syntax den Spaltenname beinhaltet, nicht nur den Spaltentyp.

  See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.
- Sie können eine Spalte mit einer `CHANGE alter_spalten_name create_definition`-Klausel umbenennen. Um das zu tun, geben Sie den alten und den neuen Spaltennamen und den Typ an, den die Spalte momentan hat. Um beispielsweise

eine INTEGER-Spalte von a nach b umzubenennen, tun Sie folgendes:

```
mysql> ALTER TABLE t1 CHANGE a b INTEGER;
```

Wenn Sie einen Spaltentyp, nicht aber den Namen ändern wollen, benötigt CHANGE dennoch zwei Spaltennamen, selbst wenn sie dieselben sind. Beispiel:

```
mysql> ALTER TABLE t1 CHANGE b b BIGINT NOT NULL;
```

Ab MySQL-Version 3.22.16a können Sie jedoch auch MODIFY benutzen, um einen Spaltentyp ohne Umbenennung zu ändern:

```
mysql> ALTER TABLE t1 MODIFY b BIGINT NOT NULL;
```

- Wenn Sie CHANGE oder MODIFY benutzen, um eine Spalte zu kürzen, für die es einen Index auf einem Teil der Spalte gibt (wenn Sie zum Beispiel einen Index auf den ersten 10 Zeichen einer VARCHAR-Spalte haben), können Sie die Spalte nicht kürzer als die Anzahl von Zeichen machen, die indiziert sind.
- Wenn Sie versuchen, einen Spaltentyp mit CHANGE oder MODIFY zu ändern, versucht MySQL, Daten so umzuwandeln, dass sie so gut wie möglich zum neuen Typ passen.
- AB MySQL-Version 3.22 können Sie FIRST oder ADD ... AFTER spalten_name benutzen, um eine Spalte an einer bestimmten Position innerhalb einer Tabellenzeile einzufügen. Vorgabemäßig wird die Spalte am Ende hinzugefügt.
- ALTER COLUMN gibt einen Vorgabewert für eine Spalte an oder entfernt den alten Vorgabewert. Wenn der alte Vorgabewert entfernt wird und die Spalte NULL sein darf, ist der neue Vorgabewert NULL. Wenn die Spalte nicht NULL sein darf, weist MySQL einen Vorgabewert zu, wie in Abschnitt 7.5.3, „CREATE TABLE-Syntax“ beschrieben.
- DROP INDEX entfernt einen Index. Das ist eine MySQL-Erweiterung zu ANSI-SQL92. See Abschnitt 7.5.8, „DROP INDEX-Syntax“.
- Wenn Spalten aus einer Tabelle gelöscht werden, werden sie auch aus jeglichen Indexen entfernt, deren Teil sie sind. Wenn alle Spalten, aus denen ein Index besteht, gelöscht werden, wird der Index ebenfalls gelöscht.
- Wenn eine Tabelle nur eine Spalte enthält, kann die Spalte nicht gelöscht werden. Wenn Sie beabsichtigen, die Tabelle zu entfernen, benutzen Sie statt dessen DROP TABLE.
- DROP PRIMARY KEY löscht den Primärschlüssel. Wenn es keinen solchen gibt, löscht es den ersten UNIQUE-Index in der Tabelle. (MySQL kennzeichnet den ersten UNIQUE-Schlüssel als PRIMARY KEY, wenn PRIMARY KEY nicht explizit angegeben wurde.)

  Wenn Sie einen UNIQUE INDEX oder PRIMARY KEY zu einer Tabelle hinzufügen, wird dieser vor jedem Nicht-UNIQUE-Index gespeichert, so dass MySQL doppelte Schlüsseleinträge so früh wie möglich feststellen kann.
- ORDER BY gestattet Ihnen, eine Tabelle mit Zeilen in einer bestimmten Reihenfolge zu erzeugen. Beachten Sie, dass die Tabelle nach INSERTs und DELETEs nicht in dieser Reihenfolge verbleibt. In einigen Fällen kann es das Sortieren für MySQL erleichtern, wenn die Tabelle nach der Spalte geordnet ist, nach der Sie sie später ordnen wollen. Diese Option ist hauptsächlich nützlich, wenn Sie wissen, dass Sie die Zeilen meistens in einer bestimmten Reihenfolge abfragen werden. Wenn Sie diese Option nach großen Änderungen in der Tabelle benutzen, können Sie möglicherweise eine höhere Performance erzielen.
- Wenn Sie ALTER TABLE auf einer MyISAM-Tabelle benutzen, werden alle nicht eindeutigen Indexe in einem separaten Stapellauf erzeugt (wie bei REPAIR). Das sollte ALTER TABLE viel schneller machen, wenn Sie viele Indexe haben.
- Ab **MySQL 4.0** kann dies explizit aktiviert werden. ALTER TABLE ... DISABLE KEYS veranlasst MySQL, mit der Aktualisierung nicht eindeutiger Indexe für MyISAM-Tabellen aufzuhören. ALTER TABLE ... ENABLE KEYS sollte dann benutzt werden, um fehlende Indexe wieder zu erzeugen. Weil MySQL das mit Algorithmen durchführt, die viel schneller sind als das Einfügen von Schlüsseln nacheinander, kann das Abschalten von Schlüsseln bei Masseneinfügeoperationen erheblich Geschwindigkeitsvorteile bringen.
- Mit der C-API-Funktion mysql_info() können Sie herausfinden, wie viele Datensätze kopiert wurden und (wenn IGNORE benutzt wird) wie viele Datensätze aufgrund der Duplizierung eindeutiger Schlüsselwerte gelöscht wurden.
- Die FOREIGN KEY-, CHECK- und REFERENCES-Klauseln machen nichts. Die Syntax für sie steht nur aus Kompatibilitätsgründen bereit, um das Portieren von Code von anderen SQL-Servern zu erleichtern und um Applikationen laufen zu lassen, die Tabellen mit Referenzen erzeugen.

  See Abschnitt 2.7.4.5, „Fremdschlüssel“.

Hier ist ein Beispiel, das einige der Anwendungsfälle von ALTER TABLE zeigt. Wir fangen mit einer Tabelle t1 an, die wie folgt erzeugt wird:

```
mysql> CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

Um die Tabelle von `t1` nach `t2` umzubenennen, geben Sie ein:

```
mysql> ALTER TABLE t1 RENAME t2;
```

Um Spalte `a` von `INTEGER` nach `TINYINT NOT NULL` zu ändern (der Name bleibt derselbe) und Spalte `b` von `CHAR(10)` nach `CHAR(20)` zu ändern und gleichzeitig von `b` nach `c` umzubenennen, geben Sie ein:

```
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

Jetzt wird eine `TIMESTAMP`-Spalte namens `d` hinzugefügt:

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Nunmehr erzeugen wir einen Index auf Spalte `d` und machen Spalte `a` zum Primärschlüssel:

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Wir entfernen Spalte `c`:

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Und fügen eine neue `AUTO_INCREMENT`-Ganzzahl-Spalte namens `c` hinzu:

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
           ADD INDEX (c);
```

Beachten Sie, dass wir `c` indiziert haben, weil `AUTO_INCREMENT`-Spalten indiziert sein müssen, und auch, dass wir `c` als `NOT NULL` deklariert haben, weil indizierte Spalten nicht `NULL` sein dürfen.

Wenn Sie eine `AUTO_INCREMENT`-Spalte hinzufügen, werden automatisch Spaltenwerte mit Zahlenfolgen eingefügt. Sie können die erste Zahl setzen, indem Sie `SET INSERT_ID=#` vor `ALTER TABLE` ausführen oder indem Sie die `AUTO_INCREMENT = #`-Tabellenoption benutzen. See Abschnitt 6.5.6, „SET-Syntax“.

Wenn Sie bei MyISAM-Tabellen nicht die `AUTO_INCREMENT`-Spalte ändern, ist die Folgezahl davon nicht betroffen. Wenn Sie eine `AUTO_INCREMENT`-Spalte löschen und dann eine weitere `AUTO_INCREMENT`-Spalte hinzufügen, fangen die Zahlen wieder bei 1 an.

See Abschnitt A.6.1, „Probleme mit `ALTER TABLE`.“.

7.5.5. `RENAME TABLE`-Syntax

```
RENAME TABLE tabelle TO neue_tabelle[, tabelle2 TO neue_tabelle2,...]
```

Das Umbenennen wird atomisch durchgeführt, was heißt, dass kein anderer Thread auf die Tabelle(n) zugreifen kann, während umbenannt wird. Das ermöglicht, eine Tabelle durch eine leere zu ersetzen:

```
CREATE TABLE neue_tabelle (...);
RENAME TABLE alte_tabelle TO datensicherung_tabelle, neue_tabelle TO alte_tabelle;
```

Das Umbenennen wird von links nach rechts durchgeführt, was bedeutet, dass Sie beim Vertauschen zweier Tabellennamen folgendes tun können:

```
RENAME TABLE alte_tabelle    TO datensicherung_tabelle,
             neue_tabelle    TO alte_tabelle,
             datensicherung_tabelle TO neue_tabelle;
```

Solange zwei Datenbanken auf derselben Platte liegen, können Sie auch von einer Datenbank in eine andere umbenennen:

```
RENAME TABLE aktuelle_datenbank.tabelle TO andere_datenbank.tabelle;
```

Wenn Sie `RENAME` ausführen, dürfen Sie keine gesperrten Tabellen oder aktive Transaktionen haben. Ausserdem benötigen Sie die `ALTER`- und `DROP`-Berechtigungen für die Original-Tabelle und die `CREATE`- und `INSERT`-Berechtigungen auf die neue Tabelle.

Wenn beim Umbenennen mehrfacher Tabellen Fehler auftreten, führt MySQL ein entgegengesetztes Umbenennen aller umbenannten Tabellen durch, um alles wieder in den Ausgangszustand zu versetzen.

7.5.6. DROP TABLE-Syntax

```
DROP TABLE [IF EXISTS] tabelle [, tabelle,...] [RESTRICT | CASCADE]
```

DROP TABLE entfernt eine oder mehrere Tabellen. Alle Tabellendaten und die Tabellendefinition werden *zerstört*, seien Sie daher **vorsichtig** mit diesem Befehl!

Ab MySQL-Version 3.22 können Sie die Schlüsselwörter IF EXISTS benutzen, um Fehler zu vermeiden, die auftreten, wenn Tabellen nicht existieren.

RESTRICT und CASCADE sind wegen leichterer Portierung zugelassen. Momentan tun sie nichts.

HINWEIS: DROP TABLE ist nicht transaktionssicher und führt automatisch jegliche aktiven Transaktionen zuende.

7.5.7. CREATE INDEX-Syntax

```
CREATE [UNIQUE|FULLTEXT] INDEX index_name ON tabelle (spalten_name[(laenge)],... )
```

Das CREATE INDEX-Statement macht vor MySQL-Version 3.22 nichts. Ab Version 3.22 ist CREATE INDEX auf ein ALTER TABLE-Statement gemappt, um Indexe zu erzeugen. See Abschnitt 7.5.4, „ALTER TABLE-Syntax“.

Normalerweise erzeugen Sie alle Indexe auf eine Tabelle zur Zeit, wo die Tabelle selbst mit CREATE TABLE erzeugt wird.

See Abschnitt 7.5.3, „CREATE TABLE-Syntax“. CREATE INDEX gestattet, bestehenden Tabellen Indexe hinzuzufügen.

A Spaltenliste der Form (spalte1,spalte2,...) erzeugt einen mehrspaltigen Index. Die Indexwerte werden durch Verkettung der Werte der angegebenen Spalten erzeugt.

Bei CHAR- und VARCHAR-Spalten können Indexe, die nur einen Teil einer Spalte benutzen, mit der spalten_name(laenge)-Syntax erzeugt werden. (Bei BLOB- und TEXT-Spalten ist die Längenangabe erforderlich.) Unten stehendes Statement zeigt, wie ein Index erzeugt wird, der die ersten 10 Zeichen der name-Spalte benutzt:

```
mysql> CREATE INDEX teil_von_name ON kunde (name(10));
```

Weil sich die meisten Namen üblicherweise in den ersten 10 Zeichen unterscheiden, sollte dieser Index nicht viel langsamer sein, als wenn der Index aus der gesamten name-Spalte erzeugt worden wäre. Die Benutzung von Teilspalten für Indexe kann die Index-Datei auch viel kleiner machen, was viel Speicherplatz sparen und zusätzlich INSERT-Operationen beschleunigen kann!

Beachten Sie, dass Sie einen Index auf eine Spalte, die NULL-Werte haben darf, oder auf eine BLOB/TEXT-Spalte erst ab MySQL-Version 3.23.2 und nur beim MyISAM-Tabellentyp erzeugen können.

Weitere Informationen darüber, wie MySQL Indexe benutzt, finden Sie unter Abschnitt 6.4.3, „Wie MySQL Indexe benutzt“.

FULLTEXT-Indexe können nur VARCHAR- und TEXT-Spalten indexieren und funktionieren nur bei MyISAM-Tabellen. FULLTEXT-Indexe sind ab MySQL-Version 3.23.23 verfügbar. Abschnitt 7.8, „MySQL-Volltextsuche“.

7.5.8. DROP INDEX-Syntax

```
DROP INDEX index_name ON tabelle
```

DROP INDEX löscht den Index namens index_name aus der Tabelle tabelle. DROP INDEX macht vor MySQL-Version 3.22 nichts. Ab Version 3.22 ist DROP INDEX auf ein ALTER TABLE-Statement gemappt, um den Index zu löschen. See Abschnitt 7.5.4, „ALTER TABLE-Syntax“.

7.6. Grundlegende Befehle des MySQL-Dienstprogramms für Benutzer

7.6.1. USE-Syntax

```
USE datenbank
```

Das USE datenbank-Statement weist MySQL an, datenbank als vorgabemäßige Datenbank für nachfolgende Anfragen zu benutzen. Die Datenbank bleibt die aktuelle, entweder bis zum Ende der Sitzung, oder bis ein weiteres USE-Statement abgesetzt wird:

```
mysql> USE datenbank1;
mysql> SELECT count(*) FROM tabelle;      # wählt aus von datenbank1.tabelle
```

```
mysql> USE datenbank2;
mysql> SELECT count(*) FROM tabelle;      # wählt aus von datenbank2.tabelle
```

Wenn Sie eine bestimmte Datenbank mit dem USE-Statement zu aktuellen machen, heißt das nicht, dass Sie nicht auf Tabellen in anderen Datenbanken zugreifen können. Das unten stehende Beispiel zeigt den Zugriff auf die autor-Tabelle in der datenbank1-Datenbank und auf die herausgeber-Tabelle in der datenbank2-Datenbank:

```
mysql> USE datenbank1;
mysql> SELECT autor_name,herausgeber_name FROM autor,datenbank2.herausgeber
           WHERE autor.herausgeber_id = datenbank2.herausgeber.herausgeber_id;
```

The USE-Statement wird für die Sybase-Kompatibilität zur Verfügung gestellt.

7.6.2. DESCRIBE-Syntax (Informationen über Spalten erhalten)

```
{DESCRIBE | DESC} tabelle {spalten_name | platzhalter}
```

DESCRIBE ist ein Kürzel für SHOW COLUMNS FROM. See Abschnitt 5.5.5.1, „Informationen über Datenbank, Tabellen, Spalten und Indexe abrufen“.

DESCRIBE stellt Informationen über die Spalten einer Tabelle bereit. spalten_name kann ein Spaltenname oder eine Zeichenkette sein, die die SQL-‘%’- und -‘_’-Platzhalterzeichen enthält.

Wenn die Spaltentypen sich von dem unterscheiden, was Sie auf der Grundlage eines CREATE TABLE-Statements erwartet hätten, beachten Sie, dass MySQL manchmal Spaltentypen ändert. See Abschnitt 7.5.3.1, „Stille Spaltentyp-Änderungen“.

Dieses Statement wird für die Oracle-Kompatibilität zur Verfügung gestellt.

Das SHOW-Statement stellt ähnliche Informationen bereit. See Abschnitt 5.5.5, „SHOW-Syntax“.

7.7. Transaktionale und Sperrbefehle von MySQL

7.7.1. BEGIN/COMMIT/ROLLBACK-Syntax

Vorgabemäßig läuft MySQL im autocommit-Modus. Das heißt, dass MySQL eine Aktualisierung auf Platte speichert, sobald Sie eine Aktualisierung ausführen.

Wenn Sie transaktionssichere Tabellen (wie InnoDB oder BDB) benutzen, können Sie MySQL mit folgendem Befehl in den Nicht-autocommit-Modus setzen:

```
SET AUTOCOMMIT=0
```

Danach müssen Sie COMMIT benutzen, um Ihre Änderungen auf Platte zu sichern, oder ROLLBACK, wenn Sie die Änderungen verwerfen wollen, die Sie seit dem Beginn der Transaktion gemacht haben.

Wenn Sie für eine Reihe von Statements zum AUTOCOMMIT-Modus umschalten wollen, können Sie das BEGIN- oder BEGIN WORK-Statement benutzen:

```
BEGIN;
SELECT @A:=SUM(gehalt) FROM tabelle1 WHERE type=1;
UPDATE tabelle2 SET zusammenfassung=@A WHERE type=1;
COMMIT;
```

Beachten Sie, dass bei der Benutzung nicht transaktionssicher Tabellen die Änderungen dennoch sofort gespeichert werden, unabhängig vom Status des autocommit-Modus.

Wenn Sie ROLLBACK bei der Aktualisierung einer nicht transaktionalen Tabelle ausführen, erhalten Sie einen Fehler (ER_WARNING_NOT_COMPLETE_ROLLBACK) als Warnung. Alle transaktionssicheren Tabellen werden zurückgesetzt, aber nicht transaktionale Tabelle ändern sich nicht.

Wenn Sie BEGIN oder SET AUTOCOMMIT=0 benutzen, sollten Sie die MySQL-Binär-Log-Datei für Datensicherungen benutzen statt der älteren Update-Log-Datei. Transaktionen werden in der Binär-Log-Datei in einem Stück gespeichert, beim COMMIT, um sicherzustellen, dass Transaktionen, die zurückgesetzt werden (Rollback), nicht gespeichert werden. See Abschnitt 5.9.4, „Die binäre Update-Log-Datei“.

Folgende Befehle beenden automatisch eine Transaktion (als ob Sie ein COMMIT vor der Ausführung des Befehls ausgeführt hätten:

ALTER TABLE	BEGIN	CREATE INDEX

DROP DATABASE	DROP TABLE	RENAME TABLE
TRUNCATE		

Sie können die Isolationsebene (Isolation Level) für Transaktionen mit SET TRANSACTION ISOLATION LEVEL ... Abschnitt 7.7.3, „SET TRANSACTION-Syntax“ ändern.

7.7.2. LOCK TABLES/UNLOCK TABLES-Syntax

```
LOCK TABLES tabelle [AS alias] {READ | [READ LOCAL] | [LOW_PRIORITY] WRITE}
            [, tabelle {READ | [LOW_PRIORITY] WRITE} ...]
...
UNLOCK TABLES
```

LOCK TABLES sperrt Tabellen für den aktuellen Thread. UNLOCK TABLES hebt alle Sperren auf, die vom aktuellen Thread veranlasst wurden. Alle Tabellen, die durch den aktuellen Thread gesperrt sind, werden automatisch entsperrt, wenn der Thread ein weiteres LOCK TABLES absetzt oder wenn die Verbindung zum Server geschlossen wird.

Die wichtigsten Gründe für die Benutzung von LOCK TABLES sind die Emulation von Transaktionen oder um mehr Geschwindigkeit bei der Aktualisierung von Tabellen zu erhalten. Das wird später detaillierter erläutert.

Wenn ein Thread eine READ-Sperre auf eine Tabelle erlangt, kann dieser Thread (und alle anderen Threads) nur aus der Tabelle lesen. Wenn ein Thread eine WRITE-Sperre auf eine Tabelle erlangt, kann nur der Thread, der die Sperre veranlasst hat, READ oder WRITE auf der Tabelle durchführen. Andere Threads werden blockiert.

Der Unterschied zwischen READ LOCAL und READ ist, dass READ LOCAL nicht kollidierende INSERT-Statements während der Dauer der Sperre zuläßt. Das kann jedoch nicht benutzt werden, wenn Sie Datenbankdateien ausserhalb von MySQL bearbeiten, während die Sperre aktiv ist.

Wenn Sie LOCK TABLES benutzen, müssen Sie alle Tabellen sperren, die Sie benutzen werden, und Sie müssen denselben Alias benutzen, den Sie in Ihren Anfragen benutzen werden! Wenn Sie eine Tabelle in einer Anfrage mehrfach (mit Aliasen) benutzen, müssen Sie für jeden Alias eine Sperre machen!

WRITE-Sperren haben normalerweise höhere Priorität als READ-Sperren, um sicherzustellen, dass Aktualisierungen so früh wie möglich bearbeitet werden. Dass heißt, wenn ein Thread eine READ-Sperre erlangt und dann ein anderer Thread eine WRITE-Sperre verlangt, dass nachfolgende READ-Sperrenanfragen warten, bis der WRITE-Thread die Sperre erhalten und freigegeben hat. Sie können LOW_PRIORITY WRITE-Sperren benutzen, um anderen Threads zu gestatten, READ-Sperren zu erlangen, während der Thread auf die WRITE-Sperre wartet. Sie sollten nur dann LOW_PRIORITY WRITE-Sperren benutzen, wenn Sie sicher sind, dass es irgendwann eine Zeit gibt, in der kein anderer Thread eine READ-Sperre haben wird.

LOCK TABLES funktioniert wie folgt:

1. Sortiert alle Tabellen, die gesperrt werden sollen, in einer intern definierten Reihenfolge (aus Benutzersicht ist die Reihenfolge undefiniert).
2. Wenn eine Tabelle mit einer Lese- und einer Schreibsperre gesperrt ist, wird die Schreibsperre vor die Lesesperre platziert.
3. Sperrt eine Tabelle nach der anderen, bis der Thread alle Sperren erhalten hat.

Diese Methode stellt sicher, dass Tabellensperren blockierungsfrei ist. Bei diesem Schema gibt es jedoch ein paar weitere Dinge, derer man sich bewusst sein muss:

Wenn Sie eine LOW_PRIORITY_WRITE-Sperre für eine Tabelle benutzen, heißt das, dass MySQL auf diese bestimmte Sperre wartet, bis es keinen Thread gibt, der eine READ-Sperre will. Wenn der Thread die WRITE-Sperre erhalten hat und darauf wartet, die Sperre für die nächste Tabelle in der Tabellensperrliste zu erhalten, warten alle anderen Threads darauf, dass die WRITE-Sperre aufgehoben wird. Wenn das bei Ihrer Applikation zu ernsthaften Problemen führt, sollten Sie in Betracht ziehen, einige Ihrer Tabelle in transaktionssichere Tabelle umzuwandeln.

Es ist sicher, einen Thread mit KILL zu killen, der auf eine Tabellensperre wartet. See Abschnitt 5.5.4, „KILL-Syntax“.

Beachten Sie, dass Sie **NICHT** irgend welche Tabellen sperren sollten, die Sie mit INSERT DELAYED benutzen. Das liegt darin, dass in diesem Fall das INSERT von einem separaten Thread durchgeführt wird.

Normalerweise müssen Sie Tabellen nicht sperren, weil alle einzelnen UPDATE-Statements atomisch sind. Kein anderer Thread kann mit einem aktuell ausgeführten SQL-Statement in die Quere kommen. Es gibt dennoch einige Fällen, in denen es wünschenswert sein kann, Tabellen zu sperren:

- Wenn Sie viele Operationen auf einer großen Zahl von Tabellen laufen lassen wollen, ist es viel schneller, die Tabellen zu sperren, die Sie benutzen werden. Der Nachteil besteht natürlich darin, dass kein anderer Thread eine `READ`-gesperrte Tabelle aktualisieren und kein anderer Thread eine `WRITE`-gesperrte Tabelle lesen kann.

  Der Grund, dass einiges mit `LOCK TABLES` schneller geht, liegt darin, dass MySQL den Schlüssel-Cache für die gesperrten Tabellen nicht auf Platte zurückschreibt (flush), bis `UNLOCK TABLES` aufgerufen wird (normalerweise wird der Schlüssel-Cache nach jedem SQL-Statement auf Platte zurückgeschrieben). Das erhöht die Geschwindigkeit bei den Operationen INSERT / UPDATE / DELETE bei `MyISAM`-Tabellen.

- Wenn Sie einen Tabellen-Handler in MySQL benutzen, der keine Transaktionen unterstützt, müssen Sie `LOCK TABLES` benutzen, wenn Sie sicherstellen wollen, dass kann anderer Thread zwischen einem `SELECT` und einem `UPDATE` dazwischen kommen kann. Das unten stehende Beispiel erfordert `LOCK TABLES`, um sicher ausgeführt zu werden:

  ```
  mysql> LOCK TABLES trans READ, kunde WRITE;
  mysql> select sum(wert) from trans where kunde_id=irgendeine_id;
  mysql> update kunde set gesamt_wert=summe_aus_vorherigem_statement
      ->       where kunde_id=irgendeine_id;
  mysql> UNLOCK TABLES;
  ```

  Ohne `LOCK TABLES` besteht die Möglichkeit, dass ein anderer Thread eine neue Zeile in die `trans`-Tabelle einfügt, zwischen der Ausführung des `SELECT`- und des `UPDATE`-Statements.

Wenn Sie inkrementelle Updates (`UPDATE kunde SET wert=wert+neuer_wert`) oder die `LAST_INSERT_ID()`-Funktion benutzen, können Sie `LOCK TABLES` in vielen Fällen vermeiden.

Einige Problemfälle können Sie auch lösen, indem Sie die Sperrfunktionen auf Benutzerebene `GET_LOCK()` und `RELEASE_LOCK()` benutzen. Diese Sperren werden in einer Hash-Tabelle im Server gespeichert und sind mit `pThread_mutex_lock()` und `pThread_mutex_unlock()` für die Erzielung höherer Geschwindigkeit implementiert. See Abschnitt 7.3.5.2, „Verschiedene Funktionen“.

Siehe Abschnitt 6.3.1, „Wie MySQL Tabellen sperrt“ wegen weiterer Informationen über Sperrmethoden.

Sie können alle Tabellen in allen Datenbanken mit Lesesperren sperren, und zwar mit dem `FLUSH TABLES WITH READ LOCK`-Befehl. See Abschnitt 5.5.3, „FLUSH-Syntax“. Das ist eine sehr bequeme Möglichkeit, Datensicherungen zu erhalten, wenn Sie ein Dateisystem wie Veritas haben, dass Schnappschüsse im Zeitverlauf aufnehmen kann.

HINWEIS: `LOCK TABLES` ist nicht transaktionssicher und schickt automatisch jegliche aktiven Transaktionen ab (Commit), bevor es versucht, die Tabellen zu sperren.

7.7.3. `SET TRANSACTION`-Syntax

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
[READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE]
```

Setzt die Transaktionsisolationsebene für die globale, gesamte Sitzung oder für die nächste Transaktion.

Das vorgabemäßige Verhalten ist das Setzen der Isolationsebene für die nächste (nicht angefangene) Transaktion.

Wenn Sie die `GLOBAL`-Berechtigung setzen, betrifft das alle neu erzeugten Threads. Sie benötigen dafür die `PROCESS`-Berechtigung.

Wenn Sie die `SESSION`-Berechtigung setzen, betrifft das die folgenden und alle zukünftigen Transaktionen.

Sie können die vorgabemäßige Isolationsebene für `mysqld` mit `--transaction-isolation=...` setzen. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

7.8. MySQL-Volltextsuche

Ab Version 3.23.23 bietet MySQL Unterstützung für Volltext-Indexierung und -Suche. Volltext-Indexe sind in MySQL Indexe vom Typ `FULLTEXT`. `FULLTEXT`-Indexe können von `VARCHAR`- und `TEXT`-Spalten zur Zeit von `CREATE TABLE` erzeugt werden oder später mit `ALTER TABLE` oder `CREATE INDEX` hinzugefügt werden. Bei großen Datenmengen ist es viel schneller, einen `FULLTEXT`-Index mit `ALTER TABLE` (oder `CREATE INDEX`) hinzuzufügen, als Zeilen in eine leere Tabelle mit einem `FULLTEXT`-Index einzufügen.

Die Volltextsuche wird mit der `MATCH`-Funktion durchgeführt.

```
mysql> CREATE TABLE artikel (
    ->   id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ->   titel VARCHAR(200),
    ->   artikeltext TEXT,
    ->   FULLTEXT (titel,artikeltext)
```

```
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO artikel VALUES
    -> (0,'MySQL-Tutorial', 'DBMS steht für DataBase-Management ...'),
    -> (0,'Wie man MySQL effizient einsetzt', 'Nachdem Sie ...'),
    -> (0,'MySQL optimieren','In diesem Tutorial wird gezeigt, wie ...'),
    -> (0,'1001 MySQL-Tricks','1. Lassen Sie mysqld nie als root laufen. 2. Normalisieren ...'),
    -> (0,'MySQL vs. YourSQL', 'Im folgenden Vergleich von Datenbank ...'),
    -> (0,'MySQL-Sicherheitsaspekte', 'Wenn er korrekt konfiguriert ist, ist MySQL ...');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM artikel WHERE MATCH (titel,artikeltext) AGAINST ('Datenbank');
+----+-------------------+------------------------------------------+
| id | titel             | artikeltext                              |
+----+-------------------+------------------------------------------+
|  5 | MySQL vs. YourSQL | Im folgenden Vergleich von Datenbank ...    |
|  1 | MySQL-Tutorial    | DBMS steht für DataBase-Management ...      |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)
```

Die Funktion `MATCH` prüft eine natürlichsprachige Anfrage gegen (`AGAINST`) eine Textsammlung (einfach ein Satz von Spalten, der vom `FULLTEXT`-Index abgedeckt wird). Für jede Zeile in einer Tabelle gibt sie eine Relevanz zurück - ein Ähnlichkeitsmaß zwischen dem Text in dieser Zeile (in den Spalten, die Teil der Textsammlung sind) und der Anfrage. Wenn sie in einer `WHERE`-Klausel benutzt wird (siehe Beispiel oben), werden die zurückgegebenen Zeilen automatisch nach absteigender Relevanz sortiert. Die Relevanz ist eine nicht negative Fließkommazahl. 0 Relevanz bedeutet keine Ähnlichkeit. Die Relevanz wird auf der Grundlage der Anzahl von Wörtern in der Zeile, der Anzahl eindeutiger Wörter in dieser Zeile, der Gesamtzahl von Wörtern in der Textsammlung und der Anzahl von Dokumenten (Zeilen) berechnet, die ein bestimmtes Wort enthalten.

Das obige Beispiel ist ein grundlegendes Beispiel der Benutzung der `MATCH`-Funktion. Die Zeilen werden nach absteigender Relevanz zurückgegeben.

```
mysql> SELECT id,MATCH (titel,artikeltext) AGAINST ('Tutorial') FROM artikel;
+----+-----------------------------------------------+
| id | MATCH (titel,artikeltext) AGAINST ('Tutorial') |
+----+-----------------------------------------------+
|  1 |                              0.64840710366884 |
|  2 |                                             0 |
|  3 |                              0.66266459031789 |
|  4 |                                             0 |
|  5 |                                             0 |
|  6 |                                             0 |
+----+-----------------------------------------------+
5 rows in set (0.00 sec)
```

Dieses Beispiel zeigt, wie man Relevanzen abruft. Weil weder die `WHERE`- noch die `ORDER BY`-Klausel vorhanden sind, werden die Zeilen unsortiert zurückgegeben.

```
mysql> SELECT id, artikeltext, MATCH (titel,artikeltext) AGAINST (
    -> 'Sicherheits-Implikationen, wenn Sie MySQL als root laufen lassen') AS rang
    -> FROM artikel WHERE MATCH (titel,artikeltext) AGAINST
    -> ('Sicherheits-Implikationen, wenn Sie MySQL als root laufen lassen');
+----+-------------------------------------------------------------+-----------------+
| id | artikeltext                                                 | rang            |
+----+-------------------------------------------------------------+-----------------+
|  4 | 1. Lassen Sie mysqld nie als root laufen. 2. Normalisieren ... | 1.5055546709332 |
|  6 | Wenn er korrekt konfiguriert ist, ist MySQL ...             |   1.31140957288 |
+----+-------------------------------------------------------------+-----------------+
2 rows in set (0.00 sec)
```

Das ist ein komplexeres Beispiel - die Anfrage gibt die Relevanz zurück und sortiert die Zeilen auch noch nach absteigender Relevanz. Um das zu erzielen, müssen Sie `MATCH` zweimal angeben. Beachten Sie, dass das keinen zusätzlichen Overhead verursacht, weil der MySQL-Optimierer bemerkt, dass diese zwei `MATCH`-Aufrufe identisch sind und daher den Code für die Volltextsuche nur einmal aufruft.

MySQL benutzt einen sehr einfachen Parser, um Text in Wörter zu zerlegen. Ein ``Wort'' ist jede Folge von Buchstaben, Zahlen, ‘'’ und ‘_’. Jedes ``Wort'', das in der Liste der Stopwords vorkommt oder einfach nur zu kurz ist (3 Zeichen oder weniger), wird ignoriert.

Jedes korrekte Wort in der Textsammlung und in der Anfrage wird nach seiner Signifikanz in der Anfrage oder der Textsammlung gewichtet. Daher hat ein Wort, dass in vielen Dokumenten vorkommt, ein geringeres Gewicht (und kann sogar 0 Gewicht haben), weil es in dieser bestimmten Textsammlung einen geringen semantischen Wert hat. Ansonsten, wenn das Wort selten vorkommt, erhält es ein höheres Gewicht. Die Gewichte der Wörter werden anschließend kombiniert, um die Relevanz der Zeile zu berechnen.

Solch eine Technik funktioniert am besten bei großen Textsammlungen (in der Tat wurde sie sorgfältig darauf optimiert). Bei sehr kleinen Tabellen spiegelt die Wortverteilung nicht adäquat seinen semantischen Wert wider, so dass dieses Modell manchmal bizarre Ergebnisse ergeben kann:

```
mysql> SELECT * FROM artikel WHERE MATCH (titel,artikeltext) AGAINST ('MySQL');
Empty set (0.00 sec)
```

Die Suche nach dem Wort `MySQL` erzeugt im obigen Beispiel keine Ergebnisse. Das Wort `MySQL` ist in mehr als der Hälfte der Zeilen vorhanden und wird deshalb als Stopword betrachtet (eins mit dem semantischen Wert 0). Das ist in der Tat das gewünschte Verhalten - eine natürlichsprachige Anfrage sollte bei einer 1 GB großen Tabelle nicht jede zweite Zeile zurückgeben.

Bei einem Wort, dass in der Hälfte der Zeilen in einer Tabelle übereinstimmt, ist es nicht sehr wahrscheinlich, dass relevante Dokumente gefunden werden, sondern statt dessen viele irrelevante Dokumente. Das kennen wir alle aus Recherchen über Suchmaschinen auf dem Internet. Das ist die Überlegung, die dahinter steht, dass solchen Wörtern ein niedriger semantischer Wert **in diesem bestimmten Satz von Daten** gegeben wird.

7.8.1. Volltext-Einschränkungen

- Alle Parameter der `MATCH`-Funktion müssen Spalten derselben Tabelle sein, die Teil desselben Volltext-Indexes ist.
- Das Argument für `AGAINST` muss eine Konstanten-Zeichenkette sein.

7.8.2. MySQL-Volltextsuche fein einstellen

Leider hat die Volltextsuche noch keine durch den Benutzer einstellbare Parameter, doch diese stehen sehr weit oben auf der TODO-Liste. Wenn Sie jedoch eine MySQL-Quelldistribution (see Abschnitt 3.3, „Installation der Quelldistribution“) haben, können Sie das Verhalten der Volltextsuche in einiger Hinsicht ändern.

Beachten Sie, dass die Volltextsuche sorgfältig auf beste Sucheffektivität eingestellt wurde. Wenn Sie dieses vorgabemäßige Verhalten ändern, wird das die Suchergebnisse in den meisten Fällen verschlechtern. Ändern Sie die MySQL-Quelltexte deshalb nur, wenn Sie genau wissen, was Sie tun!

- Die minimale zu indexierende Wortlänge wird in der `myisam/ftdefs.h`-Datei in folgender Zeile festgelegt:

  ```
  #define MIN_WORD_LEN 4
  ```

  Ändern Sie diesen Wert nach Belieben, kompilieren Sie MySQL neu und bauen Sie Ihre `FULLTEXT`-Indexe neu auf.
- Die Stopword-Liste wird in `myisam/ft_static.c` definiert. Ändern Sie sie nach Ihrem Geschmack, kompilieren Sie MySQL neu und bauen Sie Ihre `FULLTEXT`-Indexe neu auf.
- Die 50%-Schwelle wird durch das spezielle, ausgewählte Gewichtungsschema festgelegt. Um dieses abzuschalten, ändern Sie folgende Zeile in `myisam/ftdefs.h`:

  ```
  #define GWS_IN_USE GWS_PROB
  ```

  zu

  ```
  #define GWS_IN_USE GWS_FREQ
  ```

  und kompilieren Sie MySQL neu. In diesem Fall brauchen Sie die Indexe nicht neu aufzubauen.

7.8.3. Neue Features der Volltextsuche in MySQL 4.0

Dieser Abschnitt enthält eine Auflistung der Volltext-Features, die bereits im MySQL-4.0-Baum implementiert sind. Er erläutert den **More Funktionen für Volltextsuche**-Eintrag in Abschnitt 2.8, „MySQL und die Zukunft (das TODO)“.

- `REPAIR TABLE` mit `FULLTEXT`-Indexen, `ALTER TABLE` mit `FULLTEXT`-Indexen und `OPTIMIZE TABLE` mit `FULLTEXT`-Indexen läuft jetzt bis zu 100 mal schneller.
- `MATCH ... AGAINST` wird folgende **Boolesch Operatoren** unterstützen:
  - `+`wort bedeutet, dass das Wort in jeder zurückgegebenen Zeile enthalten sein **muss**.
  - `-`wort bedeutet, dass das Wort in jeder zurückgegebenen Zeile **nicht** enthalten sein darf.
  - `<` und `>` können benutzt werden, um die Wortgewichtung in der Anfrage herab- und heraufzusetzen.
  - `~` kann benutzt werden, um einem 'Rausch-Wort' ein **negatives** Gewicht zuzuweisen.
  - `*` ist ein Trunkierungsoperator.

  Die Boole'sche Suche benutzt eine vereinfachte Art, die Relevanz zu berechnen, die keine 50%-Schwelle hat.

- Suchen sind jetzt wegen optimierter Suchalgorithmen bis zu 2 mal schneller.
- Das Dienstprogramm `ft_dump` wurde für Low-Level-`FULLTEXT`-Index-Operationen hinzugefügt (Anfragen / Dumps / Statistiken).

7.8.4. Volltextsuche TODO-Liste

- Alle Operationen mit `FULLTEXT`-Index **schneller** machen.
- Unterstützung für Klammern `()` in Boole'scher Volltextsuche.
- Phrasensuche, Näherungsoperatoren
- Boole'sche Suche funktioniert ohne `FULLTEXT`-Index (ja, **sehr** langsam).
- Unterstützung für "immer indizierte Wörter". Das könnten beliebige Zeichenketten sein, die der Benutzer wie Wörter behandeln will. Beispiele sind "C++", "AS/400", "TCP/IP" usw.
- Unterstützung für Volltextsuche in `MERGE`-Tabellen.
- Unterstützung für Multi-Byte-Zeichensätze.
- Die Stopword-Liste von der Sprache der Daten abhängig machen.
- Eindämmen (Stemming, natürlich abhängig von der Sprache der Daten).
- Generischer Benutzer-unterstützbarer UDF- (?) Preparser.
- Das Modell flexibler machen (durch Hinzufügen einiger regulierbarer Parameter für `FULLTEXT` in `CREATE/ALTER TABLE`).

7.9. MySQL-Anfragen-Cache

Ab Version 4.0.1 besitzt der `MySQL-Server` einen `Anfragen-Cache`. Wenn er benutzt wird, speichert er den Text einer `SELECT`-Anfrage zusammen mit dem entsprechenden Ergebnis, das an den Client gesendet wird. Wenn eine weitere identische Anfrage empfangen wird, kann der Server die Ergebnisse aus dem Cache beziehen, statt dieselbe Anfrage zu parsen und noch einmal auszuführen.

Der Anfragen-Cache ist extrem nützlich in Umgebungen, in denen sich (einige) Tabellen nicht häufig ändern und in denen Sie viele identische Anfragen haben. Das ist eine typische Situation für viele Web-Server, die viele dynamische Inhalte benutzen.

Im folgenden finden Sie einige Performance-Daten für den Anfragen-Cache (die wir mit der MySQL-Benchmark-Suite auf einer Linux Alpha 2 x 500 MHz mit 2 GB RAM und einem 64-MB-Anfragen-Cache gewonnen haben):

- Wenn Sie den Anfragen-Cache-Code abschalten wollen, setzen Sie `query_cache_size=0`. Wenn Sie den Anfragen-Cache-Code abschalten, gibt es keinen bemerkbaren Overhead.
- Wenn alle Anfragen, die Sie ausführen, einfach sind (wie das Auswählen einer Zeile aus einer Tabelle mit einer Zeile), sich aber dennoch unterscheiden, so dass die Anfragen nicht gecachet werden können, ist der Overhead bei einem aktiven Anfragen-Cache 13%. Das sollte als Szenario für den schlechtesten Fall angesehen werden. Im echten Leben sind Anfragen jedoch meist viel komplizierter, so dass der Overhead normalerweise beträchtlich geringer ist.
- Die Suche nach einer Zeile in einer Einzeilen-Tabelle ist 238% schneller. Das kann als minimale Geschwindigkeitssteigerung für eine gecachete Anfrage betrachtet werden.

7.9.1. Wie der Anfragen-Cache funktioniert

Anfragen werden vor dem Parsen verglichen, daher werden

```
SELECT * FROM TABELLE
```

und

```
Select * from tabelle
```

als unterschiedliche Anfragen für den Anfragen-Cache betrachtet. Anfragen müssen also exakt gleich sein (Byte für Byte), um als identisch erkannt zu werden. Zusätzlich kann eine Anfrage als unterschiedlich betrachtet werden, wenn ein Client zum Beispiel ein neues Kommunikationsprotokollformat benutzt oder einen anderen Zeichensatz als ein anderer Client.

Anfragen, die unterschiedliche Datenbanken, Protokollversionen oder unterschiedliche vorgabemäßige Zeichensätze benutzen, werden als unterschiedliche Anfragen angesehen und separat gecachet.

Der Cache funktioniert auch bei Anfragen der Art `SELECT CALC_ROWS ...` und `SELECT FOUND_ROWS() ...`, weil die Anzahl der gefundenen Zeilen ebenfalls im Cache gespeichert wird.

Wenn sich eine Tabelle ändert (`INSERT`, `UPDATE`, `DELETE`, `TRUNCATE`, `ALTER` oder `DROP TABLE|DATABASE`), werden alle gecacheten Anfragen, die diese Tabelle benutzten (möglicherweise über eine `MRG_MyISAM`-Tabelle!) ungültig und werden aus dem Cache entfernt.

Momentan werden alle `InnoDB`-Tabellen beim `COMMIT` als für den Cache ungültig gekennzeichnet. In Zukunft wird das geändert, so dass nur Tabellen, die in der Transaktion geändert wurden, für die entsprechenden Cache-Einträge als ungültig markiert werden.

Eine Anfrage kann nicht gecachet werden, wenn sie eine der folgenden Funktionen enthält:

Funktion	Funktion	Funktion	Funktion
`Benutzerdefinierte Funktionen`	`CONNECTION_ID`	`FOUND_ROWS`	`GET_LOCK`
`RELEASE_LOCK`	`LOAD_FILE`	`MASTER_POS_WAIT`	`NOW`
`SYSDATE`	`CURRENT_TIMESTAMP`	`CURDATE`	`CURRENT_DATE`
`CURTIME`	`CURRENT_TIME`	`DATABASE`	`ENCRYPT` (mit einem Parameter)
`LAST_INSERT_ID`	`RAND`	`UNIX_TIMESTAMP` (ohne Parameter)	`USER`
`BENCHMARK`			

Eine Anfrage kann ebenfalls nicht gecachet werden, wenn sie Benutzer-Variablen enthält oder wenn sie in der Form `SELECT ... IN SHARE MODE` oder der Form `SELECT * FROM AUTOINCREMENT_FIELD IS NULL` (um als ODBC-Workaround die letzte eingefügte ID abzurufen) ist.

`FOUND ROWS()` gibt jedoch den korrekten Werte zurück, selbst wenn eine vorhergehende Anfrage aus dem Cache geholt wurde.

Anfragen, die keinerlei Tabellen benutzen oder solche, bei denen der Benutzer eine Spaltenberechtigung für irgend eine der beteiligten Tabellen hat, werden nicht gecachet.

Bevor eine Anfrage aus dem Anfragen-Cache geholt wird, prüft MySQL, ob der Benutzer die SELECT-Berechtigung für alle beteiligten Datenbanken und Tabellen hat. Wenn nicht, wird das Cache-Ergebnis nicht benutzt.

7.9.2. Anfragen-Cache-Konfiguration

Aufgrund des Anfragen-Caches gibt es ein paar neue `MySQL` Systemvariablen für `mysqld`, die in einer Konfigurationsdatei oder auf der Kommandozeile beim Starten von `mysqld` gesetzt werden können:

- `query_cache_limit` Keine Ergebnisse cachen, die größer als dieser Wert sind (Vorgabe 1 MB).
- `query_cache_size` Der zugewiesene Arbeitsspeicher, um Ergebnisse aus alten Anfragen zu speichern. Wenn er 0 ist, ist der Anfragen-Cache abgeschaltet (Vorgabe).
- `query_cache_startup_type` Dieser Wert (nur Zahlen) kann wie folgt gesetzt werden:

Option	Beschreibung
0	(OFF - AUS, Ergebnisse nicht cachen oder abrufen)
1	(ON - AN, alle Ergebnisse ausser `SELECT SQL_NO_CACHE ...`-Anfragen cachen)
2	(DEMAND - AUF VERLANGEN, nur `SELECT SQL_CACHE ...`-Anfragen cachen)

Innerhalb eines Threads (Verbindung) kann das Verhalten des Anfragen-Caches abweichend von der Vorgabe verändert werden. Die Syntax ist wie folgt:

`SQL_QUERY_CACHE_TYPE = OFF | ON | DEMAND` `SQL_QUERY_CACHE_TYPE = 0 | 1 | 2`

Option	Beschreibung
0 oder OFF	Keine Ergebnisse cachen oder abrufen.
1 oder ON	Alle Ergebnisse ausser `SELECT SQL_NO_CACHE ...`-Anfragen cachen.
2 oder DEMAND	Nur `SELECT SQL_CACHE ...`-Anfragen cachen.

Vorgabemäßig hängt `SQL_QUERY_CACHE_TYPE` vom Wert von `query_cache_startup_type` ab, als der Thread erzeugt wurde.

7.9.3. Anfragen-Cache-Optionen in `SELECT`

Es gibt zwei mögliche Anfragen-Cache-bezogene Parameter, die in einer `SELECT`-Anfrage angegeben werden können:

Option	Beschreibung
`SQL_CACHE`	Wenn `SQL_QUERY_CACHE_TYPE DEMAND` ist, darf die Anfrage gecachet werden. Wenn `SQL_QUERY_CACHE_TYPE ON` ist, ist das die Vorgabe. Wenn `SQL_QUERY_CACHE_TYPE OFF` ist, nichts tun.
`SQL_NO_CACHE`	Diese Anfrage wird nicht gecachet.

7.9.4. Anfragen-Cache-Status und -Wartung

Mit dem `FLUSH QUERY CACHE`-Befehl können Sie den Anfragen-Cache defragmentieren, um den Speicher besser zu benutzen. Dieser Befehl entfernt keinerlei Anfragen aus dem Cache. `FLUSH TABLES` schreibt auch den Anfragen-Cache zurück auf Platte.

Der `RESET QUERY CACHE`-Befehl entfernt alle Anfragenergebnisse aus dem Anfragen-Cache.

Sie können die Anfragen-Cache-Performance in `SHOW STATUS` beobachten:

Variable	Beschreibung
`Qcache_queries_in_cache`	Anzahl von Anfragen, die im Cache registriert sind.
`Qcache_inserts`	Anzahl von Anfragen, die zum Cache hinzugefügt wurden.
`Qcache_hits`	Anzahl von Cache-Hits.
`Qcache_not_cached`	Anzahl von nicht gecacheten Anfragen (nicht cachebar oder wegen `SQL_QUERY_CACHE_TYPE`).
`Qcache_free_memory`	Menge des freien Speichers für den Anfragen-Cache.
`Qcache_total_blocks`	Gesamtzahl von Blöcken im Anfragen-Cache.
`Qcache_free_blocks`	Anzahl freier Speicherblöcke im Anfragen-Cache.

Gesamtzahl von Anfragen = `Qcache_inserts` + `Qcache_hits` + `Qcache_not_cached`.

Der Anfragen-Cache benutzt variable Blocklängen, so dass `Qcache_total_blocks` und `Qcache_free_blocks` eine Speicherfragmentierung des Anfragen-Caches anzeigen können. Nach `FLUSH QUERY CACHE` verbleibt nur ein einzelner (großer) freier Block.

Hinweis: Jede Anfrage benötigt minimal zwei Blöcke (einen für den Anfragentext und einen weiteren für das Anfragenergebnis). Ausserdem benötigt jede Tabelle, die in einer Anfrage benutzt wurde, einen Block. Wenn allerdings zwei oder mehr Anfragen dieselbe Tabelle benutzen, muss nur ein Block zugewiesen werden.

Kapitel 8. MySQL-Tabellentypen

AB MySQL-Version 3.23.6 können Sie unter drei grundlegenden Tabellenformaten (`ISAM`, `HEAP` und `MyISAM` wählen. Neuere MySQL-Versionen können zusätzliche Tabellentypen unterstützen (`InnoDB`, oder `BDB`), abhängig davon, wie Sie sie kompilieren.

Beim Erzeugen einer neuen Tabelle können Sie MySQL mitteilen, welcher Tabellentyp dafür benutzt werden soll. MySQL erzeugt immer eine `.frm`-Datei, die die Tabellen- und Spaltendefinitionen enthält. Abhängig vom Tabellentyp werden Index und Daten in anderen Dateien gespeichert.

Beachten Sie, dass Sie für die Benutzung von `InnoDB`-Tabellen zumindest die `innodb_data_file_path`-Startoption benötigen. See Abschnitt 8.5.2, „Mit InnoDB anfangen - Optionen“.

Der vorgabemäßige Tabellentyp in MySQL ist `MyISAM`. Wenn Sie versuchen, einen Tabellentyp zu benutzen, der nicht einkompiliert oder aktiviert ist, erzeugt MySQL statt dessen eine Tabelle vom Typ `MyISAM`. Das ist ein sehr nützliches Feature, wenn Sie Tabellen zwischen unterschiedlichen SQL-Servern kopieren wollen, die unterschiedliche Tabellentypen unterstützten (zum Beispiel Tabellen zu einem Slave kopieren, der für Geschwindigkeit optimiert ist, aber keine transaktionalen Tabellen hat). Dieses automatische Ändern des Tabellentyps kann andererseits für neue MySQL-Benutzer sehr verwirrend sein. Wir planen für MySQL 4.0, das zu beheben, indem eine Warnung ausgegeben wird, wenn ein Tabellentyp automatisch geändert wird.

Sie können Tabellen zwischen unterschiedlichen Typen mit dem `ALTER TABLE`-Statement umwandeln. See Abschnitt 7.5.4, „`ALTER TABLE`-Syntax“.

MySQL unterstützt zwei unterschiedliche Arten von Tabellen: transaktionssichere Tabellen (`InnoDB` und `BDB`) und nicht transaktionssichere Tabellen (`HEAP`, `ISAM`, `MERGE` und `MyISAM`).

Vorteile transaktionssicherer Tabellen (TST):

- Sicherer. Selbst wenn MySQL abstürzt oder wenn Sie Hardware-Probleme bekommen, bekommen Sie Ihre Daten zurück, entweder über automatische Wiederherstellung oder von einer Datensicherung plus Transaktionslog-Datei.
- Sie können viele Statements kombinieren und alle in einem Rutsch mit dem `COMMIT`-Befehl akzeptieren.
- Sie können `ROLLBACK` ausführen, um Ihre Änderungen zu ignorieren (wenn Sie nicht im Auto-Commit-Modus fahren).
- Wenn eine Aktualisierung fehlschlägt, werden Ihre Änderungen zurückgesichert. (Bei nicht transaktionssicheren Tabellen sind durchgeführte Änderungen permanent.)

Vorteile nicht transaktionssicherer Tabellen (NTST):

- Viel schneller, da es keinen Transaktionsoverhead gibt.
- Benötigen aufgrund des fehlenden Transaktionsoverheads weniger Speicherplatz.
- Benötigen weniger Arbeitsspeicher für Aktualisierungen.

Sie können TST- and NTST-Tabellen in denselben Statements kombinieren, um das Beste aus beiden Welten zu bekommen.

8.1. MyISAM-Tabellen

`MyISAM` ist der vorgabemäßige Tabellentyp in MySQL-Version 3.23. Er basiert auf dem `ISAM`-Code und hat viele nützliche Erweiterungen.

Der Index wird in einer Datei mit der Endung `.MYI` (MYIndex) gespeichert, die Daten in einer Datei mit der Endung `.MYD` (MYData). Sie können `MyISAM`-Tabellen mit dem `myisamchk`-Dienstprogramm überprüfen und reparieren. See Abschnitt 5.4.6.9, „Wie Tabellen repariert werden“. Sie können `MyISAM`-Tabellen mit `myisampack` komprimieren, damit sie viel weniger Speicherplatz benötigen. See Abschnitt 5.7.4, „myisampack, MySQL-Programm zum Erzeugen komprimierter Nur-Lese-Tabellen“.

Folgende Neuerungen gibt es bei `MyISAM`:

- Es gibt einen Flag in der `MyISAM`-Datei, der anzeigt, ob die Tabelle korrekt geschlossen wurde. Wenn `mysqld` mit `--myisam-recover` gestartet wird, werden `MyISAM`-Tabellen beim Öffnen automatisch geprüft und / oder repariert, falls die Tabelle nicht korrekt geschlossen wurde.
- Sie können neue Zeilen in eine Tabelle, die keinerlei freie Blöcke mitten in der Daten-Datei hat, einfügen (`INSERT`), während

zeitgleich andere Threads aus der Tabelle lesen (zeitgleiches Einfügen). Ein freier Block kann entstehen, wenn eine Aktualisierung einer Zeile dynamischer Länge, die viele Daten enthält, mit weniger Daten durchgeführt wird, oder wenn Zeilen gelöscht werden. Wenn alle freien Blöcke aufgebraucht sind, können alle zukünftigen Einfügeoperationen auf die zeitgleiche Art erfolgen.

- Unterstützung für große Dateien (63-Bit) auf Dateisystemen / Betriebssystemen, die große Dateien unterstützen.
- Alle Daten werden mit dem niedrigen Byte zuerst gespeichert. Das macht die Daten Maschinen- und Betriebssystem-unabhängig. Die einzige Anforderung ist, dass die Maschine zweien-komplementäre vorzeichenbehaftete Ganzzahlen (two's-complement signed integers) benutzt, was bei jeder Maschine in den letzten 20 Jahren der Fall war), sowie das IEEE-Fließkomma-Format (bei Mainstream-Maschinen absolut dominierend). Die einzige Art von Maschinen, die vielleicht keine Binärkompatibilität unterstützen, sind eingebettete Systeme (Embedded Systems), weil diese manchmal eigentümliche Prozessoren haben.

  Wenn Daten mit dem niedrigen Byte zuerst gespeichert werden, ergibt sich daraus kein großer Geschwindigkeitsnachteil. Die Bytes in einer Tabellenzeile sind normalerweise unzusammenhängend und man benötigt kaum mehr Ressourcen, um ein unzusammenhängendes Byte in Reihenfolge statt in umgekehrter Reihenfolge zu lesen. Der tatsächliche Hole-Spaltenwert-Code ist im Vergleich zu sonstigem Code ebenfalls nicht zeitkritisch.
- Alle Zahlenschlüssel werden mit dem hohen Byte zuerst gespeichert, um bessere Index-Kompression zu erzielen.
- Die interne Handhabung einer `AUTO_INCREMENT`-Spalte. `MyISAM` aktualisiert diese automatisch bei `INSERT / UPDATE`. Der `AUTO_INCREMENT`-Wert kann mit `myisamchk` zurückgesetzt werden. Das macht `AUTO_INCREMENT`-Spalten schneller (mindestens 10%), und alten Zahlen werden im Gegensatz zum alten `ISAM` nicht wieder benutzt. Beachten Sie, dass das alte Verhalten immer noch da ist, wenn ein `AUTO_INCREMENT` am Ende eines mehrteiligen Schlüssels definiert wird.
- Wenn er in sortierter Reihenfolge eingefügt wird (wie bei der Benutzung einer `AUTO_INCREMENT`-Spalte), wird der Schlüsselbaum gespalten, so dass der hohe Knoten nur einen Schlüssel enthält. Das verbessert die Platzausnutzung im Schlüsselbaum.
- `BLOB`- und `TEXT`-Spalten können indiziert werden.
- `NULL`-Werte sind in indizierten Spalten erlaubt. Dafür werden 0 bis 1 Byte pro Schlüssel benötigt.
- Die maximale Schlüssellänge beträgt vorgabemäßig 500 Bytes (das kann beim Neukompilieren geändert werden). Wenn Schlüssel länger als 250 Bytes sind, wird für diese eine höhere Schlüsselblockgröße als die vorgabemäßigen 1024 Bytes benutzt.
- Die maximale Anzahl von Schlüsseln pro Tabelle beträgt vorgabemäßig 32. Diese kann bis auf 64 erhöht werden, ohne dass `myisamchk` neu kompiliert werden muss.
- `myisamchk` kennzeichnet Tabellen als geprüft, wenn es mit `--update-state` läuft. `myisamchk --fast` prüft nur die Tabellen, die diese Kennzeichnung nicht haben.
- `myisamchk -a` speichert Statistiken für Schlüsselteile (und nicht nur für gesamte Schlüssel wie bei `ISAM`).
- Zeilen dynamischer Größe werden viel weniger fragmentiert werden, wenn Lösch- mit Aktualisierungs- und Einfügeoperationen gemischt werden. Dafür wird gesorgt, indem angrenzende gelöschte Blöcke automatisch kombiniert werden und dadurch, dass Blöcke erweitert werden, wenn der nächste Block gelöscht wird.
- `myisampack` kann `BLOB`- and `VARCHAR`-Spalten komprimieren.
- Sie können die Daten-Datei und die Index-Datei in unterschiedliche Verzeichnisse legen, um mehr Geschwindigkeit zu erhalten (mit der `DATA/INDEX DIRECTORY="pfad"`-Option für `CREATE TABLE`). See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.

`MyISAM` unterstützt ausserdem die folgenden Dinge, die MySQL in naher Zukunft benutzen können wird:

- Unterstützung für einen echten `VARCHAR`-Typ. Eine `VARCHAR`-Spalte fängt mit einer in 2 Bytes gespeicherten Länge an.
- Tabellen mit `VARCHAR` können eine feste oder dynamische Datensatzlänge haben.
- `VARCHAR` und `CHAR` können bis zu 64 KB Groß sein. Alle Schlüsselsegmente haben ihre eigene Sprachdefinition. Das versetzt MySQL in die Lage, unterschiedliche Sprachdefinitionen pro Spalte zu haben.
- Ein gehashter berechneter Index kann für `UNIQUE` benutzt werden. Das erlaubt Ihnen, `UNIQUE` auf jeder beliebigen Kombination von Spalten in einer Tabelle zu haben. (Sie können jedoch auf einem `UNIQUE` berechneten Index nicht suchen.)

Beachten Sie, dass Index-Dateien bei `MyISAM` üblicherweise viel kleiner sind als bei `ISAM`. Das bedeutet, dass `MyISAM` normalerweise weniger Systemressourcen verbraucht als `ISAM`, allerdings mehr Prozessorleistung beim Einfügen von Daten in einen komprimierten Index.

Folgende Optionen für `mysqld` können benutzt werden, um das Verhalten von `MyISAM`-Tabellen zu ändern. See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“.

Option	Beschreibung
`--myisam-recover=#`	Automatische Wiederherstellung beschädigter Tabellen.
`-O myisam_sort_buffer_size=#`	Der beim Wiederherstellen von Tabellen benutzte Puffer.
`--delay-key-write-for-all-tables`	Keine Schlüsselpuffer zwischen Schreibvorgängen auf jedwede MyISAM-Tabelle zurückschreiben (flush).
`-O myisam_max_extra_sort_file_size=#`	Wird benutzt, um MySQL bei der Entscheidung zu helfen, wann die langsame, aber sichere Schlüssel-Cache-Index-Erzeugungsmethode benutzt werden sollte. **Hinweis:** Dieser Parameter wird in Megabytes angegeben!
`-O myisam_max_sort_file_size=#`	Die schnelle Index-Sortiermethode beim Erzeugen eines Indexes nicht benutzen, wenn die temporäre Datei größer als dieser Wert werden würde. **Hinweis:** Dieser Parameter wird in Megabytes angegeben! megabytes!--
`-O myisam_bulk_insert_tree_size=#`	Die Größe des Baum-Caches, der bei der Optimierung von Massen-Einfügeoperationen benutzt wird. **Hinweis:** Das ist die Begrenzung **pro Thread**!

Die automatische Wiederherstellung wird aktiviert, wenn Sie `mysqld` mit `--myisam-recover=#` starten. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“. Beim Öffnen wird geprüft, ob die Tabelle als beschädigt gekennzeichnet ist oder ob die Zählvariable für die Tabelle nicht 0 ist und Sie mit `--skip-locking` laufen lassen. Wenn eine dieser Bedingungen erfüllt ist, geschieht folgendes:

- Die Tabelle wird auf Fehler geprüft.
- Wenn ein Fehler gefunden wird, wird eine schnelle Reparatur der Tabelle versucht (mit Sortieren und ohne Neuerzeugung der Daten-Datei).
- Wenn die Reparatur wegen eines Fehlers in der Daten-Datei fehlschlägt (zum Beispiel ein Fehler wegen eines doppelten Schlüsseleintrags), wird die Reparatur noch einmal versucht, diesmal allerdings mit Neuerzeugung der Daten-Datei.
- Wenn dieser Versuch fehlschlägt, wird die Reparatur noch einmal mit der alten Reparaturoption versucht (Zeile für Zeile ohne Sortieren schreiben), was jede Sorte von Fehler beheben sollte, bei gewissen Festplatten-Erfordernissen ...

Wenn die Wiederherstellung nicht in der Lage ist, alle Zeilen aus einem vorher abgeschlossenen Statement wiederherzustellen, und Sie nicht `FORCE` als Option für `myisam-recover` angegeben haben, wird die automatische Reparatur mit einer Fehlermeldung in der Fehlerdatei abgebrochen:

```
Error: Couldn't repair table: test.g00pages
```

Hätten Sie in diesem Fall die `FORCE`-Option benutzt, würden Sie statt dessen in der Fehlerdatei eine Warnung erhalten:

```
Warning: Found 344 of 354 rows when repairing ./test/g00pages
```

Wenn Sie automatisches Wiederherstellung mit der `BACKUP`-Option laufen lassen, beachten Sie, dass Sie ein Cron-Skript haben sollten, dass automatisch Dateien mit Namen wie `tabellenname-datetime.BAK` aus den Datenbank-Verzeichnissen auf ein Sicherungsmedium verschiebt.

See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

8.1.1. Für Schlüssel benötigter Speicherplatz

MySQL unterstützt unterschiedliche Index-Typen, doch der normale Typ ist ISAM oder MyISAM. Diese benutzen einen B-Baum-Index, und Sie können die Größe der Index-Datei grob als `(schluessel_laenge+4)/0.67` kalkuliert, summiert über alle Schlüssel. (Das ist der schlechteste Fall, bei dem alle Schlüssel in sortierter Reihenfolge eingeordnet werden und es keinerlei Schlüssel-Komprimierung gibt.)

Zeichenketten-Indexe werden Leerzeichen-komprimiert. Wenn der erste Index-Teil eine Zeichenkette ist, wird er zusätzlich Präfix-komprimiert. Leerzeichen-Kompression macht die Index-Datei kleiner als in den obigen Zahlen dargestellt, wenn die

Zeichenkettenspalte viele Leerzeichen am Ende hat oder eine VARCHAR-Spalte ist, die nicht immer in voller Länge genutzt wird. Präfix-Kompression wird bei Schlüsseln benutzt, die mit einer Zeichenkette beginnen. Präfix-Kompression hilft, wenn es viele Zeichenketten mit identischem Präfix gibt.

Bei MyISAM-Tabellen können Sie auch Zahlen Präfix-komprimieren, indem Sie beim Erzeugen der Tabelle PACK_KEYS=1 angeben. Das hilft, wenn Sie viele Ganzzahl-Schlüssel mit identischem Präfix haben, wenn die Zahlen mit dem hohen Byte zuerst gespeichert werden.

8.1.2. MyISAM-Tabellenformate

MyISAM unterstützt 3 verschiedene Tabellentypen. Zwei von ihnen werden automatisch gewählt, abhängig vom Spaltentyp, den Sie benutzen. Der dritte, komprimierte Tabellen, kann nur mit dem myisampack-Dienstprogramm erzeugt werden.

Wenn Sie eine Tabelle erzeugen (CREATE) oder ändern (ALTER), können Sie bei Tabellen, die kein BLOB enthalten, ein dynamisches (DYNAMIC) oder festes (FIXED) Tabellenformat mit der ROW_FORMAT=#-Tabellenoption erzwingen. Zukünftig werden Sie in der Lage sein, Tabellen zu komprimieren / dekomprimieren, indem Sie ROW_FORMAT=compressed | default für ALTER TABLE angeben. See Abschnitt 7.5.3, „CREATE TABLE-Syntax“.

8.1.2.1. Kennzeichen statischer (Festlängen-) Tabellen

Das ist das vorgabemäßige Format. Es wird benutzt, wenn die Tabelle keine VARCHAR-, BLOB- oder TEXT-Spalten enthält.

Dieses Format ist das einfachste und sicherste Format. Es ist auch das schnellste der Formate auf Platte. Die Geschwindigkeit ergibt sich aus der einfachen Weise, wie Daten auf der Platte gefunden werden können. Wenn man etwas mit einem Index und statischem Format nachschlägt, ist es sehr einfach. Man multipliziert einfach die Zeilennummer mit der Zeilenlänge.

Wenn eine Tabelle gescannt wird, ist es ausserdem sehr einfach, mit jedem Plattenzugriff eine konstante Anzahl von Datensätzen zu lesen.

Die Sicherheit zeigt sich, wenn Ihr Computer beim Schreiben in eine MyISAM-Datei fester Länge abstürzt. In diesem Fall kann myisamchk leicht herausfinden, wo jede Zeile anfängt und aufhört. Daher kann es üblicherweise alle Datensätze mit Ausnahme desjenigen, in den nur teilweise geschrieben wurde, wieder herstellen. Beachten Sie, dass in MySQL alle Indexe in jedem Fall wiederhergestellt werden können:

- Alle CHAR-, NUMERIC- und DECIMAL-Spalten werden mit Leerzeichen auf die Spaltenbreite aufgefüllt.
- Sehr schnell.
- Leicht zu cachen.
- Nach einem Absturz leicht zu rekonstruieren, weil sich Datensätze an festen Positionen befinden.
- müssen nicht (mit myisamchk) reorganisiert werden, es sei denn, eine riesige Anzahl von Datensätzen wurde gelöscht und Sie wollen dem Betriebssystem freien Speicherplatz zurückgeben.
- Benötigen normalerweise mehr Speicherplatz als dynamische Tabellen.

8.1.2.2. Kennzeichen dynamischer Tabellen

Dieses Format wird benutzt, wenn die Tabelle irgend welche VARCHAR-, BLOB- oder TEXT-Spalten enthält, oder wenn die Tabelle mit ROW_FORMAT=dynamic erzeugt wurde.

Dieses Format ist etwas komplexer, weil jede Zeile einen Header haben muss, der aussagt, wie lang sie ist. Ein Datensatz kann ausserdem an mehr als einem Speicherplatz enden, wenn er bei einer Aktualisierung verlängert wird.

Sie können OPTIMIZE table oder myisamchk benutzen, um eine Tabelle zu defragmentieren. Wenn Sie statische Daten haben, auf die Sie oft zugreifen oder die Sie in derselben Tabelle oft ändern, als VARCHAR- oder BLOB-Spalten haben, ist es eine gute Idee, die dynamischen Spalten in andere Tabellen zu verschieben, einfach um Fragmentierung zu vermeiden:

- Alle Zeichenketten-Spalten sind dynamisch (ausser denen mit einer Länge kleiner 4).
- Jedem Datensatz ist eine Bitmap vorangestellt, die angibt, welche Spalten bei Zeichenketten-Spalten leer ('') sind oder 0 bei numerischen Spalten. (Das ist nicht dasselbe wie Spalten, die NULL-Werte enthalten.) Wenn eine Zeichenketten-Spalte nach der Entfernung von Leerzeichen am Ende eine Länge von 0 hat oder eine numerische Spalte einen Wert von 0 hat, wird sie in der Bitmap markiert und nicht auf Platte gespeichert. Nicht leere Zeichenketten werden als ein Längen-Byte plus dem Zeichenketten-Inhalt gespeichert.

- Benötigen üblicherweise weniger Plattenplatz als Festlängen-Tabellen.
- Jeder Datensatz benutzt nur so viel Speicherplatz wie erforderlich. Wenn ein Datensatz größer wird, wird er in so viele Teile wie erforderlich aufgeteilt. Hierdurch wird Datensatzfragmentierung hervorgerufen.
- Wenn Sie eine Zeile mit Informationen aktualisieren, die die Zeilenlänge überschreiten, wird die Zeile fragmentiert. In diesem Fall sollten Sie von Zeit zu Zeit `myisamchk -r` laufen lassen, um bessere Performance zu erzielen. Benutzen Sie `myisamchk -ei tabellen_name`, um einige statistische Informationen zu erhalten.
- Sind nach einem Absturz nicht so einfach zu rekonstruieren, weil ein Datensatz in viele Teile fragmentiert sein und ein Link (Fragment) fehlen kann.
- Die erwartete Zeilenlänge bei Datensätzen dynamischer Länge ist:

```
3
+ (anzahl_der_spalten + 7) / 8
+ (anzahl_der_zeichenketten_spalten)
+ komprimierte_groesse_numerischer_spalten
+ laenge_von_zeichenketten
+ (anzahl_von_NULL_spalten + 7) / 8
```

  Für jeden Link kommen 6 Bytes hinzu. Ein dynamischer Datensatz wird immer dann verknüpft (linked), wenn eine Aktualisierung eine Vergrößerung des Datensatzes bewirkt. Jede neue Verknüpfung hat mindestens 20 Bytes, so dass die nächste Vergrößerung wahrscheinlich in dieselbe Verknüpfung passt. Wenn nicht, entsteht eine weitere Verknüpfung. Sie können mit `myisamchk -ed` prüfen, wie viele Verknüpfungen es gibt. Alle Verknüpfungen können mit `myisamchk -r` entfernt werden.

8.1.2.3. Kennzeichen komprimierter Tabellen

Das ist ein Nur-Lese-Typ, der mit dem optionalen `myisampack`-Dienstprogramm (`pack_isam` für `ISAM`-Tabellen) erzeugt wird:

- All MySQL-Distributionen, selbst diejenigen, die es vor der GPL-Version von MySQL gab, können Tabellen lesen, die mit `myisampack` komprimiert wurden.
- Komprimierte Tabellen benötigen viel weniger Speicherplatz. Das minimiert Plattenzugriffe, was sehr nett ist, wenn Sie langsame Platten benutzen (wie CD-ROMs).
- Jeder Datensatz wird separat komprimiert (sehr geringer Zugriffs-Overhead). Der Header für einen Datensatz hat eine feste Länge (1 bis 3 Bytes), abhängig vom größten Datensatz in der Tabelle. Jede Spalte wird unterschiedlich komprimiert. Einige Kompressionstypen sind:
  - Für jede Spalte gibt es üblicherweise eine unterschiedliche Huffman-Tabelle.
  - Komprimierung von Leerzeichen am Ende.
  - Komprimierung von Leerzeichen am Anfang.
  - Zahlen mit dem Wert `0` werden mit 1 Bit gespeichert.
  - Wenn Werte in einer Ganzzahl-Spalte einen kleinen Wertebereich haben, wird die Spalte mit dem kleinsten möglichen Typ gespeichert. Eine `BIGINT`-Spalte (8 Bytes) kann beispielsweise als `TINYINT`-Spalte (1 Byte) gespeichert werden, wenn sich alle Werte im Bereich von `0` bis `255` befinden.
  - Wenn eine Spalte nur einen kleinen Satz möglicher Werte besitzt, wird der Spaltentyp zu `ENUM` umgewandelt.
  - Eine Spalte kann auch eine Kombination der obigen Komprimierungen benutzen.
- Kann Datensätze fester oder dynamischer Länge handhaben, aber nicht `BLOB`- oder `TEXT`-Spalten.
- Kann mit `myisamchk` dekomprimiert werden.

8.1.3. MyISAM-Tabellenprobleme

Das Dateiformat, das MySQL benutzt, um Daten zu speichern, wurde ausgiebig getestet, aber es gibt immer Umstände, die dazu führen können, dass Datenbanktabellen beschädigt werden.

8.1.3.1. Beschädigte MyISAM-Tabellen

Obwohl das MyISAM-Tabellenformat sehr zuverlässig ist (alle Änderungen an einer Tabelle werden geschrieben, bevor das SQL-Statement zurückkehrt), können Sie dennoch beschädigte Tabellen bekommen, wenn eines der folgenden Dinge passiert:

- Der `mysqld`-Prozess wird mitten in einem Schreibvorgang gekillt.
- Unerwartetes Herunterfahren des Computers (wenn der Computer beispielsweise abgeschaltet wird).
- Ein Hardware-Fehler.
- Sie benutzen ein externes Programm (wie myisamchk) auf einer benutzten Tabelle.
- Ein Software-Bug im MySQL- oder MyISAM-Code.

Typische Symptome einer beschädigten Tabelle sind:

- Sie erhalten den Fehler `Incorrect key file for table: '...'. Try to repair it`, wenn Sie Daten aus der Tabelle auswählen.
- Anfragen finden keine Zeilen in der Tabelle oder geben unvollständige Daten zurück.

Sie können mit dem Befehl `CHECK TABLE` prüfen, ob eine Tabelle in Ordnung ist. See Abschnitt 5.4.4, „`CHECK TABLE`-Syntax“.

Sie können eine beschädigte Tabelle mit `REPAIR TABLE` reparieren. See Abschnitt 5.4.5, „`REPAIR TABLE`-Syntax“. Wenn `mysqld` nicht läuft, können Sie eine Tabelle auch mit dem `myisamchk`-Befehl reparieren. `myisamchk-Syntax`.

Wenn Ihre Tabellen oft beschädigt werden, sollten Sie versuchen, den Grund dafür herauszufinden! See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“.

In diesem Fall ist es am wichtigsten zu wissen, ob die Tabelle durch einen Absturz von `mysqld` beschädigt wurde (das können Sie leicht feststellen, wenn es eine aktuelle Zeile `restarted mysqld` in der mysqld-Fehlerdatei gibt). Wenn das nicht der Fall ist, sollten Sie versuchen, daraus einen Testfall zu machen. See Abschnitt E.1.6, „Einen Testfall herstellen, wenn Sie Tabellenbeschädigung feststellen“.

8.1.3.2. Client benutzt Tabelle oder hat sie nicht korrekt geschlossen

Jede `MyISAM-.MYI`-Datei hat im Header einen Zähler, der benutzt werden kann, um zu prüfen, ob die Tabelle korrekt geschlossen wurde.

Wenn Sie folgende Warnmeldung von `CHECK TABLE` oder `myisamchk` erhalten:

```
# client is using or hasn't closed the table properly
```

heißt das, dass der Zähler nicht mehr synchron ist. Das bedeutet nicht, dass die Tabelle beschädigt ist, aber Sie sollten zumindest eine Überprüfung vornehmen, um sicherzustellen, dass die Tabelle in Ordnung ist.

Der Zähler funktioniert wie folgt:

- Wenn die Tabelle das erste Mal in MySQL aktualisiert wird, wird der Zähler im Header der Index-Dateien heraufgezählt.
- Der Zähler wird während weiterer Aktualisierungen nicht verändert.
- Wenn die letzte Instanz einer Tabelle geschlossen wird (wegen eines `FLUSH` oder weil es nicht mehr genug Platz im Tabellen-Cache gibt), wird der Zähler heruntergezählt, wenn die Tabelle zu irgend einem Zeitpunkt aktualisiert wurde.
- Wenn Sie eine Tabelle reparieren oder prüfen und sie in Ordnung ist, wird der Zähler auf 0 zurückgesetzt.
- Um Probleme zu vermeiden, die durch Interaktion mit anderen Prozessen entstehen, die vielleicht eine Prüfung der Tabelle durchführen, wird der Zähler beim Schließen nicht heruntergezählt, wenn er 0 war.

Mit anderen Worten kann der Zähler nur in folgenden Fällen nicht mehr synchron sein:

- Die `MyISAM`-Tabellen werden ohne `LOCK` und `FLUSH TABLES` kopiert.

- MySQL ist zwischen einer Aktualisierung und dem endgültigen Schließen abgestürzt. (Beachten Sie, dass die Tabelle trotzdem in Ordnung sein kann, weil MySQL stets für alles zwischen jedem Statement Schreibvorgänge durchführt.
- Jemand hat `myisamchk --repair` oder `myisamchk --update-state` auf eine Tabelle ausgeführt, die durch `mysqld` in Benutzung war.
- Viele `mysqld`-Server benutzen die Tabelle und einer davon hat `REPAIR` oder `CHECK` der Tabelle ausgeführt, während sie durch einen anderen Server in Benutzung war. Hierbei kann `CHECK` sicher ausgeführt werden (selbst wenn Sie Warnungen von anderen Servern erhalten werden), aber `REPAIR` sollte vermieden werden, weil es momentan die Daten-Datei durch eine neue ersetzt, was anderen Servern nicht signalisiert wird.

8.2. MERGE-Tabellen

`MERGE`-Tabellen sind neu seit MySQL-Version 3.23.25. Der Code ist noch Gamma, sollte aber ausreichend stabil sein.

Eine `MERGE`-Tabelle (auch bekannt als `MRG_MyISAM`-Tabelle) ist eine Sammlung identischer `MyISAM`-Tabellen, die wie eine benutzt werden können. Sie können auf dieser Sammlung von Tabellen nur `SELECT`, `DELETE` und `UPDATE` ausführen. Wenn Sie eine `MERGE`-Tabelle löschen (`DROP`), löschen Sie nur die `MERGE`-Spezifikation.

Beachten Sie, dass `DELETE FROM merge_tabelle` ohne `WHERE` nur das Mapping für die Tabelle löscht, nicht alles in den gemappten Tabellen. (Geplant ist, das in Version 4.1 zu beheben.)

Mit identischen Tabellen ist gemeint, dass alle Tabellen mit identischen Spalten- und Schlüsselinformationen erzeugt wurden. Sie können kein MERGE auf Tabellen ausführen, deren Spalten unterschiedlich komprimiert sind, nicht genau dieselben Spalten oder die Schlüssel in unterschiedlicher Reihenfolge haben. Einige der Tabellen können jedoch mit `myisampack` komprimiert sein. See Abschnitt 5.7.4, „myisampack, MySQL-Programm zum Erzeugen komprimierter Nur-Lese-Tabellen“.

Wenn Sie eine `MERGE`-Tabelle erzeugen, erhalten Sie eine `.frm`-Tabellendefinitionsdatei und eine `.MRG`-Tabellenlistendatei. Die `.MRG` enthält lediglich eine Liste der Index-Dateien (`.MYI`-Dateien), die wie eine benutzt werden sollen. Alle benutzten Tabellen müssen in derselben Datenbank wie die `MERGE`-Tabelle selbst sein.

Momentan benötigen Sie `SELECT`-, `UPDATE`- und `DELETE`-Berechtigungen für die Tabellen, die Sie auf eine `MERGE`-Tabelle mappen.

`MERGE`-Tabellen können bei der Lösung folgender Probleme helfen:

- Auf einfache Weise einen Satz von Log-Tabellen verwalten. Beispielsweise können Sie Daten aus unterschiedlichen Monaten in separaten Dateien speichern, einige davon mit `myisampack` komprimieren und dann eine `MERGE`-Tabelle erzeugen, um sie wie eine zu benutzen.
- Mehr Geschwindigkeit. Sie können eine große Nur-Lese-Tabelle nach bestimmten Kriterien aufspalten und die verschiedenen Tabellenteile auf unterschiedlichen Festplatten speichern. Eine `MERGE`-Tabelle darauf könnte viel schneller sein als die große Tabelle zu benutzen. (Natürlich können Sie auch ein RAID benutzen, um dieselben Vorteile zu erzielen.)
- Effizientere Suchen durchführen. Wenn Sie genau wissen, wonach Sie suchen, können Sie mit einigen Anfragen in lediglich einer der aufgespaltenen Tabellen suchen und die **MERGE**-Tabelle für andere benutzen. Es können sogar viele unterschiedliche `MERGE`-Tabellen aktiv sein, möglicherweise mit Dateien, die sich überlappen.
- Effizientere Reparaturen durchführen. Es ist leichter, die individuellen Dateien zu reparieren, die auf eine `MERGE`-Datei gemappt sind, als eine wirklich große Datei zu reparieren.
- Sofortiges Mappen vieler Dateien als einer. Eine `MERGE`-Tabelle benutzt den Index der individuellen Tabellen. Sie muss selbst keinen eigenen Index warten. Dadurch können Sie `MERGE`-Tabellensammlungen SEHR schnell erzeugen oder neu mappen. Beachten Sie, dass Sie die Schlüsseldefinitionen angeben, wenn Sie eine `MERGE`-Tabelle erzeugen!
- Wenn Sie einen Satz von Tabellen bei Bedarf oder im Stapel zu einer großen Tabelle vereinigen, sollten Sie statt dessen bei Bedarf eine `MERGE`-Tabelle darauf erzeugen. Das ist viel schneller und spart eine Menge Speicherplatz.
- Umgehen der Dateigrößengrenze des Betriebssystems.
- Sie können ein Alias / Synonym für eine Tabelle erzeugen, indem Sie sie einfach ein MERGE über eine Tabelle benutzen. Das sollte keine spürbaren Performance-Auswirkungen haben (nur eine Reihe indirekter Aufrufen und memcpy's bei jedem Lesen).

Die Nachteile von `MERGE`-Tabellen sind:

- Sie können nur identische `MyISAM`-Tabellen für eine `MERGE`-Tabelle benutzen.

- `AUTO_INCREMENT`-Spalten werden bei `INSERT` nicht automatisch aktualisiert.
- `REPLACE` funktioniert nicht.
- `MERGE`-Tabellen benutzen mehr Datei-Deskriptoren. Wenn Sie eine **MERGE** benutzen, die über 10 Tabellen mappt, und 10 Benutzer diese benutzen, benötigen Sie 10 * 10 + 10 Datei-Deskriptoren (10 Daten-Dateien für 10 Benutzer und 10 gemeinsam genutzte Index-Dateien).
- Lesevorgänge von Schlüsseln sind langsamer. Wenn Sie eine Leseoperation auf einen Schlüssel durchführen, muss der `MERGE`-Handler ein Lesen auf alle zugrunde liegenden Tabellen ausführen, um zu prüfen, welche am nächsten zum angegebenen Schlüssel passt. Wenn Sie ein 'Lese nächsten' ausführen, muss der `MERGE`-Handler die Lese-Puffer durchsuchen, um den nächsten Schlüssel zu finden. Erst wenn ein Schlüsselpuffer aufgebraucht ist, muss der Handler den nächsten Schlüsselblock lesen. Das macht `MERGE`-Schlüssel bei `eq_ref`-Suchen viel langsamer, aber nicht viel langsamer bei `ref`-Suchen. See Abschnitt 6.2.1, „`EXPLAIN`-Syntax (Informationen über ein `SELECT` erhalten)“.
- Sie können kein `DROP TABLE`, `ALTER TABLE` oder `DELETE FROM tabelle` ohne eine `WHERE`-Klausel auf jeder Tabelle, die von einer `MERGE`-Tabelle gemappt ist, ausführen, wenn diese 'offen' ist. Wenn Sie das tun, könnte die `MERGE`-Tabelle immer noch auf die Originaltabelle verweisen, und Sie würden unerwartete Ergebnisse erhalten.

Wenn Sie eine `MERGE`-Tabelle erzeugen, müssen Sie mit `UNION(liste-von-tabellen)` angeben, welche Tabellen Sie wie eine benutzen wollen. Optional können Sie mit `INSERT_METHOD` angeben, ob Sie wollen, dass Einfügungen in die `MERGE`-Tabelle in der ersten oder der letzten Tabelle in der `UNION`-Liste geschehen sollen. Wenn Sie keine `INSERT_METHOD` oder `NO` angeben, geben alle `INSERT`-Befehle auf die `MERGE`-Tabelle einen Fehler zurück.

Folgendes Beispiel zeigt, wie Sie `MERGE`-Tabellen benutzen:

```
CREATE TABLE t1 (a INT AUTO_INCREMENT PRIMARY KEY, nachricht CHAR(20));
CREATE TABLE t2 (a INT AUTO_INCREMENT PRIMARY KEY, nachricht CHAR(20));
INSERT INTO t1 (nachricht) VALUES ("test"),("tabelle"),("t1");
INSERT INTO t2 (nachricht) VALUES ("test"),("tabelle"),("t2");
CREATE TABLE gesamt (a INT NOT NULL, nachricht CHAR(20), KEY(a)) TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

Beachten Sie, dass wir keinen `UNIQUE`- oder `PRIMARY KEY`-Schlüssel in der `gesamt`-Tabelle angegeben haben, weil der Schlüssel in der `gesamt`-Tabelle nicht eindeutig sein wird.

Beachten Sie auch, dass Sie die `.MRG`-Datei direkt von ausserhalb des MySQL-Servers manipulieren können:

```
shell> cd /mysql-data-verzeichnis/aktuelle-datenbank
shell> ls -1 t1.MYI t2.MYI > gesamt.MRG
shell> mysqladmin flush-tables
```

Jetzt können Sie Dinge wie folgendes tun:

```
mysql> select * from gesamt;
+---+-----------+
| a | nachricht |
+---+-----------+
| 1 | test      |
| 2 | table     |
| 3 | t1        |
| 1 | test      |
| 2 | table     |
| 3 | t2        |
+---+-----------+
```

Um eine `MERGE`-Tabelle neu zu mappen, können Sie folgendes tun:

- Die Tabelle löschen (`DROP`) und neu erzeugen.
- `ALTER TABLE tabelle UNION(...)` benutzen.
- Die `.MRG`-Datei ändern und ein `FLUSH TABLE` auf die `MERGE`-Tabelle und alle zugrunde liegenden Tabellen ausführen, um den Handler zu zwingen, die neue Definitionsdatei einzulesen.

8.2.1. MERGE-Tabellenprobleme.

Folgende Probleme sind bei `MERGE`-Tabellen bekannt:

- `DELETE FROM merge_tabelle` ohne `WHERE` löscht nur das Mapping für die Tabelle, nicht alles in den gemappten Tabellen.

- `RENAME TABLE` auf eine Tabelle, die in einer aktiven `MERGE`-Tabelle benutzt wird, kann die Tabelle beschädigen. Das wird in MySQL 4.0.x behoben.
- Beim Erzeugen einer Tabelle des Typs `MERGE` wird nicht geprüft, ob die zugrunde liegenden Tabellen kompatible Typen sind. Wenn Sie `MERGE`-Tabellen in dieser Weise benutzen, ist es sehr wahrscheinlich, dass merkwürdige Probleme auftauchen.
- Wenn Sie `ALTER TABLE` benutzen, um als erstes eine `UNIQUE`-Index zu einer Tabelle hinzuzufügen, die in einer `MERGE`-Tabelle benutzt wird, und dann `ALTER TABLE` benutzen, um einen normalen Index auf die `MERGE`-Tabelle hinzuzufügen, wird die Schlüssel-Reihenfolge für die Tabellen anders sein, wenn es einen alten, nicht eindeutigen Schlüssel in der Tabelle gab. Das liegt daran, dass `ALTER TABLE` `UNIQUE`-Schlüssel vor normale Schlüssel einfügt, um in der Lage zu sein, doppelte Schlüsseleinträge so früh wie möglich zu erkennen.
- Der Bereichsoptimierer kann `MERGE`-Tabellen noch nicht effizient benutzen und kann manchmal nicht optimale Joins produzieren. Das wird in MySQL 4.0.x behoben.
- `DROP TABLE` auf eine Tabelle, die in einer `MERGE`-Tabelle benutzt wird, funktioniert unter Windows nicht, weil der `MERGE`-Handler das Tabellen-Mapping versteckt vor der oberen Ebene von MySQL durchführt. Weil Windows es nicht zuläßt, dass Dateien gelöscht werden, die offen sind, müssen Sie zuerst alle `MERGE`-Tabellen auf Platte zurückschreiben (mit `FLUSH TABLES`) oder die `MERGE`-Tabelle löschen, bevor Sie die Tabelle löschen. Das wird zu dem Zeitpunkt behoben, wenn Sichten (`VIEW`s) eingeführt werden.

8.3. ISAM-Tabellen

Sie können auch den veralteten `ISAM`-Tabellentyp benutzen. Dieser wird recht bald verschwinden (wahrscheinlich in MySQL 4.1), weil `MyISAM` eine bessere Implementation derselbe Sache ist. `ISAM` benutzt einen `B-tree`-Index. Der Index wird in einer Datei mit der Endung `.ISM` gespeichert, und die Daten in einer Datei mit der Endung `.ISD`. Sie können `ISAM`-Tabellen mit dem `isamchk`-Dienstprogramm prüfen / reparieren. See Abschnitt 5.4, „Katastrophenschutz und Wiederherstellung“.

`ISAM` hat folgende Features / Eigenschaften:

- Komprimierte und Festlängen-Schlüssel
- Feste und dynamische Datensatzlängen
- 16 Schlüssel mit 16 Schlüsselteilen pro Schlüssel
- Maximale Schlüssellänge 256 (Vorgabe)
- Daten werden im Maschinenformat gespeichert. Das ist schnell, aber Maschinen- / Betriebssystem-abhängig.

Die meisten Dinge, die für `MyISAM`-Tabellen gelten, gelten auch für `ISAM`-Tabellen. See Abschnitt 8.1, „MyISAM-Tabellen“. Die größten Unterschiede im Vergleich zu `MyISAM` sind:

- `ISAM`-Tabellen sind nicht binärportabel zwischen verschiedenen Betriebssystemen / Plattformen.
- Handhabt keine Tabellen > 4 GB.
- Unterstützt nur Präfix-Komprimierung von Zeichenketten.
- Kleinere Schlüssel-Beschränkungen.
- Dynamische Tabelle werden schneller fragmentiert.
- Tabellen werden mit `pack_isam` statt mit `myisampack` komprimiert.

Wenn Sie eine `ISAM`-Tabelle in eine `MyISAM`-Tabelle umwandeln wollen, können Sie Dienstprogramme wie `mysqlcheck` oder ein `ALTER TABLE`-Statement benutzen:

```
mysql> ALTER TABLE tabelle TYPE = MYISAM;
```

Die eingebetteten (embedded) MySQL-Versionen unterstützen keine `ISAM`-Tabellen.

8.4. HEAP-Tabellen

`HEAP`-Tabellen benutzen eine gehashten Index und werden im Arbeitsspeicher gespeichert. Das macht sie sehr schnell, aber wenn

MySQL abstürzt, verlieren Sie alle darin gespeicherten Daten. HEAP ist sehr nützlich für temporäre Tabellen.

Die MySQL-internen HEAP-Tabellen benutzen 100% dynamisches Hashen ohne Overflow-Bereiche. Es wird kein zusätzlicher Platz für freie Listen benötigt. HEAP-Tabellen haben auch keine Probleme mit Löschen plus Einfügen, was normalerweise bei gehashten Tabellen häufig vorkommt:

```
mysql> CREATE TABLE test TYPE=HEAP SELECT ip,SUM(downloads) as down
        FROM log_tabelle GROUP BY ip;
mysql> SELECT COUNT(ip),AVG(down) FROM test;
mysql> DROP TABLE test;
```

Einige Dinge sollten Sie bei der Benutzung von HEAP-Tabellen in Betracht ziehen:

- Sie sollten immer MAX_ROWS im CREATE-Statement angeben, um sicherzustellen, dass Sie nicht versehentlich den gesamten Arbeitsspeicher benutzen.
- Indexe werden nur bei = und <=> benutzt (sind aber SEHR schnell).
- HEAP-Tabellen können nur ganze Schlüssel benutzen, um nach einer Zeile zu suchen. Vergleichen Sie das mit MyISAM-Tabellen, bei denen jedes Präfix des Schlüssels für das Suchen von Zeilen benutzt werden kann.
- HEAP-Tabellen benutzen ein festes Datensatzlängenformat.
- HEAP unterstützt keine BLOB/TEXT-Spalten.
- HEAP unterstützt keine AUTO_INCREMENT-Spalten.
- HEAP unterstützt keinen Index auf eine NULL-Spalte.
- Es darf keine nicht eindeutigen Schlüssel auf eine HEAP-Tabelle geben (das ist ungebräuchlich für gehashte Tabellen).
- HEAP-Tabellen werden von allen Clients gemeinsam benutzt (so wie jede andere Tabelle).
- Sie können nicht nach dem nächsten Eintrag in der Reihenfolge suchen (also den Index benutzen, um ein ORDER BY zu machen).
- Die Daten für HEAP-Tabellen werden in kleinen Blöcken zugewiesen. Die Tabellen sind 100% dynamisch (beim Einfügen). Es werden keine Overflow-Bereiche und kein zusätzlicher Platz für Schlüssel benötigt. Gelöschte Zeilen werden in eine verknüpfte Liste geschrieben und wieder benutzt, wenn Sie neue Daten in die Tabelle einfügen.
- Sie brauchen genug zusätzlichen Arbeitsspeicher für alle HEAP-Tabellen, die Sie zugleich benutzen wollen.
- Um Speicher freizugeben, führen Sie DELETE FROM heap_tabelle, TRUNCATE heap_tabelle oder DROP TABLE heap_tabelle aus.
- MySQL kann nicht herausfinden, wie viele Zeilen es zwischen zwei Werten ungefähr gibt (das wird vom Bereichsoptimierer benötigt, um zu entscheiden, welcher Index benutzt wird). Das kann einige Anfragen betreffen, wenn Sie eine MyISAM-Tabelle in eine HEAP-Tabelle umwandeln.
- Um sicherzustellen, dass Sie nicht versehentlich etwas Unkluges tun, können Sie keine HEAP-Tabellen größer als max_heap_table_size erzeugen.

Der für eine Zeile in einer HEAP-Tabelle benötigte Speicher ist:

```
SUM_OVER_ALL_KEYS(max_length_of_key + sizeof(char*) * 2)
+ ALIGN(length_of_row+1, sizeof(char*))
```

sizeof(char*) ist 4 auf 32-Bit-Maschinen und 8 auf 64-Bit-Maschinen.

8.5. InnoDB-Tabellen

8.5.1. Überblick über InnoDB-Tabellen

InnoDB stellt MySQL einen transaktionssicheren (ACID-kompatiblen) Tabellen-Handler mit Fähigkeiten für Commit, Rollback und Reparatur nach Absturz zur Verfügung. InnoDB beherrscht Sperren auf Zeilenebene sowie ein konsistentes, nicht sperrendes Lesen in der Art von Oracle bei SELECTs. Diese Features steigern die Handhabung gleichzeitiger Verbindungen und die Performance. Es gibt bei InnoDB keine Notwendigkeit für Sperr-Eskalation, weil die Sperren auf Zeilenebene bei InnoDB in sehr wenig Speicherplatz passen. InnoDB-Tabellen unterstützen als erster Tabellentyp in MySQL FOREIGN KEY-Beschränkungen.

InnoDB wurde für maximale Performance bei der Bearbeitung großer Datenmengen entworfen. Seine Prozessor-Effizienz wird wahrscheinlich von keiner anderen Festplatten-basierenden relationalen Datenbank-Engine erreicht.

Technisch gesehen ist InnoDB ein komplettes Datenbank-Backend, das unter MySQL platziert ist. InnoDB hat seinen eigenen Puffer-Pool, um Daten und Indexe im Hauptspeicher zu cachen. InnoDB speichert seine Tabellen und Indexe in einem Tabellenplatz (Tablespace), der aus mehreren Dateien bestehen kann. Das unterscheidet sich beispielsweise von MyISAM-Tabellen, bei denen jede Tabelle als separate Datei gespeichert ist. InnoDB-Tabellen können jede beliebige Größe annehmen, sogar auf Betriebssystemen, deren Dateigröße auf 2 GB beschränkt ist.

Die neuesten Informationen über InnoDB finden Sie unter http://www.innodb.com/. Die aktuellste Version des InnoDB-Handbuchs ist immer dort zu finden, und Sie können auch kommerzielle Lizenzen und kommerziellen Support für InnoDB bestellen.

InnoDB wird momentan (Oktober 2001) für die Produktion auf mehreren großen Datenbank-Sites benutzt, die hohe Performance benötigen. Die bekannte Internet-Newssite Slashdot.org läuft auf InnoDB. Mytrix Inc. speichert über 1 TB an Daten in InnoDB, und eine andere Site handhabt eine durchschnittliche Last von 800 Einfüge- und Update-Operationen pro Sekunde mit InnoDB.

InnoDB-Tabellen sind in der MySQL-Quelldistribution ab Version 3.23.34a enthalten und in der MySQL-Max-Binärversion aktiviert. Für Windows sind die Max-Binärdateien in der Standarddistribution enthalten.

Wenn Sie eine Binärversion von MySQL herunter geladen haben, die Unterstützung für InnoDB enthält, folgen Sie einfach den Anweisungen im Handbuch für die Installation einer Binärversion von MySQL. Wenn Sie bereits MySQL-3.23 installiert haben, können Sie MySQL-Max am einfachsten installieren, indem Sie die ausführbare Datei für den Server (`mysqld`) durch die entsprechende ausführbare Datei in der Max-Distribution ersetzen. MySQL and MySQL-Max unterscheiden sich nur in Bezug auf die ausführbare Datei für den Server. See Abschnitt 3.2.6, „MySQL-Binärdistributionen, die von MySQL AB kompiliert wurden“. See Abschnitt 5.7.5, „mysqld-max, ein erweiterter mysqld-Server“.

Um MySQL mit InnoDB-Unterstützung zu kompilieren, laden Sie MySQL-3.23.34a oder neuer von http://www.mysql.com/ herunter und konfigurieren Sie MySQL mit der `--with-innodb`-Option. Sehen Sie im Handbuch unter Abschnitt 3.3, „Installation der Quelldistribution“ nach.

```
cd /pfad/zur/quelldistribution/von/mysql-3.23.37
./configure --with-innodb
```

Um InnoDB zu benutzen, müssen Sie InnoDB init in Ihrer `my.cnf`- oder `my.ini`-Datei angeben. In dieser Datei müssen Sie mindestens folgenden Zeile im `[mysqld]`-Abschnitt hinzufügen:

```
innodb_data_file_path=ibdata:30M
```

Für eine gute Performance ist es jedoch am besten, Optionen wie die unten im Abschnitt Abschnitt 8.5.2, „Mit InnoDB anfangen - Optionen“ empfohlenen anzugeben.

InnoDB wird unter der GNU-GPL-Lizenz Version 2 (vom Juni 1991) vertrieben. In den Quelldistributionen von MySQL erscheint InnoDB als Unterverzeichnis.

8.5.2. Mit InnoDB anfangen - Optionen

Um InnoDB-Tabellen in MySQL-Max-3.23 zu benutzen, **MÜSSEN** Sie Konfigurationsparameter im `[mysqld]`-Abschnitt der MySQL-Konfigurationsdatei `my.cnf` angeben. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

Der einzige erforderliche Parameter, um InnoDB in MySQL-Max-3.23 benutzen zu können, ist `innodb_data_file_path`. In MySQL-4.0 müssen Sie nicht einmal `innodb_data_file_path` angeben. Vorgabemäßig wird eine 64 MB große Daten-Datei `ibdata1` im `datadir` von MySQL erzeugt.

Um jedoch eine gute Performance zu erzielen, MÜSSEN Sie explizit die unten in Beispielen aufgeführten InnoDB-Parameter setzen.

Der Vorgabewert für `innodb_data_home_dir` ist das `datadir` von MySQL. Wenn Sie `innodb_data_home_dir` nicht angeben, können Sie in `innodb_data_file_path` keine absoluten Pfade benutzen.

Nehmen wir an, Sie haben eine Windows-NT-Maschine mit 128 MB RAM und einer einzelnen 10 GB großen Festplatte. Unten steht ein Beispiel von möglichen Konfigurationsparametern in `my.cnf` für InnoDB:

```
[mysqld]
# Hier können Ihre sonstigen MySQL-Serveroptionen stehen
# ...
#
innodb_data_home_dir = c:\ibdata
# Die Daten-Dateien müssen in der Lage sein,
# Ihre Daten und Indexe aufzunehmen
innodb_data_file_path = ibdata1:2000M;ibdata2:2000M
# Puffer-Poolgröße auf 50% bis 80%
# des Arbeitsspeichers Ihres Computers setzen
set-variable = innodb_buffer_pool_size=70M
```

```
set-variable = innodb_additional_mem_pool_size=10M
innodb_log_group_home_dir = c:\iblogs
# ..._log_arch_dir muss dasselbe sein wie
# ..._log_group_home_dir
innodb_log_arch_dir = c:\iblogs
innodb_log_archive=0
set-variable = innodb_log_files_in_group=3
# Die Log-Dateigröße auf ungefähr 15%
# der Puffer-Poolgröße setzen
set-variable = innodb_log_file_size=10M
set-variable = innodb_log_buffer_size=8M
# ..flush_log_at_trx_commit auf 0 setzen,
# wenn Sie es sich leisten können,
# ein paar der letzten Transaktionen zu verlieren
innodb_flush_log_at_trx_commit=1
set-variable = innodb_file_io_threads=4
set-variable = innodb_lock_wait_timeout=50
```

Beachten Sie, dass die Daten-Dateien bei einigen Betriebssystemen kleiner als 2 GB sein müssen! Die Gesamtgröße von Daten-Dateien muss größer oder gleich 10 MB sein. Die Gesamtgröße der Log-Dateien MUSS auf 32-Bit-Computern kleiner als 4 GB sein.

InnoDB legt keine Verzeichnisse an. Diese müssen Sie selbst erzeugen! Stellen Sie auch sicher, dass der MySQL-Server Rechte hat, Dateien in den Verzeichnissen anzulegen, die Sie angeben.

Wenn Sie zum ersten Mal eine InnoDB-Datenbank erzeugen, sollten Sie den MySQL-Server am besten von der Kommandozeilen-Eingabeaufforderung starten. InnoDB gibt dann Informationen über die Datenbank-Erzeugung auf dem Bildschirm aus und Sie sehen, was passiert. Unten in Abschnitt 3 sehen Sie, wie die Ausgaben aussehen sollten. Unter Windows können Sie `mysqld-max.exe` so starten:

```
ihr-pfad-zu-mysqld>mysqld-max --standalone --console
```

Nehmen wir an, Sie haben einen Linux-Computer mit 512 MB RAM und drei Festplatten mit jeweils 20 GB (in Verzeichnispfaden `/`, `/dr2` and `/dr3`). Unten ist ein Beispiel möglicher Konfigurationsparameter in `my.cnf` für InnoDB:

```
[mysqld]
# Hier können Ihre sonstigen MySQL-Serveroptionen stehen
# ...
#
innodb_data_home_dir = /
# Die Daten-Dateien müssen in der Lage sein,
# Ihre Daten und Indexe aufzunehmen
innodb_data_file_path = ibdata/ibdata1:2000M;dr2/ibdata/ibdata2:2000M
# Puffer-Poolgröße auf 50% bis 80%
# des Arbeitsspeichers Ihres Computers setzen
set-variable = innodb_buffer_pool_size=350M
set-variable = innodb_additional_mem_pool_size=20M
innodb_log_group_home_dir = /dr3/iblogs
# ..._log_arch_dir muss dasselbe sein wie
# ..._log_group_home_dir
innodb_log_arch_dir = /dr3/iblogs
innodb_log_archive=0
set-variable = innodb_log_files_in_group=3
# Die Log-Dateigröße auf ungefähr 15%
# der Puffer-Poolgröße setzen
set-variable = innodb_log_file_size=50M
set-variable = innodb_log_buffer_size=8M
# ..flush_log_at_trx_commit auf 0 setzen,
# wenn Sie es sich leisten können,
# ein paar der letzten Transaktionen zu verlieren
innodb_flush_log_at_trx_commit=1
set-variable = innodb_file_io_threads=4
set-variable = innodb_lock_wait_timeout=50
#innodb_flush_method=fdatasync
#innodb_fast_shutdown=1
#set-variable = innodb_thread_concurrency=5
```

Beachten Sie, dass die beiden Daten-Dateien auf unterschiedliche Platten platziert wurden. Der Grund für den Namen `innodb_data_file_path` ist, dass Sie auch Pfade zu Ihren Daten-Dateien angeben können und dass `innodb_data_home_dir` nur textlich mit Ihren Daten-Datei-Pfaden verkettet wird, wobei ein möglicher Schrägstrich oder Backslash dazwischen hinzugefügt wird. InnoDB füllt den Tabellenplatz (Tablespace), der durch die Daten-Dateien gebildet wird, von unten nach oben. In manchen Fällen verbessert es die Performance der Datenbank, wenn nicht alle Daten auf derselben physikalischen Festplatte platziert sind. Es verbessert häufig die Performance, Log-Dateien auf anderen Platten als die Daten zu platzieren.

Die Bedeutung der Konfigurationsparameter ist wie folgt:

Option	**Beschreibung**
`innodb_data_home_dir`	Der allgemeine Teil des Verzeichnispfads für alle InnoDB-Daten-Dateien. Die Vorgabe für diesen Parameter ist das `datadir` von MySQL.
`innodb_data_file_path`	Pfade zu individuellen Daten-Dateien und ihre Größen. Der volle Verzeichnispfad zu jeder

	Daten-Datei wird durch Verkettung von innodb_data_home_dir mit den hier angegebenen Pfaden hergestellt. Die Dateigrößen werden in Megabytes angegeben, daher das 'M' nach der obigen Angabe. InnoDB versteht auch die Abkürzung 'G', 1G bedeutet 1024M. Ab 3.23.44 können Sie die Dateigröße auf mehr als 4 GB setzen, wenn das Betriebssystem große Dateien unterstützt. Auf einige Betriebssystemen müssen Dateien kleiner als 2 GB sein. Die Summe der Dateigrößen muss mindestens 10 MB betragen.
`innodb_mirrored_log_groups`	Anzahl identischer Kopien von Log-Gruppen, die für die Datenbank gehalten werden. Momentan sollte dieser Parameter auf 1 gesetzt werden.
`innodb_log_group_home_dir`	Verzeichnispfad zu den InnoDB-Log-Dateien.
`innodb_log_files_in_group`	Anzahl von Log-Dateien in der Log-Gruppe. InnoDB schreibt in zirkulärer Weise in die Dateien. Hier wird ein Wert 3 empfohlen.
`innodb_log_file_size`	Größe jeder Log-Datei in einer Log-Gruppe in Megabytes. Sinnvolle Werte reichen von 1 MB bis 1/n-tel der Größe des Puffer-Pools, die unten angegeben wird, wobei n die Anzahl der Log-Dateien in der Gruppe ist. Je größer der Wert, desto weniger Checkpoint-Flush-Aktivität wird im Puffer benötigt, was Festplatten-Ein- und -Ausgaben erspart. Größere Log-Dateien bedeutet jedoch auch, dass die Wiederherstellung im Fall eines Absturzes langsamer ist. Die Gesamtgröße aller Log-Dateien muss auf 32-Bit-Computern kleiner als 4 GB sein.
`innodb_log_buffer_size`	Die Größe des Puffers, den InnoDB benutzt, um in die Log-Dateien auf Platte zu schreiben. Sinnvolle Werte liegen im Bereich von 1 MB bis zur Hälfte der Gesamtgröße der Log-Dateien. Ein großer Log-Puffer erlaubt, dass große Transaktionen laufen können, ohne dass die Notwendigkeit besteht, das Log auf Platte zu schreiben, bis die Transaktion abgeschickt (commit) wird. Wenn Sie daher große Transaktionen haben, sparen Sie Festplatten-Ein- und Ausgaben, wenn Sie den Log-Puffer Groß machen.
`innodb_flush_log_at_trx_commit`	Normalerweise wird dieser Parameter auf 1 gesetzt, was bedeutet, dass beim Abschicken (commit) einer Transaktion das Log auf Platte geschrieben wird (flush) und die durch die Transaktion gemachten Änderungen permanent werden und einen Datenbankabsturz überleben. Wenn Sie willens sind, in Bezug auf diese Sicherheit Kompromisse einzugeben und eher kleine Transaktionen laufen lassen, können Sie diesen Wert auf 0 setzen, um Festplatten-Ein- und -Ausgaben in Bezug auf die Log-Dateien zu verringern.
`innodb_log_arch_dir`	Das Verzeichnis, in dem komplett geschriebene Log-Dateien archiviert werden, wenn Archivierung benutzt wird. Der Wert dieses Parameters sollte momentan derselbe sein wie `innodb_log_group_home_dir`.
`innodb_log_archive`	Dieser Wert sollte momentan auf 0 gesetzt werden. Weil MySQL die Wiederherstellung aus einer Datensicherung unter Benutzung seiner eigenen Log-Dateien durchführt, gibt es momentan keine Notwendigkeit, InnoDB-Log-Dateien zu archivieren.
`innodb_buffer_pool_size`	Die Größe des Speicherpuffers, den InnoDB benutzt, um Daten und Indexe seiner Tabellen zu cachen. Je größer Sie diesen Wert setzen, desto weniger Festplatten-Ein- und -Ausgaben werden für den Zugriff auf Daten in Tabellen benötigt. Auf einem dedizierten Datenbank-Server können Sie diesen Parameter auf bis zu 80% des physikalischen Arbeitsspeichers der Maschine setzen. Setzen Sie ihn allerdings nicht zu hoch, weil bei manchen Betriebssystemen der Wettbewerb um Arbeitsspeicher zu Paging führt.
`innodb_additional_mem_pool_size`	Die Größe des Speicher-Pools, den InnoDB für die Speicherung von Daten-Wörterbuchinformationen und anderen internen Datenstrukturen benutzt. Ein sinnvoller Wert hierfür könnte 2 MB sein. Je mehr Tabellen Sie jedoch in Ihrer Applikation haben, desto mehr müssen Sie hier zuweisen. Wenn InnoDB in diesem Pool keinen Speicherplatz mehr hat, läßt es sich Speicherplatz vom Betriebssystem zuweisen und schreibt Warnmeldungen in die MySQL-Fehler-Log-Datei.
`innodb_file_io_threads`	Die Anzahl der Datei-Ein- und -Ausgabe-Threads in InnoDB. Normalerweise sollte dieser Wert 4 sein, aber Windows-Festplatten könnten von einer höheren Zahl profitieren.
`innodb_lock_wait_timeout`	Timeout in Sekunden. Solange wartet eine InnoDB-Transaktion auf eine Sperre, bevor sie abgebrochen (Rollback) wird. InnoDB erkennt automatisch Transaktionsblockierungen in seiner eigenen Sperr-Tabelle und bricht die Transaktion ab (Rollback). Wenn Sie den `LOCK TABLES`-Befehl oder andere transaktionssichere Tabellen-Handler als InnoDB in derselben Transaktion benutzen, kann eine Blockierung auftreten, die InnoDB nicht erkennen kann. In solchen Fällen ist ein Timeout nützlich, um die Situation zu bereinigen.
`innodb_flush_method`	(Verfügbar ab Version 3.23.40.) Der Vorgabewert hierfür ist `fdatasync`. Ein andere Option ist `O_DSYNC`.

8.5.3. InnoDB-Tabellenplatz (Tablespace) erzeugen

Angenommen, Sie haben MySQL installiert und `my.cnf` so editiert, dass sie die notwendigen InnoDB Konfigurationsparameter

enthält. Bevor Sie MySQL starten, sollten Sie überprüfen, dass die für InnoDB-Daten- und Log-Dateien angegebenen Verzeichnisse existieren und dass Sie auf diese Zugriffsrechte haben. InnoDB kann keine Verzeichnisse anlegen, nur Dateien. Überprüfen Sie auch, ob Sie auf der Festplatte genug Platz für Daten- und Log-Dateien haben.

Wenn Sie jetzt MySQL starten, fängt InnoDB an, Ihre Daten- und Log-Dateien zu erzeugen. InnoDB gibt dabei etwas wie das folgende aus:

```
~/mysqlm/sql > mysqld
InnoDB: The first specified data file /home/stefan/data/ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file /home/stefan/data/ibdata1 size to 134217728
InnoDB: Database physically writes the file full: wait...
InnoDB: Data file /home/stefan/data/ibdata2 did not exist: new to be created
InnoDB: Setting file /home/stefan/data/ibdata2 size to 262144000
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file /home/stefan/data/logs/ib_logfile0 did not exist: new to be created
InnoDB: Setting log file /home/stefan/data/logs/ib_logfile0 size to 5242880
InnoDB: Log file /home/stefan/data/logs/ib_logfile1 did not exist: new to be created
InnoDB: Setting log file /home/stefan/data/logs/ib_logfile1 size to 5242880
InnoDB: Log file /home/stefan/data/logs/ib_logfile2 did not exist: new to be created
InnoDB: Setting log file /home/stefan/data/logs/ib_logfile2 size to 5242880
InnoDB: Started
mysqld: ready for connections
```

Jetzt wurde eine neue InnoDB-Datenbank erzeugt. Sie können sich mit den üblichen MySQL-Client-Programmen wie `mysql` mit dem MySQL-Server verbinden. Wenn Sie den MySQL-Server mit `mysqladmin shutdown` herunter fahren, gibt InnoDB etwa wie das folgende aus:

```
010321 18:33:34  mysqld: Normal shutdown
010321 18:33:34  mysqld: Shutdown Complete
InnoDB: Starting shutdown...
InnoDB: Shutdown completed
```

Wenn Sie jetzt einen Blick auf die Daten-Dateien und Log-Verzeichnisse werfen, sehen Sie die erzeugten Dateien. Das Log-Verzeichnis enthält auch eine kleine Datei namens `ib_arch_log_0000000000`. Diese Datei resultiert aus der Datenbank-Erzeugung, nach der InnoDB die Log-Archivierung ausgeschaltet hat. Wenn MySQL noch einmal gestartet wird, sieht die Ausgabe etwa wie folgt aus:

```
~/mysqlm/sql > mysqld
InnoDB: Started
mysqld: ready for connections
```

8.5.3.1. Falls etwas bei der Datenbank-Erzeugung schiefgeht

Falls etwas bei der Datenbank-Erzeugung schiefgeht, sollten Sie alle durch InnoDB erzeugten Dateien löschen. Das heißt alle Daten-Dateien, alle Log-Dateien, die kleine archivierte Log-Datei und - falls Sie bereits InnoDB-Tabellen erzeugt haben, auch die entsprechenden `.frm`-Dateien für diese Tabellen in den MySQL-Datenbankverzeichnissen. Danach können Sie die InnoDB-Datenbankerzeugung erneut versuchen.

8.5.4. InnoDB-Tabellen erzeugen

Angenommen, Sie haben den MySQL-Client mit dem Befehl `mysql test` gestartet. Um eine Tabelle im InnoDB-Format zu erzeugen, müssen Sie im SQL-Befehl zur Tabellenerzeugung `TYPE = InnoDB` angeben:

```
CREATE TABLE kunde (A INT, B CHAR (20), INDEX (A)) TYPE = InnoDB;
```

Dieser SQL-Befehl erzeugt eine Tabelle und einen Index auf die Spalte `A` im InnoDB-Tabellenplatz (Tablespace), der aus den Daten-Dateien besteht, die Sie in `my.cnf` angegeben haben. MySQL erzeugt zusätzlich eine Datei `kunde.frm` im MySQL-Datenbankverzeichnis `test`. Intern fügt InnoDB seinem eigenen Datenwörterbuch einen Eintrag für die Tabelle `'test/kunde'` hinzu. Wenn Sie daher eine Tabelle namens `kunde` in einer anderen Datenbank von MySQL erzeugen, kollidieren die Tabellennamen innerhalb InnoDB nicht.

Sie können den freien Speicherplatz im InnoDB-Tabellenplatz (Tablespace) mit dem Tabellen-Status-Befehl von MySQL für jede Tabelle, die Sie mit `TYPE = InnoDB` erzeugt haben, abfragen. Die Menge freien Platzes im Tabellenplatz (Tablespace) erscheint im Kommentar-Abschnitt der Tabelle in der Ausgabe von `SHOW`. Beispiel:

```
SHOW TABLE STATUS FROM test LIKE 'kunde'
```

Beachten Sie, dass die Statistiken, die `SHOW` über InnoDB-Tabellen ausgibt, nur Näherungswerte sind: Sie werden für die SQL-Optimierung benutzt. Die für Tabelle und Indexe reservierten Größen in Bytes sind allerdings genau.

8.5.4.1. MyISAM-Tabellen in InnoDB-Tabellen umwandeln

InnoDB hat keine spezielle Optimierung für separate Index-Erzeugung. Daher lohnt es sich nicht, die Tabelle zu exportieren und importieren und die Indexe danach zu erzeugen. Die schnellste Art, eine Tabelle in InnoDB zu ändern, ist, die Einfügungen direkt in eine InnoDB-Tabelle vorzunehmen, das heißt, `ALTER TABLE ... TYPE=INNODB` zu benutzen oder eine leere InnoDB-Tabelle mit identischen Definitionen zu nehmen und die Zeilen mit `INSERT INTO ... SELECT * FROM ...` einzufügen.

Um eine bessere Kontrolle über den Einfügeprozess zu erhalten, kann es besser sein, große Tabellen in Teilstücken einzufügen:

```
INSERT INTO neue_tabelle SELECT * FROM alte_tabelle WHERE schluessel > etwas
                                          AND schluessel <= etwas_anderes;
```

Nachdem alle Daten eingefügt wurden, können Sie die Tabellen umbenennen.

Während der Umwandlung großer Tabellen sollten Sie den InnoDB-Puffer-Pool hoch setzen, um Festplatten-Ein- und -Ausgaben zu verringern, allerdings nicht höher als 80% des physikalischen Arbeitsspeichers. Sie sollten die InnoDB-Log-Dateien Groß machen und auch den Log-Puffer.

Stellen Sie sicher, dass Sie genug Tabellenplatz (Tablespace) haben! InnoDB-Tabellen benötigen viel mehr Platz als MyISAM-Tabellen. Wenn ein `ALTER TABLE` nicht mehr genug Platz hat, wird ein Rollback gestartet, das Stunden dauern kann, wenn es auf der Festplatte stattfindet. Bei Einfügeoperationen verwendet InnoDB den Einfügepuffer, um sekundäre Index-Datensätze mit Indexen in Stapeln zu vermischen. Das spart eine Menge an Festplatten-Ein- und -Ausgaben. Beim Rollback wird kein solcher Mechanismus benutzt, weshalb das Rollback bis zu 30 mal länger als das Einfügen dauern kann.

Falls Sie keine wertvollen Daten in Ihren InnoDB-Dateien haben, ist es im Fall eines 'festgefahrenen' Rollback besser, den Datenbank-Prozess zu killen und alle InnoDB-Daten- und Log-Dateien sowie alle InnoDB-Tabellen (`.frm`-Dateien) zu löschen und noch einmal anzufangen, statt darauf zu warten, dass Millionen von Festplatten-Ein- und -Ausgaben beendet werden.

8.5.4.2. Fremdschlüssel-(Foreign Key)-Beschränkungen

InnoDB-Version 3.23.44 hat Fremdschlüssel-(Foreign Key)-Beschränkungen. InnoDB ist der erste MySQL-Tabellentyp, der die Definition von Fremdschlüssel-Beschränkungen zuläßt, um die Integrität Ihrer Daten zu überwachen.

Die Syntax einer Fremdschlüsseldefinition in InnoDB:

```
FOREIGN KEY (index_spalten_name, ...) REFERENCES tabellen_name (index_spalten_name, ...)
```

Beispiel:

```
CREATE TABLE eltern(id INT NOT NULL, PRIMARY KEY (id)) TYPE=INNODB;
CREATE TABLE kind(id INT, eltern_id INT, INDEX par_ind (eltern_id),
           FOREIGN KEY (eltern_id) REFERENCES eltern(id)) TYPE=INNODB;
```

Beide Tabellen müssen vom Typ InnoDB sein und es muss einen Index geben, bei dem der Fremdschlüssel und der referenzierte Schlüssel als erste Spalten aufgeführt sind. Jegliches `ALTER TABLE` entfernt momentan alle Fremdschlüsselbeschränkungen, die für die Tabelle definiert wurden, aber nicht die Beschränkungen, die die Tabelle referenzieren. Korrespondierende Spalten im Fremdschlüssel und dem referenzierten Schlüssel müssen ähnliche interne Datentypen innerhalb InnoDB sein, so dass sie ohne Typumwandlung verglichen werden können. Die Längen von Zeichenkettentypen müssen nicht dieselben sein. Die Größe und Vorzeichen / kein Vorzeichen von Ganzzahltypen müssen dieselben sein.

Beim Prüfen von Fremdschlüsseln setzt InnoDB gemeinsame Sperren auf Zeilenebene auf kind- und eltern-Datensätze, die es betrachten muss. InnoDB prüft Fremdschlüssel-(Foreign Key)-Beschränkungen sofort: Die Prüfung wird nicht bis zu einem Transaktions-Commit verschoben.

InnoDB läßt zu, dass jegliche Tabelle gelöscht wird, selbst wenn das die Fremdschlüssel-(Foreign Key)-Beschränkungen durchbrechen würde, die die Tabelle referenzieren. Wenn Sie eine Tabelle löschen, werden die Beschränkungen, die in ihrem CREATE-Statement definiert wurden, ebenfalls gelöscht.

Wenn Sie eine gelöschte Tabelle neu erzeugen, muss sie eine Definition haben, die mit den Fremdschlüssel-(Foreign Key)-Beschränkungen konform ist, die sie referenzieren. Sie muss die richten Spaltennamen und -typen haben, und sie muss - wie oben angegeben - Indexe auf die referenzierten Schlüssel haben.

Sie können die Fremdschlüssel-(Foreign Key)-Beschränkungen für eine Tabelle wie folgt auflisten: `T` with

```
SHOW TABLE STATUS FROM ihr_datenbank_name LIKE 'T';
```

Die Fremdschlüssel-(Foreign Key)-Beschränkungen werden im Tabellen-Kommentar der Ausgabe aufgelistet.

InnoDB unterstützt noch kein `CASCADE ON DELETE` oder andere spezielle Optionen für diese Beschränkungen.

8.5.5. Hinzufügen und Entfernen von InnoDB-Daten- und -Log-Dateien

Sie können die Größe einer InnoDB-Daten-Datei nicht vergrößern. Um Ihrem Tabellenplatz (Tablespace) mehr hinzuzufügen, müssen Sie eine neue Daten-Datei hinzufügen. Um das zu tun, müssen Sie Ihre MySQL-Datenbank herunter fahren, die `my.cnf`-Datei editieren und eine neue Datei zu `innodb_data_file_path` hinzufügen. Dann starten Sie MySQL erneut.

Momentan können Sie keine Daten-Datei aus InnoDB entfernen. Um die Größe Ihrer Datenbank zu verringern, müssen Sie `mysqldump` benutzen, um alle Ihre Tabellen zu dumpen, eine neue Datenbank erzeugen und Ihre Tabellen in die neue Datenbank importieren.

Wenn Sie die Anzahl oder die Größe Ihrer InnoDB-Log-Dateien ändern wollen, müssen Sie MySQL herunter fahren und sicher stellen, dass er ohne Fehler herunter fuhr. Dann kopieren Sie die alten Log-Dateien an eine sichere Stelle, falls etwas beim Herunterfahren schiefging und Sie die Datenbank wiederherstellen müssen. Löschen Sie die alten Log-Dateien aus dem Log-Datei-Verzeichnis, editieren Sie `my.cnf` und starten Sie MySQL noch einmal. InnoDB meldet beim Starten, dass es neue Log-Dateien anlegt.

8.5.6. Datensicherung und Wiederherstellung einer InnoDB-Datenbank

Der Schlüssel zur sicheren Datenbankverwaltung sind regelmäßige Datensicherungen. Im eine 'binäre' Sicherung Ihrer Datenbank zu machen, tun Sie folgendes:

- Fahren Sie Ihre MySQL-Datenbank herunter und stellen Sie sicher, dass dabei keine Fehler auftraten.
- Kopieren Sie Ihre Daten-Dateien an eine sichere Stelle.
- Kopieren Sie alle InnoDB-Log-Dateien an eine sichere Stelle.
- Kopieren Sie Ihre `my.cnf` Konfigurationsdatei(en) an eine sichere Stelle.
- Kopieren Sie alle `.frm`-Dateien für Ihre InnoDB-Tabellen an eine sichere Stelle.

Momentan gibt es kein Online- oder inkrementelles Datensicherungsprogramm für InnoDB, obwohl diese auf der TODO-Liste sind.

Zusätzlich zu den beschriebenen Binär-Datensicherungen sollten Sie ausserdem regelmäßig Dumps Ihrer Tabellen mit `mysqldump` machen. Der Grund ist, dass eine Binärdatei beschädigt sein kann, ohne dass Sie das bemerken. Gedumpte Tabellen werden in Textdateien gespeichert, die Menschen-lesbar und viel einfacher als binäre Datenbankdateien sind. Aus gedumpten Dateien läßt sich Tabellenbeschädigung leichter erkennen und da ihr Format einfacher ist, ist das Risiko ernsthafter Datenbeschädigung in ihnen geringer.

Es ist eine gute Idee, Dumps zur gleichen Zeit zu machen wie die binäre Datensicherung Ihrer Datenbank. Sie müssen alle Clients aus Ihrer Datenbank ausschließen, um konsistente Schnappschüsse aller Ihrer Tabellen im Dump zu bekommen. Danach können Sie die binäre Datensicherung machen, so dass Sie einen konsistenten Schnappschuss Ihrer Datenbank in zwei Formaten haben.

Um in der Lage zu sein, Ihre InnoDB-Datenbank aus den beschriebenen binären Datensicherungen wiederherzustellen, müssen Sie Ihre MySQL-Datenbank mit allgemeinem Loggen und angeschalteter Log-Archivierung von MySQL laufen lassen. Mit allgemeinem Loggen ist hier der Log-Mechanismus des MySQL-Servers gemeint, der unabhängig von den InnoDB-Logs ist.

Zum Wiederherstellen nach einem Absturz des MySQL-Serverprozesses ist es lediglich nötig, diesen erneut zu starten. InnoDB prüft automatisch die Log-Dateien und führt ein Roll-Forward der Datenbank bis zum aktuellen Stand durch. InnoDB macht ein automatisches Rollback nicht abgeschlossener (committed) Transaktionen, die zur Zeit des Absturzes anhängig waren. Während der Wiederherstellung gibt InnoDB etwa folgendes aus:

```
~/mysqlm/sql > mysqld
InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 13674004
InnoDB: Doing recovery: scanned up to log sequence number 0 13739520
InnoDB: Doing recovery: scanned up to log sequence number 0 13805056
InnoDB: Doing recovery: scanned up to log sequence number 0 13870592
InnoDB: Doing recovery: scanned up to log sequence number 0 13936128
...
InnoDB: Doing recovery: scanned up to log sequence number 0 20555264
InnoDB: Doing recovery: scanned up to log sequence number 0 20620800
InnoDB: Doing recovery: scanned up to log sequence number 0 20664692
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 16745
InnoDB: Rolling back of trx no 16745 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections
```

Wenn Ihre Datenbank beschädigt wird oder Ihre Festplatte Fehler hat, müssen Sie eine Wiederherstellung aus einer Datensicherung durchführen. Im Falle der Beschädigung sollten Sie zunächst eine Datensicherung finden, die nicht beschädigt ist. Machen Sie aus der Datensicherung eine Wiederherstellung aus den allgemeinen Log-Dateien von MySQL unter Beachtung der Anleitungen im MySQL-Handbuch.

8.5.6.1. Checkpoints

InnoDB hat einen Checkpoint-Mechanismus implementiert, der sich Fuzzy Checkpoint nennt. InnoDB schreibt veränderten Datenbankseiten aus dem Puffer-Pool in kleinen Stapeln (Batch) auf Platte (flush), daher besteht keine Notwendigkeit, den Puffer-Pool in einem einzelnen Stapel zurückzuschreiben, was in der Praxis dazu führen würde, dass SQL-Statements von Benutzern für eine Weile angehalten würden.

Bei der Reparatur nach Abstürzen sucht InnoDB nach einem Checkpoint-Label in den Log-Dateien. Es weiß, dass alle Änderungen an der Datenbank vor dem Label bereits im Platten-Image der Datenbank enthalten sind. InnoDB scannt anschließend die Log-Dateien ab dem Checkpoint vorwärts und wendet die geloggten Änderungen auf die Datenbank an.

InnoDB schreibt in zirkulärer Art in die Log-Dateien. Alle abgeschickten (committed) Änderungen, die dazu führen, dass sich die Datenbankseiten im Puffer-Pool vom Image auf der Platte unterscheiden, müssen in den Log-Dateien verfügbar sein, für den Fall, dass InnoDB eine Wiederherstellung durchführen muss. Das heißt, wenn InnoDB anfängt, eine Log-Datei auf zirkuläre Weise wieder zu benutzen, muss es sicherstellen, dass die Datenbankseiten-Images auf der Festplatte bereits die Änderungen enthalten, die in der Log-Datei mitgeschrieben sind, die InnoDB benutzen wird. Mit anderen Worten muss InnoDB einen Checkpoint machen, was oft das Zurückschreiben auf Platte (flush) geänderter Datenbankseiten beinhaltet.

Das erklärt, warum es Festplatten-Ein- und -Ausgaben sparen kann, wenn man die Log-Dateien sehr Groß macht. Es kann sinnvoll sein, die Gesamtgröße der Log-Dateien so Groß wie den Puffer-Pool oder sogar noch größer zu machen. Der Nachteil großer Log-Dateien ist, dass eine Reparatur nach Absturz länger dauern kann, weil mehr Log-Einträge auf die Datenbank angewendet werden müssen.

8.5.7. Eine InnoDB-Datenbank auf eine andere Maschine verschieben

InnoDB-Daten- und Log-Dateien sind auf allen Plattformen binärkompatibel, wenn das Fließkommazahlenformat auf den Maschinen dasselbe ist. Sie können eine InnoDB-Datenbank einfach verschieben, indem Sie alle relevanten Dateien kopieren, die im vorherigen Abschnitt über Datensicherung erwähnt wurden. Wenn sich das Fließkommaformat auf den Maschinen unterscheidet, sie aber keine `FLOAT`- oder `DOUBLE`-Datentypen in Ihren Tabellen benutzt haben, ist die Prozedur dieselbe: Kopieren Sie einfach die relevanten Dateien. Wenn die Formate unterschiedlich sind und Ihre Tabellen Fließkomma-Daten enthalten, müssen Sie `mysqldump` und `mysqlimport` benutzen, um diese Tabellen zu verschieben.

Ein Tipp zur Performance: Schalten Sie Auto-Commit aus, wenn Sie Daten in Ihre Datenbank importieren (unter der Annahme, dass Ihr Tabellenplatz (Tablespace) genug Platz für das große Rollback-Segment enthält, den die große Import-Transaktion erzeugen wird). Machen Sie das Commit erst nach dem Import einer ganzen Tabelle oder eines Segments einer Tabelle.

8.5.8. InnoDB-Transaktionsmodell

Im InnoDB-Transaktionsmodell war das Ziel, die besten Eigenschaften einer multiversionsfähigen Datenbank mit dem traditionellen Zwei-Phasen-Sperren zu verbinden. InnoDB führt Sperren auf Zeilenebene durch und läßt Anfragen vorgabemäßig als nicht sperrende konsistente Leseoperationen laufen, im Stil von Oracle. Das Tabellensperren ist in InnoDB so platzsparend gespeichert, dass keine Sperr-Eskalation benötigt wird: Typischerweise dürfen mehrere Benutzer jede Zeile in der Datenbank oder eine beliebige Teilmenge der Zeilen sperren, ohne dass InnoDB keinen Speicher mehr hat.

Bei InnoDB findet jede Benutzeraktivität innerhalb von Transaktionen statt. Wenn der Auto-Commit-Modus in MySQL benutzt wird, stellt jedes SQL-Statement eine einzelne Transaktion dar. Wenn der Auto-Commit-Modus ausgeschaltet wird, kann man sich vorstellen, dass ein Benutzer stets eine Transaktion offen hat. Wenn er das SQL-`COMMIT`- oder `ROLLBACK`-Statement absetzt, beendet das die aktuelle Transaktion und eine neue beginnt. Beide Statements heben alle InnoDB-Sperren auf, die während der aktuellen Transaktion gesetzt wurden. Ein `COMMIT` bedeutet, dass die in der aktuellen Transaktion gemachten Änderungen permanent und sichtbar für andere Benutzer gemacht werden. Auf der anderen Seite bricht ein `ROLLBACK` alle Änderungen ab, die in der aktuellen Transaktion gemacht wurden.

8.5.8.1. Konsistentes Lesen

Konsistentes Lesen bedeutet, dass InnoDB seine Multiversionsfähigkeiten nutzt, um einer Anfrage einen Schnappschuss der Datenbank zu einem bestimmten Zeitpunkt zu zeigen. Die Anfrage sieht genau die Änderungen, die von Transaktionen durchgeführt wurden, die bis zu diesem Zeitpunkt abgeschlossen wurden (committed), und keine Änderungen, die später gemacht wurden oder die noch nicht abgeschlossen sind. Die Ausnahme von der Regel ist, dass die Anfrage die Änderungen sieht, die durch die Transaktion selbst durchgeführt wurde, die die Anfrage absetzt.

Wenn eine Transaktion ihr erstes Konsistentes Lesen durchführt, weist InnoDB den Schnappschuss oder Zeitpunkt zu, den jedes Konsistente Lesen in derselben Transaktion benutzen wird. Im Schnappschuss sind alle Transaktionen enthalten, die vor der Zuweisung zum Schnappschuss abgeschlossen (committed) wurden. Daher ist Konsistentes Lesens innerhalb derselben Transaktion

auch untereinander konsistent. Sie können einen frischeren Schnappschuss für Ihre Anfragen erhalten, indem Sie die aktuelle Transaktion beenden (commit) und danach neue Anfragen absetzen.

Konsistentes Lesen ist der vorgabemäßige Modus, in dem InnoDB `SELECT`-Statements abarbeitet. Konsistentes Lesen setzt keinerlei Sperren auf die Tabellen, auf die es zugreift. Daher können andere Benutzer zur selben Zeit, wie Konsistentes Lesen auf die Tabelle durchgeführt wird, diese verändern.

8.5.8.2. Lesevorgänge sperren

Unter manchen Umständen ist Konsistentes Lesen nicht wünschenswert. Angenommen, Sie wollen eine neue Zeile in die Tabelle `kind` einfügen und dabei sicherstellen, dass das Kind bereits Eltern in der Tabelle `eltern` hat.

Wenn Sie Konsistentes Lesen benutzen, um die Tabelle `eltern` zu lesen und in der Tat die Eltern des Kindes in der Tabelle sehen, können Sie dann sicher die Kind-Zeile zur Tabelle `kind` hinzufügen? Nein, denn es kann sein, dass zwischenzeitlich jemand anderes die Eltern-Zeile aus der Tabelle `eltern` gelöscht hat und Sie das nicht sehen.

Die Lösung besteht darin, das `SELECT` im Sperrmodus durchzuführen. `LOCK IN SHARE MODE`.

```
SELECT * FROM eltern WHERE NAME = 'Hinz' LOCK IN SHARE MODE;
```

Wenn Sie ein Lesen im Share-Modus durchführen, heißt das, dass die letzten verfügbaren Daten gelesen werden und eine Shared-Modus-Sperre auf die Zeile gesetzt wird, die gelesen wird. Wenn die letzten Daten zu einer noch nicht abgeschlossenen Transaktion eines anderen Benutzers gehören, wird gewartet, bis die Transaktion abgeschlossen (committed) ist. Eine Shared-Modus-Sperre verhindert, dass andere die Zeile aktualisieren oder löschen, die gerade gelesen wurde. Nachdem festgestellt wurde, dass die obige Anfrage die Eltern `'Hinr'` zurückgibt, kann das Kind sicher zur Tabelle `kind` hinzugefügt und die Transaktion abgeschlossen werden. Dieses Beispiel zeigt, wie Sie in Ihren Applikations-Code referentielle Integrität integrieren können.

Sehen wir uns ein weiteres Beispiel an. Wir haben ein ganzzahliges Zählerfeld in einer Tabelle `kind_codes`, was benutzt wird, um jedem Kinde, das wir der Tabelle `kind` hinzufügen, eine eindeutige Kennung zuzuweisen. Es ist offensichtlich, dass Konsistentes Lesen oder Shared-Modus-Lesen kein geeignetes Mittel ist, um den aktuellen Wert des Zählers zu ermitteln, weil nämlich zwei Benutzer der Datenbank denselben Wert des Zählers sehen können und wir daher einen Fehler wegen doppelter Schlüsseleinträge erhalten, wenn wir zwei Kinder mit derselben Kennung in die Tabelle einfügen.

In diesem Fall gibt es zwei geeignete Möglichkeiten, das Lesen und Heraufzählen des Zählers zu implementieren: (1) Zuerst den Zähler um eins erhöhen und erst danach lesen. (2) Zuerst den Zähler im Sperr-Modus `FOR UPDATE` lesen und danach heraufzählen:

```
SELECT COUNTER_FIELD FROM kind_codes FOR UPDATE;
UPDATE kind_codes SET COUNTER_FIELD = COUNTER_FIELD + 1;
```

`SELECT ... FOR UPDATE` liest die letzten verfügbaren Daten und setzt exklusive Sperren auf jede Zeile, die es liest. Daher setzt es dieselben Sperren, die ein gesuchtes SQL-`UPDATE` auf die Zeilen setzen würde.

8.5.8.3. Nächsten Schlüssel sperren: Wie das Phantom-Problem vermieden wird

Beim Sperren auf Zeilenebene benutzt InnoDB einen Algorithmus, der Nächsten-Schlüssel-Sperren genannt wird. InnoDB führt das Sperren auf Zeilenebene so durch, dass es beim Suchen oder Scannen eines Indexes auf eine Tabelle gemeinsam genutzte (shared) oder exklusive Sperren auf die Index-Datensätze setzt, die es findet. Daher werden die Sperren auf Zeilenebene genauer Index-Datensatz-Sperren genannt.

Die Sperren, die InnoDB auf Index-Datensätze setzt, betreffen auch die 'Lücke' vor diesem Index-Datensatz. Wenn ein Benutzer eine gemeinsam benutzte (shared) oder exklusive Sperre auf den Datensatz R in einem Index hat, kann ein anderen Benutzer keinen Datensatz direkt vor R (in der Index-Reihenfolge) einfügen. Dieses Sperren von Lücken wird durchgeführt, um das so genannte Phantom-Problem zu vermeiden. Angenommen, man will alle Kinder aus der Tabelle `kind` lesen und sperren, die eine Kennung größer 100 haben, und irgend ein Feld in der ausgewählten Zeile aktualisieren:

```
SELECT * FROM kind WHERE ID > 100 FOR UPDATE;
```

Angenommen, es gibt einen Index auf der Tabelle `kind` auf der Spalte `ID`. Unsere Anfrage scannt diesen Index ab dem ersten Datensatz, bei dem `ID` größer als 100 ist. Wenn jetzt die auf den Index-Datensatz gesetzten Sperren nicht Einfügeoperationen sperren würden, die in die Lücken ausgeführt würden, könnte zwischenzeitlich ein neues Kind in die Tabelle eingefügt werden. Wenn jetzt unsere Transaktion noch einmal folgendes ausführen würde:

```
SELECT * FROM kind WHERE ID > 100 FOR UPDATE;
```

Sehen wir ein neues Kind in der Ergebnismenge, die die Anfrage zurückgibt. Das verstößt gegen das Isolationsprinzip von Transaktionen: Eine Transaktion sollte in der Lage sein, so abzulaufen, dass die Daten, die sie gelesen hat, sich nicht während der Transaktion ändern. Wenn wir einen Satz von Zeilen als Daten-Posten betrachten, würde das neue 'Phantom'-Kind dieses

Isolationsprinzip durchbrechen.

Wenn InnoDB einen Index scannt, kann es auch die Lücke nach dem letzten Datensatz im Index sperren. Genau das passiert im vorherigen Beispiel: Die Sperren, die von InnoDB gesetzt werden, verhindert jedes Einfügen in die Tabelle an Stellen, wo `ID` größer als 100 ist.

Sie können Nächsten-Schlüssel-Sperren dazu benutzen, eine Eindeutigkeitsprüfung in Ihre Applikation zu implementieren: Wenn Sie Ihre Daten im Share-Modus lesen und kein Duplikat für eine Zeile sehen, die Sie einfügen werden, können Sie Ihre Zeile sicher einfügen und wissen, dass das Nächsten-Schlüssel-Sperren verhindern wird, dass zwischenzeitlich jemand eine Duplikatzeile Ihrer Zeile einfügt. Daher gestattet Ihnen das Nächsten-Schlüssel-Sperren, die Nicht-Existenz von irgend etwas in Ihrer Tabelle zu 'sperren'.

8.5.8.4. Sperren, die in InnoDB durch unterschiedliche SQL-Statements gesetzt werden

- `SELECT ... FROM ...` : Das ist Konsistentes Lesen, es wird ein Schnappschuss einer Datenbank gelesen und es werden keine Sperren gesetzt.
- `SELECT ... FROM ... LOCK IN SHARE MODE` : setzt gemeinsam genutztes (shared) Nächsten-Schlüssel-Sperren auf alle Index-Datensätze, die beim Lesen gefunden werden.
- `SELECT ... FROM ... FOR UPDATE` : setzt exklusives Nächsten-Schlüssel-Sperren auf alle Index-Datensätze, die beim Lesen gefunden werden.
- `INSERT INTO ... VALUES (...)` : setzt eine exklusive Sperre auf die eingefügte Zeile. Beachten Sie, dass diese Sperre kein Nächsten-Schlüssel-Sperren ist und andere Benutzer nicht davon abhält, etwas in die Lücke vor der eingefügten Zeile einzufügen. Wenn ein Fehler wegen doppelter Schlüsseleinträge auftritt, setzt dieser Befehl eine gemeinsam genutzte (shared) Sperre auf den doppelten (Duplikat) Index-Datensatz.
- `INSERT INTO T SELECT ... FROM S WHERE ...` setzt eine exklusive Sperre (kein Nächsten-Schlüssel-Sperren) auf jede Zeile, die in `T` eingefügt wurde. Sucht nach `S` in Form von Konsistentem Lesen, aber setzt Nächsten-Schlüssel-Sperren auf `S`, wenn bei MySQL das Loggen angeschaltet ist. InnoDB muss in letzterem Fall Sperren setzen, weil bei einer Roll-Forward-Wiederherstellung aus einer Datensicherung jedes SQL-Statement auf genau dieselbe Weise ausgeführt werden muss, wie es ursprünglich ausgeführt wurde.
- `CREATE TABLE ... SELECT ...` führt `SELECT` als Konsistentes Lesen oder mit gemeinsam genutzten (shared) Sperren aus, wie im vorherigen Punkt.
- `REPLACE` wird wie Einfügen ausgeführt, wenn es keine Kollision auf einem eindeutigen Schlüssel gibt. Ansonsten wird ein exklusives Nächsten-Schlüssel-Sperren auf die Reihe gesetzt, die aktualisiert werden muss.
- `UPDATE ... SET ... WHERE ...` setzt ein exklusives Nächsten-Schlüssel-Sperren auf jeden Datensatz, der beim Suchen gefunden wird.
- `DELETE FROM ... WHERE ...` setzt ein exklusives Nächsten-Schlüssel-Sperren auf jeden Datensatz, der beim Suchen gefunden wird.
- Wenn auf der Tabelle eine `FOREIGN KEY`-Beschränkung definiert ist, setzt jedes Einfügen, Aktualisieren oder Löschen, was die Überprüfung der Beschränkungsbedingung erfordert, gemeinsam genutzte (shared) Sperren auf Datensatzebene auf die Datensätze, die bei der Überprüfung der Beschränkung betrachtet werden. Auch im Falle, dass die Beschränkung fehlschlägt, setzt InnoDB diese Sperren.
- `LOCK TABLES ...` : setzt Tabellensperren. In der Implementation setzt die MySQL-Ebene des Codes diese Sperren. Die automatische Blockierungserkennung von InnoDB kann keine Blockierungen bemerken, bei denen solche Tabellensperren involviert sind, siehe nächster Abschnitt weiter unten. Sehen Sie auch im Abschnitt 13 ('InnoDB-Einschränkungen') wegen folgendem nach: Weil MySQL keine Sperren auf Zeilenebene erkennt, ist es möglich, dass Sie eine Sperre auf eine Tabelle erhalten, auf der ein anderer Benutzer momentan Sperren auf Zeilenebene hat. Das gefährdet allerdings nicht die Transaktionsintegrität.

8.5.8.5. Blockierungserkennung und Rollback

InnoDB erkennt automatisch eine Blockierung von Transaktionen und rollt die Transaktion zurück, deren Sperranforderung diejenige war, die die Blockierung aufbaute, also einen Kreis im Warte-Diagramm von Transaktionen. InnoDB kann keine Blockierungen erkennen, bei denen eine Sperre im Spiel ist, die durch ein MySQL-`LOCK TABLES`-Statement verursacht wurde, oder wenn eine Sperre durch einen anderen Tabellen-Handler als InnoDB gesetzt wurde. Solche Situationen müssen Sie mit `innodb_lock_wait_timeout`, das in `my.cnf` gesetzt wird.

Wenn InnoDB ein komplettes Rollback einer Transaktion durchführt, werden alle Sperren der Transaktion aufgehoben. Wenn jedoch nur ein einzelnes SQL-Statement als Ergebnis eines Fehlers zurückgerollt wird, können einige der Sperren, die durch das

SQL-Statement gesetzt wurde, verbleiben. Das liegt daran, dass InnoDB Zeilensperren in einem Format speichert, die ihm unmöglich machen, im Nachhinein zu erkennen, welche Sperre durch welches SQL-Statement gesetzt wurde.

8.5.8.6. Ein Beispiel, wie konsistentes Lesen bei InnoDB funktioniert

Wenn Sie ein Konsistentes Lesen ausführen, also ein gewöhnliches `SELECT`-Statement, gibt InnoDB Ihrer Transaktion einen Zeitpunkt (Timepoint), gemäß dem Ihre Anfrage die Datenbank sieht. Wenn daher Transaktion B eine Zeile löscht und das wirksam wird (commit), nachdem Ihr Zeitpunkt zugewiesen wurde, werden Sie die Zeile nicht als gelöscht sehen. Gleiches gilt für Einfüge- und Aktualisierungsoperationen.

Sie können Ihren Zeitpunkt 'vorstellen', indem Sie Ihre Transaktion abschicken (commit) und dann ein weiteres `SELECT` ausführen.

Das nennt sich Multiversioned Concurrency Control (multiversionierte Gleichzeitigkeitskontrolle):

```
                    Benutzer A              Benutzer B

                set autocommit=0;       set autocommit=0;
zeit
|               SELECT * FROM t;
|               empty set
|                                         INSERT INTO t VALUES (1, 2);
|
v               SELECT * FROM t;
                empty set
                                         COMMIT;

                SELECT * FROM t;
                empty set;

                COMMIT;

                SELECT * FROM t;
                ---------------------
                |    1    |    2    |
                ---------------------
```

Daher sieht Benutzer A die durch B eingefügte Zeile erst, wenn B das Einfügen und A seine eigene Transaktion abgeschickt hat (commit), so dass der Zeitpunkt hinter das Commit von B 'vorgestellt' ist.

Wenn Sie den 'frischsten' Zustand der Datenbank sehen wollen, sollten Sie ein sperrendes Lesen (Locking Read) benutzen:

```
SELECT * FROM t LOCK IN SHARE MODE;
```

8.5.9. Tipps zur Performance-Steigerung

1. Wenn das Unix-`top` oder der Windows-`Task-Manager` zeigen, dass die CPU-Auslastung weniger als 70% beträgt, ist Ihre Auslastung wahrscheinlich Platten-gebunden. Das kann daran liegen, dass Sie zu viele Transaktionen abschicken (commit) oder dass der Puffer-Pool zu klein ist. Dann kann es helfen, den Puffer-Pool zu vergrößern. Setzen Sie ihn aber nicht höher als 80% des physikalischen Arbeitsspeichers.

2. Packen Sie mehrere Änderungen in eine Transaktion. InnoDB muss das Log jedes Mal auf Platte zurückschreiben (flush), wenn eine Transaktion abgeschickt wird (commit), wenn diese Transaktion irgend welche Änderungen an der Datenbank vorgenommen hat. Weil die Rotationsgeschwindigkeit einer Platte typischerweise höchsten 167 Umdrehungen pro Sekunde beträgt, beschränkt das die Anzahl von Commits auf eben diese Zahl pro Sekunde, wenn die Festplatte nicht das Betriebssystem täuscht.

3. Wenn Sie es sich leisten können, einige der zuletzt abgeschickten (committed) Transaktionen zu verlieren, können Sie den `my.cnf`-Parameter `innodb_flush_log_at_trx_commit` auf 0 setzen. InnoDB versucht dann trotzdem, das Log einmal pro Sekunde auf Platte zurückzuschreiben (flush), doch dieses Zurückschreiben ist nicht garantiert.

4. Machen Sie Ihre Log-Dateien Groß, selbst so Groß wie den Puffer-Pool. Wenn InnoDB seine Log-Dateien vollgeschrieben hat, muss es die veränderten Inhalte des Puffer-Pools in einem Checkpoint auf Platte schreiben. Kleine Log-Dateien verursachen daher unnötige Festplatten-Schreibzugriffe. Der Nachteil großer Log-Dateien liegt darin, dass die Wiederherstellungszeit länger wird.

5. Ausserdem sollte der Log-Puffer recht Groß sein, sagen wir 8 MB.

6. (Relevant from 3.23.39 up.) In einigen Versionen von Linux und Unix ist das Zurückschreiben von Dateien auf Platte (flush) mit dem Unix-`fdatasync` und anderen ähnlichen Methoden überraschend langsam. InnoDB benutzt vorgabemäßig die `fdatasync`-Funktion. Wenn Sie mit der Datenbank-Schreib-Performance nicht zufrieden sind, können Sie versuchen, die `innodb_flush_method` in `my.cnf` auf `O_DSYNC` zu setzen, obwohl O_DSYNC auf den meisten Systemen langsamer zu sein scheint.

7. Wenn Sie Daten in InnoDB importieren, stellen Sie sicher, dass MySQL `autocommit=1` nicht angeschaltet hat, denn dann benötigt jedes Einfügen ein Zurückschreiben des Logs auf Platte (flush). Setzen Sie vor Ihre SQL-Importdatei die Zeile

```
set autocommit=0;
```

und danach

```
commit;
```

Wenn Sie die `mysqldump`-Option `--opt` benutzen, erhalten Sie Dump-Dateien, die sich sehr schnell auch in eine InnoDB-Tabelle importieren lassen, selbst ohne sie in die oben erwähnten `set autocommit=0; ... commit;`-Wrapper zu verpacken.

8. Hüten Sie sich vor großen Rollbacks beim Einfügen von Massendaten: InnoDB benutzt den Einfüge-Puffer, um beim Einfügen Festplatten-Ein- und -Ausgaben zu sparen, doch beim entsprechenden Rollback wird kein solcher Mechanismus benutzt. Ein Festplatten-gebundenes Rollback kann die 30-fache Zeit des entsprechenden Einfügevorgangs in Anspruch nehmen. Es hilft nicht, den Datenbankprozess zu killen, weil der Rollback erneut starten wird, wenn die Datenbank hochfährt. Die einzige Möglichkeit, ein aus dem Ruder gelaufenes Rollback loszuwerden, besteht darin, den Puffer-Pool zu erhöhen, so dass das Rollback CPU-gebunden wird und damit schnell läuft, oder indem die gesamte InnoDB-Datenbank gelöscht wird.

9. Seien Sie auch vor anderen großen Festplatten-gebundenen Operationen auf der Hut. Benutzen Sie `DROP TABLE` oder `TRUNCATE` (ab MySQL-4.0), um eine Tabelle zu löschen, nicht `DELETE FROM tabelle`.

10. Benutzen Sie das mehrzeilige `INSERT`, um den Kommunikations-Overhead zwischen Client und Server zu verringern, wenn Sie viele Zeilen einfügen müssen:

```
INSERT INTO tabelle VALUES (1, 2), (5, 5);
```

Dieser Tipp gilt natürlich für jeden Tabellentyp, nicht nur für InnoDB.

8.5.9.1. Der InnoDB-Monitor

Ab Version 3.23.41 beinhaltet InnoDB den InnoDB-Monitor, der Informationen über den internen Zustand von InnoDB ausgibt. Wenn er angeschaltet ist, veranlasst der InnoDB-Monitor den MySQL-Server `mysqld`, etwa alle 15 Sekunden Daten an die Standardausgabe auszugeben (Hinweis: der MySQL-Client gibt nichts aus). Diese Daten sind nützlich, um die Performance zu tunen. Unter Windows müssen Sie `mysqld-max` von einer DOS-Kommandozeile aus mit `--standalone --console` starten, um die Ausgabe auf das DOS-Fenster umzuleiten.

Es gibt einen separaten `innodb_lock_monitor`, der dieselben Informationen ausgibt wie `innodb_monitor`, aber zusätzlich Informationen über Sperren, die durch jede Transaktion gesetzt werden.

Die ausgegebene Information enthält Daten über:

- Sperren, die auf eine Transaktion warten,
- Semaphore, die auf Threads warten,
- anhängige Datei-Ein- und -Ausgabeanforderungen,
- Puffer-Pool-Statistiken und
- Bereinigungs- (purge) und Einfüge-Puffer-Vermengungs- (merge) Aktivität des Haupt-Threads von InnoDB.

Sie können den InnoDB-Monitor mit folgendem SQL-Befehl starten:

```
CREATE TABLE innodb_monitor(a int) type = innodb;
```

Und ihn mit folgendem Befehl anhalten:

```
DROP TABLE innodb_monitor;
```

Die `CREATE TABLE`-Syntax ist nur eine Möglichkeit, einen Befehl durch den MySQL-SQL-Parser an die InnoDB-Engine durchzureichen. Wenn Sie die Datenbank herunter fahren, während der Monitor läuft, und Sie den Monitor erneut starten wollen, müssen Sie die Tabelle löschen, bevor Sie ein erneutes `CREATE TABLE` absetzen können, um den Monitor zu starten. Diese Syntax wird sich in zukünftigen Releases möglicherweise ändern.

Beispiel für die Ausgabe des InnoDB-Monitors:

```
================================
010809 18:45:06 INNODB MONITOR OUTPUT
================================
```

```
------------------------
LOCKS HELD BY transactions
------------------------
LOCK INFO:
Number of locks in the record hash table 1294
LOCKS FOR TRANSACTION ID 0 579342744
TABLE LOCK table test/tabelle trx id 0 582333343 lock_mode IX

RECORD LOCKS space id 0 page no 12758 n bits 104 table test/tabelle index
PRIMARY trx id 0 582333343 lock_mode X
Record lock, heap no 2 PHYSICAL RECORD: n_fields 74; 1-byte offs FALSE;
info bits 0
 0: len 4; hex 0001a801; asc ;; 1: len 6; hex 000022b5b39f; asc ";; 2: len 7;
hex 000002001e03ec; asc ;; 3: len 4; hex 00000001;
...
--------------------------------------------
CURRENT SEMAPHORES RESERVED AND SEMAPHORE WAITS
--------------------------------------------
SYNC INFO:
Sorry, cannot give mutex list info in non-debug version!
Sorry, cannot give rw-lock list info in non-debug version!
----------------------------------------------------
SYNC ARRAY INFO: reservation count 6041054, signal count 2913432
4a239430 waited for by thread 49627477 op. S-LOCK file NOT KNOWN line 0
Mut ex 0 sp 5530989 r 62038708 sys 2155035; rws 0 8257574 8025336; rwx 0 1121090 1848344
----------------------------------------------------
CURRENT PENDING FILE I/O'S
--------------------------
Pending normal aio reads:
Reserved slot, messages 40157658 4a4a40b8
Reserved slot, messages 40157658 4a477e28
...
Reserved slot, messages 40157658 4a4424a8
Reserved slot, messages 40157658 4a39ea38
Total of 36 reserved aio slots
Pending aio writes:
Total of 0 reserved aio slots
Pending insert buffer aio reads:
Total of 0 reserved aio slots
Pending log writes or reads:
Reserved slot, messages 40158c98 40157f98
Total of 1 reserved aio slots
Pending synchronous reads or writes:
Total of 0 reserved aio slots
-----------
BUFFER POOL
-----------
LRU list length 8034
Free list length 0
Flush list length 999
Buffer pool size in pages 8192
Pending reads 39
Pending writes: LRU 0, flush list 0, single page 0
Pages read 31383918, created 51310, written 2985115
----------------------------
END OF INNODB MONITOR OUTPUT
============================
010809 18:45:22 InnoDB starts purge
010809 18:45:22 InnoDB purged 0 pages
```

Einige Anmerkungen zur Ausgabe:

- Wenn der Abschnitt `LOCKS HELD BY transactions` warten auf Sperren berichtet, kann es sein, dass Ihre Applikation Sperr-Konflikte hat. Die Ausgabe kann auch helfen, Gründe für Transaktions-Blockierungen aufzuspüren.
- Der Abschnitt `SYNC INFO` berichtet reservierte Semaphore, wenn Sie InnoDB mit `UNIV_SYNC_DEBUG` kompilieren, definiert in `univ.i`.
- Der Abschnitt `SYNC ARRAY INFO` berichtet Threads, die auf ein Semaphor warten, und Statistiken, wie viele Male Threads ein Spin oder ein Warten auf einem Mutex oder einem Lese-/Schreibe-Sperr-Semaphor benötigten. Eine große Anzahl auf Semaphore wartender Threads kann ein Ergebnis von Festplatten-Ein- und -Ausgaben oder Konfliktproblemen innerhalb von InnoDB sein. Konflikte können durch starke Parallelen von Anfragen oder durch Probleme des Betriebssystems beim Thread Scheduling hervorgerufen werden.
- Der Abschnitt `CURRENT PENDING FILE I/O'S` listet anhängige Datei-Ein- und -Ausgabeanforderungen auf. Eine große Anzahl davon zeigt an, dass die Auslastung Festplatten-Ein- und -Ausgabe-gebunden ist.
- Der Abschnitt `BUFFER POOL` gibt statistische Informationen über gelesene und geschriebene Seiten. Aus diesen Zahlen können Sie errechnen, wie viele Daten-Datei-Ein- und Ausgaben Ihre Anfragen aktuell durchführen.

8.5.10. Implementation des Multiversionings

Weil InnoDB eine multiversionierte Datenbank ist, muss es Informationen über alte Versionen von Zeilen im Tabellenplatz

(Tablespace) aufbewahren. Diese Informationen werden in einer Datenstruktur gespeichert, die wir in Anlehnung an eine analoge Struktur in Oracle Rollback-Segment nennen.

InnoDB fügt jeder Zeile, die in der Datenbank gespeichert wird, intern zwei Felder hinzu. Ein 6 Byte großes Feld enthält den Transaktions-Identifikator der letzten Transaktion, die die Zeile eingefügt oder aktualisiert hat. Ein Löschen wir intern als eine Aktualisierung behandelt, wobei ein spezielles Bit in die Zeile eingefügt wird, um sie als gelöscht zu markieren. Jede Zeile enthält ausserdem ein 7 Byte großes Feld, das Roll-Zeiger genannt wird. Der Roll-Zeiger zeigt auf einen Rückgängig-Log-Datensatz, der in das Rollback-Segment geschrieben wird. Wenn die Zeile aktualisiert wurde, enthält der Rückgängig-Log-Datensatz die Informationen, die notwendig sind, um den Inhalt der Zeile wieder herzustellen, bevor sie aktualisiert wurde.

InnoDB benutzt die Informationen im Rollback-Segment, um die Rückgängig-Operationen durchzuführen, die bei einem Transaktions-Rollback notwendig sind. Diese Informationen benutzt es auch dafür, um frühere Informationen einer Zeile beim Konsistenten Lesen aufzubauen.

Rückgängig-Logs im Rollback-Segment lassen sich in Logs für Einfügen und für Aktualisieren unterteilen. Einfüge-Rückgängig-Logs werden nur für Transaktions-Rollbacks benötigt und können verworfen werden, sobald die Transaktion abgeschickt ist (commit). Aktualisierungs-Rückgängig-Logs werden auch für Konsistentes Lesens benutzt und können daher erst verworfen werden, wenn keine Transaktion mehr vorhanden ist, für die InnoDB einen Schnappschuss zugewiesen hat, dessen Informationen beim Konsistenten Lesen benötigt werden könnten, um daraus eine frühere Version der Datenbank-Zeile aufzubauen.

Sie müssen daran denken, Ihre Transaktionen regelmäßig abzuschicken (commit), auch die Transaktionen, die nur Konsistentes Lesens ausführen. Ansonsten kann InnoDB Daten aus dem Aktualisierungs-Rückgängig-Log nicht verwerfen und das Rollback-Segment könnte zu Groß werden und Ihren Tabellenplatz (Tablespace) komplett füllen.

Die physikalische Größe eines Rückgängig-Log-Datensatzes im Rollback-Segment ist typischerweise kleiner als die entsprechende eingefügte oder aktualisierte Zeile. Sie können diese Informationen benutzen, um den Platzbedarf für Ihr Rollback-Segment zu berechnen.

In diesem multiversionierten Schema wird eine Zeile nicht unmittelbar physikalisch aus der Datenbank entfernt, wenn Sie sie mit einem SQL-Statement löschen. Erst wenn InnoDB den Datensatz des Aktualisierungs-Rückgängig-Logs löschen kann, der für das Löschen geschrieben wurde, kann es die entsprechende Zeile und ihre Index-Datensätze auch physikalisch aus der Datenbank entfernen. Diese Entfernungsoperation wird Purge genannt und ist recht schnell, wobei sie überschlägig dieselbe Zeit benötigt wie das SQL-Statement, das das Löschen ausführte.

8.5.11. Tabellen- und Index-Strukturen

MySQL speichert seine Daten-Wörterbuch-Informationen über Tabellen in `.frm`-Dateien in den Datenbank-Verzeichnissen. Jedoch hat auch jede Tabelle vom Typ InnoDB ihren eigenen Eintrag, in InnoDB-internen Daten-Wörterbüchern innerhalb des Tabellenplatzes (Tablespace). Wenn MySQL eine Tabelle oder Datenbank löscht, muss er sowohl eine oder mehrere `.frm`-Datei(en) als auch die entsprechenden Einträge im InnoDB-Daten-Wörterbuch löschen. Das ist der Grund, warum Sie InnoDB-Tabellen nicht einfach zwischen Datenbanken verschieben können, indem Sie die `.frm`-Dateien verschieben und warum `DROP DATABASE` bei InnoDB-Tabellen in MySQL-Versionen bis 3.23.43 nicht funktionierte.

Jede InnoDB-Tabelle hat einen speziellen Index, der Cluster-Index genannt wird, in dem die Daten der Zeilen gespeichert sind. Wenn Sie auf Ihre Tabelle einen `PRIMARY KEY` definieren, ist der Index des Primärschlüssels der Cluster-Index.

Wenn Sie für Ihre Tabelle keinen Primärschlüssel definieren, erzeugt InnoDB intern einen Cluster-Index, bei dem die Zeilen nach der Zeilen-Kennung (ID) geordnet sind, die InnoDB Zeilen in einer solchen Tabelle zuweist. Die Zeilen-Kennung ist ein 6 Byte großes Feld, das monoton erhöht wird, wenn neue Zeilen eingefügt werden. Daher liegen nach der Zeilen-Kennung geordnete Zeile physikalisch in der Einfüge-Reihenfolge vor.

Der Zugriff auf eine Zeile über den Cluster-Index ist schnell, weil die Zeilendaten auf derselben Seite sind, auf die die Index-Suche führt. In vielen Datenbanken werden die Daten traditionell auf einer anderen Seite als derjenigen, wo sich der Index-Datensatz befindet, gespeichert. Wenn die Tabelle Groß ist, spart die Cluster-Index-Architektur im Vergleich zur traditionellen Lösung auf Festplatten-Ein- und -Ausgaben.

In InnoDB enthalten die Datensätze in Nicht-Cluster-Indexen (die wir auch sekundäre Indexe nennen) den Primärschlüsselwert für die Zeile. InnoDB benutzt diesen Primärschlüsselwert, um vom Cluster-Index aus nach der Zeile zu suchen. Beachten Sie, dass die sekundären Indexe mehr Platz benötigen, wenn der Primärschlüssel lang ist.

8.5.11.1. Physikalische Struktur eines Indexes

Alle Indexe in InnoDB sind B-Bäume, in denen die Index-Datensätze in den Blätter-Seiten des Baums gespeichert sind. Die vorgabemäßige Größe einer Index-Seite ist 16 KB. Wenn neue Datensätze eingefügt werden, versucht InnoDB, 1/16 der Seite für zukünftige Einfügungen und Aktualisierungen des Index-Datensatzes freizuhalten.

Wenn Index-Datensätze in sequentieller (aufsteigender oder absteigender) Reihenfolge eingefügt werden, sind die resultierenden Index-Seiten ungefähr zu 15/16 gefüllt. Wenn der Füllfaktor einer Index-Seite unter 1/12 fällt, versucht InnoDB, den Index-Baum

zusammenzuziehen, um die Seite freizugeben.

8.5.11.2. Einfügepufferung

Häufig wird der Primärschlüssel in Datenbank-Applikationen als eindeutiger Identifizierer benutzt und neue Zeilen in aufsteigender Reihenfolge des Primärschlüssels eingefügt. Daher erfordern Einfügungen in den Cluster-Index keine wahlfreien (random) Lesezugriffe auf die Platte.

Sekundäre Indexe auf der anderen Seite sind üblicherweise nicht eindeutig und Einfügungen in sekundäre Indexe erfolgen in einer relativ wahlfreien Reihenfolge. Wenn InnoDB keinen speziellen Mechanismus hierfür benutzen würde, würden diese viele wahlfreie Festplatten-Ein- und -Ausgaben verursachen.

Wenn ein Index-Datensatz in einen nicht eindeutigen sekundären Index eingefügt werden soll, prüft InnoDB, ob die sekundäre Index-Seite bereits im Puffer-Pool ist. Wenn das der Fall ist, führt InnoDB das Einfügen direkt in die Index-Seite durch. Wenn die Index-Seite aber nicht im Puffer-Pool gefunden wird, fügt InnoDB den Datensatz in eine spezielle Einfüge-Puffer-Struktur ein. Der Einfüge-Puffer wird so klein gehalten, dass er komplett in den Puffer-Pool passt, so dass Einfügungen sehr schnell durchgeführt werden können.

Der Einfüge-Puffer wird periodisch mit den sekundären Index-Bäumen in der Datenbank vermengt. Oft können mehrere Einfügeoperationen auf derselben Seite im Index-Baum zusammengefasst werden, so dass Festplatten-Ein- und -Ausgaben eingespart werden. Messungen ergaben, dass der Einfüge-Puffer Einfügungen in eine Tabelle bis zu 15 mal schneller machen kann.

8.5.11.3. Anpassungsfähige Hash-Indexe

Wenn eine Datenbank fast komplette in den Hauptspeicher passt, können Anfragen am schnellsten unter Verwendung von Hash-Indexen ausgeführt werden. InnoDB hat einen automatischen Mechanismus, der Index-Suchen beobachtet, die auf den Indexen durchgeführt werden, die für eine Tabelle definiert wurden. Wenn InnoDB bemerkt, dass Anfragen vom Aufbauen eines Hash-Indexes profitieren könnten, wird ein solcher Index automatisch aufgebaut.

Beachten Sie aber, dass der Hash-Index immer auf der Grundlage eines bestehenden B-Baum-Indexes auf die Tabelle aufgebaut wird. InnoDB kann einen Hash-Index auf einem Präfix beliebiger Länge des Schlüssels aufbauen, der für den B-Baum definiert wurde, abhängig vom Suchmuster, das InnoDB auf dem Index-Baum beobachtet. Ein Hash-Index kann partiell sein: Es ist nicht erforderlich, dass der gesamte Index-Baum im Puffer-Pool zwischengespeichert ist. InnoDB baut Hash-Indexe bei Bedarf automatisch für die Index-Seiten auf, auf die oft zugegriffen wird.

In gewisser Hinsicht kommt InnoDB durch den anpassungsfähigen Hash-Index-Mechanismus (wobei sich InnoDB üppig verfügbarem Hauptspeicher anpasst) der Architektur von Hauptspeicher-Datenbanken nahe.

8.5.11.4. Physikalische Datensatzstruktur

- Jeder Index-Datensatz in InnoDB enthält einen Header von 6 Bytes. Der Header wird benutzt, um nachfolgende Datensätze zu verknüpfen, sowie beim Sperren auf Zeilenebene.
- Datensätze im Cluster-Index enthalten Felder für alle benutzerdefinierten Spalten. Zusätzlich gibt es ein 6 Byte großes Feld für die Transaktions-Kennung und ein 7 Byte großes Feld für den Roll-Zeiger.
- Wenn der Benutzer keinen Primärschlüssel für eine Tabelle definiert hat, enthält jeder Cluster-Index-Datensatz zusätzlich ein 6 Byte großes Zeilenkennungsfeld.
- Jeder sekundäre Index-Datensatz enthält auch alle Felder, die für den Cluster-Index-Schlüssel definiert wurden.
- Ein Datensatz enthält auch einen Zeiger zu jedem Feld des Datensatzes. Wenn die Gesamtlänge des Feldes in einem Datensatz kleiner als 128 Bytes ist, ist der Zeiger 1 Byte lang, ansonsten 2 Bytes.

8.5.11.5. Wie eine Auto-Increment-Spalte in InnoDB funktioniert

Wenn der Benutzer nach einem Datenbankstart zuerst einen Datensatz in eine Tabelle `T` einfügt, in der eine Auto-Increment-Spalte definiert wurde, und er keinen expliziten Wert für die Spalte angibt, führt InnoDB `SELECT MAX(auto-inc-column) FROM T` aus und weist den um 1 hochgezählten Wert der Spalte und dem Auto-Increment-Zähler der Tabelle zu. Wir sagen dazu, dass der Auto-Increment-Zähler für Tabelle `T` initialisiert wurde.

InnoDB führt dieselbe Prozedur der Initialisierung des Auto-Increment-Zählers für eine frisch erzeugte Tabelle durch.

Wenn Sie für die Auto-Increment-Spalte einen Wert von 0 angeben, beachten Sie, dass InnoDB die Zeile so behandelt, als hätten Sie den Wert nicht angegeben.

Wenn nach der Initialisierung des Auto-Increment-Zählers der Benutzer eine Zeile eingibt, in der er explizit den Spaltenwert angibt, und dieser größer als der aktuelle Zählerwert ist, wird der Zähler auf den angegebenen Spaltenwert gesetzt. Wenn der

Benutzer nicht explizit einen Wert angibt, zählt InnoDB den Zähler um 1 hoch und weist der Spalte diesen neuen Wert zu.

Der Auto-Increment-Mechanismus umgeht beim Zuweisen von Werten vom Zähler Sperren und Transaktionshandhabung. Daher können Lücken in der Nummernfolge entstehen, wenn Sie Transaktionen zurückrollen (Rollback), die Nummern vom Zähler erhalten haben.

Das Verhalten von Auto-Increment ist für die Fälle undefiniert, in denen ein Benutzer der Spalte einen negativen Wert gibt oder wenn der Wert größer als die größte Ganzzahl wird, die im festgelegten Ganzzahl-Typ gespeichert werden kann.

8.5.12. Verwaltung von Datei-Speicherplatz und Festplatten-Eingaben / -Ausgaben

8.5.12.1. Festplatten-Ein- und -Ausgaben

Bei Festplatten-Ein- und -Ausgaben benutzt InnoDB asynchrone Ein- und Ausgaben. Unter Windows NT benutzt es die nativen Ein- und Ausgaben, die vom Betriebssystem zur Verfügung gestellt werden. Unter Unix benutzt InnoDB simulierte asynchrone Ein- und Ausgaben, die in InnoDB eingebaut sind: InnoDB erzeugt eine Reihe von Ein-/Ausgabe-Threads, die sich um Ein- und Ausgabeoperationen kümmern, zum Beispiel Vorwärts-Lesen (Read-Ahead). Zukünftig werden wir auch für Windows NT simulierte Ein-/Ausgaben unterstützen sowie für die Unix-Versionen, die so etwas besitzen, native Ein-/Ausgaben.

Unter Windows NT benutzt InnoDB ungepufferte Ein- und Ausgaben. Das heißt, dass die Festplatten-Seiten, die InnoDB liest oder schreibt, nicht im Datei-Cache des Betriebssystems gepuffert werden. Das spart einiges an Arbeitsspeicher-Bandbreite.

Ab Version 3.23.41 benutzt InnoDB eine neuartige Datei-Flush-Technik, die Doublewrite heißt. Sie erhöht die Sicherheit bei Reparaturen nach Absturz, wenn ein Betriebssystemabsturz oder ein Stromausfall aufgetreten sind, und verbessert auf den meisten Unix-Versionen die Performance, indem die Notwendigkeit von Fsync-Operationen verringert wird.

Doublewrite bedeutet, dass InnoDB zuerst in einen zusammenhängenden Tabellenplatz (Tablespace) namens Doublewrite-Puffer schreibt, bevor Seiten in eine Daten-Datei geschrieben werden. Erst nachdem das Schreiben und Zurückschreiben (Flush) in den Doublewrite-Puffer fertig sind, schreibt InnoDB die Seiten an ihre korrekten Positionen in der Daten-Datei. Wenn das Betriebssystem mitten in einem Seiten-Schreiben abstürzt, findet InnoDB bei der Wiederherstellung eine gute Kopie der Seite im Doublewrite-Puffer.

Ab Version 3.23.41 können Sie auch eine Raw-Disk-Partition als Daten-Datei benutzen, obwohl das bisher noch nicht getestet wurde. Wenn Sie eine neue Daten-Datei erzeugen, müssen Sie das Schlüsselwort `newraw` unmittelbar nach der Daten-Datei-Größe in `innodb_data_file_path` angeben. Die Partition muss größer oder gleich der Größe sein, die Sie angeben. Beachten Sie, dass in InnoDB 1 MB 1024 x 1024 Bytes ist, während 1 MB in Festplatten-Spezifikationen üblicherweise 1.000.000 Bytes bedeutet.

```
innodb_data_file_path=hdd1:5Gnewraw;hdd2:2Gnewraw
```

Wenn Sie die Datenbank wieder starten, **müssen** -sue das Schlüsselwort in `raw` ändern. Ansonsten schreibt InnoDB über Ihre Partition!

```
innodb_data_file_path=hdd1:5Graw;hdd2:2Graw
```

Wenn Sie Raw-Disk benutzen, können Sie unter einigen Unixen ungepufferte Ein- und Ausgaben ausführen.

Es gibt zwei Vorwärts-Lesen-(Read-Ahead-)Heuristiken in InnoDB: sequentielles Vorwärts-Lesen und wahlfreies (random) Vorwärts-Lesen. Beim sequentiellen Vorwärts-Lesen bemerkt InnoDB, dass das Zugriffsschema auf ein Segment im Tabellenplatz (Tablespace) sequentiell ist. InnoDB schickt dann vorab einen Stapel von Lesevorgängen von Datenbankseiten an das Ein-/Ausgabesystem. Beim wahlfreien Vorwärts-Lesen bemerkt InnoDB, dass ein bestimmter Bereich im Tabellenplatz (Tablespace) im Zustand des vollständig Eingelesenwerdens in den Puffer-Pool zu sein scheint. Dann schickt InnoDB die verbleibenden Lesevorgänge an das Ein-/Ausgabesystem.

8.5.12.2. Speicherplatzverwaltung

Die Daten-Dateien, die Sie in der Konfigurationsdatei definieren, formen den Tabellenplatz (Tablespace) von InnoDB. Die Dateien werden einfach verkettet, um den Tabellenplatz (Tablespace) zu formen, es wird kein Striping benutzt. Momentan können Sie nicht direkt angeben, wo der Platz für Ihre Tabellen zugewiesen werden soll, ausser wenn Sie folgende Tatsache benutzen: InnoDB weist Speicherplatz von einem neu erzeugten Tabellenplatz (Tablespace) vom niedrigen Ende ausgehend zu.

Der Tabellenplatz (Tablespace) besteht aus Datenbankseiten, deren vorgabemäßige Größe 16 KB beträgt. Diese Seiten werden bis zu einer Ausdehnung von 64 aufeinander folgenden Seiten gruppiert. Die 'Dateien' innerhalb eines Tabellenplatzes (Tablespace) werden in InnoDB Segmente genannt. Der Name des Rollback-Segments ist in gewisser Hinsicht irreführend, weil dieses tatsächlich viele Segmente im Tabellenplatz enthält.

Für jeden Index in InnoDB werden zwei Segmente zugewiesen: eins für die Nicht-Blätter-Knoten (Non-Leaf-Nodes) des B-Baum,

das andere für die Blätter-Knoten. Die Idee dahinter ist, für die Blätter-Knoten, die die Daten enthalten, bessere Sequentialität zu erzielen.

Wenn ein Segment innerhalb des Tabellenplatzes anwächst, weist ihm InnoDB die ersten 32 Seiten individuell zu. Danach fängt InnoDB an, dem Segment ganze Ausdehnungen zuzuweisen. InnoDB kann einem großen Segment bis zu vier Ausdehnungen auf einmal hinzufügen, um gute Sequentialität für die Daten sicherzustellen.

Einige Seiten im Tabellenplatz enthalten Bitmaps anderer Seiten. Daher können einige Ausdehnungen in einem InnoDB-Tabellenplatz (Tablespace) nicht Segmenten als Ganzes zugewiesen werden, sondern nur als individuelle Seiten.

Wenn Sie eine Anfrage `SHOW TABLE STATUS FROM ... LIKE ...` ausführen, um den verfügbaren freien Platz im Tabellenplatz festzustellen, berichtet InnoDB den Platz, der in völlig freien Ausdehnungen im Tabellenplatz sicher benutzt werden kann. InnoDB reserviert immer einige Ausdehnungen für Säuberungs- und interne Zwecke. Diese Ausdehnungen werden nicht in den freien Platz einbezogen.

Wenn Sie Daten aus einer Tabelle löschen, zieht InnoDB die entsprechenden B-Baum-Indexe zusammen. Es hängt vom Schema der Löschvorgänge ab, ob das individuelle Seiten oder Ausdehnungen im Tabellenplatz freigibt, so dass der freigegebene Platz anderen Benutzern zur Verfügung steht. Wenn eine Tabelle gelöscht wird oder alle Zeilen aus ihr gelöscht werden, gibt das garantiert Platz frei für andere Benutzer, aber denken Sie daran, dass gelöschte Zeile physikalisch nur durch eine Purge-Operation entfernt werden können, nachdem Sie nicht mehr für ein Transaktions-Rollback oder für Konsistentes Lesen benötigt werden.

8.5.12.3. Eine Tabelle defragmentieren

Wenn es wahlfreie (random) Einfüge- oder Löschvorgänge in die Indexe einer Tabelle gibt, können die Indexe fragmentiert werden. Unter Fragmentierung verstehen wird, dass die physikalische Reihenfolge der Index-Seiten auf der Platte der alphabetischen Reihenfolge der Datensätze auf den Seiten nicht nahe kommt oder dass es viele unbenutzte Seiten in den 64-Seiten-Blöcken gibt, die dem Index zugewiesen wurden.

Index-Scans können beschleunigt werden, wenn Sie von Zeit zu Zeit `mysqldump` benutzen, um die Tabelle in eine Textdatei zu dumpen, dann die Tabelle zu löschen und sie aus dem Dump neu aufzubauen. Eine weitere Möglichkeit zur Defragmentierung besteht darin, den Tabellentyp in `MyISAM` zu ändern (`ALTER`) und danach wieder in `InnoDB` zurück. Beachten Sie, dass die `MyISAM`-Tabelle auf Ihrem Betriebssystem in eine einzige Datei passen muss.

Wenn die Einfügungen in einen Index immer aufsteigend sind und Datensätze nur vom Ende gelöscht werden, garantiert der Speicherplatzverwaltungs-Algorithmus von InnoDB, dass keine Fragmentierung im Index auftritt.

8.5.13. Fehlerbehandlung

Die Fehlerbehandlung in InnoDB ist nicht immer so, wie es die ANSI-SQL-Standards festlegen. Nach ANSI-Standard sollte jeder Fehler während eines SQL-Statements ein Rollback des Statements verursachen. InnoDB rollt manchmal nur Teile des Statements oder auch die gesamte Transaktion zurück. Folgende Liste gibt die Fehlerbehandlung von InnoDB an:

- Wenn es keinen Speicherplatz mehr im Tabellenplatz (Tablespace) gibt, bekommen Sie den MySQL-Fehler `'Table is full'` und InnoDB rollt das SQL-Statement zurück.
- Eine Transaktions-Blockierung oder eine Zeitüberschreitung beim Warten auf eine Sperre führen dazu, dass InnoDB die gesamte Transaktion zurückrollt.
- Ein Fehler wegen doppelter Schlüsseleinträge rollt das Einfügen dieser Zeile zurück, selbst in einem Statement wie `INSERT INTO ... SELECT ...`. Das wird sich voraussichtlich ändern, so dass das SQL-Statement zurückgerollt wird, wenn Sie die `IGNORE`-Option in Ihrem Statement nicht angegeben haben.
- Ein Fehler 'row too long' rollt das SQL-Statement zurück.
- Andere Fehler werden zumeist durch die MySQL-Code-Ebene entdeckt und rollen das entsprechende SQL-Statement zurück.

8.5.14. Beschränkungen von InnoDB-Tabellen

- **ACHTUNG:** Konvertieren Sie **KEINE** MySQL-Systemtabellen von MyISAM in InnoDB-Tabellen! Das wird nicht unterstützt. Wenn Sie es dennoch tun, startet MySQL nicht mehr, bis Sie die alten Systemtabellen aus einer Datensicherung wiederhergestellt haben oder sie mit dem mysql_install_db-Skript neu erzeugen.
- `SHOW TABLE STATUS` gibt keine genauen Statistiken über InnoDB-Tabellen, ausser über die physikalische Größe, die durch die Tabelle reserviert wird. Der Zeilenzähler ist nur eine grobe Schätzung, die bei der SQL-Optimierung benutzt wird.
- Wenn Sie versuchen, einen eindeutigen Index auf ein Präfix einer Spalte zu erzeugen, erhalten Sie einen Fehler:

```
CREATE TABLE T (A CHAR(20), B INT, UNIQUE (A(5))) TYPE = InnoDB;
```

Wenn Sie einen nicht eindeutigen Index auf ein Spaltenpräfix erzeugen, erzeugt InnoDB einen Index über die gesamte Spalte.

- `INSERT DELAYED` wird für InnoDB-Tabellen nicht unterstützt.
- Die MySQL-`LOCK TABLES`-Operation weiß nichts von InnoDB-Sperren auf Zeilenebene, die in bereits fertigen SQL-Statements gesetzt sind. Das bedeutet, dass Sie eine Tabellensperre auf eine Tabelle selbst dann erhalten können, wenn es noch Transaktionen anderer Benutzer gibt, die Sperren auf Zeilenebene auf dieselbe Tabelle haben. Daher kann es sein, dass Ihre Operationen auf die Tabelle warten müssen, wenn sie mit diesen Sperren anderer Benutzer kollidieren. Auch eine Blockierung ist möglich. Dennoch gefährdet das nicht die Transaktionsintegrität, weil sich die Sperren auf Zeilenebene, die InnoDB setzt, um die Integrität kümmern. Zusätzlich hindert eine Tabellensperren andere Transaktionen daran, weitere Sperren auf Zeilenebene (in einem konfliktbehafteten Sperrmodus) auf die Tabelle zu erlangen.
- Sie können keinen Schlüssel auf eine `BLOB`- oder `TEXT`-Spalte setzen.
- Eine Tabelle kann nicht mehr als 1.000 Spalten enthalten.
- `DELETE FROM TABLE` erzeugt die Tabelle nicht neu, sondern löscht statt dessen alle Zeilen, eine nach der anderen, was nicht sehr schnell ist. In zukünftigen MySQL-Versionen können Sie `TRUNCATE` benutzen, was schnell ist.
- Die vorgabemäßige Datenbank-Seitengröße in InnoDB beträgt 16 KB. Indem Sie den Code neu kompilieren, können Sie sie auf Werte zwischen 8 KB und 64 KB setzen. Die maximale Zeilenlänge beträgt etwas weniger als die Hälfte der Datenbank-Seite in den InnoDB-Versionen kleiner oder gleich 3.23.40. Ab Quelldistribution 3.23.41 dürfen BLOB- und TEXT-Spalten bis zu 4 GB Groß sein, die gesamte Zeilenlänge kann auch < 4 GB betragen. InnoDB speichert Felder, deren Größe kleiner oder gleich 128 Bytes beträgt, nicht auf separaten Seiten. Nachdem InnoDB die Zeile geändert hat, indem lange Felder auf separaten Seiten gespeichert werden, muss die restliche Zeilenlänge weniger als die Hälfte einer Datenbank-Seite betragen. Die maximale Schlüssellänge beträgt 7.000 Bytes.
- Auf einigen Betriebssystemen müssen Daten-Dateien kleiner als 2 GB sein. Die Gesamtgröße der Log-Dateien muss auf 32-Bit-Computern kleiner als 4 GB sein.
- Die maximale Größe des Tabellenplatzes (Tablespace) beträgt 4 Milliarden Datenbank-Seiten. Das ist auch die maximale Größe für eine Tabelle. Die minimale Größe des Tabellenplatzes (Tablespace) beträgt 10 MB.

8.5.15. InnoDB-Kontaktinformationen

Kontaktinformationen von Innobase Oy, Hersteller der InnoDB-Engine: Website: http://www.innodb.com/. E-Mail: `<Heikki.Tuuri@innodb.com>`

```
Telefon: 358-9-6969 3250 (Büro) 358-40-5617367 (mobil)
Innobase Oy Inc.
World Trade Center Helsinki
Aleksanterinkatu 17
P.O.Box 800
00101 Helsinki
Finnland
```

8.6. BDB- oder Berkeley_db-Tabellen

8.6.1. Überblick über BDB-Tabellen

Unterstützung für BDB-Tabellen ist in der MySQL-Quelldistribution seit Version 3.23.34 enthalten und in der MySQL-Max-Binärdistribution aktiviert.

BerkeleyDB, erhältlich unter http://www.sleepycat.com/, stattet MySQL mit einem transaktionalen Tabellen-Handler aus. Wenn Sie BerkeleyDB-Tabellen benutzen, haben Ihre Tabellen eine höhere Chance, Abstürze zu überleben. Zusätzlich stehen `COMMIT` und `ROLLBACK` für Transaktionen zur Verfügung. Die MySQL-Quelldistribution enthält eine BDB-Distribution, die eine Reihe kleiner Patches hat, damit sie glatter mit MySQL zusammen arbeitet. Sie können keine nicht gepatchte `BDB`-Version für MySQL verwenden.

Wir bei MySQL AB arbeiten in enger Kooperation mit Sleepycat, um die hohe Qualität der MySQL-/BDB-Schnittstelle zu halten.

Was den Support für BDB-Tabellen angeht, sehen wir uns in der Pflicht, unseren Benutzern zu helfen, Probleme zu lokalisieren und Ihnen zu helfen, einen reproduzierbaren Testfall für jegliche Probleme mit BDB-Tabellen zu erstellen. Solche ein Fall wird an Sleepycat weiter geleitet, die sich dann an uns wenden, um uns zu helfen, das Problem zu finden und zu beheben. Weil das also in zwei Schritten abläuft, kann es bei jeglichen Problemen mit BDB-Tabellen etwas länger dauern, diese zu lösen, als das bei anderen Tabellen-Handlern der Fall ist. Weil jedoch der BerkeleyDB-Code selbst auch von vielen sonstigen Applikationen benutzt wird,

sind hierbei keine großen Probleme zu erwarten. See Abschnitt 2.4.1, „Support den MySQL AB anbietet“.

8.6.2. BDB installieren

Wenn Sie eine Binärdistribution von MySQL herunter geladen haben, die Unterstützung für BerkeleyDB enthält, folgen Sie einfach den Anweisungen zur Installation einer Binärversion von MySQL. See Abschnitt 3.2.6, „MySQL-Binärdistributionen, die von MySQL AB kompiliert wurden“. See Abschnitt 5.7.5, „mysqld-max, ein erweiterter mysqld-Server“.

Um MySQL mit BerkeleyDB-Unterstützung zu kompilieren, laden Sie MySQL-Version 3.23.34 oder neuer herunter und konfigurieren Sie `MySQL` mit der `--with-berkeley-db`-Option.

See Abschnitt 3.3, „Installation der Quelldistribution“.

```
cd /pfad/zur/quelle/von/mysql-3.23.34
./configure --with-berkeley-db
```

Bitte sehen Sie wegen aktuellerer Informationen im Handbuch nach, das mit der `BDB`-Distribution mitgeliefert wird.

Obwohl BerkeleyDB selbst sehr gut getestet und zuverlässig ist, wird die MySQL-Schnittstelle noch als Beta-Qualität erachtet. Wir verbessern diese aktiv und optimieren sie, um sie sehr bald stabil zu bekommen.

8.6.3. BDB-Startoptionen

Wenn Sie mit `AUTOCOMMIT=0` fahren, werden Ihre Änderungen in `BDB`-Tabellen erst aktualisiert, wenn Sie `COMMIT` ausführen. Statt dessen können Sie `ROLLBACK` ausführen, um Ihre Änderungen zu verwerfen. See Abschnitt 7.7.1, „`BEGIN/COMMIT/ROLLBACK`-Syntax“.

Wenn Sie mit `AUTOCOMMIT=1` fahren (der Vorgabe), werden Ihre Änderungen sofort abgeschickt. Sie können eine ausgedehnte Transaktion mit dem SQL-Befehl `BEGIN WORK` starten. Danach werden Ihre Änderungen solange nicht abgeschickt, bis Sie `COMMIT` ausführen (oder sich für `ROLLBACK` entscheiden, um Ihre Änderungen zu verwerfen).

Folgende Optionen für `mysqld` können benutzt werden, um das Verhalten von BDB-Tabellen zu ändern:

Option	Beschreibung
`--bdb-home=directory`	Base Verzeichnis für BDB-Tabellen. Das sollte dasselbe Verzeichnis sein, das Sie für --datadir benutzen.
`--bdb-lock-detect=#`	Berkeley-Sperr-Erkennung. # steht für DEFAULT, OLDEST, RANDOM oder YOUNGEST.
`--bdb-logdir=Verzeichnis`	BerkeleyDB-Log-Datei-Verzeichnis.
`--bdb-no-sync`	Flush-Logs nicht synchronisieren.
`--bdb-no-recover`	BerkeleyDB nicht im Wiederherstellungsmodus starten.
`--bdb-shared-data`	BerkeleyDB im Multi-Prozess-Modus starten (`DB_PRIVATE` bei der Initialisierung von BerkeleyDB nicht verwenden).
`--bdb-tmpdir=verzeichnis`	Name der temporären Datei von BerkeleyDB.
`--skip-bdb`	BerkeleyDB nicht benutzen.
`-O bdb_max_lock=1000`	Setzt die höchste Anzahl möglicher Sperren. See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“.

Wenn Sie `--skip-bdb` benutzen, initialisiert MySQL nicht die BerkeleyDB-Bibliothek und spart deshalb viel Speicher. Natürlich können Sie `BDB`-Tabellen nicht benutzen, wenn Sie diese Option verwenden.

Normalerweise sollten Sie `mysqld` ohne `--bdb-no-recover` starten, wenn Sie vorhaben, BDB-Tabellen zu verwenden. Das kann allerdings zu Problemen führen, wenn Sie `mysqld` starten und die BDB-Log-Dateien beschädigt sind. See Abschnitt 3.4.2, „Probleme mit dem Start des MySQL-Servers“.

Mit `bdb_max_lock` können Sie die maximale Anzahl von Sperren festlegen (vorgabemäßig 10.000), die auf einer BDB-Tabelle aktiv sein können. Sie sollten diesen Wert herauf setzen, wenn Sie Fehler vom Typ `bdb: Lock table is out of available locks` oder `Got error 12 from ...` erhalten, wenn Sie lange Transaktionen ausführen oder wenn `mysqld` viele Zeilen untersuchen muss, um die Anfrage zu berechnen.

Sie könnten auch `binlog_cache_size` und `max_binlog_cache_size` ändern, wenn Sie große, vielzeilige Transaktionen benutzen. See Abschnitt 7.7.1, „`BEGIN/COMMIT/ROLLBACK`-Syntax“.

8.6.4. Kennzeichen von BDB-Tabellen

- Um Transaktionen zurückrollen zu können, unterhält BDB Log-Dateien. Um maximale Performance zu erzielen, sollten Sie diese auf andere Festplatten platzieren als Ihre Datenbanken, indem Sie die `--bdb_log_dir`-Option benutzen.
- MySQL macht jedes Mal, wenn eine neue BDB-Log-Datei gestartet wird, einen Checkpoint und entfernt alle Log-Dateien, die nicht für aktuelle Transaktionen benötigt werden. Sie können auch jederzeit `FLUSH LOGS` laufen lassen, um einen Checkpoint für die BerkeleyDB-Tabellen anzulegen.

  Für die Wiederherstellung nach Abstürzen sollten Sie Datensicherungen der Tabellen plus das Binär-Log von MySQL benutzen. See Abschnitt 5.4.1, „Datenbank-Datensicherungen“.

  Achtung: Wenn Sie alte Log-Dateien löschen, die in Benutzung sind, ist BDB nicht in der Lage, Wiederherstellungen durchzuführen, und Sie könnten Daten verlieren, wenn etwas schief geht.
- MySQL erfordert einen `PRIMARY KEY` in jeder BDB-Tabelle, um auf vorher gelesene Zeilen verweisen zu können. Wenn Sie keine Primärschlüssel anlegen, erzeugt MySQL einen versteckten `PRIMARY KEY`. Der versteckte Schlüssel hat eine Länge von 5 Bytes und wird bei jedem Einfügeversuch um 1 hochgezählt.
- Wenn alle Spalten, auf die Sie in einer `BDB`-Tabelle zugreifen, Teil desselben Indexes oder Teil des Primärschlüssels sind, kann MySQL die Anfrage ausführen, ohne auf die tatsächliche Zeile zugreifen zu müssen. Bei einer `MyISAM`-Tabelle gilt das nur, wenn die Spalten Teil desselben Indexes sind.
- Der `PRIMARY KEY` ist schneller als jeder andere Schlüssel, weil `PRIMARY KEY` zusammen mit den Zeilendaten gespeichert wird. Weil die anderen Schlüssel als Schlüsseldaten plus `PRIMARY KEY` gespeichert werden, ist es wichtig, den `PRIMARY KEY` so kurz wie möglich zu halten, um Plattenplatz zu sparen und bessere Geschwindigkeit zu erzielen.
- `LOCK TABLES` funktioniert bei `BDB`-Tabellen wie bei anderen Tabellen. Wenn Sie `LOCK TABLE` nicht benutzen, führt MySQL einer interne mehrfache Schreibsperre auf die Tabelle aus, um sicherzustellen, dass die Tabelle korrekt gesperrt ist, wenn ein anderer Thread eine Tabellensperre ausführt.
- Internes Sperren in `BDB`-Tabellen wird auf Seitenebene durchgeführt.
- `SELECT COUNT(*) FROM tabelle` ist langsam, weil `BDB`-Tabellen keinen Zähler für die Anzahl der Zeilen in der Tabelle unterhalten.
- Scannen ist langsamer als bei `MyISAM`-Tabellen, weil Daten in BDB-Tabellen in B-Bäumen und nicht in separaten Daten-Dateien gespeichert werden.
- Die Applikation muss stets darauf vorbereitet sein, Fälle zu handhaben, bei denen jegliche Änderung einer `BDB`-Tabelle zu einem automatischen Rollback führen kann und jegliches Lesen fehlschlagen kann, weil ein Blockierungsfehler auftritt.
- Schlüssel werden nicht auf vorherige Schlüssel komprimiert, wie das bei ISAM- und MyISAM-Tabellen der Fall ist. Mit anderen Worten benötigt die Schlüsselinformation etwas mehr Platz bei `BDB`-Tabellen im Vergleich zu MyISAM-Tabellen, die nicht `PACK_KEYS=0` benutzen.
- Oft gibt es Löcher in der BDB-Tabelle, damit Sie neue Zeilen in der Mitte des Schlüsselbaums einfügen können. Das macht BDB-Tabellen etwas größer als MyISAM-Tabellen.
- Der Optimierer muss näherungsweise die Anzahl von Zeilen in der Tabelle kennen. MySQL löst dieses Problem, indem Einfügeoperationen gezählt werden, und unterhält diese in einem separaten Segment in jeder BDB-Tabelle. Wenn Sie nicht viele `DELETE` oder `ROLLBACK` ausführen, sollte diese Zahl ausreichend genau für den MySQL-Optimierer sein. Weil MySQL die Zahl nur beim Schließen speichert, kann sie falsch sein, wenn MySQL unerwartet stirbt. Das sollte kein schwerer Fehler sein, selbst wenn die Zahl nicht 100% korrekt ist. Man kann die Anzahl von Zeilen aktualisieren, indem man `ANALYZE TABLE` oder `OPTIMIZE TABLE` ausführt.

  See Abschnitt 5.5.2, „`ANALYZE TABLE`-Syntax“. See Abschnitt 5.5.1, „`OPTIMIZE TABLE`-Syntax“.
- Wenn die Platte bei einer `BDB`-Tabelle voll wird, erhalten Sie einen Fehler (wahrscheinlich Fehler 28) und die Transaktion sollte zurückgerollt werden. Das steht im Gegensatz zu `MyISAM`- and `ISAM`-Tabellen, bei denen `mysqld` wartet, bis genug Plattenplatz frei ist, bevor weiter gemacht wird.

8.6.5. Was in naher Zukunft bei BDB in Ordnung gebracht werden muss

- Viele BDB-Tabellen zur gleichen Zeit öffnen ist sehr langsam. Wenn Sie BDB-Tabellen benutzen wollen, sollten Sie einen sehr großen Tabellen-Cache haben (evtl. größer als 256) und beim `mysql`-Client `--no-auto-rehash` benutzen. Das soll partiell in Version 4.0 behoben werden.

- `SHOW TABLE STATUS` gibt momentan noch nicht viele Informationen über BDB-Tabellen aus.
- Performance optimieren.
- Es sollten überhaupt keine Seitensperren mehr benutzt werden, wenn Tabellen gescannt werden.

8.6.6. Betriebssysteme, die von BDB unterstützt werden

Wenn Sie MySQL mit Unterstützung für BDB-Tabellen gebaut haben und folgenden Fehler in der Log-Datei sehen, wenn Sie `mysqld` starten:

```
bdb: architecture lacks fast mutexes: applications cannot be threaded
Can't init databases
```

Bedeutet das, dass `BDB`-Tabellen für Ihre Architektur nicht unterstützt werden. In diesem Fall müssen Sie MySQL erneut bauen, ohne Unterstützung für BDB-Tabellen.

HINWEIS: Folgende Liste ist nicht komplett. Sie wird aktualisiert, sobald wir mehr Informationen darüber haben.

Momentan wissen wir, dass BDB-Tabellen auf folgenden Betriebssystemen laufen:

- Linux 2.x intel
- Solaris sparc
- Caldera (SCO) OpenServer
- Caldera (SCO) UnixWare 7.0.1

Auf folgenden Betriebssystemen läuft BDB nicht:

- Linux 2.x Alpha
- Mac OS X

8.6.7. Fehler, die bei der Benutzung von BDB-Tabellen auftreten können

- Wenn Sie folgenden Fehler in der `hostname.err`-Log-Datei beim Start von `mysqld` erhalten:

  ```
  bdb:  Ignoring log file: .../log.XXXXXXXXXX: unsupported log version #
  ```

  Bedeutet das, dass die neue `BDB`-Version das alte Log-Dateiformat nicht unterstützt. In diesem Fall müssen Sie alle `BDB`-Log-Dateien aus Ihrem Datenbankverzeichnis löschen (die Dateien haben das Format `log.XXXXXXXXXX`) und `mysqld` neu starten. Wir empfehlen ausserdem, dass Sie `mysqldump --opt` auf Ihre alten `BDB`-Tabellen ausführen, die alten Tabellen löschen und aus dem Dump wiederherstellen.

- Wenn Sie im `auto_commit`-Modus fahren und eine Tabelle löschen, die durch einen anderen Thread benutzt wird, erhalten Sie womöglich folgende Fehlermeldungen in der MySQL-Fehlerdatei:

  ```
  001119 23:43:56  bdb:  Missing log fileid entry
  001119 23:43:56  bdb:  txn_abort: Log undo failed for LSN: 1 3644744: Invalid
  ```

  Das ist kein schwerer Fehler, aber wir empfehlen, alle Tabellen zu löschen, wenn Sie nicht im `auto_commit`-Modus sind, bis dieses Problem behoben ist (die Behebung ist nicht trivial).

Kapitel 9. MySQL-APIs

Dieses Kapitel beschreibt die APIs, die für MySQL bereitstehen, wo man sie bekommt und wie man sie benutzt. Die C-API ist am ausführlichsten beschrieben, da sie vom MySQL-Team stammt und als Basis für die meisten anderen APIs dient.

9.1. MySQL-PHP-API

PHP ist eine serverseitige Skriptsprache, die in HTML eingebettet werden kann und mit der man dynamische Webseiten erstellen kann. PHP unterstützt eine Vielzahl von Datenbanken. Darunter befindet sich auch MySQL. PHP kann als alleinstehendes Programm oder als Teil des Apache Webservers eingesetzt werden.

Die Distribution und die Dokumentation gibt es unter PHP-Website.

9.1.1. Allgemeine Probleme mit MySQL und PHP

- Error: "Maximum Execution Time Exceeded" Dies ist eine PHP-Beschränkung. Ändern sie den Wert für die maximale Ausführungszeit in der `php3.ini`-Datei. Es ist ausserdem keine schlechte Idee, die Beschränkung für die maximale Benutzung von RAM von 8 MB auf 16 MB per Skript zu verdoppeln.
- Error: "Fatal error: Call to unsupported oder undefined function mysql_connect() in .." Das bedeutet, dass Ihre PHP-Version nicht mit MySQL-Unterstützung ausgestattet ist. Sie können entweder ein dynamisches MySQL-Modul für PHP kompilieren oder PHP mit seiner eingebautet MySQL-Unterstützung neu kompilieren. Im PHP-Manual ist dies ausführlich beschrieben.
- Error: "undefined reference to `uncompress'" Die Client-Bibliothek wurde mit der Unterstützung für ein komprimiertes Client-/Server-Protokoll kompiliert. Um den Fehler zu beheben, müssen Sie `-lz` als letztes angeben, wenn Sie gegen `-lmysqlclient` linken.

9.2. MySQL-Perl-API

Dieser Abschnitt dokumentiert die Perl-`DBI`-Schnittstelle. Die frühere Schnittstelle hieß `mysqlperl`. `DBI`/`DBD` ist jetzt die empfohlene Perl-Schnittstelle. `mysqlperl` ist überflüßig und deshalb hier nicht näher beschrieben.

9.2.1. `DBI` mit `DBD::mysql`

`DBI` ist eine allgemeine Schnittstelle für viele Datenbanken. Das bedeutet, Sie können ein Skript schreiben, dass viele verschiedene Datenbanken unterstützt, ohne es zu ändern. Sie brauchen für jeden Datenbanktyp einen Datenbank-Treiber (DBD). Für MySQL heißt dieser Treiber `DBD::mysql`. Für weitere Informationen über Perl5 DBI besuchen Sie bitte die `DBI`-Website und lesen Sie die Dokumentation:

```
http://www.symbolstone.org/technology/perl/DBI/index.html
```

Für weitere Informationen über objektorientierte Programmierung (OOP) in Perl5 besuchen Sie die OOP-Seite:

```
http://language.perl.com/info/documentation.html
```

Beachten Sie, dass Sie, wenn Sie Transaktionen mit Perl einsetzen wollen, `Msql-Mysql-modules` der Version 1.2216 oder neuer benötigen.

Installationsanweisungen für MySQL-Perl-Unterstützung finden Sie unter Abschnitt 9.2, „MySQL-Perl-API“.

9.2.2. Die `DBI`-Schnittstelle

Portable DBI-Methoden

`connect`	Errichtet eine Verbindung zum Datenbankserver.
`disconnect`	Trennt eine Verbindung zum Datenbankserver.
`prepare`	Bereitet ein SQL-Statement zur Abfrage vor.
`execute`	Führt eine vorbereitetes Statement aus.
`do`	Bereitet ein SQL-Statement vor und führt es aus.
`quote`	Quotet eine Zeichenkette oder einen `BLOB`-Wert zum Einfügen.
`fetchrow_array`	Holt die nächste Zeile als einen Array aus Feldern.

`fetchrow_arrayref`	Holt die nächste Zeile als eine Referenz eines Arrays aus Feldern.
`fetchrow_hashref`	Holt die nächste Zeile als eine Referenz einer Hash-Tabelle.
`fetchall_arrayref`	Holt alle Zeilen als einen Array von Arrays.
`finish`	Beendet ein Statement und läßt das System Resourcen freigeben.
`rows`	Gibt die Anzahl der betroffenen Zeilen zurück.
`data_sources`	Gibt einen Array mit den verfügbaren Daten auf localhost zurück.
`ChopBlanks`	Kontroliert, ob die `fetchrow_*`-Methoden Leerzeichen entfernen.
`NUM_OF_PARAMS`	Die Anzahl der Platzhalter in einem vorbereiteten Statement.
`NULLABLE`	Welche Spalten `NULL` sein können.
`trace`	Tracen zum Debuggen ausführen.

MySQL-spezifische Methoden

`insertid`	Der letzte `AUTO_INCREMENT`-Wert.
`is_blob`	Welche Spalten `BLOB`-Werte sind.
`is_key`	Welche Spalten Schlüssel sind.
`is_num`	Welche Spalten numerisch sind.
`is_pri_key`	Welche Spalten Primärschlüssel sind.
`is_not_null`	Welche Spalten NICHT `NULL` sein können. Siehe auch `NULLABLE`.
`length`	Maximal mögliche Spaltengröße.
`max_length`	Maximale Spaltengröße, die im aktuellen Ergebnis enthalten ist.
`NAME`	Spaltennamen.
`NUM_OF_FIELDS`	Anzahl der zurückgegebenen Felder.
`table`	Tabellennamen im zurückgegebenen Ergebnis.
`type`	Alle Spaltentypen.

Die Perl-Methoden werden im Folgenden detaillierter erläutert. Die Variablen für die zurückgegebenen Werte haben folgende Bedeutung:

- `$dbh`

  Datenbank-Handle

- `$sth`

  Statement-Handle

- `$rc`

  Rückgabe-Code (oft ein Status)

- `$rv`

  Rückgabewert (oft ein Status)

Portable DBI-Methoden

- `connect($datenquelle, $benutzername, $passwort)`

  Benutzen Sie die `connect`-Methode, um eine Verbindung zur Datenbank der Datenquelle herzustellen. Der `$datenquelle`-Wert sollte mit `DBI:Treiber_name:` beginnen. Beispielanwendungen von `connect` mit dem `DBD::mysql` Treiber:

```
$dbh = DBI->connect("DBI:mysql:$datenbank", $benutzer, $passwort);
$dbh = DBI->connect("DBI:mysql:$datenbank:$hostname",
                    $benutzer, $passwort);
$dbh = DBI->connect("DBI:mysql:$datenbank:$hostname:$port",
                    $benutzer, $passwort);
```

Wenn der Benutzername und / oder das Passwort nicht angegeben werden, verwendet `DBI` die Werte der `DBI_USER`- und `DBI_PASS`- Umgebungsvariablen. Wen Sie keinen Hostnamen angeben, wird `'localhost'` verwendet. Wenn Sie keine Portnummer angeben, wird der MySQL-Port (3306) verwendet.

Seit `Msql-Mysql-modules`-Version 1.2009 erlaubt der `$datenquelle`-Wert bestimmte Modifikatoren:

- `mysql_read_default_file=datei`

  Liest `datei` als eine Optionsdatei. Für weitere Informationen zu Optionsdateien beachten Sie bitte Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

- `mysql_read_default_group=group_name`

  Beim Lesen einer Optionsdatei ist die Standardgruppe normalerweise die `[client]`-Gruppe. Wenn Sie die `mysql_read_default_group`- Option angeben, wird die Standardgruppe `[gruppenname]`.

- `mysql_compression=1`

  Aktiviert die Kompression während der Kommunikation zwischen Client und Server (ab Version 3.22.3).

- `mysql_socket=/pfad/zur/socket`

  Gibt den Pfad des Unix-Sockets an, der zum Verbinden mit dem Server verwendet wird (MySQL-Version 3.21.15 oder neuer).

Sie können mehrere Modifikatoren angeben, dabei muss jedem ein Semikolon vorangestellt sein.

Wenn Sie zum Beispiel vermeiden wollen, dass sie Benutzername und Passwort im `DBI`-Skript angeben müssen, können Sie sie aus der `~/.my.cnf`-Optionsdatei nehmen. Ihr `connect`-Aufruf sieht folgendermaßen aus:

```
$dbh = DBI->connect("DBI:mysql:$datenbank"
                . ";mysql_read_default_file=$ENV{HOME}/.my.cnf",
                $benutzer, $passwort);
```

Dieser Aufruf liest die Optionen für die `[client]`-Gruppe aus der Optionsdatei. Wenn Sie dasselbe für die `[perl]`-Gruppe tun wollen, könnte Ihr Code so aussehen:

```
$dbh = DBI->connect("DBI:mysql:$Datenbank"
                . ";mysql_read_default_file=$ENV{HOME}/.my.cnf"
                . ";mysql_read_default_group=perl",
                $benutzer, $passwort);
```

- `disconnect`

  Die `disconnect`-Methode beendet die Verbindung mit der Datenbank. Dies wird typischerweise kurz vor dem Ende eines Scripts ausgeführt. Beispiel:

  ```
  $rc = $dbh->disconnect;
  ```

- `prepare($statement)`

  Bereitet ein SQL-Statement zum Ausführen durch den Datenbankserver vor und gibt ein "Statement-Handle" `($sth)` zurück, mit der Sie die `execute`-Methode aufrufen. Normalerweise werden Sie `SELECT`-Statements (und `SELECT`-ähnliche Statements so wie `SHOW`, `DESCRIBE` und `EXPLAIN`) mit der Bedeutung von `prepare` und `execute` verwenden. Beispiel:

  ```
  $sth = $dbh->prepare($statement)
      or die "$statement: $dbh->errstr kann nicht vorbereitet werden\n";
  ```

- `execute`

  Die `execute`-Methode führt ein vorbereitetes Statement aus. Bei Nicht-`SELECT`-Statements gibt `execute` die Anzahl der betroffenen Zeilen zurück. Wenn Zeilen betroffen sind, gibt `execute "0E0"` zurück, was in Perl als 0 und true erkannt wird. Wenn ein Fehler auftritt, gibt `execute undef` zurück. Bei `SELECT`-Statements beginnt `execute` die SQL-Anfrage in der Datenbank; Sie müssen eine der `fetch_*`-Methoden nutzen, die weiter unten beschrieben sind, um Daten erhalten. Beispiel:

```
$rv = $sth->execute
          or die "Die Query: $sth->errstr kann nicht ausgeführt werden.";
```

- `do($statement)`

  Die `do`-Methode bereitet ein Statement vor, führt es aus und gibt die Anzahl der betroffenen Zeilen zurück. Wenn Zeilen betroffen sind, gibt `execute "0E0"` zurück, was in Perl als 0 und true erkannt wird. Diese Methode wird normalerweise verwendet, um Nicht-`SELECT`-Statements zu bearbeiten, die (z. B. wegen Treiber-Beschränkungen) nicht vorbereitet werden können, oder die nicht mehr als einmal vorbereitet werden müssen (INSERTS, DELETE usw.). Beispiel:

```
$rv = $dbh->do($statement)
        or die "$statement: $dbh- >errstr kann nicht vorbereitet werden\n";
```

  Im Allgemeinen ist die `do`-Methode VIEL schneller (und vorzuziehen) als die `prepare`/`execute`-Methoden, die ohne Parameter aufgerufen werden.

- `quote($string)`

  Die `quote`-Methode wird verwendet, um Sonderzeichen zu "escapen", die in Zeichenketten enthalten sein können, und um notwendige äußere Anführungszeichen hinzuzufügen. Beispiel:

```
$sql = $dbh->quote($string)
```

- `fetchrow_array`

  Die Methode holt die nächste Datenzeile und gibt sie als ein Array mit den Feldwerten zurück. Beispiel:

```
while(@row = $sth->fetchrow_array) {
        print qw($row[0]\t$row[1]\t$row[2]\n);
}
```

- `fetchrow_arrayref`

  Die Methode holt die nächste Datenzeile und gibt sie als eine Referenz auf ein Array mit den Feldwerten zurück. Beispiel:

```
while($row_ref = $sth->fetchrow_arrayref) {
        print qw($row_ref->[0]\t$row_ref->[1]\t$row_ref->[2]\n);
}
```

- `fetchrow_hashref`

  Diese Methode holt eine Datenzeile und gibt eine Referenz auf einen Hash zurück, der Name-/Wert-Paare enthält. Die Methode ist lange nicht so performant wie das Verwenden von Referenzen auf ein Array, wie weiter oben beschrieben ist. Beispiel:

```
while($hash_ref = $sth->fetchrow_hashref) {
        print qw($hash_ref->{vorname}\t$hash_ref->{nachname}\t\
                $hash_ref- > title}\n);
}
```

- `fetchall_arrayref`

  Diese Methode gibt alle Zeilen eines Ergebnisses einer SQL-Anfrage zurück. Sie gibt eine Referenz auf ein Array mit Referenzen auf Arrays mit den Werten der einzelnen Zeilen zurück. Sie können mit zwei verschachtelten Schleifen auf die Werte zugreifen. Beispiel:

```
my $table = $sth->fetchall_arrayref
                or die "$sth->errstr\n";
my($i, $j);
for $i ( 0 .. $#{$table} ) {
        für $j ( 0 .. $#{$table->[$i]} ) {
                print "$table->[$i][$j]\t";
        }
        print "\n";
}
```

- `finish`

  Bewirkt, dass keine weiteren Daten von dem SQL-Anfrageergebnis geholt werden. Sie können diese Methode aufrufen, um Systemressourcen freizugeben. Beispiel:

```
s$rc = $sth->finish;
```

- `rows`

  Gibt die Anzahl der veränderten Zeilen (die aktualisiert oder gelöscht wurden) des letzten Befehls zurück. Dies wird normalerweise nach Nicht-`SELECT`-`execute`-Statements verwendet. Beispiel:

```
$rv = $sth->rows;
```

- `NULLABLE`

  Gibt eine Referenz auf ein Array mit Boole'schen Werten zurück; für jedes Element TRUE kann die Spalte `NULL`-Werte enthalten. Beispiel:

```
$null_possible = $sth->{NULLABLE};
```

- `NUM_OF_FIELDS`

  Dieses Attribut enthält die Anzahl der Zeilen, die eine `SELECT`- oder `SHOW FIELDS`-SQL-Anfrage zurückgibt. Sie können es verwenden, um zu prüfen, ob eine Anfrage ein Ergebnis zurückgegeben hat: 0 weist auf eine Nicht-`SELECT`-Anfrage hin, wie `INSERT`, `DELETE` oder `UPDATE`. Beispiel:

```
$nr_of_fields = $sth->{NUM_OF_FIELDS};
```

- `datasource($Treiber_name)`

  Diese Methode gibt einen Array zurück, der die Namen der verfügbaren Datenbanken auf `'localhost'` enthält. Beispiel:

```
@dbs = DBI->datasource("mysql");
```

- `ChopBlanks`

  Dieses Attribut gibt an, ob die `fetchrow_*`-Methoden vor- und nachstehende Leerzeichen entfernen. Beispiel:

```
$sth->{'ChopBlanks'} =1;
```

- `trace($trace_ebene)`, `trace($trace_ebene, $trace_dateiname)`

  `trace` aktiviert oder deaktiviert "Tracing". Wenn `DBI` als eine Klassenmethode aufgerufen wird, steuert es das "Tracing" mit allen Datenbankverbindungen. Wenn es als Datenbank- oder Statement-Handle-Methode aufgerufen wird, steuert es nur die verwendete Verbindung (und deren spätere Ableitungen). Wenn Sie `$trace_ebene` auf 2 setzen, bewirkt es detaillierte Informationen. Der Wert 0 stellt "Tracing" ab. Die Ausgabe des "Tracing" wird vorgabemäßig nach "standard error" geleitet. Wenn `$trace_dateiname` angegeben ist, wird die Ausgabe für *alle* "getraceten" Verbindungen an das Ende dieser Datei geschrieben. Beispiel:

```
DBI->trace(2);               # alles tracen
DBI->trace(2,"/tmp/dbi.out"); # alles nach /tmp/dbi.out tracen
$dth->trace(2);              # diese Datenbankverbindung tracen
$sth->trace(2);              # dieses Statement-Handle tracen.
```

  Sie können `DBI`-Tracing auch anschalten, indem Sie die `DBI_TRACE`-Umgebungsvariable setzen. Wenn Sie sie auf einen numerischen Wert setzen, ist das dasselbe, wie `DBI->(wert)` aufzurufen. Wenn Sie sie auf einen Pfadnamen setzen, ist das dasselbe, wie `DBI->(2,wert)` aufzurufen.

MySQL-spezifische Methoden

Die unten stehenden Methoden sind MySQL-spezifisch und nicht Teil des `DBI`-Standards. Mehrere von ihnen sind veraltet:

`is_blob`, `is_key`, `is_num`, `is_pri_key`, `is_not_null`, `length`, `max_length` und `table`. Wo immer es `DBI`-Standard-Alternativen gibt, ist das unten angemerkt:

- `insertid`

  Wenn Sie das `AUTO_INCREMENT`-Feature von MySQL benutzen, werden neue, automatisch heraufgezählte Werte hier gespeichert. Beispiel:

  ```
  $new_id = $sth->{insertid};
  ```

  Alternativ können Sie `$dbh->{'mysql_insertid'}` verwenden.

- `is_blob`

  Gibt eine Referenz auf einen Array mit Boole'schen Werten zurück; für jedes Element des Arrays bedeutet der Wert TRUE, dass die entsprechende Spalte ein `BLOB` ist. Beispiel:

  ```
  $keys = $sth->{is_blob};
  ```

- `is_key`

  Gibt eine Referenz auf einen Array mit Boole'schen Werten zurück; für jedes Element des Arrays bedeutet der Wert TRUE, dass die entsprechende Spalte ein Schlüssel ist. Beispiel:

  ```
  $keys = $sth->{is_key};
  ```

- `is_num`

  Gibt eine Referenz auf einen Array mit Boole'schen Werten zurück; für jedes Element des Arrays bedeutet der Wert TRUE, dass die entsprechende Spalte numerische Werte enthält. Beispiel:

  ```
  $nums = $sth->{is_num};
  ```

- `is_pri_key`

  Gibt eine Referenz auf einen Array mit Boole'schen Werten zurück; für jedes Element des Arrays bedeutet der Wert TRUE, dass die entsprechende Spalte ein Primärschlüssel ist. Beispiel:

  ```
  $pri_keys = $sth->{is_pri_key};
  ```

- `is_not_null`

  Gibt eine Referenz auf einen Array mit Boole'schen Werten zurück; für jedes Element des Arrays bedeutet der Wert FALSE, dass die entsprechende Spalte NULL enthalten kann. Beispiel:

  ```
  $not_nulls = $sth->{is_not_null};
  ```

  Das oben beschriebene `NULLABLE`-Attribut ist `is_not_null` in jedem Fall vorzuziehen, da es zum DBI-Standard gehört.

- `length`, `max_length`

  Beide Methoden geben je einen Array mit Spaltenlängen zurück. Der `length`-Array gibt die maximal mögliche Länge jeder Spalte an (wie es in der Tabellendefinition festgelegt wurde). Der `max_length`-Array gibt die Länge des aktuell längsten Wertes in den Spalten an. Beispiel:

  ```
  $lengths = $sth->{length};
  $max_lengths = $sth->{max_length};
  ```

- `NAME`

  Gibt eine Referenz auf ein Array mit den Spaltennamen zurück. Beispiel:

```
$names = $sth->{NAME};
```

- `table`

  Gibt eine Referenz auf ein Array mit den Tabellennamen zurück. Beispiel:

  ```
  $tables = $sth->{table};
  ```

- `type`

  Gibt eine Referenz auf ein Array mit den Spaltentypen zurück. Beispiel:

  ```
  $types = $sth->{type};
  ```

9.2.3. Weitere `DBI`/`DBD`-Informationen

Bitte verwenden Sie den `perldoc`-Befehl, um weitere Informationen über `DBI` zu erhalten.

```
perldoc DBI
perldoc DBI::FAQ
perldoc DBD::mysql
```

Sie können ausserdem `pod2man`, `pod2html` usw. verwenden, um in andere Formate zu wandeln.

Die neuesten `DBI`-Informationen finden Sie auf der `DBI` Website:

```
http://www.symbolstone.org/technology/perl/DBI/index.html
```

9.3. MySQL-ODBC-Unterstützung

MySQL unterstützt ODBC mit Hilfe des **MyODBC**-Programms. Dieses Kapitel erläutert, wie Sie **MyODBC** installieren und benutzen. Hier werden Sie außerdem eine Liste von Programmen finden, die mit **MyODBC** zusammenarbeiten.

9.3.1. Wie Sie MyODBC installieren

MyODBC ist ein 32-Bit-ODBC- (2.50) -Level-0- (mit Level-1- und Level-2-Features) Treiber für die Anbindung an ODBC-fähige Applikationen an MySQL. **MyODBC** funktioniert unter Windows95, Windows98, NT, und auf den meisten Unix-Plattformen.

MyODBC ist "public domain". Sie finden die neueste Version bei http://www.mysql.com/downloads/api-myodbc.html.

Wenn Sie ein Problem mit **MyODBC** haben und Ihr Programm auch mit OLEDB arbeitet, sollten sie den OLEDB Treiber probieren, den sie im "Contrib"-Abschnitt finden.

Normalerweise müssen Sie **MyODBC** nur auf Windows-Maschinen installieren. Sie brauchen **MyODBC** für Unix nur, wenn sie ein Programm wie ColdFusion haben, das auf einer Unix-Maschine läuft und ODBC für die Datenbankverbindung nutzt.

Wenn Sie **MyODBC** unter Unix installieren wollen, brauchen Sie noch einen **ODBC**-Manager. **MyODBC** arbeitet mit den meisten Unix-ODBC-Managern zusammen.

Um **MyODBC** unter Windows zu installieren, sollten sie die passende **MyODBC** Zip-Datei (für Windows 95/98 oder NT / Windows 2000) herunterladen, es mit `WINZIP` oder einem ähnlichen Programm entpacken, und die `SETUP.EXE`-Datei ausführen.

Unter Windows NT kann folgender Fehler während der Installation auftreten (**MyODBC**):

```
Während des Kopiervorgangs ist ein Fehler aufgetreten:
C:\WINDOWS\SYSTEM\MFC30.DLL. Starten Sie Windows neu und beginnen die
Installation erneut, noch bevor sie ein anderes Programm starten, das ODBC
verwendet.
```

Das Problem in diesem Fall ist, dass ein anderes Programm ODBC verwendet und dass unter Windows zwei Programme nicht gleichzeitig auf eine Datei zugreifen können. Deshalb kann es sein, dass Sie nicht in der Lage sind, die ODBC-Treiber mit Microsofts ODBC Setup Programm zu installieren. In den meisten Fällen genügt es, den `Ignorieren`-Knopf zu drücken, um die restlichen Dateien zu installieren und die Installation abzuschließen. Wenn das nicht funktioniert, booten Sie Ihren Rechner im Abgesicherten Modus, indem sie F8 vor dem Starten von Windows drücken und den Abgesicherten Modus auswählen. Installieren

sie **MyODBC**, und starten Sie wieder im normalen Modus.

- Um eine Verbindung mit einer ODBC-Applikation, die MySQL nicht nativ unterstützt, von Windows zu Unix herzustellen, müssen Sie zunächst **MyODBC** unter Windows installieren.
- Der Windows-Benutzer muss Zugriffsrechte auf den MySQL-Server der Unix-Maschine besitzen. Diese richten Sie mit dem `GRANT`-Befehl ein. See Abschnitt 5.3.1, „`GRANT`- und `REVOKE`-Syntax“.
- Sie müssen wie folgt einen ODBC-DSN-Eintrag erstellen:
  - Öffnen Sie die Systemsteuerung der Windows-Maschine.
  - Doppelklicken Sie das ODBC-Datenquellen-Symbol.
  - Klicken Sie auf die Registerkarte Benutzer-DSN.
  - Klicken Sie auf Hinzufügen.
  - Wählen Sie MySQL im Fenster "Neue Datenquelle hinzufügen" und klicken Sie auf den Fertig-Knopf.
  - Das MySQL-Treiber-Standard-Konfigurationsfenster wird nun angezeigt. See Abschnitt 9.3.2, „Wie Sie die verschiedenen Felder im ODBC-Administrator Programm ausfüllen“.
- Starten Sie nun ihre Applikation und wählen Sie den ODBC-Treiber mit der von ihnen im ODBC angegebenen DSN.

Bitte beachten Sie, dass weitere Konfigurationsoptionen im MySQL-Fenster vorhanden sind (trace, don't prompt on connect usw.). Probieren Sie diese aus, wenn Sie Probleme haben.

9.3.2. Wie Sie die verschiedenen Felder im ODBC-Administrator Programm ausfüllen

Es gibt drei Möglichkeiten, den Server unter Windows 95 anzugeben:

- Verwenden Sie die IP-Adresse des Servers.
- Fügen Sie der Datei `\windows\lmhosts` folgende Informationen an:

  ```
  ip hostname
  ```

  Beispiel:

  ```
  194.216.84.21 mein_hostname
  ```

- Konfigurieren Sie DNS:

Beispiel: Wie Sie das `ODBC setup` ausfüllen:

```
Windows DSN Name:   test
Beschreibung:       Das ist meine Datenbank
MySql Datenbank:    test
Server:             194.216.84.21
User:               monty
Password:           mein_passwort
Port:
```

Der Wert für `Windows DSN Namen` muss in ihrem Windows-ODBC-Setup eindeutig sein.

Sie müssen die Werte für `Server`, `User`, `Password` oder `Port` im ODBC-Setup-Fenster nicht angeben. Wenn Sie es jedoch tun, werden diese Werte als Standardwerte verwendet, wenn Sie versuchen, eine Verbindung aufzubauen. Sie können die Werte auch zur Laufzeit ihres Programms angeben.

Wenn Sie die Portnummer nicht angeben, wird der Standard-Port (3306) verwendet.

Wenn Sie die Option `Optionen aus C:\my.cnf lesen` angeben, werden die Gruppen `client` und `odbc` aus der `C:\my.cnf`-Datei gelesen. Sie können alle Optionen verwenden, die für `mysql_options()` gültig sind. See Abschnitt 9.4.3.38, „`mysql_options()`“.

9.3.3. Verbindungsparameter für MyODBC

Man kann die folgenden Parameter für **MyODBC** im `[Servername]`-Abschnitt in der `ODBC.INI`-Datei oder über das `InConnectionString`-Argument im `SQLDriverConnect()`-Aufruf angeben:

Parameter	Standardwert	Bedeutung
user	ODBC (unter Windows)	Der Benutzername, der verwendet wird, um zu MySQL zu verbinden.
server	localhost	Der Hostname des MySQL-Servers.
database		Die Standarddatenbank
option	0	Eine Ganzzahl, die angibt, wie **MyODBC** arbeiten soll. Siehe unten.
port	3306	Der TCP/IP-Port, der verwendet werden soll, wenn der `server` nicht `localhost` ist.
stmt		Ein Statement, das bei der Verbindung zu `MySQL` ausgeführt wird.
password		Das Passwort für die `server`-`user`-Kombination.
socket		Der Socket oder die Windows-Pipe, über die verbunden werden soll.

Die Option "argument" wird verwendet, um **MyODBC** zu sagen, dass der Client nicht 100% ODBC-kompatibel ist. Unter Windows setzt man diese Option normalerweise im Verbindungsdialog, Sie können aber auch das "option"-Argument verwenden. Die folgenden Optionen sind in derselben Reihenfolge wie im **MyODBC**-Verbindungsdialog:

Bit	Bedeutung
1	Der Client kann nicht damit umgehen, dass **MyODBC** die wirkliche Breite einer Spalte zurückgibt.
2	Der Client kann nicht damit umgehen, dass MySQL die wirkliche Anzahl an "affected rows" zurückgibt. Wenn dieses Bit gesetzt ist, wird MySQL statt dessen 'found rows' zurückgeben. Dies wird erst ab MySQL 3.21.14 unterstützt.
4	Erstellt ein Debug-Log in c:\myodbc.log. Das ist dasselbe, als wenn Sie `MYSQL_DEBUG=d:t:O,c:\myodbc.log` in Ihre `AUTOEXEC.BAT` schreiben.
8	Entfernt jede Paket-Beschränkung für Ergebnisse und Parameter.
16	Nicht auf Eingaben warten, sogar wenn der Treiber dies verlangt.
32	Einen ODBC 1.0 Treiber simulieren.
64	Die Angabe 'datenbank' in 'datenbank.tabelle.spalte' ignorieren.
128	Die Verwendung von ODBC-Manager-Zeigern erzwingen (experimentell).
256	Die Verwendung des erweiterten 'fetch' verbieten (experimentell).
512	CHAR-Felder bis zur vollen Spaltenlänge füllen.
1024	SQLDescribeCol() wird voll qualifizierte Spaltennamen zurückgeben.
2048	Verwendet das komprimierte Client-/Server Protokoll.
4096	Weist den Server an, Leerzeichen nach einem Funktionsnamen und vor `'('` zu ignorieren (wird von PowerBuilder benötigt). So werden alle Funktionsnamen zu Schlüsselwörtern!
8192	Über "Named Pipes" zu einem `mysqld`-Server verbinden, der unter Windows NT läuft.
16384	Ändert LONGLONG-Spalten zu INT-Spalten (einige Applikationen können mit LONGLONG nicht umgehen).
32768	Gibt 'user' als Tabellenqualifizierer und Tabellen-Besitzer von SQL-Tabellen zurück (experimentell).
65536	Liest die Parameter `client` und `odbc`-Gruppen aus der `my.cnf`-Datei.
131072	Fügt einige Sicherheitsüberprüfungen hinzu (sollte nicht nötig sein, aber ...).

Wenn Sie viele Optionen haben wollen, sollten Sie die obigen Flags hinzufügen. Zum Beispiel gibt Ihnen die Option 12 (4+8) Debugging und keine Paketbeschränkungen.

Die Standard-`MYODBC.DLL`-Datei wird für optimale Performance kompiliert. Wenn Sie **MyODBC** debuggen wollen (um zum Beispiel "tracing" zu aktivieren), sollten Sie stattdessen `MYODBCD.DLL` verwenden. Um diese Datei zu installieren, kopieren Sie `MYODBCD.DLL` einfach über die installierte `MYODBC.DLL`-Datei.

9.3.4. Wie Sie Probleme mit MyODBC berichten

MyODBC wurde mit Access, Admndemo.exe, C++-Builder, Borland Builder 4, Centura Team Developer (vorher Gupta SQL/Windows), ColdFusion (unter Solaris und NT mit Service Pack 5), Crystal Reports, DataJunction, Delphi, ERwin, Excel, iHTML,

FileMaker Pro, FoxPro, Notes 4.5/4.6, SBSS, Perl DBD-ODBC, Paradox, Powerbuilder, Powerdesigner 32 bit, VC++ und Visual Basic getestet.

Wenn Sie weitere Applikationen kennen, die mit **MyODBC** zusammenarbeiten, sagen Sie uns bitte unter `<myodbc@lists.mysql.com>` Bescheid!

Mit einigen Programmen können Fehler wie diese auftreten: `Another user hat modifies the record that you have modified`. Meistens lösen Sie das folgendermaßen:

- Fügen Sie der Tabelle einen Primärschlüssel hinzu, wenn noch keiner existiert.
- Fügen Sie eine TIMESTAMP-Spalte hinzu, wenn noch keine existiert.
- Verwenden Sie ausschließlich 'Double Float'-Felder. Manche Programme kommen mit 'Single Float'-Feldern nicht klar.

Wenn das nicht helfen sollte, dann erstellen Sie eine `MyODBC` 'Trace'-Datei und versuchen Sie, die Fehlerquelle so zu erschließen.

9.3.5. Programme, die bekanntermaßen mit MyODBC zusammenarbeiten

Die meisten Programme sollten mit **MyODBC** zusammenarbeiten. Für die unten aufgeführten haben wir es selbst getestet oder haben die Bestätigung eines Benutzers, dass es läuft.

- **Programm**

  Anmerkung

- Access

  Um Access zum Laufen zu bringen:

  - Wenn Sie Access 2000 verwenden, sollten Sie die neuesten (Version 2.6 oder höher) Microsoft-MDAC (`Microsoft Data Access Components`) von http://www.microsoft.com/data herunterladen. Dies wird folgenden Bug in Access beheben: Wenn Sie Daten nach MySQL exportieren, werden Tabellen- und Spaltennamen nicht spezifiziert. Ein anderer Weg, diesen Bug zu umgehen, ist, MyODBC auf Version 2.50.33 und MySQL auf Version 3.23.x zu aktualisieren, welche beide zusammen einen Workaround für diesen Bug implementiert haben.

    Ausserdem sollten Sie das Microsoft-Jet-4.0-Service-Pack 5 (SP5) einspielen, welches hier http://support.microsoft.com/support/kb/articles/Q 239/1/14.ASP gefunden werden kann. Dies behebt einige Fälle, in denen Spalten als `#deleted#` in Access markiert sind.

    Beachten Sie, dass Sie, wenn Sie die MySQL-Version 3.22 verwenden, den MDAC-Patch einspielen und MyODBC 2.50.32 oder 2.50.34 und höher benutzen müssen, um dieses Problem zu umgehen.

  - Für alle Access-Versionen sollten Sie die MyODBC-Optionen auf `Return matching rows` setzen. Für Access 2.0 sollten Sie ausserdem `Simulate ODBC 1.0` einschalten.
  - Sie sollten einen Timestamp in alle Tabellen einfügen, die Sie aktualisieren wollen. Für maximale Portablilität werden `TIMESTAMP(14)` oder einfach `TIMESTAMP` anstelle von `TIMESTAMP(X)`-Variationen empfohlen.
  - Sie sollten einen Primärschlüssel in Ihren Tabellen haben. Falls nicht, können neue oder geänderte Zeilen als `#DELETED#` erscheinen.
  - Verwenden sie ausschließlich `DOUBLE`-Float-Felder. Access kann nicht richtig mit "Single Floats" vergleichen. Die Symptome sind, dass entweder neue oder geänderte Zeilen als `#DELETED#` erscheinen oder Sie keine Zeilen finden oder ändern können.
  - Wenn Sie eine Tabelle mit MyODBC verbinden, die eine `BIGINT`-Spalte hat, werden die Ergebnisse als `#DELETED` angezeigt. Sie umgehen das Problem folgendermaßen:
    - Fügen Sie eine weitere `TIMESTAMP`-"Dummy-Spalte" hinzu, am besten `TIMESTAMP(14)`.
    - Wählen Sie `'BIGINT Spalten zu INT wandeln'` im Verbindungsdialog des ODBC-DSN-Administrators.
    - Entfernen Sie die Tabellenverknüpfung aus Access und stellen Sie sie wieder her.

    Die vorherigen Zeilen werden weiterhin als `#DELETED#` angezeigt, aber neue/geänderte Zeilen werden korrekt dargestellt.

  - Wenn Sie weiterhin den Fehler `Ein anderer Benutzer hat Ihre Daten geändert` erhalten, nachdem Sie die `TIMESTAMP`-Spalte hinzugefügt haben, könnte Ihnen der folgende Trick helfen:

Verwenden Sie anstelle von `Datenblattansicht` ein Formular mit den von Ihnen gewünschten Feldern und benutzen Sie dann `Formularblattansicht`. Sie sollten den `StandardWert` für die `TIMESTAMP`-Spalte auf `NOW()` setzen. Zusätzlich ist es sicher nützlich, die `TIMESTAMP`-Spalte zu verstecken, damit Ihre Anwender nicht erschrecken.

- Manchmal erstellt Access ungültige SQL-Anfragen, die MySQL nicht versteht.

  Wählen Sie zur Lösung dieses Problems `"Abfrage|SQL-spezifisch|Pass-Through"` aus dem Access-Menü.

- Wenn Sie statt dessen `MEMO`-Spalten haben wollen, sollten Sie die Spalte mit `ALTER TABLE` in `TEXT` ändern.

- Access kann nicht immer sauber mit `DATE`-Spalten umgehen. Wenn Sie ein solches Problem haben, ändern Sie die entsprechenden Spalten in `DATETIME`.

- Wenn Sie in Access eine Spalte `BYTE` haben, wird Access versuchen, diese in `TINYINT` anstelle von `TINYINT UNSIGNED` zu exportieren. Das führt zu Problemen, wenn Sie Werte in der Spalte haben, die größer als 127 sind!

- ADO

  Wenn Sie mit der ADO-API und **MyODBC** kodieren, müssen Sie auf einige vorgabemäßige Eigenschaften achten, die vom MySQL-Server nicht unterstützt werden. Die Benutzung von `CursorLocationProperty` als `adUseServer` zum Beispiel gibt für `RecordCountProperty` ein Ergebnis von -1 zurück. Um den richtigen Wert zu erhalten, müssen Sie diese Eigenschaft auf `adUseClient` setzen, wie im unten stehenden Visual-Basic-Code gezeigt:

```
Dim myconn As New ADODB.Connection
Dim myrs As New Recordset
Dim mySQL As String
Dim myrows As Long

myconn.Open "DSN=MyODBCsample"
mySQL = "SELECT * from user"
myrs.Source = mySQL
Set myrs.ActiveConnection = myconn
myrs.CursorLocation = adUseClient
myrs.Open
myrows = myrs.RecordCount

myrs.Close
myconn.Close
```

  Ein weiterer Workaround besteht darin, ein `SELECT COUNT(*)`-Statement für eine ähnliche Anfrage zu benutzen, um das korrekte Zählen der Zeilen zu erreichen.

- Active server pages (ASP)

  Sie sollten den Option-Flag `Return matching rows` benutzen.

- BDE-Applikationen

  Damit diese funktionieren, sollten Sie die Option-Flags `Don't optimize column widths` und `Return matching rows` benutzen.

- Borland Builder 4

  Wenn Sie eine Anfrage starten, können Sie die Eigenschaft `Active` oder die Methode `Open` benutzen. Beachten Sie, dass `Active` automatisch mit einer `SELECT * FROM ...`-Anfrage startet, was keine gute Idee ist, wenn Ihre Tabellen Groß sind!

- ColdFusion (unter Unix)

  Die folgenden Informationen sind der ColdFusion-Dokumentation entnommen:

  Lesen Sie folgende Informationen, um den ColdFusion-Server für Linux so zu konfigurieren, dass er den unixODBC-Treiber bei **MyODBC** für MySQL-Datenquellen benutzt. Allaire kann bestätigen, dass die **MyODBC**-Version 2.50.26 mit MySQL-Version 3.22.27 und ColdFusion für Linux funktioniert. (Jede neuere Version sollte ebenfalls funktionieren.) Sie können **MyODBC** von http://www.mysql.com/downloads/api-myodbc.html herunter laden.

  Bei ColdFusion Version 4.5.1 können Sie den ColdFusion Administrator benutzen, um die MySQL-Datenquelle hinzuzufügen. Der Treiber liegt der ColdFusion Version 4.5.1 jedoch nicht bei. Bevor der MySQL-Treiber in der Auswahlliste der ODBC-Datenquellen erscheint, müssen Sie den **MyODBC**-Treiber bauen und nach `/opt/coldfusion/lib/libmyodbc.so` kopieren.

  Das Contrib-Verzeichnis enthält das Programm mydsn-xxx.zip, mit dem Sie die DSN-Registrierungs-Datei für den MyODBC-

Treiber auf Coldfusion-Applikationen bauen können.

- DataJunction

  Sie müssen es ändern, damit es `VARCHAR` statt `ENUM` ausgibt, weil es Letzteres in einer Art ausgibt, die MySQL nicht versteht.

- Excel

  Funktioniert. Einige Tipps:

  - Wenn Sie Probleme mit Datumsangaben haben, versuchen Sie, sie als Zeichenketten mit der `CONCAT()`-Funktion abzurufen. Beispiel:

    ```
    select CONCAT(sonnenaufgang), CONCAT(sonnenuntergang)
        from aufgang_untergang;
    ```

    Werte, die auf diese Art als Zeichenketten abgerufen werden, sollten korrekt als Zeitwerte von Excel97 erkannt werden.

    Der Zweck von `CONCAT()` in diesem Beispiel ist, ODBC auszutricksen, so dass es denkt, dass die Spalte vom Typ "Zeichenkette" sei. Ohne `CONCAT()` weiß ODBC, dass die Spalte vom Typ "Zeit" ist, und Excel versteht das nicht.

    Beachten Sie, dass das ein Bug in Excel ist, weil es eine Zeichenkette automatisch in eine Zeitangabe umwandelt. Das wäre sehr gut, wenn die Quelle eine Textdatei wäre, ist aber einfach nur dumm, wenn die Quelle eine ODBC-Verbindung ist, die exakte Typen für jede Spalte übermittelt.

- Word

  Um Daten von MySQL in Word- / Excel-Dokumente abzurufen, müssen Sie den `MyODBC`-Treiber benutzen und das Add-in Microsoft Query hinzufügen.

  Erzeugen Sie zum Beispiel eine Datenbank mit einer Tabelle, die 2 Text-Spalten enthält:

  - Fügen Sie Zeilen mit dem `mysql`-Kommandozeilenwerkzeug ein.
  - Erzeugen Sie eine DSN-Datei mit dem MyODBC-Treiber, die Sie zum Beispiel my nennen, für die oben genannten Datenbank.
  - Öffnen Sie Word.
  - Erzeugen Sie ein leeres Dokument.
  - Öffnen Sie die Symbolleiste 'Datenbank' und klicken Sie auf die Schaltfläche 'Datenbank einfügen'.
  - Klicken Sie auf die Schaltfläche 'Daten abrufen'.
  - Klicken Sie auf die Schaltfläche 'MS Query'.
  - Erzeugen Sie in MS Query eine neue Datenquelle unter Benutzung der DSN-Datei my.
  - Wählen Sie die neue Anfrage aus.
  - Wählen Sie die Spalten aus, die Sie haben wollen.
  - Legen Sie bei Bedarf einen Filter fest.
  - Legen Sie bei Bedarf eine Sortierung fest.
  - Wählen Sie 'Daten an Word zurückgeben'.
  - Klicken Sie auf 'Beenden'.
  - Klicken Sie auf 'Daten einfügen' und wählen Sie die Datensätze aus.
  - Klicken Sie auf 'OK', und Sie sehen die Zeilen in Ihrem Word-Dokument.

- odbcadmin

  Test-Programm für ODBC.

- Delphi

Sie müssen BDE-Version 3.2 oder neuer benutzen. Setzen Sie die `Don't optimize column width'-Option, wenn Sie sich mit MySQL verbinden.

Hier ist möglicherweise nützlicher Delphi-Code, der sowohl einen ODBC-Eintrag als auch einen BDE-Eintrag für **MyODBC** setzt (der BDE-Eintrag erfordert einen BDE-Alias-Editor, der kostenlos auf einer Delphi Super Page in Ihrer Nähe herunter geladen werden kann (Dank dafür an Bryan Brunton <bryan@flesherfab.com>:

```
fReg:= TRegistry.Create;
  fReg.OpenKey('\Software\ODBC\ODBC.INI\DocumentsFab', True);
  fReg.WriteString('Database', 'Documents');
  fReg.WriteString('Description', ' ');
  fReg.WriteString('Driver', 'C:\WINNT\System32\myodbc.dll');
  fReg.WriteString('Flag', '1');
  fReg.WriteString('Password', '');
  fReg.WriteString('Port', ' ');
  fReg.WriteString('Server', 'xmark');
  fReg.WriteString('User', 'winuser');
  fReg.OpenKey('\Software\ODBC\ODBC.INI\ODBC Data Sources', True);
  fReg.WriteString('DocumentsFab', 'MySQL');
  fReg.CloseKey;
  fReg.Free;

  Memo1.Lines.Add('DATABASE NAME=');
  Memo1.Lines.Add('USER NAME=');
  Memo1.Lines.Add('ODBC DSN=DocumentsFab');
  Memo1.Lines.Add('OPEN MODE=READ/WRITE');
  Memo1.Lines.Add('BATCH COUNT=200');
  Memo1.Lines.Add('LANGTreiber=');
  Memo1.Lines.Add('MAX ROWS=-1');
  Memo1.Lines.Add('SCHEMA CACHE DIR=');
  Memo1.Lines.Add('SCHEMA CACHE SIZE=8');
  Memo1.Lines.Add('SCHEMA CACHE TIME=-1');
  Memo1.Lines.Add('SQLPASSTHRU MODE=SHARED AUTOCOMMIT');
  Memo1.Lines.Add('SQLQRYMODE=');
  Memo1.Lines.Add('ENABLE SCHEMA CACHE=FALSE');
  Memo1.Lines.Add('ENABLE BCD=FALSE');
  Memo1.Lines.Add('ROWSET SIZE=20');
  Memo1.Lines.Add('BLOBS TO CACHE=64');
  Memo1.Lines.Add('BLOB SIZE=32');

  AliasEditor.Add('DocumentsFab','MySQL',Memo1.Lines);
```

- C++-Builder

  Getestet mit BDE-Version 3.0. Das einzige bekannte Problem ist, dass Anfragefelder nicht aktualisiert werden, wenn sich die Tabellenstruktur ändert. BDE scheint jedoch keine Primärschlüssel zu erkennen, sondern nur den Index PRIMARY, obwohl das eigentlich kein Problem darstellt.

- Vision

  Sie sollten den Option-Flag `Return matching rows` benutzen.

- Visual Basic

  Damit Sie eine Tabelle aktualisieren können, müssen Sie für die Tabelle einen Primärschlüssel definieren.

  Visual Basic mit ADO kann keine großen Ganzzahlen handhaben. Das heißt, dass einige Anfragen wie `SHOW PROCESSLIST` nicht korrekt funktionieren. Das läßt sich beheben, indem man die Option `OPTION=16834` in der ODBC-Verbindungs-Zeichenkette hinzufügt oder die `Change BIGINT columns to INT`-Option im MySQL-Verbindungsbildschirm setzt. Eventuell sollten Sie auch die `Return matching rows`-Option setzen.

- VisualInterDev

  Wenn Sie den Fehler `[Microsoft][ODBC Driver Manager] Driver does not support this parameter` erhalten, kann es daran liegen, dass Sie ein `BIGINT` in Ihrem Ergebnis haben. Versuchen Sie, die `Change BIGINT columns to INT`-Option im MySQL-Verbindungsbildschirm zu setzen.

- Visual Objects

  Sie sollten den Option-Flag `Don't optimize column widths` setzen.

9.3.6. Wie man den Wert einer `AUTO_INCREMENT`-Spalte in ODBC erhält

Ein häufiges Problem ist es, den Wert einer automatisch erzeugten Kennung von einem `INSERT` zu erhalten. Bei ODBC können Sie etwas wie folgendes tun (unter der Annahme, dass `auto` ein `AUTO_INCREMENT`-Feld ist):

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
SELECT LAST_INSERT_ID();
```

Oder, wenn Sie die Kennung in eine andere Tabelle einfügen wollen:

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
INSERT INTO foo2 (id,text) VALUES(LAST_INSERT_ID(),'text');
```

See Abschnitt 9.4.6.3, „Wie erhalte ich die eindeutige Kennung für die letzte eingefügte Zeile?“.

Bei einigen ODBC-Applikationen (zumindest Delphi und Access) kann folgende Anfrage benutzt werden, um eine neu eingefügte Zeile zu finden:

```
SELECT * FROM tabelle WHERE auto IS NULL;
```

9.3.7. Probleme mit MyODBC berichten

Wenn Sie Probleme mit **MyODBC** bekommen, sollten Sie als erstes eine Log-Datei durch den ODBC-Manager anlegen lassen (das Log, das Sie erhalten, wenn Sie Logs von ODBCADMIN abfragen) sowie ein **MyODBC**-Log.

Um ein **MyODBC**-Log zu erhalten, tun Sie folgendes:

1. Stellen Sie sicher, dass Sie `myodbcd.dll` und nicht `myodbc.dll` benutzen. Am einfachsten ist es, wenn Sie sich `myodbcd.dll` aus der MyODBC-Distribution holen und es über `myodbc.dll` kopieren, die sich wahrscheinlich in Ihrem `C:\windows\system32`- oder `C:\winnt\system32`-Verzeichnis befindet.

   Denken Sie daran, dass Sie wahrscheinlich die alten myodbc.dll nach dem Testen wiederherstellen wollen, weil Sie um einiges schneller ist als `myodbcd.dll`.

2. Kreuzen Sie `Trace MyODBC' im **MyODBC**-Verbindungs- bzw. Konfigurationsfenster an. Das Log wird in die Datei `C:\myodbc.log` geschrieben.

   Wenn Sie zu diesem Fenster zurückkehren und feststellen, dass die Trace-Option nicht mehr angekreuzt ist, heißt das, dass Sie nicht den `myodbcd.dll`-Treiber benutzen (siehe oben).

3. Starten Sie Ihre Applikation und versuchen Sie, eine Fehlfunktion zu bekommen.

Untersuchen Sie die `MyODBC-Trace-Datei`, um herauszufinden, was möglicherweise schief geht. Sie können die abgesetzten Anfragen finden, indem Sie nach der Zeichenkette `>mysql_real_query` in der `myodbc.log`-Datei suchen.

Sie sollten die Anfragen auch zusätzlich im `mysql`-Monitor oder in `admndemo` laufen lassen, um herauszufinden, ob der Fehler bei MyODBC oder bei MySQL liegt.

Wenn Sie herausgefunden haben, was schief läuft, schicken Sie bitte nur die relevanten Zeilen (maximal 40 Zeilen) an `<myodbc@lists.mysql.com>`. Bitte schicken Sie nie die gesamte MyODBC- oder ODBC-Log-Datei!

Wenn Sie nicht herausfinden können, was schief läuft, besteht die letzte Option darin, eine Archivdatei anzulegen (tar oder zip), die eine MyODBC-Trace-Datei, die ODBC-Log-Datei und eine README-Datei enthält, die das Problem erläutert. Schicken Sie diese an ftp://support.mysql.com/pub/mysql/secret. Nur wir bei MySQL AB haben Zugriff auf die Dateien, die Sie hochspielen, und wir gehen mit den Daten sehr diskret um!

Wenn Sie ein Programm erzeugen können, das dieses Problem ebenfalls zeigt, laden Sie dieses bitte ebenfalls hoch!

Wenn das Programm mit irgend einem anderen SQL-Server funktioniert, sollten Sie eine ODBC-Log-Datei anlegen, in der Sie dasselbe in dem anderen SQL-Server ausführen.

Bedenken Sie, dass wir umso eher das Problem beheben können, desto mehr Informationen Sie uns zur Verfügung stellen!

9.4. MySQL-C-API

Der C-API-Code wird mit MySQL ausgeliefert. Er ist in der `mysqlclient`-Bibliothek enthalten und erlaubt C-Programmen, auf eine Datenbank zuzugreifen.

Viele Clients in der MySQL-Quelldistribution sind in C geschrieben. Wenn Sie nach Beispielen für den Gebrauch der C-API suchen, schauen Sie sich diese Clients an. Sie finden Sie im `clients`-Verzeichnis in der MySQL-Quelldistribution.

Viele andere Client-APIs (alle ausser Java) benutzen die `mysqlclient`-Bibliothek, um mit dem MySQL-Server zu

kommunizieren. Das heißt zum Beispiel, dass Sie viele derselben Umgebungsvariablen nutzen können, die von anderen Client-Programmen benutzt werden, weil sie von der Bibliothek referenziert werden. Eine Liste dieser Variablen findet sich unter Abschnitt 5.8, „Clientseitige Skripte und Hilfsprogramme von MySQL“.

Der Client hat eine maximale Kommunikationspuffergröße. Die anfänglich zugewiesene Puffergröße (16 KB) wird automatisch bis zur maximale Größe (16 MB) vergrößert. Weil Puffergrößen nur bei Bedarf vergrößert werden, bedeutet die einfache Erhöhung der maximalen Größe nicht per se, dass mehr Ressourcen benutzt werden. Die Überprüfung der Größe ist hauptsächlich eine Prüfung auf irrtümliche Anfragen und Kommunikationspakete.

Der Kommunikationspuffer muss Groß genug sein, um ein einzelnes SQL-Statement aufzunehmen (für den Client-Server-Verkehr) und eine Zeile zurückgegebener Daten (für den Server-Client-Verkehr). Der Kommunikationspuffer jedes Threads wird dynamisch vergrößert, um jede Anfrage oder Zeile bis zur maximalen Größe zu handhaben. Wenn Sie beispielsweise `BLOB`-Werte haben, die bis zu 16 MB an Daten beinhalten, müssen Sie eine Kommunikationspuffergrenze von zumindest 16 MB haben (sowohl beim Server als auch beim Client). Die vorgabemäßige maximale Größe beim Client liegt bei 16 MB, aber die vorgabemäßige maximale Grenze beim Server liegt bei 1 MB. Das können Sie vergrößern, indem Sie den Wert des `max_allowed_packet`-Parameters setzen, wenn der Server gestartet wird. See Abschnitt 6.5.2, „Serverparameter tunen“.

Der MySQL-Server verringert den Kommunikationspuffer auf `net_buffer_length` Bytes nach jeder Anfrage. Bei Clients wird die Größe des zugewiesenen Puffers bei einer Verbindung nicht herabgesetzt, bis die Verbindung geschlossen wird. Dann wird der Client-Speicher wieder freigesetzt.

Zum Programmieren mit Threads siehe Abschnitt 9.4.8, „Wie man einen threaded Client herstellt“. Um eine Standalone-Applikation herzustellen, die "Server" und "Client" im selben Programm beinhaltet (und nicht mit einem externen MySQL-Server kommuniziert), siehe Abschnitt 9.4.9, „libmysqld, die eingebettete MySQL-Server-Bibliothek“.

9.4.1. C-API-Datentypen

- `MYSQL`

  This structure represents a handle to one Datenbank connection. It is used für almost all MySQL Funktionen.

- `MYSQL_RES`

  Diese Struktur repräsentiert das Ergebnis einer Anfrage, die Zeilen zurückgibt (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`). Die von der Anfrage zurückgegebene Informationen wird im Weiteren *result set* (Ergebnismenge) genannt.

- `MYSQL_ROW`

  Das ist eine Typ-sichere Repräsentation einer Zeile von Daten. Momentan ist sie als ein Array gezählter Byte-Zeichenketten implementiert. (Sie können diese nicht als NULL-begrenzte Zeichenketten behandeln, falls Feldwert binäre Daten enthalten können, welche solche Werte intern NULL-Bytes enthalten können.) Zeilen werden mit dem Aufruf von `mysql_fetch_row()` abgeholt.

- `MYSQL_FIELD`

  Diese Struktur enthält Informationen über ein Feld, wie Feldname, Feldtyp und Feldgröße. Seine Elemente werden weiter unten genauer beschrieben. Sie erhalten die `MYSQL_FIELD`-Strukturen für jedes Feld durch den wiederholten Aufruf von `mysql_fetch_field()`. Feldwerte sind nicht Teil dieser Struktur, sondern in der `MYSQL_ROW`-Struktur enthalten.

- `MYSQL_FIELD_OFFSET`

  Das ist eine Typ-sichere Repräsentation eines Offsets in einer MySQL-Feldliste (benutzt von `mysql_field_seek()`.) Offsets sind Feldnummern innerhalb einer Zeile, beginnend mit 0.

- `my_ulonglong`

  Der Typ, der für die Anzahl von Zeilen und für `mysql_affected_rows()`, `mysql_num_rows()`, und `mysql_insert_id()` benutzt wird. Dieser Typ stellt einen Bereich von `0` bis `1.84e19` zur Verfügung.

  Auf manchen Systemen funktioniert der Versuch, einen Wert des Typs `my_ulonglong` auszugeben, nicht. Um einen solchen Wert auszugeben, wandeln Sie ihn in `unsigned long` um und benutzen Sie ein `%lu`-Ausgabeformat. Beispiel:

```
printf (Anzahl von Zeilen: %lu\n", (unsigned long) mysql_num_rows(result));
```

Die `MYSQL_FIELD`-Struktur enthält die unten aufgeführten Elemente:

- `char * name`

Der Name des Feldes, als NULL-begrenzte Zeichenkette.

- `char * table`

  Der Name der Tabelle, die dieses Feld enthält, falls es kein berechnetes Feld ist. Bei berechneten Feldern ist der `table`-Wert eine leere Zeichenkette.

- `char * def`

  Der Vorgabewert dieses Felds als eine NULL-begrenzte Zeichenkette. Dieser wird nur gesetzt, wenn Sie `mysql_list_fields()` benutzen.

- `enum enum_field_types-Typ`

  Der Typ des Felds. Der `type`-Wert kann einer der folgenden sein:

Typwert	Typbedeutung
`FIELD_TYPE_TINY`	`TINYINT`-Feld
`FIELD_TYPE_SHORT`	`SMALLINT`-Feld
`FIELD_TYPE_LONG`	`INTEGER`-Feld
`FIELD_TYPE_INT24`	`MEDIUMINT`-Feld
`FIELD_TYPE_LONGLONG`	`BIGINT`-Feld
`FIELD_TYPE_DECIMAL`	`DECIMAL`- oder `NUMERIC`-Feld
`FIELD_TYPE_FLOAT`	`FLOAT`-Feld
`FIELD_TYPE_DOUBLE`	`DOUBLE`- oder `REAL`-Feld
`FIELD_TYPE_TIMESTAMP`	`TIMESTAMP`-Feld
`FIELD_TYPE_DATE`	`DATE`-Feld
`FIELD_TYPE_TIME`	`TIME`-Feld
`FIELD_TYPE_DATETIME`	`DATETIME`-Feld
`FIELD_TYPE_YEAR`	`YEAR`-Feld
`FIELD_TYPE_STRING`	`CHAR`- oder `VARCHAR`-Feld
`FIELD_TYPE_BLOB`	`BLOB`- oder `TEXT`-Feld (benutzen Sie `max_length`, um die maximale Länge festzulegen)
`FIELD_TYPE_SET`	`SET`-Feld
`FIELD_TYPE_ENUM`	`ENUM`-Feld
`FIELD_TYPE_NULL`	`NULL`-Feld
`FIELD_TYPE_CHAR`	Veraltet; benutzen Sie statt dessen `FIELD_TYPE_TINY`

  Sie können das `IS_NUM()`-Makro benutzen, um zu testen, ob ein Feld einen numerischen Typ besitzt oder nicht. Übergeben Sie den `type`-Wert an `IS_NUM()`, und Sie erhalten WAHR (true), wenn das Feld numerisch ist:

```
if (IS_NUM(field->type))
    printf("Feld ist numerisch\n");
```

- `unsigned int length`

  Die Breite des Felds, wie in der Tabellendefinition festgelegt.

- `unsigned int max_length`

  Die maximale Breite des Felds für die Ergebnismenge (die Länge des längsten Feldwerts für die Zeilen, die tatsächlich in der Ergebnismenge enthalten sind). Wenn Sie `mysql_store_result()` oder `mysql_list_fields()` benutzen, enthält die Variable die maximale Länge für das Feld. Wenn Sie `mysql_use_result()` benutzen, ist sie 0.

- `unsigned int flags`

  Unterschiedliche Bit-Flags für das Feld. Der `flags`-Wert kann 0 oder mehr der folgenden Bits gesetzt haben:

Flag-Wert	Flag-Bedeutung

NOT_NULL_FLAG	Feld darf nicht NULL sein
PRI_KEY_FLAG	Feld ist Teil eines Primärschlüssels
UNIQUE_KEY_FLAG	Feld ist Teil eines eindeutigen Schlüssels
MULTIPLE_KEY_FLAG	Feld ist Teil eines nicht eindeutigen Schlüssels
UNSIGNED_FLAG	Feld hat das UNSIGNED-Attribute
ZEROFILL_FLAG	Feld hat das ZEROFILL-Attribute
BINARY_FLAG	Feld hat das BINARY-Attribute
AUTO_INCREMENT_FLAG	Feld hat das AUTO_INCREMENT-Attribut
ENUM_FLAG	Feld ist ein ENUM (veraltet)
BLOB_FLAG	Feld ist ein BLOB oder TEXT (veraltet)
TIMESTAMP_FLAG	Feld ist ein TIMESTAMP (veraltet)

Die Benutzung der BLOB_FLAG-, ENUM_FLAG- und TIMESTAMP_FLAG-Flags ist veraltet, weil sie den Feldtyp statt eines Attributs seines Typs angeben. Statt dessen sollten Sie field->type gegen FIELD_TYPE_BLOB, FIELD_TYPE_ENUM oder FIELD_TYPE_TIMESTAMP testen.

Das unten stehende Beispiel zeigt eine typische Benutzung des flags-Werts:

```
if (field->flags & NOT_NULL_FLAG)
    printf("Feld darf nicht NULL sein\n");
```

Sie können aus Bequemlichkeitsgründen folgende Makros benutzen, um den Bool'schen Status des flags-Werts zu bestimmen:

IS_NOT_NULL(flags)	WAHR, wenn der Feldwert als NOT NULL definiert ist
IS_PRI_KEY(flags)	WAHR, wenn der Feldwert ein Primärschlüssel ist
IS_BLOB(flags)	WAHR, wenn der Feldwert ein BLOB oder TEXT ist (veraltet; testen Sie statt dessen field->type)

- unsigned int decimals

  Die Anzahl von Dezimalstellen für numerische Felder.

9.4.2. C-API-Funktionsüberblick

Die in der C-API verfügbaren Funktionen sind unten aufgeführt und im nächsten Abschnitt detaillierter beschrieben. See Abschnitt 9.4.3, „C-API-Funktionsbeschreibungen“.

mysql_affected_rows()	Gibt die Anzahl von Zeilen zurück, die durch die letzte UPDATE-, DELETE- oder INSERT-Anfrage geändert, gelöscht bzw. hinzugefügt wurden.
mysql_close()	Schließt eine Server-Verbindung.
mysql_connect()	Stellt die Verbindung mit einem MySQL-Server her. Diese Funktion ist veraltet, benutzen Sie statt dessen mysql_real_connect().
mysql_change_user()	Ändert Benutzer und Datenbank bei einer geöffneten Verbindung.
mysql_character_set_name()	Gibt den Namen des vorgabemäßigen Zeichensatzes für die Verbindung zurück.
mysql_create_db()	Erzeugt eine Datenbank. Diese Funktion ist veraltet, benutzen Sie statt dessen den SQL-Befehl CREATE DATABASE.
mysql_data_seek()	Sucht bis zu einer beliebigen Zeile in einer Anfrage-Ergebnismenge.
mysql_debug()	Macht ein DBUG_PUSH mit der angegebenen Zeichenkette.
mysql_drop_db()	Löscht eine Datenbank. Diese Funktion ist veraltet, benutzen Sie statt dessen den SQL-Befehl DROP DATABASE.
mysql_dump_debug_info()	Veranlasst den Server, Debug-Informationen in die Log-Datei zu schreiben.
mysql_eof()	Stellt fest, ob die letzte Zeile der Ergebnismenge gelesen wurde oder nicht. Diese Funktion ist veraltet, benutzen Sie statt dessen mysql_errno() oder mysql_error().

mysql_errno()	Gibt die Fehlernummer der zuletzt aufgerufenen MySQL-Funktion zurück.
mysql_error()	Gibt die Fehlermeldung der zuletzt aufgerufenen MySQL-Funktion zurück.
mysql_real_escape_string()	Escapet Sonderzeichen in einer Zeichenkette, die für ein SQL-Statement benutzt wird, wobei der aktuelle Zeichensatz der Verbindung berücksichtigt wird.
mysql_escape_string()	Escapet Sonderzeichen in einer Zeichenkette, die für ein SQL-Statement benutzt wird.
mysql_fetch_field()	Gibt den Typ des nächsten Tabellenfelds zurück.
mysql_fetch_field_direct()	Gibt den Typ eines Tabellenfelds zurück, angegeben durch eine Feldnummer.
mysql_fetch_fields()	Gibt ein Array aller Feldstrukturen zurück.
mysql_fetch_lengths()	Gibt die Länge aller Spalten in der aktuellen Zeile zurück.
mysql_fetch_row()	Holt die nächste Zeile aus der Ergebnismenge.
mysql_field_seek()	Setzt den Spaltencursor auf eine bestimmte Spalte.
mysql_field_count()	Gibt die Anzahl der Ergebnisspalten für die letzte Anfrage zurück.
mysql_field_tell()	Gibt die Position des Feldcursors zurück, der für das letzte `mysql_fetch_field()` benutzt wurde.
mysql_free_result()	Gibt Speicher frei, der von einer Ergebnismenge benutzt wird.
mysql_get_client_info()	Gibt Client-Versionsinformationen zurück.
mysql_get_host_info()	Gibt eine Zeichenkette zurück, die die Verbindung beschreibt.
mysql_get_proto_info()	Gibt die Protokollversion zruück, die von der Verbindung benutzt wird.
mysql_get_server_info()	Gibt die Server-Versionsnummer zurück.
mysql_info()	Gibt Informationen über die zuletzt ausgeführte Anfrage zurück.
mysql_init()	Holt oder initialisiert eine `MYSQL`-Struktur.
mysql_insert_id()	Gibt die Kennung zurück, die für eine `AUTO_INCREMENT`-Spalte durch die letzte Anfrage erzeugt wurde.
mysql_kill()	Tötet einen angegebenen Thread.
mysql_list_dbs()	Gibt die Datenbanknamen zurück, die mit einem einfachen regulären Ausdruck übereinstimmen.
mysql_list_fields()	Gibt die Feldnamen zurück, die mit einem einfachen regulären Ausdruck übereinstimmen.
mysql_list_processes()	Gibt eine Liste der aktuellen Server-Threads zurück.
mysql_list_tables()	Gibt Tabellenamen zurück, die mit einem einfachen regulären Ausdruck übereinstimmen.
mysql_num_fields()	Gibt die Anzahl von Spalten in einer Ergebnismenge zurück.
mysql_num_rows()	Gibt die Anzahl von Zeilen in einer Ergebnismenge zurück.
mysql_options()	Setzt Verbindungsoptionen für `mysql_connect()`.
mysql_ping()	Prüft, ob die Verbindung zum Server funktioniert oder nicht und verbindet sich erneut, falls notwendig.
mysql_Anfrage()	Führt eine SQL-Anfrage aus, die als NULL-begrenzte Zeichenkette angegeben wird.
mysql_real_connect()	Verbindet sich mit einem MySQL-Server.
mysql_real_query()	Führt eine SQL-Anfrage aus, die als gezählte Zeichenkette angegeben wird.
mysql_reload()	Weist den Server an, die Berechtigungstabellen erneut zu laden.
mysql_row_seek()	Sucht bis zu einer Zeile in einer Ergebnismenge, indem sie den Wert benutzt, der von `mysql_row_tell()` zurückgegeben wird.
mysql_row_tell()	Gibt die Zeilencursorposition zurück.
mysql_select_db()	Wählt eine Datenbank aus.
mysql_shutdown()	Fährt den Datenbankserver herunter.
mysql_stat()	Gibt den Serverstatus als Zeichenkette zurück.
mysql_store_result()	Ruft eine komplette Ergebnismenge zum Client ab.
mysql_thread_id()	Gibt die aktuelle Thread-Kennung zurück.
mysql_thread_safe()	Gibt 1 zurück, wenn die Clients Thread-sicher kompiliert sind.
mysql_use_result()	Initialisiert den zeilenweisen Abruf einer Ergebnismenge.

Um sich mit dem Server zu verbinden, rufen Sie `mysql_init()` auf, um einen Verbindungs-Handler zu initialisieren. Rufen Sie

dann `mysql_real_connect()` mit diesem Handler auf (mit Informationen wie Hostname, Benutzername und Passwort). Beim Verbinden setzt `mysql_real_connect()` den `reconnect`-Flag (Teil der MYSQL-Struktur) auf einen Wert von `1`. Dieser Flag legt bei einer Anfrage, die wegen einer verloren gegangenen Serververbindung nicht ausgeführt werden kann, fest, dass ein erneutes Verbinden versucht wird, bevor aufgegeben wird. Wenn Sie mit der Verbindung fertig sind, rufen Sie `mysql_close()` auf, um sie zu beenden.

Während eine Verbindung aktiv ist, kann der Client SQL-Anfragen an den Server schicken, indem er `mysql_query()` oder `mysql_real_query()` benutzt. Der Unterschied zwischen beiden ist, dass `mysql_query()` erwartet, dass die Anfrage als NULL-separierte Zeichenkette angegeben wird, während `mysql_real_query()` eine gezählte Zeichenkette erwartet. Wenn die Zeichenkette Binärdaten enthält (was NULL-Bytes beinhalten kann), müssen Sie `mysql_real_query()` benutzen.

Bei jeder Nicht-`SELECT`-Anfrage (wie `INSERT`, `UPDATE`, `DELETE`) finden Sie heraus, wie viele Zeilen geändert (betroffen) wurden, indem Sie `mysql_affected_rows()` aufrufen.

Bei `SELECT`-Anfragen rufen Sie die ausgewählten Zeilen als Ergebnismenge ab. (Beachten Sie, dass einige Statements ähnlich wie `SELECT` sind, weil auch sie Zeilen zurückgeben. Das sind `SHOW`, `DESCRIBE` und `EXPLAIN`. Sie werden auf dieselbe Weise behandelt wie `SELECT`-Statements.)

Es gibt für einen Client zwei Möglichkeiten, Ergebnismengen zu verarbeiten. Eine Möglichkeit besteht darin, die gesamte Ergebnismenge auf einmal abzurufen, indem `mysql_store_result()` aufgerufen wird. Diese Funktion holt alle Zeilen vom Server ab, die von der Anfrage zurückgegeben werden, und speichert sie im Client. Die zweite Möglichkeit besteht darin, dass der Client die Ergebnismenge zeilenweise abruft, indem er `mysql_use_result()` aufruft. Diese Funktion initialisiert den Abruf, holt aber keinerlei Zeilen vom Server ab.

In beiden Fällen können Sie auf Zeilen zugreifen, indem Sie `mysql_fetch_row()` aufrufen. Bei `mysql_store_result()` greift `mysql_fetch_row()` auf Zeilen zurück, die bereits vom Server geholt wurden. Bei `mysql_use_result()` ruft `mysql_fetch_row()` die Zeilen direkt vom Server ab. Informationen über die Größe der Daten in jeder Zeile sind durch Aufruf von `mysql_fetch_lengths()` verfügbar.

Wenn Sie mit einer Ergebnismenge fertig sind, rufen Sie `mysql_free_result()` auf, um den hierfür benutzten Speicher freizugeben.

Die beiden Abrufmechanismen sind komplementär. Client-Programme sollten entscheiden, welcher Ansatz der für ihre Erfordernisse geeignetste ist. In der Praxis wird für Clients häufiger `mysql_store_result()` verwendet.

Ein Vorteil von `mysql_store_result()` ist, dass bereits alle Zeilen zum Client geholt wurden. Deshalb können Sie nicht nur sequentiell auf Zeilen zugreifen, sondern sich in der Ergebnismenge vorwärts und rückwärts bewegen, indem Sie `mysql_data_seek()` oder `mysql_row_seek()` benutzen, um die aktuelle Position innerhalb der Ergebnismenge zu ändern. Sie können auch herausfinden, wie viele Zeilen es gibt, indem Sie `mysql_num_rows()` aufrufen. Auf der anderen Seite kann der Speicherbedarf für `mysql_store_result()` sehr hoch sein, wenn Sie große Ergebnismengen abrufen, so dass Speichermangel eintreten kann.

Ein Vorteil von `mysql_use_result()` ist, dass der Client weniger Arbeitsspeicher für die Ergebnismenge benötigt, weil er nur eine Zeile zugleich erhält (und weil weniger Zuweisungs-Overhead da ist, kann `mysql_use_result()` schneller sein). Die Nachteile liegen darin, dass Sie jede Zeile schnell verarbeiten müssen, um zu vermeiden, den Server zu blockieren. Ausserdem haben Sie keinen wahlfreien (random) Zugriff auf die Zeilen innerhalb einer Ergebnismenge (Sie können auf die Zeilen nur sequentiell zugreifen), und Sie wissen nicht, wie viele Zeilen sich in der Ergebnismenge befinden, bis Sie sie alle abgerufen haben. Darüber hinaus *müssen* Sie alle Zeilen abrufen, selbst wenn Sie während des Abrufs feststellen, dass Sie die Information gefunden haben, nach der Sie suchen.

Die API ermöglicht Clients, auf die Anfragen entsprechend zu antworten (Zeilen nur wenn nötig abzurufen), ohne zu wissen, ob die Anfragen ein `SELECT` ist oder nicht. Das erreichen Sie durch Aufruf von `mysql_store_result()` nach jedem `mysql_query()` (oder `mysql_real_query()`). Wenn der Ergebnismengenaufruf erfolgreich ist, war die Anfrage ein `SELECT` und Sie können die Zeilen lesen. Wenn der Ergebnismengenaufruf fehlschlägt, rufen Sie `mysql_field_count()` auf, um festzustellen, ob ein Ergebnis erwartet wurde oder nicht. Wenn `mysql_field_count()` 0 zurückgibt, gab die Anfrage keine Daten zurück (was anzeigt, dass sie kein `INSERT`, `UPDATE`, `DELETE` usw. war), und es wurde nicht erwartet, dass sie Zeilen zurückgibt. Wenn `mysql_field_count()` ungleich 0 ist, sollte die Anfrage Zeilen zurückgegeben haben, tat das aber nicht. Das zeigt an, dass die Anfrage ein `SELECT` war, das fehlschlug. Sehen Sie in der Beschreibung von `mysql_field_count()` wegen eines Beispiels nach, wie das gemacht wird.

Sowohl `mysql_store_result()` als auch `mysql_use_result()` gestatten Ihnen, Informationen über die Felder zu erlangen, aus denen die Ergebnismenge besteht (die Anzahl der Felder, ihre Namen, Typen usw.). Sie können sequentiell auf Feldinformationen innerhalb der Zeile zugreifen, indem Sie `mysql_fetch_field()` wiederholt aufrufen, oder direkt auf die Feldnummer innerhalb einer Zeile durch Aufruf von `mysql_fetch_field_direct()`. Die aktuelle Feldcursorposition kann durch den Aufruf von `mysql_field_seek()` geändert werden. Wenn Sie den Feldcursor setzen, betrifft das nachfolgende Aufrufe von `mysql_fetch_field()`. Sie erhalten alle Feldinformationen auf einmal, wenn Sie `mysql_fetch_fields()` aufrufen.

Um Fehler zu erkennen und zu berichten, stellt MySQL den Zugriff auf Fehlerinformationen durch die `mysql_errno()`- und `mysql_error()`-Funktionen zur Verfügung. Diese geben den Fehlercode oder die Fehlermeldung für die zuletzt aufgerufenen Funktionen zur Verfügung, die erfolgreich sein oder fehlschlagen können, so dass Sie feststellen können, wann ein Fehler auftrat

und welcher es war.

9.4.3. C-API-Funktionsbeschreibungen

In den unten stehenden Beschreibungen bedeutet ein Parameter oder Rückgabewert von `NULL` `NULL` im Sinne der C-Programmier-Sprache, nicht einen MySQL-`NULL`-Wert.

Funktionen, die einen Wert zurückgeben, geben allgemein einen Zeiger oder eine Ganzzahl zurück. Falls nicht anders angegeben geben Funktionen, die einen Zeiger zurückgeben, einen Nicht-`NULL`-Wert zurück, um Erfolg anzuzeigen, oder einen `NULL`-Wert, um einen Fehler anzuzeigen. Funktionen, die eine Ganzzahl zurückgeben, geben 0 zurück, um Erfolg anzuzeigen, und Nicht-0, um einen Fehler anzuzeigen. Beachten Sie, dass ``Nicht-0" genau das bedeutet. Wenn die Funktionsbeschreibung nichts anderes aussagt, testen Sie nicht gegen einen anderen Wert als 0:

```
if (ergebnis)                   /* korrekt */
    ... FEHLER ...

if (ergebnis < 0)               /* nicht korrekt */
    ... FEHLER ...

if (ergebnis == -1)             /* nicht korrekt */
    ... FEHLER ...
```

Wenn eine Funktion einen Fehler zurückgibt, listet der Unterabschnitt **Errors** der Funktionsbeschreibung die möglichen Fehlertypen auf. Sie finden heraus, welcher davon auftrat, indem Sie `mysql_errno()` aufrufen. Eine Zeichenketten-Darstellung des Fehler kann durch Aufruf von `mysql_error()` erlangt werden.

9.4.3.1. `mysql_affected_rows()`

```
my_ulonglong mysql_affected_rows(MYSQL *mysql)
```

Beschreibung

Gibt die Anzahl von Zeilen zurück, die durch das letzte `UPDATE` geändert, durch das letzte `DELETE` gelöscht oder durch das letzte `INSERT` eingefügt wurden. Kann direkt nach `mysql_query()` aufgerufen werden, bei `UPDATE`-, `DELETE`- oder `INSERT`-Statements. Bei `SELECT`-Statements funktioniert `mysql_affected_rows()` wie `mysql_num_rows()`.

Rückgabewerte

Eine Ganzzahl größer als 0 gibt die Anzahl von Zeilen an, die betroffen oder abgerufen wurden. 0 gibt an, dass keine Datensätze bei einem `UPDATE`-Statement geändert wurden, keine Zeilen der `WHERE`-Klausel in der Anfrage entsprachen oder dass bislang keine Anfrage ausgeführt wurde. -1 gibt an, dass die Anfrage einen Fehler zurückgab oder dass - bei einer `SELECT`-Anfrage - `mysql_affected_rows()` vor `mysql_store_result()` aufgerufen wurde.

Fehler

Keine.

Beispiel

```
mysql_query(&mysql,"UPDATE produkte SET kosten=kosten*1.25 WHERE gruppe=10");
printf("%ld produkte updated",(long) mysql_affected_rows(&mysql));
```

Wenn man den Flag `CLIENT_FOUND_ROWS` angibt, wenn man sich mit `mysqld` verbindet, gibt `mysql_affected_rows()` die Anzahl von Zeilen zurück, die mit dem `WHERE`-Statement bei `UPDATE`-Statements übereinstimmten.

Beachten Sie bei der Benutzung des `REPLACE`-Befehls, dass `mysql_affected_rows()` 2 zurückgibt, wenn die neue Zeile eine alte Zeile ersetzte. Das liegt daran, dass in diesem Fall eine neue Zeile eingefügt und dann das alte Duplikat gelöscht wurde.

9.4.3.2. `mysql_close()`

```
void mysql_close(MYSQL *mysql)
```

Beschreibung

Schließt eine vorher geöffnete Verbindung. `mysql_close()` gibt auch den Verbindungs-Handle frei, der von `mysql` zugewiesen wurde, wenn der Handle automatisch mit `mysql_init()` oder `mysql_connect()` zugewiesen wurde.

Rückgabewerte

Keine.

Fehler

Keine.

9.4.3.3. mysql_connect()

```
MYSQL *mysql_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd)
```

Beschreibung

Diese Funktion ist veraltet. Sie sollten statt dessen `mysql_real_connect()` benutzen.

`mysql_connect()` versucht, eine Verbindung zu einer MySQL-Datenbankmaschine aufzubauen, die auf `host` läuft. `mysql_connect()` muss erfolgreich beendet werden, bevor Sie irgend welche weiteren API-Funktionen aufrufen können, mit Ausnahme von `mysql_get_client_info()`.

Die Bedeutung der Parameter ist dieselbe wie die entsprechenden Parameter bei `mysql_real_connect()`, mit dem Unterschied, dass die Verbindungsparameter `NULL` sein dürfen. In diesem Fall weist die C-API automatisch Speicher für die Verbindungsstruktur zu und gibt diesen frei, wenn Sie `mysql_close()` aufrufen. Der Nachteil dieses Ansatzes besteht darin, dass Sie keine Fehlermeldung abrufen können, wenn die Verbindung fehlschlägt. (Um Fehlerinformationen von `mysql_errno()` oder `mysql_error()` abrufen zu können, müssen Sie einen gültigen `MYSQL`-Zeiger angeben.)

Rückgabewerte

Dieselben wie für `mysql_real_connect()`.

Fehler

Dieselben wie für `mysql_real_connect()`.

9.4.3.4. mysql_change_user()

```
my_bool mysql_change_user(MYSQL *mysql, const char *user, const char *password, const char *db)
```

Beschreibung

Ändert den Benutzer und veranlasst, dass die Datenbank, die mit `db` angegeben wurde, die vorgabemäßige (aktuelle) Datenbank für die Verbindung wird, die durch `mysql` festgelegt wurde. In nachfolgenden Anfragen ist diese Datenbank die Vorgabe für Tabellenverweise, bei denen nicht explizit eine Datenbank angegeben wird.

Diese Funktion wurde in MySQL-Version 3.23.3 eingeführt.

`mysql_change_user()` schlägt fehl, wenn sich der Benutzer nicht authentifizieren kann oder wenn er keine Zugriffsrechte auf die Datenbank hat. In diesem Fall werden Benutzer und Datenbank nicht geändert.

Der `db`-Parameter kann auf `NULL` gesetzt werden, wenn Sie keine vorgabemäßige Datenbank haben wollen.

Rückgabewerte

0 für Erfolg. Nicht-0, wenn ein Fehler auftrat.

Fehler

Dieselben, die Sie von `mysql_real_connect()` erhalten.

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurde in nicht korrekter Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler ist aufgetreten.

- ER_UNKNOWN_COM_ERROR

  Der MySQL-Server hat diesen Befehl nicht implementiert (wahrscheinlich ein alter Server).

- ER_ACCESS_DENIED_ERROR

  Benutzername oder Passwort sind falsch.

- ER_BAD_DB_ERROR

  Die Datenbank existiert nicht.

- ER_DBACCESS_DENIED_ERROR

  Der Benutzer hat keine Zugriffsrechte auf die Datenbank.

- ER_WRONG_DB_NAME

  Der Datenbankname war zu lang.

Beispiel

```
if (mysql_change_user(&mysql, "benutzer", "passwort", "neue_datenbank"))
{
   fprintf(stderr, "Änderung des Benutzers fehlgeschlagen. Fehler: %s\n",
           mysql_error(&mysql));
}
```

9.4.3.5. mysql_character_set_name()

```
const char *mysql_character_set_name(MYSQL *mysql)
```

Beschreibung

Gibt den vorgabemäßigen Zeichensatz für die aktuelle Verbindung zurück.

Rückgabewerte

Der vorgabemäßige Zeichensatz

Fehler

Keine.

9.4.3.6. mysql_create_db()

```
int mysql_create_db(MYSQL *mysql, const char *db)
```

Beschreibung

Erzeugt die Datenbank, die durch den db-Parameter angegeben wird.

Diese Funktion ist veraltet. Vorzugsweise sollten Sie mysql_query() benutzen, um statt dessen ein SQL-CREATE DATABASE-Statement abzusetzen.

Rückgabewerte

0, wenn die Datenbank erfolgreich erzeugt wurde. Nicht-0, wenn ein Fehler auftrat.

Fehler

- CR_COMMANDS_OUT_OF_SYNC

  Befehle wurden in nicht korrekter Reihenfolge ausgeführt.

- CR_SERVER_GONE_ERROR

  Der MySQL-Server ist weg.

- CR_SERVER_LOST

Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

Beispiel

```
if(mysql_create_db(&mysql, "meine_datenbank"))
{
   fprintf(stderr, "Erzeugung der neuen Datenbank fehlgeschlagen. Fehler: %s\n",
           mysql_error(&mysql));
}
```

9.4.3.7. `mysql_data_seek()`

```
void mysql_data_seek(MYSQL_RES *result, unsigned long long offset)
```

Beschreibung

Sucht bis zu einer beliebigen Zeile in einer Anfrageergebnismenge. Das setzt voraus, dass die Ergebnismengenstruktur das gesamte Anfrageergebnis enthält. Daher kann `mysql_data_seek()` nur in Verbindung mit `mysql_store_result()` benutzt werden, nicht in Verbindung mit `mysql_use_result()`.

Der Offset sollte ein Wert im Bereich von 0 bis `mysql_num_rows(ergebnis)-1` sein.

Rückgabewerte

Keine.

Fehler

Keine.

9.4.3.8. `mysql_debug()`

```
void mysql_debug(char *debug)
```

Beschreibung

Führt ein `DBUG_PUSH` mit der angegebenen Zeichenkette durch. `mysql_debug()` benutzt die Debug-Bibliothek von Fred Fish. Um diese Funktion benutzen zu können, müssen Sie die Client-Bibliothek so kompilieren, dass sie Debuggen unterstützt. See Abschnitt E.1, „Einen MySQL-Server debuggen“. See Abschnitt E.2, „Einen MySQL-Client debuggen“.

Rückgabewerte

Keine.

Fehler

Keine.

Beispiel

Der unten stehende Aufruf führt dazu, dass die Client-Bibliothek eine Trace-Datei in `/tmp/client.trace` auf der Client-Maschine erzeugt:

```
mysql_debug("d:t:O,/tmp/client.trace");
```

9.4.3.9. `mysql_drop_db()`

```
int mysql_drop_db(MYSQL *mysql, const char *db)
```

Beschreibung

Löscht die Datenbank, die durch den `db`-Parameter angegeben wird.

Diese Funktion ist veraltet. Benutzen Sie vorzugsweise `mysql_query()`, um statt dessen ein SQL-`DROP DATABASE`-Statement abzusetzen.

Rückgabewerte

0, wenn die Datenbank erfolgreich gelöscht wurde. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

Beispiel

```
if(mysql_drop_db(&mysql, "meine_datenbank"))
  fprintf(stderr, "Löschen der Datenbank fehlgeschlagen: Fehler: %s\n",
          mysql_error(&mysql));
```

9.4.3.10. `mysql_dump_debug_info()`

```
int mysql_dump_debug_info(MYSQL *mysql)
```

Beschreibung

Weist den Server an, Debug-Informationen ins Log zu schreiben. Damit das funktioniert, muss der verbundene Benutzer die **process**-Berechtigung haben.

Rückgabewerte

0, wenn der Befehl erfolgreich war. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.11. `mysql_eof()`

```
my_bool mysql_eof(MYSQL_RES *result)
```

Beschreibung

Diese Funktion ist veraltet. Benutzen Sie statt dessen `mysql_errno()` oder `mysql_error()`.

`mysql_eof()` stellt fest, ob die letzte Zeile einer Ergebnismenge gelesen wurde oder nicht.

Wenn Sie eine Ergebnismenge durch einen erfolgreichen Aufruf von `mysql_store_result()` erhalten, erhält der Client den gesamten Satz in einer Operation. In diesem Fall bedeutet eine `NULL`-Rückgabe von `mysql_fetch_row()` immer, dass das Ende der Ergebnismenge erreicht wurde und es unnötig ist, `mysql_eof()` aufzurufen.

Wenn Sie auf der anderen Seite `mysql_use_result()` aufrufen, um den Abruf einer Ergebnismenge zu initialisieren, werden die Zeilen des Satzes Zeile für Zeile vom Server erlangt, indem Sie `mysql_fetch_row()` wiederholt aufrufen. Weil während dieses Prozesses ein Verbindungsfehler auftreten kann, bedeutet ein `NULL`-Rückgabewert von `mysql_fetch_row()` nicht notwendigerweise, dass das Ende der Ergebnismenge auf normale Weise erreicht wurde. In diesem Fall können Sie `mysql_eof()` benutzen, um festzustellen, was passiert ist. `mysql_eof()` gibt einen Nicht-0-Wert zurück, wenn das Ende der Ergebnismenge erreicht wurde, und 0, wenn ein Fehler auftrat.

Historisch liegt `mysql_eof()` vor den Standard-MySQL-Fehlerfunktionen `mysql_errno()` und `mysql_error()`. Weil diese Fehlerfunktionen dieselben Informationen zur Verfügung stellen, wird ihre Benutzung des des veralteten `mysql_eof()` empfohlen. (Sie stellen in der Tat sogar mehr Informationen zur Verfügung, weil `mysql_eof()` nur einen Bool'schen Wert zurückgibt, während die Fehlerfunktionen den Grund angeben, warum der Fehler auftrat.)

Rückgabewerte

0, wenn kein Fehler auftrat. Nicht-0, wenn das Ende der Ergebnismenge erreicht wurde.

Fehler

Keine.

Beispiel

Folgendes Beispiel zeigt, wie Sie `mysql_eof()` benutzen können:

```
mysql_query(&mysql,"SELECT * FROM tabelle");
ergebnis = mysql_use_result(&mysql);
while((zeile = mysql_fetch_row(ergebnis)))
{
    // Daten verarbeiten usw.
}
if(!mysql_eof(ergebnis))  // mysql_fetch_row() schlug wegen eines Fehlers fehl
{
    fprintf(stderr, "Fehler: %s\n", mysql_error(&mysql));
}
```

Sie können denselben Effekt jedoch auch mit den Standard-MySQL-Fehlerfunktionen erreichen:

```
mysql_query(&mysql,"SELECT * FROM tabelle");
result = mysql_use_result(&mysql);
while((zeile = mysql_fetch_row(ergebnis)))
{
    // Daten verarbeiten usw.
}
if(mysql_errno(&mysql))  // mysql_fetch_row() schlug wegen eines Fehlers fehl
{
    fprintf(stderr, "Fehler: %s\n", mysql_error(&mysql));
}
```

9.4.3.12. `mysql_errno()`

```
unsigned int mysql_errno(MYSQL *mysql)
```

Beschreibung

Für die von `mysql` angegebene Verbindung gibt `mysql_errno()` den Fehlercode für die zuletzt aufgerufene API-Funktion zurück, die erfolgreich sein oder fehlschlagen kann. Ein Rückgabewert von 0 bedeutet, dass kein Fehler auftrat. Client-Fehlermeldungsnummern sind in der MySQL-`errmsg.h`-Header-Datei aufgelistet. Server-Fehlermeldungsnummern sind in `mysqld_error.h` aufgelistet. In der MySQL-Quelldistribution finden Sie eine komplette Liste der Fehlermeldungen und Fehlernummern in der Datei `Docs/mysqld_error.txt`.

Rückgabewerte

Ein Fehlercode-Wert. 0, wenn kein Fehler auftrat.

Fehler

Keine.

9.4.3.13. `mysql_error()`

```
char *mysql_error(MYSQL *mysql)
```

Beschreibung

Für die von `mysql` angegebene Verbindung gibt `mysql_error()` die Fehlermeldung für die zuletzt aufgerufene API-Funktion zurück, die erfolgreich sein oder fehlschlagen kann. Eine leere Zeichenkette (`""`) wird zurückgegeben, wenn kein Fehler auftrat. Das bedeutet, dass folgende zwei Tests äquivalent sind:

```
if(mysql_errno(&mysql))
{
    // Ein Fehler trat auf
}

if(mysql_error(&mysql)[0] != '\0')
{
    // Ein Fehler trat auf
}
```

Die Sprache der Client-Fehlermeldungen kann durch erneutes Kompilieren der MySQL-Client-Bibliothek geändert werden. Sie können Fehlermeldungen in mehreren unterschiedlichen Sprachen auswählen. See Abschnitt 5.6.2, „Nicht englische Fehlermeldungen“.

Rückgabewerte

Eine Zeichenkette, die den Fehler beschreibt. Eine leere Zeichenkette, wenn kein Fehler auftrat.

Fehler

Keine.

9.4.3.14. `mysql_escape_string()`

Statt dessen sollten Sie `mysql_real_escape_string()` benutzen!

Das ist identisch mit `mysql_real_escape_string()`, ausser dass die Verbindung als erstes Argument genommen wird. `mysql_real_escape_string()` escapet die Zeichenkette gemäß dem aktuellen Zeichensatz, wohingegen `mysql_escape_string()` die aktuelle Zeichensatzeinstellung ignoriert.

9.4.3.15. `mysql_fetch_field()`

```
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)
```

Beschreibung

Gibt die Definition einer Spalte der Ergebnismenge als `MYSQL_FIELD`-Struktur zurück. Rufen Sie diese Funktion wiederholt auf, um Informationen über alle Spalten in der Ergebnismenge zu erhalten. `mysql_fetch_field()` gibt `NULL` zurück, wenn es keine weiteren Felder mehr gibt.

`mysql_fetch_field()` wird zurückgesetzt, so dass sie Informationen über das erste Feld zurückgibt, und zwar jedes Mal, wenn Sie eine neue `SELECT`-Anfrage ausführen. Das von `mysql_fetch_field()` zurückgegebene Feld wird auch durch Aufrufe von `mysql_field_seek()` betroffen.

Wenn Sie `mysql_query()` aufgerufen haben, um ein `SELECT` auf eine Tabelle auszuführen, aber nicht `mysql_store_result()` aufgerufen haben, gibt MySQL die vorgabemäßige BLOB-Länge (8 KB) zurück, wenn Sie `mysql_fetch_field()` aufrufen, um nach der Länge eines `BLOB`-Felds zu fragen. (Die Größe von 8 KB wird gewählt, weil MySQL die maximale Länge des `BLOB` nicht kennt. Das wird irgendwann einmal konfigurierbar gemacht.) Nachdem Sie die Ergebnismenge erst einmal abgerufen haben, enthält `field->max_length` die Länge des längsten Werts dieser Spalte in der bestimmten Anfrage.

Rückgabewerte

Die `MYSQL_FIELD`-Struktur der aktuellen Spalte. `NULL`, wenn keine Spalten mehr übrig sind.

Fehler

Keine.

Beispiel

```
MYSQL_FIELD *field;

while((field = mysql_fetch_field(ergebnis)))
{
    printf("Feldname %s\n", field->name);
```

```
}
```

9.4.3.16. mysql_fetch_fields()

```
MYSQL_FIELD *mysql_fetch_fields(MYSQL_RES *result)
```

Beschreibung

Gibt ein Array aller `MYSQL_FIELD`-Strukturen für eine Ergebnismenge zurück. Jede Struktur stellt Felddefinitionen für eine Spalte der Ergebnismenge zur Verfügung.

Rückgabewerte

Ein Array von `MYSQL_FIELD`-Strukturen für alle Spalten einer Ergebnismenge.

Fehler

Keine.

Beispiel

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

num_fields = mysql_num_fields(ergebnis);
fields = mysql_fetch_fields(ergebnis);
for(i = 0; i < num_fields; i++)
{
   printf("Feld %u ist %s\n", i, fields[i].name);
}
```

9.4.3.17. mysql_fetch_field_direct()

```
MYSQL_FIELD *mysql_fetch_field_direct(MYSQL_RES *result, unsigned int feldnr)
```

Beschreibung

Mit der Angabe einer Feldnummer `feldnr` für eine Spalte innerhalb einer Ergebnismenge gibt sie die Felddefinition dieser Spalte als `MYSQL_FIELD`-Struktur zurück. Sie können diese Funktion verwenden, um die Definition für eine beliebige Spalte abzurufen. Der Wert von `feldnr` sollte im Bereich von 0 bis `mysql_num_fields(ergebnis)-1` liegen.

Rückgabewerte

Die `MYSQL_FIELD`-Struktur für die angegebene Spalte.

Fehler

Keine.

Beispiel

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *field;

num_fields = mysql_num_fields(ergebnis);
for(i = 0; i < num_fields; i++)
{
    field = mysql_fetch_field_direct(ergebnis, i);
    printf("Feld %u ist %s\n", i, field->name);
}
```

9.4.3.18. mysql_fetch_lengths()

```
unsigned long *mysql_fetch_lengths(MYSQL_RES *result)
```

Beschreibung

Gibt die Länge der Spalten der aktuellen Zeile innerhalb der Ergebnismenge zurück. Wenn Sie vorhaben, Feldwerte zu kopieren, sind diese Längeninformationen auch nützlich für Optimierungen, weil Sie vermeiden können, `strlen()` aufzurufen. Wenn die Ergebnismenge Binärdaten enthält, kommt hinzu, dass Sie diese Funktion benutzen *müssen*, um die Größe der Daten zu bestimmen, weil `strlen()` falsche Ergebnisse für Felder zurückgibt, die NULL-Zeichen enthalten.

Die Länge leerer Spalten und von Spalten, die NULL-Werte enthalten, ist 0. Um zu sehen, wie man diese beiden Fälle auseinander hält, sehen Sie in der Beschreibung von mysql_fetch_row() nach.

Rückgabewerte

Ein Array vorzeichenloser langer Ganzzahlen (long integer), die die Größe jeder Spalte darstellen (ohne irgend welche begrenzenden NULL-Zeichen). NULL, wenn ein Fehler auftrat.

Fehler

mysql_fetch_lengths() ist nur für die aktuelle Zeile der Ergebnismenge gültig. Sie gibt NULL zurück, wenn Sie sie vor mysql_fetch_row() oder nach dem Abruf aller Zeilen im Ergebnis aufrufen.

Beispiel

```
MYSQL_ROW zeile;
unsigned long *laengen;
unsigned int anzahl_felder;
unsigned int i;

zeile = mysql_fetch_row(ergebnis);
if (zeile)
{
    anzahl_felder = mysql_num_fields(ergebnis);
    laengen = mysql_fetch_lengths(ergebnis);
    for(i = 0; i < anzahl_felder; i++)
    {
         printf("Spalte %u ist %lu Bytes lang.\n", i, lengths[i]);
    }
}
```

9.4.3.19. mysql_fetch_row()

```
MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)
```

Beschreibung

Ruft die nächste Zeile einer Ergebnismenge ab. Wenn sie nach mysql_store_result() benutzt wird, gibt mysql_fetch_row() NULL zurück, wenn es keine weiteren Zeilen zum Abruf mehr gibt. Wenn sie nach mysql_use_result() benutzt wird, gibt mysql_fetch_row() NULL zurück, wenn es keine Zeilen mehr zum Abruf gibt oder wenn ein Fehler auftrat.

Die Anzahl von Werten in der Zeile wird durch mysql_num_fields(ergebnis) gegeben. Wenn zeile den Rückgabewert eines Aufrufs von mysql_fetch_row() enthält, wird auf Zeiger auf die Werte als zeile[0] bis zeile[mysql_num_fields(ergebnis)-1] zugegriffen. NULL-Werte in der Zeile werden durch NULL-Zeiger angezeigt.

Die Längen der Feldwerte in der Zeile können durch Aufruf von mysql_fetch_lengths() bestimmt werden. Leere Felder und Felder, die NULL enthalten, haben beide die Länge 0. Sie können diese auseinanderhalten, indem Sie den Zeiger für den Feldwert überprüfen. Wenn der Zeiger NULL ist, ist das Feld NULL, ansonsten ist das Feld leer.

Rückgabewerte

Eine MYSQL_ROW-Struktur für die nächste Zeile. NULL, wenn keine weiteren Zeilen abzurufen sind oder wenn ein Fehler auftrat.

Fehler

- CR_SERVER_LOST

  Die Verbindung zum Server ging während der Anfrage verloren.

- CR_UNKNOWN_ERROR

  Ein unbekannter Fehler trat auf.

Beispiel

```
MYSQL_ROW zeile;
unsigned int anzahl_felder;
unsigned int i;

anzahl_felder = mysql_num_fields(ergebnis);
while ((zeile = mysql_fetch_row(ergebnis)))
{
   unsigned long *laengen;
   laengen = mysql_fetch_lengths(ergebnis);
```

```
   for(i = 0; i < anzahl_felder; i++)
   {
       printf("[%.*s] ", (int) laengen[i], zeile[i] ? zeile[i] : "NULL");
   }
   printf("\n");
}
```

9.4.3.20. `mysql_field_count()`

```
unsigned int mysql_field_count(MYSQL *mysql)
```

Wenn Sie eine Version von MySQL vor Version 3.22.24 benutzen, sollten Sie statt dessen `unsigned int mysql_num_fields(MYSQL *mysql)` benutzen.

Beschreibung

Gibt die Anzahl von Spalten der letzten Anfrage auf der Verbindung zurück.

Normalerweise wird diese Funktion benutzt, wenn `mysql_store_result()` `NULL` zurückgab (und Sie daher keinen Ergebnismengen-Zeiger haben). In diesem Fall können Sie `mysql_field_count()` aufrufen, um festzustellen, ob `mysql_store_result()` ein leeres Ergebnis hätte zurückgeben sollen oder nicht. Das gestattet dem Client-Programm, die richtigen Aktionen zu ergreifen, ohne wissen zu müssen, ob die Anfrage ein `SELECT` war oder nicht (oder ein `SELECT`-ähnliches Statement). Das unten stehende Beispiel zeigt, wie man das machen kann.

See Abschnitt 9.4.6.1, „Warum gibt `mysql_store_result()` manchmal `NULL` zurück, nachdem `mysql_query()` Erfolg zurückgegeben hat?“.

Rückgabewerte

Eine vorzeichenlose Ganzzahl, die die Anzahl von Feldern in einer Ergebnismenge darstellt.

Fehler

Keine.

Beispiel

```
MYSQL_RES *result;
unsigned int anzahl_felder;
unsigned int anzahl_zeilen;

if (mysql_query(&mysql,anfrage))
{
    // FEHLER
}
else // Anfrage war erfolgreich, zurückgegebene Daten verarbeiten
{
    ergebnis = mysql_store_result(&mysql);
    if (ergebnis)  // Es gibt Zeilen
    {
        anzahl_felder = mysql_num_fields(ergebnis);
        // Zeilen abrufen, dann mysql_free_result(result) aufrufen
    }
    else  // mysql_store_result() gab nichts zurück, hätte es etwas zurückgeben sollen?
    {
        if(mysql_field_count(&mysql) == 0)
        {
            // Anfrage gibt keine Daten zurück
            // (Anfrage war kein SELECT)
            anzahl_zeilen = mysql_affected_rows(&mysql);
        }
        else // mysql_store_result() hätte Daten zurückgeben sollen
        {
            fprintf(stderr, "Fehler: %s\n", mysql_error(&mysql));
        }
    }
}
```

Eine Alternative besteht darin, den `mysql_field_count(&mysql)`-Aufruf durch `mysql_errno(&mysql)` zu ersetzen. In diesem Fall überprüfen Sie direkt auf einen Fehler von `mysql_store_result()`, statt aus dem Wert von `mysql_field_count()` zu schlussfolgern, ob das Statement ein `SELECT` war oder nicht.

9.4.3.21. `mysql_field_seek()`

```
MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *result, MYSQL_FIELD_OFFSET offset)
```

Beschreibung

Setzt den Feldcursor auf den angegebenen Offset. Der nächste Aufruf von `mysql_fetch_field()` ruf die Felddefinition der

Spalte ab, die mit diesem Offset verknüpft ist.

Um bis zum Anfang einer Zeile zu suchen, geben Sie einen `offset`-Wert von 0 an.

Rückgabewerte

Der vorherige Wert des Feldcursors.

Fehler

Keine.

9.4.3.22. `mysql_field_tell()`

```
MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *result)
```

Beschreibung

Gibt die Position des Feldcursors für die letzte `mysql_fetch_field()` zurück. Dieser Wert kann als Argument für `mysql_field_seek()` benutzt werden.

Rückgabewerte

Der aktuelle Offset des Feldcursors.

Fehler

Keine.

9.4.3.23. `mysql_free_result()`

```
void mysql_free_result(MYSQL_RES *result)
```

Beschreibung

Gibt den Speicher frei, der für eine Ergebnismenge von `mysql_store_result()`, `mysql_use_result()`, `mysql_list_dbs()` usw. zugewiesen wurde. Wenn Sie mit einer Ergebnismenge fertig sind, müssen Sie den von ihr benutzten Speicher durch Aufruf von `mysql_free_result()` freigeben.

Rückgabewerte

Keine.

Fehler

Keine.

9.4.3.24. `mysql_get_client_info()`

```
char *mysql_get_client_info(void)
```

Beschreibung

Returns a string that represents the client Bibliothek version.

Rückgabewerte

A character string that represents the MySQL-Client Bibliothek version.

Fehler

Keine.

9.4.3.25. `mysql_get_host_info()`

```
char *mysql_get_host_info(MYSQL *mysql)
```

Beschreibung

Gibt eine Zeichenkette zurück, die den Typ der benutzten Verbindung beschreibt, inklusive des Server-Hostnamens.

Rückgabewerte

Eine Zeichenkette, die den Server-Hostnamen und den Verbindungstyp bezeichnet.

Fehler

Keine.

9.4.3.26. mysql_get_proto_info()

```
unsigned int mysql_get_proto_info(MYSQL *mysql)
```

Beschreibung

Gibt die Protokollversion zurück, die von der aktuellen Verbindung benutzt wird.

Rückgabewerte

Eine vorzeichenlose Ganzzahl, die die Protokollversion bezeichnet, die von der aktuellen Verbindung benutzt wird.

Fehler

Keine.

9.4.3.27. mysql_get_server_info()

```
char *mysql_get_server_info(MYSQL *mysql)
```

Beschreibung

Gibt eine Zeichenkette zurück, die die Server-Versionsnummer bezeichnet.

Rückgabewerte

Eine Zeichenkette, die die Server-Versionsnummer bezeichnet.

Fehler

Keine.

9.4.3.28. mysql_info()

```
char *mysql_info(MYSQL *mysql)
```

Beschreibung

Ruft eine Zeichenkette ab, die Informationen über die zuletzt ausgeführte Anfrage zurückgibt, aber nur für die unten aufgeführten Statements. Bei anderen Statements gibt `mysql_info()` `NULL` zurück. Das Format der Zeichenkette variiert in Abhängigkeit vom Anfragetyp, wie unten beschrieben. Die Nummern dienen nur der Veranschaulichung; die Zeichenkette enthält die der Anfrage entsprechenden Werte.

- `INSERT INTO ... SELECT ...`

  Zeichenkettenformat: `Records: 100 Duplicates: 0 Warnings: 0`

- `INSERT INTO ... VALUES (...),(...),(...)...`

  Zeichenkettenformat: `Records: 3 Duplicates: 0 Warnings: 0`

- `LOAD DATA INFILE ...`

  Zeichenkettenformat: `Records: 1 Deleted: 0 Skipped: 0 Warnings: 0`

- `ALTER TABLE`

  Zeichenkettenformat: `Records: 3 Duplicates: 0 Warnings: 0`

- `UPDATE`

  Zeichenkettenformat: `Rows matched: 40 Changed: 40 Warnings: 0`

Beachten Sie, dass `mysql_info()` einen Nicht-`NULL`-Wert für das `INSERT ... VALUES`-Statement nur dann zurückgibt, wenn im Statement mehrfache Wertlisten angegeben sind.

Rückgabewerte

Eine Zeichenkette, die zusätzliche Informationen über die zuletzt ausgeführte Anfrage bereitstellt. `NULL`, wenn für die Anfrage keine Information verfügbar ist.

Fehler

Keine.

9.4.3.29. `mysql_init()`

```
MYSQL *mysql_init(MYSQL *mysql)
```

Beschreibung

Alloziert oder initialisiert ein `MYSQL`-Objekt, das für `mysql_real_connect()` geeignet ist. Wenn `mysql` ein `NULL`-Zeiger ist, alloziert, initialisiert und gibt diese Funktion ein neues Objekt zurück. Ansonsten wird das Objekt initialisiert und die Adresse des Objekts zurückgegeben. Wenn `mysql_init()` ein neues Objekt alloziert, wird es freigegeben, wenn `mysql_close()` aufgerufen wird, um die Verbindung zu schließen.

Rückgabewerte

Ein initialisiertes `MYSQL*`-Handle. `NULL`, wenn der Speicher nicht ausreichte, um ein neues Objekt zu allozieren.

Fehler

Im Falle von ungenügendem Speicher wird `NULL` zurückgegeben.

9.4.3.30. `mysql_insert_id()`

```
my_ulonglong mysql_insert_id(MYSQL *mysql)
```

Beschreibung

Gibt die Kennung zurück, die für eine `AUTO_INCREMENT`-Spalte durch die vorherige Anfrage erzeugt wurde. Benutzen Sie diese Funktion, nachdem Sie eine `INSERT`-Anfrage für eine Tabelle durchgeführt haben, die ein `AUTO_INCREMENT`-Feld enthält.

Beachten Sie, dass `mysql_insert_id()` `0` zurückgibt, wenn die vorherige Anfrage keinen `AUTO_INCREMENT`-Wert erzeugt hat. Wenn Sie den Wert für spätere Benutzung speichern wollen, stellen Sie sicher, dass Sie `mysql_insert_id()` unmittelbar nach der Anfrage aufrufen, die den Wert erzeugt.

`mysql_insert_id()` wird nach `INSERT`- und `UPDATE`-Statements aktualisiert, die einen `AUTO_INCREMENT`-Wert erzeugen oder einen Spaltenwert auf `LAST_INSERT_ID(ausdruck)` setzen. See Abschnitt 7.3.5.2, „Verschiedene Funktionen“.

Beachten Sie auch, dass der Wert der SQL-`LAST_INSERT_ID()`-Funktion immer den aktuellsten erzeugten `AUTO_INCREMENT`-Wert enthält, und zwischen Anfragen nicht zurückgesetzt wird, weil der Wert dieser Funktion im Server gewartet wird.

Rückgabewerte

Der Wert des `AUTO_INCREMENT`-Felds, das durch die vorherige Anfrage aktualisiert wurde. Gibt 0 zurück, wenn es keine vorherige Anfrage auf der Verbindung gab oder wenn die Anfrage keinen `AUTO_INCREMENT`-Wert aktualisierte.

Fehler

Keine.

9.4.3.31. `mysql_kill()`

```
int mysql_kill(MYSQL *mysql, unsigned long pid)
```

Beschreibung

Bittet den Server, den Thread zu töten, der durch `pid` angegeben wurde.

Rückgabewerte

0 für Erfolg. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.32. `mysql_list_dbs()`

```
MYSQL_RES *mysql_list_dbs(MYSQL *mysql, const char *wild)
```

Beschreibung

Gibt eine Ergebnismenge zurück, die aus den Datenbanknamen auf dem Server besteht, die mit dem einfachen regulären Ausdruck übereinstimmen, der durch den `wild`-Parameter angegeben wird. `wild` darf die Platzhalterzeichen '`%`' oder '`_`' enthalten oder ein `NULL`-Zeiger sein, der mit allen Datenbanken übereinstimmt. Der Aufruf von `mysql_list_dbs()` ist ähnlich der Ausführung der Anfrage `SHOW DATABASES [LIKE wild]`.

Sie müssen die Ergebnismenge mit `mysql_free_result()` freigeben.

Rückgabewerte

Eine `MYSQL_RES`-Ergebnismenge bei Erfolg. `NULL`, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_OUT_OF_MEMORY`

  Kein Speicher mehr.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.33. `mysql_list_fields()`

```
MYSQL_RES *mysql_list_fields(MYSQL *mysql, const char *table, const char *wild)
```

Beschreibung

Gibt eine Ergebnismenge zurück, die aus Feldnamen in der angegebenen Tabelle bestehen, die mit einem einfachen regulären Ausdruck übereinstimmen, der durch den `wild`-Parameter angegeben wird. `wild` darf die Platzhalterzeichen '`%`' oder '`_`' enthalten oder ein `NULL`-Zeiger sein, der mit allen Datenbanken übereinstimmt. Der Aufruf von `mysql_list_fields()` ist ähnlich der Ausführung der Anfrage `SHOW COLUMNS FROM tabelle [LIKE wild]`.

Beachten Sie, dass empfohlen wird, `SHOW COLUMNS FROM tabelle` statt `mysql_list_fields()` zu benutzen.

Sie müssen die Ergebnismenge mit `mysql_free_result()` freigeben.

Rückgabewerte

Eine `MYSQL_RES`-Ergebnismenge bei Erfolg. `NULL`, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.34. `mysql_list_processes()`

```
MYSQL_RES *mysql_list_processes(MYSQL *mysql)
```

Beschreibung

Gibt eine Ergebnismenge zurück, die die aktuellen Server-Threads beschreibt. Das ist dieselbe Art von Information, die von `mysqladmin processlist` oder einer `SHOW PROCESSLIST`-Anfrage zur Verfügung gestellt wird.

Sie müssen die Ergebnismenge mit `mysql_free_result()` freigeben.

Rückgabewerte

Eine `MYSQL_RES`-Ergebnismenge bei Erfolg. `NULL`, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.35. `mysql_list_tables()`

```
MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char *wild)
```

Beschreibung

Gibt eine Ergebnismenge zurück, die aus Tabellennamen in der aktuellen Datenbank besteht, die mit einem einfachen regulären Ausdruck übereinstimmen, der durch den `wild`-Parameter angegeben wird. `wild` darf die Platzhalterzeichen '`%`' oder '`_`' enthalten oder ein `NULL`-Zeiger sein, der mit allen Tabellen übereinstimmt. Der Aufruf von `mysql_list_tables()` ist ähnlich

der Ausführung der Anfrage `SHOW tables [LIKE wild]`.

Sie müssen die Ergebnismenge mit `mysql_free_result()` freigeben.

Rückgabewerte

Eine `MYSQL_RES`-Ergebnismenge bei Erfolg. `NULL`, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.36. `mysql_num_fields()`

```
unsigned int mysql_num_fields(MYSQL_RES *result)
```

oder

```
unsigned int mysql_num_fields(MYSQL *mysql)
```

Die zweite Form funktioniert nicht bei MySQL-Version 3.22.24 oder neuer. Um ein `MYSQL*`-Argument zu übergeben, müssen Sie statt dessen `unsigned int mysql_field_count(MYSQL *mysql)` benutzen.

Beschreibung

Gibt die Anzahl von Spalten in einer Ergebnismenge zurück.

Beachten Sie, dass Sie die Anzahl von Spalten entweder durch einen Zeiger auf die Ergebnismenge oder auf ein Verbindungs-Handle erhalten. Das Verbindungs-Handle benutzen Sie, wenn `mysql_store_result()` oder `mysql_use_result()` `NULL` zurückgibt (und Sie daher keinen Ergebnismengen-Zeiger haben). In diesem Fall können Sie `mysql_field_count()` aufrufen, um festzustellen, ob `mysql_store_result()` eine leere Ergebnismenge produziert haben sollte oder nicht. Das erlaubt dem Client-Programm, die korrekten Aktionen vorzunehmen, ohne wissen zu müssen, ob die Anfrage ein `SELECT`- (oder `SELECT`-ähnliches) Statement war oder nicht. Das unten stehende Beispiel zeigt, wie das gemacht wird.

See Abschnitt 9.4.6.1, „Warum gibt `mysql_store_result()` manchmal `NULL` zurück, nachdem `mysql_query()` Erfolg zurückgegeben hat?“.

Rückgabewerte

Eine vorzeichenlose Ganzzahl, die die Anzahl von Feldern in einer Ergebnismenge darstellt.

Fehler

Keine.

Beispiel

```
MYSQL_RES *ergebnis;
unsigned int anzahl_felder;
unsigned int anzahl_zeilen;

if (mysql_query(&mysql,anfrage_zeichenkette))
{
    // FEHLER
}
else // Anfrage erfolgreich, zurückgegebene Daten verarbeiten
{
    ergebnis = mysql_store_result(&mysql);
    if (ergebnis)  // Es gibt Zeilen
    {
```

```
        anzahl_felder = mysql_num_fields(ergebnis);
        // Zeilen abrufen, dann mysql_free_result(ergebnis) aufrufen
    }
    else  // mysql_store_result() gab nichts zurück, hätte es das tun sollen?
    {
        if (mysql_errno(&mysql))
        {
           fprintf(stderr, "Fehler: %s\n", mysql_error(&mysql));
        }
        else if (mysql_field_count(&mysql) == 0)
        {
            // Anfrage gibt keine Daten zurück
            // (war kein SELECT)
            anzahl_zeilen = mysql_affected_rows(&mysql);
        }
    }
}
```

Eine Alternative (wenn Sie WISSEN, dass Ihre Anfrage eine Ergebnismenge hätte zurückgeben sollen) ist es, den `mysql_errno(&mysql)`-Aufruf durch eine Prüfung zu ersetzen, ob `mysql_field_count(&mysql)` gleich 0 ist. Das passiert nur, wenn etwas schief lief.

9.4.3.37. `mysql_num_rows()`

```
my_ulonglong mysql_num_rows(MYSQL_RES *result)
```

Beschreibung

Gibt die Anzahl von Zeilen in der Ergebnismenge zurück.

Die Benutzung von `mysql_num_rows()` hängt davon ab, ob Sie `mysql_store_result()` oder `mysql_use_result()` benutzen, um die Ergebnismenge zurückzugeben.. Wenn Sie `mysql_store_result()` benutzen, kann `mysql_num_rows()` unmittelbar aufgerufen werden. Wenn Sie `mysql_use_result()` benutzen, gibt `mysql_num_rows()` nicht den richtigen Wert zurück, bis alle Zeilen in der Ergebnismenge abgerufen wurden.

Rückgabewerte

Die Anzahl von Zeilen in der Ergebnismenge.

Fehler

Keine.

9.4.3.38. `mysql_options()`

```
int mysql_options(MYSQL *mysql, enum mysql_option option, const char *arg)
```

Beschreibung

Kann benutzt werden, um zusätzliche Optionen zu setzen und das Verhalten einer Verbindung zu beeinflussen. Diese Funktion kann mehrfach aufgerufen werden, um mehrere Optionen zu setzen.

`mysql_options()` sollte nach `mysql_init()` und vor `mysql_connect()` oder `mysql_real_connect()` aufgerufen werden.

Das `option`-Argument ist die Option, die Sie setzen wollen. Das `arg`-Argument ist der Wert für die Option. Wenn die Option eine Ganzzahl ist, sollte `arg` auf den Wert der Ganzzahl zeigen.

Mögliche Optionswerte:

Option	Argumenttyp	Funktion
`MYSQL_OPT_CONNECT_TIMEOUT`	`unsigned int *`	Verbindungszeitüberschreitung (Timeout) in Sekunden.
`MYSQL_OPT_COMPRESS`	Unbenutzt	Das komprimierte Client-/Server-Protokoll verwenden.
`MYSQL_OPT_NAMED_PIPE`	Unbenutzt	Named Pipes benutzen, um sich mit einem MySQL-Server unter NT zu verbinden.
`MYSQL_INIT_COMMAND`	`char *`	Befehl, der beim Verbinden mit dem MySQL-Server ausgeführt werden soll. Wird beim erneuten Verbinden automatisch wieder ausgeführt.
`MYSQL_READ_DEFAULT_FILE`	`char *`	Optionen aus der benannten Optionsdatei einlesen anstelle von `my.cnf`.

MYSQL_READ_DEFAULT_GROUP	char *	Optionen aus der benannten Gruppe von my.cnf oder der Datei einlesen, die durch MYSQL_READ_DEFAULT_FILE angegeben wurde.

Beachten Sie, dass die Gruppe client immer gelesen wird, wenn Sie MYSQL_READ_DEFAULT_FILE oder MYSQL_READ_DEFAULT_GROUP benutzen.

Die angegebene Gruppe in der Optionsdatei kann folgende Optionen enthalten:

connect_timeout	Zeitüberschreitung (Timeout) für die Verbindung in Sekunden. Unter Linux wird dieser Wert zusätzlich für die Wartezeit auf die erste Antwort vom Server benutzt.
compress	Das komprimierte Client-/Server-Protokoll benutzen.
database	Mit dieser Datenbank verbinden, wenn im Verbindungsbefehl keine Datenbank angegeben wurde.
debug	Debug-Optionen.
host	Vorgabemäßiger Hostname.
init-commund	Befehl, der bei der Verbindung zum MySQL-Server ausgeführt wird. Wird automatisch beim erneuten Verbinden erneut ausgeführt.
interactive-timeout	Dasselbe wie die Angabe von CLIENT_INTERACTIVE für mysql_real_connect(). See Abschnitt 9.4.3.41, „mysql_real_connect()“.
password	Vorgabemäßiges Passwort.
pipe	Named Pipes benutzen, um sich mit einem MySQL-Server unter NT zu verbinden.
port	Vorgabemäßige Port-Nummer.
return-found-rows	Weist mysql_info() an, gefundene Zeilen anstelle von aktualisierten Zeilen zurückzugeben, wenn UPDATE benutzt wird.
socket	Vorgabemäßige Socket-Nummer.
user	Vorgabemäßiger Benutzer.

Beachten Sie, dass timeout durch connect_timeout ersetzt wurde. Dennoch wird timeout noch für eine Weile funktionieren.

Weitere Informationen über Optionsdateien finden Sie unter Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

Rückgabewerte

0 bei Erfolg. Nicht-0, wenn Sie eine unbekannte Option verwenden.

Beispiel

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_OPT_COMPRESS,0);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"odbc");
if (!mysql_real_connect(&mysql,"host","benutzer","passwort","datenbank",0,NULL,0))
{
    fprintf(stderr, "Keine Verbindung zur Datenbank: Fehler: %s\n",
          mysql_error(&mysql));
}
```

Im obigen Beispiel wird der Client angewiesen, das komprimierte Client-/Server-Protokoll zu benutzen und zusätzliche Optionen aus dem odbc-Abschnitt in my.cnf zu lesen.

9.4.3.39. mysql_ping()

```
int mysql_ping(MYSQL *mysql)
```

Beschreibung

Prüft, ob die Verbindung zum Server funktioniert oder nicht. Wenn diese weg ist, wird automatisch eine erneute Verbindung versucht.

Diese Funktion kann von Clients benutzt werden, die für lange Zeit im Leerlauf laufen, um zu prüfen, ob der Server die

Verbindung geschlossen hat, und sich bei Bedarf erneut zu verbinden.

Rückgabewerte

0, wenn der Server da ist. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.40. `mysql_query()`

```
int mysql_query(MYSQL *mysql, const char *anfrage)
```

Beschreibung

Führt die SQL-Anfrage aus, auf die durch die NULL-begrenzte Zeichenkette `anfrage` gezeigt wird. Die Anfrage muss aus einem einzelnen SQL-Statement bestehen. Sie dürfen kein Semikolon (‘`;`’) oder `\g` zum Statement hinzufügen.

`mysql_query()` kann nicht für Anfragen benutzt werden, die Binärdaten enthalten. Hierfür sollten Sie statt dessen `mysql_real_query()` benutzen. (Binärdaten können das ‘`\0`’-Zeichen enthalten, was `mysql_query()` als Ende der Anfrage-Zeichenkette interpretiert.)

Wenn Sie wissen wollen, ob die Anfrage eine Ergebnismenge zurückgeben sollte oder nicht, können Sie `mysql_field_count()` benutzen, um hierauf zu prüfen. See Abschnitt 9.4.3.20, „`mysql_field_count()`“.

Rückgabewerte

0, wenn die Anfrage erfolgreich war. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.41. `mysql_real_connect()`

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned int client_flag)
```

Beschreibung

`mysql_real_connect()` versucht, eine Verbindung zu einer MySQL-Datenbankmaschine aufzubauen, die auf `host` läuft.

`mysql_real_connect()` muss erfolgreich verlaufen sein, bevor Sie irgend eine andere API-Funktion ausführen können, mit Ausnahme von `mysql_get_client_info()`.

Die Parameter werden wie folgt angegeben:

- Der erste Parameter sollte die Adresse einer existierenden `MYSQL`-Struktur sein. Vor dem Aufruf von `mysql_real_connect()` müssen Sie `mysql_init()` aufrufen, um die `MYSQL`-Struktur zu initialisieren. Sie können viele der Verbindungsoptionen mit dem `mysql_options()`-Aufruf ändern. See Abschnitt 9.4.3.38, „`mysql_options()`“.
- Der Wert von `host` kann entweder ein Hostname oder eine IP-Adresse sein. Wenn `host` `NULL` oder die Zeichenkette `"localhost"` ist, wird eine Verbindung zum lokalen Host angenommen. Wenn das Betriebssystem Sockets (Unix) oder Named Pipes (Windows NT) unterstützt, werden diese statt TCP/IP benutzt, um sich mit dem Server zu verbinden.
- Der `user`-Parameter enthält die MySQL-Login-Benutzerkennung. Wenn `user` `NULL` ist, wird der aktuelle Benutzer angenommen. Unter Unix ist das der aktuelle Login-Name. Unter Windows-ODBC muss der aktuelle Benutzername explizit angegeben werden. See Abschnitt 9.3.2, „Wie Sie die verschiedenen Felder im ODBC-Administrator Programm ausfüllen“.
- Der `passwd`-Parameter enthält das Passwort für `user`. Wenn `passwd` `NULL` ist, werden nur Einträge in der `user`-Tabelle für Benutzer auf Übereinstimmung überprüft, die ein leeres Passwort-Feld haben. Das erlaubt dem Datenbank-Administrator, das MySQL-Berechtigungssystem so einzurichten, dass Benutzer unterschiedliche Berechtigungen haben, je nachdem, ob sie ein Passwort angegeben haben oder nicht.

  HINWEIS: Versuchen Sie nicht, dass Passwort zu verschlüsseln, bevor Sie `mysql_real_connect()` aufrufen. Die Passwortverschlüsselung wird automatisch durch die Client-API gehandhabt.
- `db` ist der Datenbankname. Wenn `db` nicht `NULL` ist, wird die vorgabemäßige Datenbank für die Verbindung auf diesen Wert gesetzt.
- Wenn `port` nicht 0 ist, wird dieser Wert als Port-Nummer für die TCP/IP-Verbindung benutzt. Beachten Sie, dass der `host`-Parameter den Verbindungstyp festlegt.
- Wenn `unix_socket` nicht `NULL` ist, legt die Zeichenkette den Socket oder die Named Pipe fest, die benutzt werden sollen. Beachten Sie, dass der `host`-Parameter den Verbindungstyp festlegt.
- Der Wert von `client_flag` ist üblicherweise 0, kann aber unter sehr speziellen Umständen auf eine Kombination folgender Flags gesetzt werden:

Flag-Name	**Flag-Bedeutung**
`CLIENT_COMPRESS`	Komprimiertes Protokoll benutzen.
`CLIENT_FOUND_ROWS`	Die Anzahl gefundener (übereinstimmender) Zeilen zurückgeben, nicht die Anzahl betroffener Zeilen.
`CLIENT_IGNORE_SPACE`	Leerzeichen nach Funktionsnamen zulassen. Macht alle Funktionsnamen zu reservierten Wörter.
`CLIENT_INTERACTIVE`	`interactive_timeout` Sekunden zulassen (anstelle von `wait_timeout` Sekunden) von Inaktivität, bevor die Verbindung geschlossen wird.
`CLIENT_NO_SCHEMA`	Die `datenbank.tabelle.spalte`-Syntax nicht zulassen. Das ist für ODBC. Der Flag veranlasst den Parser, einen Fehler zu erzeugen, wenn Sie diese Syntax benutzen, was für die Fehlersuche in einigen ODBC-Programmen hilfreich ist.
`CLIENT_ODBC`	Der Client ist ein ODBC-Client. Das ändert
`CLIENT_SSL`	SSL benutzen (verschlüsseltes Protokoll).

Rückgabewerte

Ein `MYSQL*`-Verbindungs-Handle, wenn die Verbindung erfolgreich war, `NULL`, wenn die Verbindung nicht erfolgreich war. Bei einer erfolgreichen Verbindung ist der Rückgabewert derselbe wie der Wert des ersten Parameters, es sei denn, Sie übergeben für diesen Parameter `NULL`.

Fehler

- `CR_CONN_HOST_ERROR`

  Verbindung zum MySQL-Server fehlgeschlagen.

- `CR_CONNECTION_ERROR`

  Verbindung zum lokalen MySQL-Server fehlgeschlagen.

- `CR_IPSOCK_ERROR`

  IP-Socket konnte nicht erzeugt werden.

- `CR_OUT_OF_MEMORY`

  Kein Speicher mehr.

- `CR_SOCKET_CREATE_ERROR`

  Unix-Socket konnte nicht erzeugt werden.

- `CR_UNKNOWN_HOST`

  IP-Adresse für den Hostnamen konnte nicht gefunden werden.

- `CR_VERSION_ERROR`

  Eine Protokollunverträglichkeit resultierte aus dem Versuch, sich mit einer Client-Bibliothek mit einem Server zu verbinden, die eine andere Protokollversion benutzt. Das kann passieren, wenn Sie eine sehr alte Client-Bibliothek benutzen und sich mit einem neuen Server verbinden, der nicht mit der `--old-protocol`-Option gestartet wurde.

- `CR_NAMEDPIPEOPEN_ERROR`

  Named Pipe unter Windows konnte nicht erzeugt werden.

- `CR_NAMEDPIPEWAIT_ERROR`

  Fehler beim Warten auf eine Named Pipe unter Windows.

- `CR_NAMEDPIPESETSTATE_ERROR`

  Pipe-Handler unter Windows konnte nicht erlangt werden.

- `CR_SERVER_LOST`

  Wenn `connect_timeout` > 0 ist und die Verbindung zum Server länger als `connect_timeout` benötigte, oder wenn der Server während der Ausführung von `init-command` starb.

Beispiel

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"your_prog_name");
if (!mysql_real_connect(&mysql,"host","benutzer","passwort","datenbank",0,NULL,0))
{
    fprintf(stderr, "Verbindung zur Datenbank fehlgeschlagen: Fehler: %s\n",
          mysql_error(&mysql));
}
```

Wenn Sie `mysql_options()` benutzen, liest die MySQL-Bibliothek die `[client]`- und `ihr_programm_name`-Abschnitte in der `my.cnf`-Datei. Das stellt sicher, dass Ihr Programm funktioniert, selbst wenn jemand MySQL auf Nicht-Standard-Weise eingerichtet hat.

Beachten Sie, dass `mysql_real_connect()` beim Verbinden den `reconnect`-Flag (Teil der MySQL-Struktur) auf einen Wert von `1` setzt. Dieser Flag gibt an, dass ein erneuter Verbindungsversuch zum Server gemacht wird, bevor aufgegeben wird, wenn eine Anfrage wegen einer verloren gegangenen Verbindung nicht ausgeführt werden kann.

9.4.3.42. `mysql_real_escape_string()`

```
unsigned int mysql_real_escape_string(MYSQL *mysql, char *nach, const char *von,
unsigned int laenge)
```

Beschreibung

Diese Funktion wird benutzt, um eine zulässige SQL-Zeichenkette zu erzeugen, die Sie in einem SQL-Statement benutzen können. See Abschnitt 7.1.1.1, „Zeichenketten“.

Die Zeichenkette in `von` wird in eine escapete SQL-Zeichenkette kodiert, wobei der aktuelle Zeichensatz der Verbindung berücksichtigt wird. Das Ergebnis wird in `nach` platziert und ein Null-Byte am Ende angefügt. Kodierte Zeichen sind `NUL` (ASCII 0), '`\n`', '`\r`', '`\`', '`'`', '`"`' und Control-Z (see Abschnitt 7.1.1, „Literale: Wie Zeichenketten und Zahlen geschrieben werden“).

Die Zeichenkette, auf die von `von` gezeigt wird, muss `laenge` Bytes lang sein. Sie müssen den `nach`-Puffer so zuweisen, dass er mindestens `laenge*2+1` Bytes lang ist. (Im schlimmsten Fall muss jedes Zeichen mit zwei Bytes kodiert werden, und Sie brauchen Platz für das Null-Byte am Ende.) Wenn `mysql_escape_string()` zurückgibt, sind die Inhalte von `nach` eine Null-begrenzte Zeichenkette. Der Rückgabewert ist die Länge der kodierten Zeichenkette, ohne das Null-Zeichen am Ende.

Beispiel

```
char anfrage[1000],*end;

end = strmov(anfrage,"INSERT INTO tabelle values(");
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"Was is'n das?",12);
*end++ = '\'';
*end++ = ',';
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"Binärdaten: \0\r\n",17);
*end++ = '\'';
*end++ = ')';

if (mysql_real_query(&mysql,anfrage,(unsigned int) (end - anfrage)))
{
   fprintf(stderr, "Einfügen der Zeile fehlgeschlagen, Fehler: %s\n",
           mysql_error(&mysql));
}
```

Die im Beispiel benutzte `strmov()`-Funktion ist in der `mysqlclient`-Bibliothek enthalten und funktioniert wie `strcpy()`, gibt aber einen Zeiger auf Null am Ende des ersten Parameters zurück.

Rückgabewerte

Die Länge des Wertes in `nach`, ohne das Null-Zeichen am Ende.

Fehler

Keine.

9.4.3.43. `mysql_real_query()`

```
int mysql_real_query(MYSQL *mysql, const char *anfrage, unsigned int laenge)
```

Beschreibung

Führt die SQL-Anfrage aus, auf die von `anfrage` gezeigt wird, was eine `laenge` Bytes lange Zeichenkette sein sollte. Die0 Anfrage muss aus einem einzelnen SQL-Statement bestehen. Sie dürfen kein Semikolon ('`;`') oder `\g` zum Statement hinzufügen.

Sie *müssen* `mysql_real_query()` statt `mysql_query()` für Anfragen benutzen, die Binärdaten enthalten, weil Binärdaten das '`\0`'-Zeichen enthalten können. Ausserdem ist `mysql_real_query()` schneller als `mysql_query()`, weil es in der Anfragezeichenkette nicht `strlen()` aufruft.

Wenn Sie wissen wollen, ob die Anfrage eine Ergebnismenge zurückgeben sollte oder nicht, können Sie hierfür `mysql_field_count()` benutzen. See Abschnitt 9.4.3.20, „`mysql_field_count()`“.

Rückgabewerte

0, wenn die Anfrage erfolgreich war. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.44. `mysql_reload()`

```
int mysql_reload(MYSQL *mysql)
```

Beschreibung

Weist den MySQL-Server an, die Berechtigungstabellen neu zu laden. Der verbundene Benutzer muss die **reload**-Berechtigung haben.

Diese Funktion ist veraltet. Vorzugsweise sollten Sie `mysql_query()` benutzen, um statt dessen ein SQL-`FLUSH PRIVILEGES`-Statement auszuführen.

Rückgabewerte

0 bei Erfolg. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.45. `mysql_row_seek()`

```
MYSQL_ROW_OFFSET mysql_row_seek(MYSQL_RES *ergebnis, MYSQL_ROW_OFFSET offset)
```

Beschreibung

Setzt den Zeilencursor auf eine beliebige Zeile in einer Anfrageergebnismenge. Dafür ist erforderlich, dass die Ergebnismengenstruktur das gesamte Ergebnis der Anfrage enthält, so dass `mysql_row_seek()` nur in Verbindung mit `mysql_store_result()` benutzt werden kann, nicht mit `mysql_use_result()`.

Der Offset sollte ein Wert sein, der von einem Aufruf von `mysql_row_tell()` oder `mysql_row_seek()` zurückgegeben wird. Dieser Wert ist nicht einfach eine Zeilennummer; wenn Sie eine Zeile innerhalb einer Ergebnismenge mittels einer Zeilennummer suchen wollen, benutzen Sie statt dessen `mysql_data_seek()`.

Rückgabewerte

Der vorherige Wert des Zeilencursors. Dieser Wert kann an einen nachfolgenden Aufruf von `mysql_row_seek()` übergeben werden.

Fehler

Keine.

9.4.3.46. `mysql_row_tell()`

```
MYSQL_ROW_OFFSET mysql_row_tell(MYSQL_RES *ergebnis)
```

Beschreibung

Gibt die aktuelle Position des Zeilencursors für das letzte `mysql_fetch_row()` zurück. Dieser Wert kann als Argument für

`mysql_row_seek()` benutzt werden.

Sie sollten `mysql_row_tell()` nur nach `mysql_store_result()`, nicht nach `mysql_use_result()` benutzen.

Rückgabewerte

Der aktuelle Offset des Zeilencursors.

Fehler

Keine.

9.4.3.47. `mysql_select_db()`

```
int mysql_select_db(MYSQL *mysql, const char *db)
```

Beschreibung

Führt dazu, dass die Datenbank, die durch `db` angegeben wird, die vorgabemäßige (aktuelle) Datenbank auf der von `mysql` angegebenen Verbindung wird. Bei nachfolgenden Anfragen ist diese Datenbank die Vorgabe für Tabellenverweise, die nicht explizit einen Datenbank-Spezifizierer enthalten.

`mysql_select_db()` schlägt fehl, wenn der verbundene Benutzer keine Zugriffsrechte auf die Datenbank hat.

Rückgabewerte

0 bei Erfolg. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.48. `mysql_shutdown()`

```
int mysql_shutdown(MYSQL *mysql)
```

Beschreibung

Führt dazu, dass der Datenbankserver herunter fährt. Der verbundene Benutzer muss die **shutdown**-Berechtigung haben.

Rückgabewerte

0 bei Erfolg. Nicht-0, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.49. `mysql_stat()`

```
char *mysql_stat(MYSQL *mysql)
```

Beschreibung

Gibt eine Zeichenkette zurück, die Informationen enthält, die ähnlich denen sind, die vom `mysqladmin status`-Befehl zur Verfügung gestellt werden. Das schließt die Betriebszeit (Uptime) in Sekunden und die Anzahl laufender Threads, Anfragen (Questions), Neuladen (Reloads) und offener Tabellen ein.

Rückgabewerte

Eine Zeichenkette, die den Serverstatus beschreibt. `NULL`, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.50. `mysql_store_result()`

```
MYSQL_RES *mysql_store_result(MYSQL *mysql)
```

Beschreibung

Sie müssen `mysql_store_result()` oder `mysql_use_result()` für jede Anfrage aufrufen, die erfolgreich Daten abruft (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`).

Für andere Anfragen müssen Sie `mysql_store_result()` oder `mysql_use_result()` nicht aufrufen, es schadet aber auch nicht, noch führt es zu wahrnehmbaren Performance-Störungen, wenn Sie `mysql_store_result()` in jedem Fall aufrufen. Sie können feststellen, ob die Anfrage keine Ergebnismenge hatte, wenn Sie prüfen, ob `mysql_store_result()` 0 zurückgibt (mehr darüber später).

Wenn Sie wissen wollen, ob die Anfrage eine Ergebnismenge zurückgeben sollte oder nicht, können Sie hierfür `mysql_field_count()` benutzen. See Abschnitt 9.4.3.20, „`mysql_field_count()`“.

`mysql_store_result()` liest das gesamte Ergebnis einer Anfrage zum Client ein, weist eine `MYSQL_RES`-Struktur zu und platziert das Ergebnis in diese Struktur.

`mysql_store_results()` gibt einen NULL-Zeiger zurück, wenn die Anfrage keine Ergebnismenge zurückgab (wenn die Anfrage zum Beispiel ein `INSERT`-Statement war).

`mysql_store_results()` gibt auch einen NULL-Zeiger zurück, wenn das Lesen der Ergebnismenge fehlschlug. Sie können prüfen, ob Sie einen Fehler erhielten, wenn `mysql_error()` keinen NULL-Zeiger zurückgibt, wenn `mysql_errno()` ungleich 0 zurückgibt oder wenn `mysql_field_count()` ungleich 0 zurückgibt.

Eine leere Ergebnismenge wird zurückgegeben, wenn keine Zeilen zurückgegeben werden. (Eine leere Ergebnismenge unterscheidet sich als Rückgabewert von einem NULL-Zeiger.)

Nachdem Sie erst einmal `mysql_store_result()` aufgerufen und ein Ergebnis erhalten haben, das kein NULL-Zeiger ist, können Sie `mysql_num_rows()` aufrufen, um herauszufinden, wie viele Zeilen es in der Ergebnismenge gibt.

Sie können `mysql_fetch_row()` aufrufen, um Zeilen aus der Ergebnismenge zu holen, oder `mysql_row_seek()` und `mysql_row_tell()`, um die aktuelle Zeilenposition innerhalb der Ergebnismenge zu erhalten oder zu setzen.

Sie müssen `mysql_free_result()` aufrufen, wenn Sie mit der Ergebnismenge fertig sind.

See Abschnitt 9.4.6.1, „Warum gibt `mysql_store_result()` manchmal `NULL` zurück, nachdem `mysql_query()` Erfolg zurückgegeben hat?“.

Rückgabewerte

Eine `MYSQL_RES`-Ergebnisstruktur mit den Ergebnissen. `NULL`, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_OUT_OF_MEMORY`

  Kein Speicher mehr.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.3.51. `mysql_thread_id()`

```
unsigned long mysql_thread_id(MYSQL *mysql)
```

Beschreibung

Gibt die Thread-Kennung der aktuellen Verbindung zurück. Der Wert kann als Argument für `mysql_kill()` benutzt werden, um den Thread zu töten.

Wenn die Verbindung verloren ging und Sie sich mit `mysql_ping()` erneut verbinden, ändert sich die Thread-Kennung. Das heißt, dass Sie nicht die Thread-Kennung holen und für spätere Benutzung speichern sollten. Sie sollten sie holen, wenn Sie sie benötigen.

Rückgabewerte

Die Thread-Kennung der aktuellen Verbindung.

Fehler

Keine.

9.4.3.52. `mysql_use_result()`

```
MYSQL_RES *mysql_use_result(MYSQL *mysql)
```

Beschreibung

Sie müssen `mysql_store_result()` oder `mysql_use_result()` für jede Anfrage aufrufen, die erfolgreich Daten abruft (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`).

`mysql_use_result()` initiiert einen Ergebnismengen-Abruf, aber liest die Ergebnismenge nicht tatsächlich in den Client wie `mysql_store_result()`. Statt dessen muss jede Zeile individuell abgerufen werden, indem Aufrufe von

`mysql_fetch_row()` durchgeführt werden. Das liest das Ergebnis einer Anfrage direkt vom Server, ohne es in einer temporären Tabelle oder einem lokalen Puffer zu speichern, was manchmal schneller ist und viel weniger Speicher benutzt als `mysql_store_result()`. Dem Client wird nur Speicher für die aktuelle Zeile zugewiesen sowie ein Kommunikationspuffer, der bis zu `max_allowed_packet` Bytes Groß werden kann.

Auf der anderen Seite sollten Sie `mysql_use_result()` nicht benutzen, wenn Sie viele Verarbeitungen für jede Zeile auf der Client-Seite durchführen oder wenn die Ausgabe auf den Bildschirm geschickt wird, auf dem der Benutzer `^S` (stop scroll) eingeben kann. Das bindet den Server und verhindert, dass andere Threads irgend welche Tabellen aktualisieren können, von denen gerade Daten geholt werden.

Wenn Sie `mysql_use_result()` benutzen, müssen Sie `mysql_fetch_row()` ausführen, bis ein `NULL`-Wert zurückgegeben wird, denn ansonsten werden die nicht geholten Zeilen als Teil der Ergebnismenge bei Ihrer nächsten Anfrage zurückgegeben. Die C-API gibt den Fehler `Commands out of sync; You can't run this command now` aus, wenn Sie das vergessen!

Sie können `mysql_data_seek()`, `mysql_row_seek()`, `mysql_row_tell()`, `mysql_num_rows()` oder `mysql_affected_rows()` nicht bei einem Ergebnis verwenden, das von `mysql_use_result()` zurückgegeben wird. Ausserdem dürfen Sie keine anderen Anfragen absetzen, bis `mysql_use_result()` beendet ist. (Nachdem Sie alle Zeilen abgeholt haben, wird `mysql_num_rows()` jedoch exakt die Anzahl der geholten Zeilen zurückgeben.)

Sie müssen `mysql_free_result()` aufrufen, wenn Sie mit der Ergebnismenge fertig sind.

Rückgabewerte

Eine `MYSQL_RES`-Ergebnisstruktur. `NULL`, wenn ein Fehler auftrat.

Fehler

- `CR_COMMANDS_OUT_OF_SYNC`

  Befehle wurden nicht in der korrekten Reihenfolge ausgeführt.

- `CR_OUT_OF_MEMORY`

  Kein Speicher mehr.

- `CR_SERVER_GONE_ERROR`

  Der MySQL-Server ist weg.

- `CR_SERVER_LOST`

  Die Verbindung zum Server ging während der Anfrage verloren.

- `CR_UNKNOWN_ERROR`

  Ein unbekannter Fehler trat auf.

9.4.4. C-Threaded-Funktionsbeschreibungen

Sie benötigen folgende Funktionen, wenn Sie einen threaded Client erstellen wollen. See Abschnitt 9.4.8, „Wie man einen threaded Client herstellt“.

9.4.4.1. `my_init()`

Beschreibung

Diese Funktion muss einmal im Programm aufgerufen werden, bevor Sie irgend eine MySQL-Funktion aufrufen. Sie initialisiert einige globale Variablen, die MySQL braucht. Wenn Sie eine Thread-sichere Client-Bibliothek benutzen, wird diese ebenfalls `mysql_thread_init()` für diesen Thread aufrufen.

Diese Funktion wird automatisch von `mysql_init()`, `mysql_server_init()` und `mysql_connect()` aufgerufen.

Rückgabewerte

Keine.

9.4.4.2. `mysql_thread_init()`

Beschreibung

Diese Funktion muss für jeden erzeugten Thread aufgerufen werden, um Thread-spezifische Variablen zu initialisieren.

Diese Funktion wird automatisch von `my_init()` und `mysql_connect()` aufgerufen.

Rückgabewerte

Keine.

9.4.4.3. `mysql_thread_end()`

Beschreibung

Diese Funktion muss vor dem Aufruf von `pthread_exit()` aufgerufen werden, um den von `mysql_thread_init()` zugewiesenen Speicher freizusetzen.

Beachten Sie, dass diese Funktion **nicht automatisch** von der Client-Bibliothek aufgerufen wird. Sie muss explizit aufgerufen werden, um Speicherlecks zu vermeiden.

Rückgabewerte

Keine.

9.4.4.4. `mysql_thread_safe()`

```
unsigned int mysql_thread_safe(void)
```

Description

This function indicates whether the client is compiled as thread safe.

Return Values

1 is the client is thread safe, 0 otherwise.

9.4.5. C-Embedded-Server-Funktionsbeschreibungen

Sie müssen folgende Funktionen benutzen, wenn Sie wollen, dass Ihre Applikation gegen die eingebettete MySQL-Server-Bibliothek gelinkt werden kann. See Abschnitt 9.4.9, „libmysqld, die eingebettete MySQL-Server-Bibliothek“.

Wenn das Programm mit `-lmysqlclient` anstelle von `-lmysqld` gelinkt wird, tun diese Funktionen nicht. Das ermöglicht die Auswahl zwischen der Benutzung des eingebetteten MySQL-Servers und eines Standalone-Servers, ohne irgend welchen Code zu verändern.

9.4.5.1. `mysql_server_init()`

```
void mysql_server_init(int argc, const char **argv, const char **groups)
```

Beschreibung

Diese Funktion **muss** einmal im Programm aufgerufen werden, bevor irgend eine andere MySQL-Funktion aufgerufen wird. Sie startet den Server und initialisiert jegliche Subsysteme (`mysys`, InnoDB usw.), die der Server benutzt. Wenn diese Funktion nicht aufgerufen wird, stürzt das Programm ab.

Die `argc`- und `argv`-Argumente sind analog zu den Argumenten für `main()`. Das erste Element von `argv` wird ignoriert (es enthält typischerweise den Programmnamen). Aus Bequemlichkeitsgründen kann `argc` `0` sein, wenn es keine Kommandozeilenargumente für den Server gibt.

Die `NULL`-begrenzte Liste von Zeichenketten in `groups` wählt aus, welche Gruppen in den Optionsdateien aktiv sind. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“. Aus Bequemlichkeitsgründen kann `groups` `NULL` sein. In diesem Fall ist die `[server]`-Gruppe aktiv.

Beispiel

```
#include <mysql.h>
#include <stdlib.h>

static char *server_args[] = {
  "Mein Programm",       /* Diese Zeichenkette ist unbenutzt */
  "--datadir=.",
  "--set-variable=key_buffer_size=32M"
```

```
};
static char *server_groups[] = {
  "server",
  "Dieser_Programm_SERVER",
  (char *)NULL
};

int main(void) {
  mysql_server_init(sizeof(server_args) / sizeof(char *),
                    server_args, server_groups);

  /* Hier können Sie irgend welche MySQL-API-Funktionen benutzen */

  mysql_server_end();

  return EXIT_SUCCESS;
}
```

Rückgabewerte

Keine.

9.4.5.2. `mysql_server_end()`

Beschreibung

Diese Funktion **muss** einmal im Programm nach allen anderen MySQL-Funktionen aufgerufen werden. Sie fährt den eingebetteten Server herunter.

Rückgabewerte

Keine.

9.4.6. Häufige Fragen und Probleme bei der Benutzung der C-API

9.4.6.1. Warum gibt `mysql_store_result()` manchmal `NULL` zurück, nachdem `mysql_query()` Erfolg zurückgegeben hat?

`mysql_store_result()` kann `NULL` zurückgeben, auch nach einem erfolgreichen Aufruf von `mysql_query()`. Wenn das passiert, bedeutet das, dass eine der folgenden Bedingungen eingetreten ist:

- Es gab einen `malloc()`-Fehler (zum Beispiel, wenn die Ergebnismenge zu Groß war).
- Die Daten konnten nicht gelesen werden (ein Fehler mit der Verbindung trat auf).
- Die Anfrage gab keine Daten zurück (sie war zum Beispiel ein `INSERT`, `UPDATE` oder `DELETE`).

Sie können jederzeit prüfen, ob das Statement eine leere Ergebnismenge geliefert haben sollte oder nicht, indem Sie `mysql_field_count()` aufrufen. Wenn `mysql_field_count()` 0 zurückliefert, ist das Ergebnis leer und die letzte Anfrage war ein Statement, die keine Rückgabewerte liefert (zum Beispiel ein `INSERT` oder ein `DELETE`). Wenn `mysql_field_count()` einen Nicht-0-Wert zurückgibt, hätte das Statement ein nicht leeres Ergebnis zurückliefern sollen. Sehen Sie in der Beschreibung von `mysql_field_count()`-Funktion wegen eines Beispiels nach.

Sie können durch Aufruf von `mysql_error()` oder `mysql_errno()` auf einen Fehler überprüfen.

9.4.6.2. Welche Ergebnisse kann ich von einer Anfrage bekommen?

Zusätzlich zur Ergebnismenge, die von einer Anfrage zurückgegeben wird, können Sie auch folgende Informationen bekommen:

- `mysql_affected_rows()` gibt die Anzahl von Zeilen zurück, die durch die letzte Anfrage betroffen wurden, wenn Sie ein `INSERT`, `UPDATE` oder `DELETE` ausführen. Eine Ausnahme besteht darin, wenn `DELETE` ohne eine `WHERE`-Klausel benutzt wird. In diesem Fall wird die Tabelle leer neu erzeugt, was viel schneller ist! Daher gibt `mysql_affected_rows()` 0 für die Anzahl betroffener Datensätze zurück.
- `mysql_num_rows()` gibt die Anzahl von Zeilen in einer Ergebnismenge zurück. Bei `mysql_store_result()` kann `mysql_num_rows()` aufgerufen werden, sobald `mysql_store_result()` etwas zurückgibt. Bei `mysql_use_result()` kann `mysql_num_rows()` erst aufgerufen werden, nachdem Sie alle Zeilen mit `mysql_fetch_row()` geholt haben.
- `mysql_insert_id()` gibt die Kennung zurück, die von der letzten Anfrage erzeugt wurde, die eine Zeile in eine Tabelle

mit einem AUTO_INCREMENT-Index einfügte. See Abschnitt 9.4.3.30, „mysql_insert_id()“.

- Einige Anfragen (LOAD DATA INFILE ..., INSERT INTO ... SELECT ..., UPDATE) geben zusätzliche Informationen zurück. Das Ergebnis wird von mysql_info() zurückgegeben. Siehe die Beschreibung für mysql_info() hinsichtlich des Formats der Zeichenkette, die diese Funktion zurückgibt. mysql_info() gibt einen NULL-Zeiger zurück, wenn es keine zusätzlichen Informationen gibt.

9.4.6.3. Wie erhalte ich die eindeutige Kennung für die letzte eingefügte Zeile?

Wenn Sie einen Datensatz in eine Tabelle einfügen, der eine Spalte enthält, die das AUTO_INCREMENT-Attribut hat, erhalten Sie die letzte erzeugte Kennung durch Aufruf der mysql_insert_id()-Funktion.

Sie können die Kennung auch dadurch abrufen, dass Sie die LAST_INSERT_ID()-Funktion in einer Anfrage-Zeichenkette verwenden, die Sie an mysql_query() übergeben.

Sie können überprüfen, ob ein AUTO_INCREMENT-Index benutzt wird, wenn Sie folgenden Code ausführen. Er prüft auch, ob die Anfrage ein INSERT mit einem AUTO_INCREMENT-Index war:

```
if (mysql_error(&mysql)[0] == 0 &&
    mysql_num_fields(ergebnis) == 0 &&
    mysql_insert_id(&mysql) != 0)
{
    used_id = mysql_insert_id(&mysql);
}
```

Die letzte erzeugte Kennung wird im Server auf der Grundlage der jeweiligen Verbindung gewartet. Sie wird nicht durch andere Clients geändert. Sie wird nicht einmal geändert, wenn Sie eine andere AUTO_INCREMENT-Spalte mit einem nicht magischen Wert aktualisieren (einem Wert, der nicht NULL und nicht 0 ist).

Wenn Sie die Kennung benutzen wollen, die für eine Tabelle erzeugt wurde, um sie in eine zweite Tabelle einzufügen, können Sie SQL-Statements wie folgt benutzen:

```
INSERT INTO foo (auto,text)
    VALUES(NULL,'text');             # Kennung durch Einfügen von NULL erzeugen
INSERT INTO foo2 (id,text)
    VALUES(LAST_INSERT_ID(),'text');  # Kennung in zweiter Tabelle benutzen
```

9.4.6.4. Probleme beim Linken mit der C-API

Wenn Sie mit der C-API linken, können auf manchen Systemen folgende Fehler auftreten:

```
gcc -g -o client test.o -L/usr/local/lib/mysql -lmysqlclient -lsocket -lnsl

Undefined       first referenced
 symbol         in file
floor           /usr/local/lib/mysql/libmysqlclient.a(password.o)
ld: fatal: Symbol referencing errors. No output written to client
```

Wenn das auf Ihrem System passiert, müssen Sie die math-Bibliothek einschließen, indem Sie -lm am Ende der Kompilier- / Link-Zeile hinzufügen.

9.4.7. Client-Programme bauen

Wenn Sie MySQL-Clients kompilieren, die Sie selbst geschrieben oder von Dritten erhalten haben, müssen diese mit der -lmysqlclient -lz-Option für den Link-Befehl gelinkt werden. Eventuell sollten Sie auch eine -L-Option verwenden, um dem Linker mitzuteilen, wo sich die Bibliothek befindet. Wenn zum Beispiel die Bibliothek in /usr/local/mysql/lib installiert ist, benutzen Sie -L/usr/local/mysql/lib -lmysqlclient -lz für den Link-Befehl.

Für Clients, die MySQL-Header-Dateien benutzen, müssen Sie eventuell eine -I-Option angeben, wenn Sie sie kompilieren (zum Beispiel -I/usr/local/mysql/include), so dass der Kompiler die Header-Dateien finden kann.

9.4.8. Wie man einen threaded Client herstellt

Die Client-Bibliothek ist fast Thread-sicher. Das größte Problem besteht darin, dass die Subroutinen in net.c, die von Sockets lesen, nicht Interrupt-sicher sind. Das wurde mit dem Hintergedanken gemacht, dass Sie eventuell Ihre eigenen Alarme haben möchten, die ein langes Lesen vom Server unterbrechen können. Wenn Sie Interrupt-Handler für den SIGPIPE-Interrupt installieren, sollte die Socket-Handhabung Thread-sicher sein.

In den älteren Binärdistributionen wurden die Client-Bibliotheken normalerweise nicht mit der Thread-sicheren Option kompiliert (die Windows-Binärdateien sind vorgabemäßig Thread-sicher kompiliert). Neuere Binärdistributionen sollten sowohl eine normale

als auch eine Thread-sichere Client-Bibliothek haben.

Um einen threaded Client zu erhalten, bei dem Sie den Client durch andere Threads unterbrechen (interrupt) und Zeitüberschreitungen (Timeouts) setzen können, wenn Sie mit dem MySQL-Server kommunizieren, sollten Sie die `-lmysys`-, `-lstring`-, und `-ldbug`-Bibliotheken und den `net_serv.o`-Code benutzen, den der Server benutzt.

Wenn Sie keine Unterbrechungen (Interrupts) oder Zeitüberschreitungen (Timeouts) benötigen, können Sie einfach eine Thread-sicher Client-Bibliothek (`mysqlclient_r`) kompilieren und diese benutzen. See Abschnitt 9.4, „MySQL-C-API“. In diesem Fall müssen Sie sich nicht um die `net_serv.o`-Objektdatei oder die anderen MySQL-Bibliotheken kümmern.

Wenn Sie einen threaded Client benutzen und Unterbrechungen (Interrupts) und Zeitüberschreitungen (Timeouts) benutzen wollen, können Sie in umfangreicher Weise die Routinen in der `thr_alarm.c`-Datei benutzen. Wenn Sie Routinen aus der `mysys`-Bibliothek benutzen, müssen Sie lediglich daran denken, `my_init()` zuerst aufzurufen! See Abschnitt 9.4.4, „C-Threaded-Funktionsbeschreibungen“.

Alle Funktionen ausser `mysql_real_connect()` sind vorgabemäßig Thread-sicher. Die folgenden Hinweise beschreiben, wie man eine Thread-sichere Client-Bibliothek kompiliert und sie auf Thread-sichere Weise benutzt. (Die unten stehenden Hinweise für `mysql_real_connect()` beziehen sich in der Tat auch auf `mysql_connect()`. Weil aber `mysql_connect()` veraltet ist, sollten Sie in jedem Fall `mysql_real_connect()` benutzen.)

Um `mysql_real_connect()` Thread-sicher zu machen, müssen Sie die Client-Bibliothek mit diesem Befehl neu kompilieren:

```
shell> ./configure --enable-thread-safe-client
```

Das erzeugt eine Thread-sichere Client-Bibliothek `libmysqlclient_r`. `--enable-thread-safe-client`. Diese Bibliothek ist pro Verbindung Thread-sicher. Sie können zwei Threads dieselbe Verbindung benutzen lassen, solange Sie folgendes tun:

- Zwei Threads können zur gleichen Zeit keine Anfrage an MySQL über dieselbe Verbindung schicken. Insbesondere müssen Sie sicherstellen, dass zwischen einem `mysql_query()` und einem `mysql_store_result()` kein anderer Thread dieselbe Verbindung benutzt.
- Viele Threads können auf unterschiedliche Ergebnismengen zugreifen, die mit `mysql_store_result()` abgerufen wurden.
- Wenn Sie `mysql_use_result` benutzen, müssen Sie sicherstellen, dass kein anderer Thread irgend etwas über dieselbe Verbindung anfragt, bis die Ergebnismenge geschlossen wurde. Für threaded Clients, die dieselbe Verbindung benutzen, ist es jedoch am besten, `mysql_use_result()` zu benutzen.
- Wenn Sie mehrfache Threads über dieselbe Verbindung benutzen wollen, müssen Sie eine mutex-Sperre um Ihre `mysql_query()`- und `mysql_store_result()`-Aufruf-Kombination haben. Sobald `mysql_store_result()` fertig ist, kann die Sperre aufgehoben werden und andere Threads können über dieselbe Verbindung anfragen.
- Wenn Sie mit POSIX-Threads programmieren, können Sie `pthread_mutex_lock()` und `pthread_mutex_unlock()` benutzen, um eine mutex-Sperre aufzubauen und aufzuheben.

Sie müssen folgendes wissen, wenn Sie einen Thread haben, der MySQL-Funktionen aufruft, dieser Thread aber keine Verbindung zur MySQL-Datenbank aufgebaut hat:

Wenn Sie `mysql_init()` oder `mysql_connect()` aufrufen, erzeugt MySQL eine Thread-spezifische Variable für diesen Thread, die von der Debug-Bibliothek benutzt wird (unter anderem).

Wenn Sie in einem Thread-Aufruf eine MySQL-Funktion haben, bevor ein Thread `mysql_init()` oder `mysql_connect()` aufgerufen hat, hat der Thread nicht notwendigerweise Thread-spezifische Variablen zur Hand, und Sie werden wahrscheinlich früher oder später einen Coredump erhalten.

Damit alles reibungslos funktioniert, müssen Sie folgendes tun:

1. Rufen Sie bei Programmbeginn `my_init()` auf, wenn Ihr Programm irgend welche MySQL-Funktion vor dem Aufruf von `mysql_real_connect()` benutzt.
2. Rufen Sie `mysql_thread_init()` im Thread-Handler auf, bevor Sie irgend welche MySQL-Funktionen aufrufen.
3. Rufen Sie im Thread `mysql_thread_end()` auf, bevor Sie `pthread_exit()` aufrufen. Das gibt Speicher frei, der von MySQL-Thread-spezifischen Variablen benutzt wird.

Eventuell erhalten Sie Fehler wegen undefinierter Symbole, wenn Sie Ihren Client mit `mysqlclient_r` linken. In den meisten

Fällen liegt das daran, dass Sie die Thread-Bibliotheken nicht auf der Link- / Kompilierzeile eingeschlossen haben.

9.4.9. libmysqld, die eingebettete MySQL-Server-Bibliothek

9.4.9.1. Überblick über die eingebettete MySQL-Server-Bibliothek

Die eingebettete MySQL-Server-Bibliothek ermöglicht es, einen MySQL-Server mit allen Features innerhalb einer Client-Applikation laufen zu lassen. Die hauptsächlichen Vorteile sind erhöhte Geschwindigkeit und einfachere Verwaltung eingebetteter Applikationen.

9.4.9.2. Programme mit `libmysqld` kompilieren

Momentan müssen alle unterstützten Bibliotheken explizit aufgelistet werden, wenn Sie mit `-lmysqld` linken. In Zukunft wird `mysql_config --libmysqld-libs` die Bibliotheken benennen, um das zu erleichtern. Darüber hinaus werden alle unterstützten Bibliotheken wahrscheinlich in libmysqld eingeschlossen werden, um dies noch weiter zu vereinfachen.

Die korrekten Flags zum Kompilieren und Linken eines threaded Programms müssen benutzt werden, selbst wenn Sie nicht direkt irgend welche Thread-Funktionen in Ihrem Code aufrufen.

9.4.9.3. Ein einfaches Embedded-Server-Beispiel

Dieses Beispiel-Programm und makefile sollten ohne Änderungen auf einem Linux- oder FreeBSD-System funktionieren. Bei anderen Betriebssystemen sind kleinere Änderungen notwendig. Dieses Beispiel ist so angelegt, dass genügend Details dargestellt werden, um die Problematik zu verstehen, ohne zu viel "Verwirrendes" einzubringen, das Teil einer echten Applikation ist.

Um das Beispiel auszuprobieren, erzeugen Sie ein `example`-Verzeichnis auf derselben Ebene wie das mysql-4.0-Quell-Verzeichnis. Speichern Sie die `example.c`-Quelle und das `GNUmakefile` im Verzeichnis und lassen Sie GNU-`make` innerhalb des `example`-Verzeichnisses laufen.

`example.c`

```
/*
 * Ein einfacher Beispiel-Client, der die eingebettete
 * MySQL-Server-Bibliothek benutzt
 */

#include <mysql.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

enum on_error { E_okay, E_warn, E_fail };

static void   die(MYSQL *db, char *fmt, ...);
MYSQL *db_connect(const char *dbname);
void db_disconnect(MYSQL *db);
void db_do_Anfrage(MYSQL *db, const char *query, enum on_error on_error);

const char *server_groups[] = { "test_client_SERVER", "server", NULL };

int
main(int argc, char **argv)
{
  MYSQL *one, *two;

  /* Das muss vor allen weiteren mysql-Funktionen aufgerufen werden.
   *
   * Sie können mysql_server_init(0, NULL, NULL) benutzen,
   * was den Server initialisiert und die Gruppen
   * groups = { "server", NULL } benutzt.
   *
   * In Ihre $HOME/.my.cnf-Datei sollten Sie folgendes eintragen:

[test_client_SERVER]
language = /pfad/zur/quelle/von/mysql/sql/share/english

   * Natürlich können Sie auch argc und argv ändern,
   * bevor Sie sie an diese Funktion übergeben.
   * Oder erzeugen Sie neue auf jede Art, die Sie wollen.
   * Alle Argumente in argv (ausser argv[0], was der Programmname ist)
   * müssen allerdings gültige Optionen für den MySQL-Server sein.
   * Wenn Sie diesen Client gegen die normale mysqlclient-
   * Bibliothek linken, ist diese Funktion nur ein Stumpf, der nichts tut.
   */
  mysql_server_init(argc, argv, server_groups);

  one = db_connect("test");
  two = db_connect(NULL);

  db_do_query(one, "show table status", E_fail);
  db_do_query(two, "show databases", E_fail);

  mysql_close(two);
  mysql_close(one);
```

```
  /* Folgendes muss nach allen anderen mysql-Funktionen aufgerufen werden */
  mysql_server_end();

  exit(EXIT_SUCCESS);
}

void
die(MYSQL *db, char *fmt, ...)
{
  va_list ap;
  va_start(ap, fmt);
  vfprintf(stderr, fmt, ap);
  va_end(ap);
  putc('\n', stderr);
  if (db)
    db_disconnect(db);
  exit(EXIT_FAILURE);
}

MYSQL *
db_connect(const char *dbname)
{
  MYSQL *db = mysql_init(NULL);
  if (!db)
    die(db, "mysql_init fehlgeschlagen: kein Speicher mehr");
  mysql_options(db, MYSQL_READ_DEFAULT_GROUP, "simple");
  if (!mysql_real_connect(db, NULL, NULL, NULL, dbname, 0, NULL, 0))
    die(db, "mysql_real_connect fehlgeschlagen: %s", mysql_error(db));

  return db;
}

void
db_disconnect(MYSQL *db)
{
  mysql_close(db);
}

/*
 * show_query: Dieser Code ist aus mysql.cc. Diese Funktion
 * ist dafür gedacht, intern für db_do_query() benutzt zu werden.
 */
static char *
show_query(MYSQL *db)
{
  MYSQL_RES   *res;
  MYSQL_FIELD *field;
  MYSQL_ROW    row;
  char         sep[256], *psep = sep;
  char        *is_num = 0;
  char        *err = 0;
  unsigned int length = 1;      /* anfangs "|" */
  unsigned int off;

  if (!(res = mysql_store_result(db)))
    return mysql_error(db);

  if (!(is_num = malloc(mysql_num_fields(res))))
  {
    err = "Kein Speicher mehr";
    goto err;
  }

  /* set up */
  *psep++ = '+';
  while ((field = mysql_fetch_field(res)))
  {
    unsigned int len = strlen(field->name);
    if (len < field->max_length)
      len = field->max_length;
    if (len < 2 && !IS_NOT_NULL(field->flags))
      len = 2;  /* \N */
    field->max_length = len + 1;        /* die API verbiegen ... */
    len += 2; length += len + 1;        /* " " davor, " |" danach */
    if (length >= 255)
    {
      err = "Zeile zu lang";
      goto err;
    }
    memset(psep, '-', len); psep += len;
    *psep++ = '+';
    *psep   = '\0';
  }

  /* Spaltenüberschriften */
  puts(sep);
  mysql_field_seek(res,0);
  fputc('|',stdout);
  für (off=0; (field = mysql_fetch_field(res)) ; off++)
  {
    printf(" %-*s|",field->max_length, field->name);
    is_num[off]= IS_NUM(field->type);
  }
  fputc('\n',stdout);
  puts(sep);

  /* Zeilen */
```

```
  while ((row = mysql_fetch_row(res)))
  {
    (void) fputs("|",stdout);
    mysql_field_seek(res,0);
    for (off=0 ; off < mysql_num_fields(res); off++)
    {
      field = mysql_fetch_field(res);
      printf(is_num[off] ? "%*s |" : " %-*s|",
             field->max_length, row[off] ? (char*) row[off] : "NULL");
    }
    (void) fputc('\n',stdout);
  }
  puts(sep);

err:
  if (is_num)
    free(is_num);
  mysql_free_result(res);

  return err;
}

void
db_do_query(MYSQL *db, const char *query, enum on_error on_error)
{
  char *err = 0;
  if (mysql_query(db, query) != 0)
    goto err;

  if (mysql_field_count(db) > 0)
  {
    if ((err = show_query(db)))
      goto err;
  }
  else if (mysql_affected_rows(db))
    printf("Betroffene Zeilen: %lld [%s]\n", mysql_affected_rows(db), query);

  return;

err:
  switch (on_error) {
  case E_okay:
    break;
  case E_warn:
    fprintf(stderr, "db_do_query fehlgeschlagen: %s [%s]\n",
        err ? err : mysql_error(db), query);
    break;
  case E_fail:
    die(db, "db_do_query fehlgeschlagen: %s [%s]",
        err ? err : mysql_error(db), query);
    break;
  }
}
```

GNUmakefile

```
# Platzieren Sie diese in Ihr mysql-Quell-Verzeichnis
m        := ../mysql-4.0

CC       := cc
CPPFLAGS := -I$m/include -D_thread_SAFE -D_REENTRANT
CFLAGS   := -g -W -Wall
LDFLAGS  := -static
LDLIBS    = $(embed_libs) -lz -lm -lcrypt

ifneq (,$(shell grep FreeBSD /COPYRIGHT 2>/dev/null))
# FreeBSD
LDFLAGS += -pThread
else
# Linux wird angenommen
LDLIBS += -lpThread
endif


# Standard-Bibliotheken

embed_libs := \
        $m/libmysqld/.libs/libmysqld.a \
        $m/isam/libnisam.a \
        $m/myisam/libmyisam.a \
        $m/heap/libheap.a \
        $m/merge/libmerge.a \
        $m/myisammrg/libmyisammrg.a


# Optional gebaute Bibliotheken

ifneq (,$(shell test -r $m/innobase/usr/libusr.a && echo "yes"))
embed_libs += \
        $m/innobase/usr/libusr.a \
        $m/innobase/odbc/libodbc.a \
        $m/innobase/srv/libsrv.a \
        $m/innobase/que/libque.a \
        $m/innobase/srv/libsrv.a \
        $m/innobase/dict/libdict.a \
```

```
        $m/innobase/ibuf/libibuf.a \
        $m/innobase/row/librow.a \
        $m/innobase/pars/libpars.a \
        $m/innobase/btr/libbtr.a \
        $m/innobase/trx/libtrx.a \
        $m/innobase/read/libread.a \
        $m/innobase/usr/libusr.a \
        $m/innobase/buf/libbuf.a \
        $m/innobase/ibuf/libibuf.a \
        $m/innobase/eval/libeval.a \
        $m/innobase/log/liblog.a \
        $m/innobase/fsp/libfsp.a \
        $m/innobase/fut/libfut.a \
        $m/innobase/fil/libfil.a \
        $m/innobase/lock/liblock.a \
        $m/innobase/mtr/libmtr.a \
        $m/innobase/page/libpage.a \
        $m/innobase/rem/librem.a \
        $m/innobase/thr/libthr.a \
        $m/innobase/com/libcom.a \
        $m/innobase/sync/libsync.a \
        $m/innobase/data/libdata.a \
        $m/innobase/mach/libmach.a \
        $m/innobase/ha/libha.a \
        $m/innobase/dyn/libdyn.a \
        $m/innobase/mem/libmem.a \
        $m/innobase/sync/libsync.a \
        $m/innobase/ut/libut.a \
        $m/innobase/os/libos.a \
        $m/innobase/ut/libut.a
endif

ifneq (,$(shell test -r $m/bdb/build_unix/libdb.a && echo "yes"))
embed_libs += $m/bdb/build_unix/libdb.a
endif


# Unterstützte Bibliotheken

embed_libs += \
        $m/mysys/libmysys.a \
        $m/strings/libmystrings.a \
        $m/dbug/libdbug.a \
        $m/regex/libregex.a


# Optional gebaute unterstützte Bibliotheken

ifneq (,$(shell test -r $m/readline/libreadline.a && echo "yes"))
embed_libs += $m/readline/libreadline.a
endif

# Das funktioniert bei einfachen Ein-Datei-Test-Programmen
sources := $(wildcard *.c)
objects := $(patsubst %c,%o,$(sources))
targets := $(basename $(sources))

all: $(targets)

clean:
        rm -f $(targets) $(objects) *.core
```

9.4.9.4. Lizensierung des eingebetteten Servers

Der MySQL-Quelltext wird von der GNU-GPL-Lizenz abgedeckt (see Anhang H, *GNU GENERAL PUBLIC LICENSE*). Eine Folge davon ist, dass jegliches Programm, das durch Linken mit `libmysqld` den MySQL-Quelltext enthält, als freie Software (unter einer mit der GPL kompatiblen Lizenz) veröffentlicht werden muss.

Wir ermutigen jeden, freie Software durch Veröffentlichung von Code unter der GPL oder einer kompatiblen Lizenz zu fördern. Für diejenigen, die dazu nicht in der Lage sind, ist eine weitere Option, den MySQL-Code von MySQL AB unter einer lockereren Lizenz zu erwerben. Wegen Details betreffs dieses Themas siehe unter Abschnitt 2.4.4, „MySQL-Lizenzpolitik“.

9.5. MySQL-C++-APIs

9.5.1. Borland C++

Sie können den MySQL-Windows-Quellcode mit Borlund C++ 5.02 kompilieren. (Der Windows-Quellcode beinhaltet nur Projekte für Microsoft VC++, für Borland C++ müssen Sie die Projektdateien selbst erstellen).

Ein bekanntes Problem bei Borland C++ ist, dass es eine andere Strukturanordnung benutzt als VC++. Das bedeutet, dass Sie Probleme bekommen, wenn Sie versuchen, die vorgabemäßigen `libmysql.dll`-Bibliotheken (die mit VC++ kompiliert wurden) mit Borland C++ zu verwenden. Sie können eins der folgenden Dinge tun, um dieses Problem zu vermeiden:

- Sie können statische MySQL-Bibliotheken für Borland C++ verwenden, die Sie unter

http://www.mysql.com/downloads/os-win32.html finden.

- Rufen Sie `mysql_init()` nur mit `NULL` als Argument auf, kein vorher zugewiesenes (prä-alloziertes) MySQL-Strukt.

9.6. MySQL Java Connectivity (JDBC)

Es gibt zwei unterstützte JDBC-Treiber für MySQL:

- `MySQL Connector/J` von MySQL AB, implementiert in 100% nativem Java. Dieses Produkt hieß ursprünglich `mm.mysql`-Treiber. Sie können `MySQL Connector/J` von http://www.mysql.com/products/connector-j/ herunterladen.
- Der Resin JDBC-Treiber, den Sie auf http://www.caucho.com/projects/jdbc-mysql/index.xtp finden.

9.7. MySQL-Python-APIs

`MySQLdb` bietet MySQL-Unterstützung für Python, konform mit Python DB API Version 2.0. `MySQLdb` finden Sie hier: http://sourceforge.net/projects/mysql-python/.

9.8. MySQL-Tcl-APIs

MySQLtcl ist eine einfache API zum Zugriff auf einen MySQL-Datenbankserver von der Tcl-Programmiersprache aus. Sie finden die API hier: http://www.xdobry.de/mysqltcl/.

9.9. MySQL-Eiffel-Wrapper

Eiffel MySQL ist eine Schnittstelle zum MySQL-Datenbankserver, die die von Michael Ravits geschriebene Programmiersprache benutzt. Sie finden die Schnittstelle hier: http://efsa.sourceforge.net/archive/ravits/mysql.htm.

Kapitel 10. MySQL erweitern

10.1. Hinzufügen neuer Funktionen zu MySQL

Es gibt zwei Möglichkeiten, MySQL neue Funktionen hinzuzufügen:

- Sie können die Funktion über die benutzerdefinierbare Funktions- (UDF-) Schnittstelle hinzufügen. Benutzerdefinierbare Funktionen werden dynamisch mittels `CREATE FUNCTION` und `DROP FUNCTION`-Statements hinzugefügt bzw. gelöscht. See Abschnitt 10.1.1, „`CREATE FUNCTION / DROP FUNCTION`-Syntax“.
- Sie können die Funktion als native (eingebaute) MySQL-Funktion hinzufügen. Native Funktionen werden in den `mysqld`-Server kompiliert und stehen dann dauerhaft zur Verfügung.

Jede Methode hat Vorteile und Nachteile:

- Wenn Sie eine benutzerdefinierte Funktion schreiben, müssen Sie die Objekt-Datei zusätzlich zum Server selbst installieren. Wenn Sie Ihre Funktion in den Server einkompilieren, brauchen Sie das nicht zu tun.
- Sie können der binären MySQL-Distribution benutzerdefinierte Funktionen hinzufügen. Native Funktionen erfordern, dass Sie eine Quelldistribution verändern.
- Wenn Sie Ihre MySQL-Distribution aktualisieren, können Sie weiterhin Ihre vorher installierten benutzerdefinierten Funktionen benutzen. Bei nativen Funktionen müssen Sie Ihre Änderungen jedes Mal wiederholen, wenn Sie aktualisieren.

Gleich welche Methode Sie zum Hinzufügen neuer Funktionen verwenden, können Sie diese genau wie die nativen Funktionen, z. B. `ABS()` oder `SOUNDEX()`, benutzen.

10.1.1. `CREATE FUNCTION / DROP FUNCTION`-Syntax

```
CREATE [AGGREGATE] FUNCTION funktion RETURNS {STRING|REAL|INTEGER}
       SONAME gemeinsame_bibliothek

DROP FUNCTION funktion
```

Eine benutzerdefinierte Funktion (UDF) ist eine Möglichkeit, MySQL durch eine neue Funktion zu erweitern, die wie die nativen (eingebauten) MySQL-Funktionen, z. B. `ABS()` und `CONCAT()`, funktioniert.

`AGGREGATE` ist eine neue Option für MySQL-Version 3.23. Eine `AGGREGATE`-Funktion funktioniert genau wie eine native MySQL- `GROUP`-Funktion wie `SUM` oder `COUNT()`.

`CREATE FUNCTION` speichert den Funktionnamen, -typ und die gemeinsam genutzte Bibliothek in der `mysql.func`-Systemtabelle. Sie benötigen die **insert**- und **delete**-Berechtigungen für die `mysql`-Datenbank, um Funktionen zu erzeugen und zu löschen.

Alle aktiven Funktionen werden jedes Mal wieder geladen, wenn der Server startet, es sei denn, Sie starten ihn mit der `--skip-grant-tables`-Option. In diesem Fall wird die UDF-Initialisierung übersprungen, so dass UDFs nicht verfügbar sind. (Eine aktive Funktion ist eine, die mit `CREATE FUNCTION` geladen und nicht mit `DROP FUNCTION` entfernt wurde.)

Wegen weiterer Anleitungen zum Schreiben benutzerdefinierte Funktionen siehe Abschnitt 10.1, „Hinzufügen neuer Funktionen zu MySQL“. Damit der UDF-Mechanismus funktioniert, müssen Funktionen in C oder C++ geschrieben sein. Ihr Betriebssystem muss dynamisches Laden unterstützen und Sie müssen `mysqld` dynamisch (nicht statisch) kompiliert haben.

Beachten Sie, dass Sie für das Funktionieren von `AGGREGATE` eine `mysql.func`-Tabelle benötigen, die die Spalte `typ` enthält. Wenn das nicht der Fall ist, sollten Sie das Skript `mysql_fix_privilege_tables` laufen lassen, um diesen Mangel zu beheben.

10.1.2. Hinzufügen einer neuen benutzerdefinierten Funktion

Damit der UDF-Mechanismus funktioniert, müssen Funktionen in C oder C++ geschrieben sein. Ihr Betriebssystem muss dynamisches Laden unterstützen und Sie müssen `mysqld` dynamisch (nicht statisch) kompiliert haben. Die MySQL-Quelldistribution enthält eine Datei `sql/udf_example.cc`, die 5 neue Funktionen definiert. Sehen Sie in dieser Datei nach, wie die UDF-Aufruf-Konventionen funktionieren.

Damit `mysqld` UDF-Funktionen benutzen kann, sollten Sie MySQL mit `--with-mysqld-ldflags=-rdynamic` konfigurieren. Der Grund liegt darin, dass Sie auf vielen Plattformen (inklusive Linux) eine dynamische Bibliothek (mit

`dlopen()`) von einem statisch gelinkten Programm laden können, was Sie erhalten würden, wenn Sie `--with-mysqld-ldflags=-all-static` benutzen. Wenn Sie eine UDF benutzen wollen, die auf Symbole von `mysqld` zugreifen muss (wie das `methaPhone`-Beispiel in `sql/udf_example.cc`, das `default_charset_info` benutzt), müssen Sie das Programm mit `-rdynamic` benutzen (siehe `man dlopen`).

Für jede Funktion, die Sie in SQL-Statements benutzen wollen, sollten Sie die entsprechenden C- (oder C++-) Funktionen benutzen. In den unten stehenden Ausführungen wird ``xxx" als Beispiel-Funktionsname benutzt. Um zwischen SQL- und C-/C++-Benutzung zu unterscheiden, kennzeichnet `XXX()` (Großschreibung) einen SQL-Funktionsaufruf und `xxx()` (Kleinschreibung) einen C-/C++-Funktionsaufruf.

The C-/C++-Funktionen, die Sie für die Implementierung der Schnittstelle für `XXX()` schreiben, sind:

- `xxx()` (required)

  Die Hauptfunktion. Hier wird das Funktionsergebnis berechnet. Der Zusammenhang zwischen dem SQL-Typ und dem Rückgabe-Typ Ihrer C-/C++-Funktion ist unten dargestellt:

SQL-Typ	C-/C++-Typ
`STRING`	`char *`
`INTEGER`	`long long`
`REAL`	`double`

- `xxx_init()` (optional)

  Die Initialisierungsfunktion für `xxx()`. Sie kann für folgendes benutzt werden:

  - Um die Anzahl von Argumenten für `XXX()` zu prüfen.
  - Um zu prüfen, ob die Argumente vom erforderlichen Typ sind oder, alternativ, MySQL mitzuteilen, den Argumenttyp zu erzwingen, den Sie beim Aufruf der Hauptfunktion brauchen.
  - Um jeglichen Speicher zuzuweisen, der von der Hauptfunktion benötigt wird.
  - Um die maximale Länge des Ergebnisses anzugeben.
  - Um (für `REAL`-Funktionen) die maximale Anzahl von Dezimalstellen anzugeben.
  - Um festzulegen, ob das Ergebnis `NULL` sein darf oder nicht.

- `xxx_deinit()` (optional)

  Die Deinitialisierungsfunktion für `xxx()`. Sie sollte jeglichen Speicher freigeben (deallozieren), der durch die Initialisierungsfunktion zugewiesen wurde.

Wenn ein SQL-Statement `XXX()` aufruft, ruft MySQL die Initialisierungsfunktion `xxx_init()` auf, damit diese die notwendige Einrichtung vornehmen kann wie Argumente prüfen oder Speicherzuweisung. Wenn `xxx_init()` einen Fehler zurückgibt, wird das SQL-Statement mit einer Fehlermeldung abgebrochen, die Haupt- und Deinitialisierungsfunktionen werden nicht aufgerufen. Ansonsten wird die Hauptfunktion `xxx()` für jede Zeile aufgerufen. Nachdem alle Zeilen abgearbeitet sind, wird die Deinitialisierungsfunktion `xxx_deinit()` aufgerufen, damit sie die erforderlichen Aufräumarbeiten ausführen kann.

Alle Funktionen müssen Thread-sicher sein (nicht nur die Hauptfunktion, sondern auch die Initialisierungs- und Deinitialisierungsfunktionen). Das heißt, dass Sie keinerlei globale oder statische Variablen zuweisen dürfen, die sich ändern! Wenn Sie Speicher brauchen, sollten Sie ihn in `xxx_init()` zuweisen und in `xxx_deinit()` freigeben.

10.1.2.1. UDF-Aufruf-Sequenzen

Die Hauptfunktion sollte wie unten dargestellt deklariert werden. Beachten Sie, dass sich der Rückgabetyp und der Parameter unterscheiden, abhängig davon, wie Sie die SQL-Funktion `XXX()` deklarieren, damit sie `STRING`, `INTEGER` oder `REAL` im `CREATE FUNCTION`-Statement zurückgibt:

Bei `STRING`-Funktionen:

```
char *xxx(UDF_INIT *initid, UDF_ARGS *args,
          char *result, unsigned long *length,
          char *is_null, char *error);
```

Bei `INTEGER`-Funktionen:

```
long long xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Bei `REAL`-Funktionen:

```
double xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Die Initialisierungs- und Deinitialisierungsfunktionen werden wie folgt deklariert:

```
my_bool xxx_init(UDF_INIT *initid, UDF_ARGS *args, char *message);

void xxx_deinit(UDF_INIT *initid);
```

Der `initid`-Parameter wird an alle drei Funktionen übergeben. Er zeigt auf eine `UDF_INIT`-Struktur, die benutzt wird, um Informationen zwischen den Funktionen zu übermitteln. Die `UDF_INIT`-Strukturmitglieder sind unten aufgelistet. Die Initialisierungsfunktion sollte alle Mitglieder ausfüllen, die sie ändern will. (Um für ein Mitglied den Vorgabewert zu verwenden, lassen Sie es unverändert.)

- `my_bool maybe_null`

  `xxx_init()` sollte `maybe_null` auf `1` setzen, wenn `xxx()` `NULL` zurückgeben kann. Der Vorgabewert ist `1`, wenn irgend eins der Argumente als `maybe_null` deklariert ist.

- `unsigned int Dezimalstellen`

  Anzahl von Dezimalstellen. Der Vorgabewert ist die maximale Anzahl von Dezimalstellen in den Argumenten, die an die Hauptfunktion übergeben werden. (Wenn der Funktion beispielsweise die Argumente `1.34`, `1.345` und `1.3` übergeben werden, wäre der Vorgabewert 3, weil `1.345` 3 Dezimalstellen hat.

- `unsigned int max_length`

  Die maximale Länge des Zeichenkettenergebnisses. Der Vorgabewert ist unterschiedlich, abhängig vom Ergebnistyp der Funktion. Bei Zeichenketten-Funktionen ist die Vorgabe die Länge des längsten Arguments. Bei Ganzzahl-Funktionen ist die Vorgabe 21 Ziffern. Bei REAL-Funktionen ist die Vorgabe 13 plus die Anzahl von Dezimalstellen, die von `initid->Dezimalstellen` angezeigt werden. (Bei numerischen Funktionen enthält die Länge jedes Vorzeichen- oder Dezimalpunkt-Zeichen.)

  Wenn Sie einen Blob zurückgeben wollen, können Sie diesen auf 65 KB oder 16MB setzen. Der Speicher wird nicht zugewiesen, aber dazu verwendet, um zu entscheiden, welcher Spaltentyp benutzt werden soll, falls es notwendig werden sollte, Daten temporär zu speichern.

- `char *ptr`

  Ein Zeiger, den die Funktion für eigene Zwecke verwenden kann. Beispielsweise können Funktionen `initid->ptr` benutzen, um Informationen über den zugewiesenen Speicher zwischen den Funktionen zu kommunizieren. Beispiel, um in `xxx_init()` Speicher zuzuweisen und ihn diesem Zeiger zuzuordnen:

  ```
  initid->ptr = allocated_memory;
  ```

  In `xxx()` und `xxx_deinit()` verweisen Sie auf `initid->ptr`, um Speicher zu verwenden oder freizugeben.

10.1.2.2. Verarbeitung von Argumenten

Der `args`-Parameter zeigt auf eine `UDF_ARGS`-Struktur, die unten aufgelistete Mitglieder hat:

- `unsigned int arg_count`

  Die Anzahl von Argumenten. Prüfen Sie diesen Wert in der Initialisierungsfunktion, wenn Sie wollen, dass Ihre Funktion mit einer bestimmten Anzahl von Argumenten aufgerufen wird. Beispiel:

  ```
  if (args->arg_count != 2)
  {
      strcpy(message,"XXX() benoetigt zwei Argumente");
      return 1;
  ```

```
}
```

- `enum Item_result *arg_type`

  Die Typen für jedes Argument. Die möglichen Typenwerte sind `STRING_RESULT`, `INT_RESULT` und `REAL_RESULT`.

  Um sicherzustellen, dass die Argumente vom angegebenen Typ sind und einen Fehler zurückgeben, falls nicht, prüfen Sie das `arg_type`-Array in der Initialisierungsfunktion. Beispiel:

```
if (args->arg_type[0] != STRING_RESULT ||
    args->arg_type[1] != INT_RESULT)
{
    strcpy(message,"XXX() erfordert eine Zeichenkette und eine Ganzzahl");
    return 1;
}
```

  Als Alternative dazu, dass Ihre Funktionsargumente von bestimmten Typen sein müssen, können Sie die Initialisierungsfunktion benutzen, um die `arg_type`-Elemente auf die Typen zu setzen, die Sie wollen. Das veranlasst MySQL, die Typen der Argumente bei jedem Aufruf von `xxx()` zu erzwingen. Um beispielsweise zu erzwingen, dass die ersten zwei Argumente Zeichenkette und Ganzzahl sind, geben Sie in `xxx_init()` folgendes ein:

```
args->arg_type[0] = STRING_RESULT;
args->arg_type[1] = INT_RESULT;
```

- `char **args`

  `args->args` kommuniziert der Initialisierungsfunktion Informationen über die allgemeine Natur der Argumente, mit der Ihre Funktion aufgerufen wurde. Bei einem Konstanten-Argument `i` zeigt `args->args[i]` auf den Argumentwert. (Siehe unten wegen Anleitungen, wie auf diesen Wert korrekt zugegriffen wird.) Bei einem Nicht-Konstanten-Argument ist `args->args[i]` `0`. Ein Konstanten-Argument ist ein Ausdruck, der nur Konstanten wie `3` oder `4*7-2` oder `SIN(3.14)` benutzt. Ein Nicht-Konstanten-Argument ist ein Ausdruck, der auf Werte verweist, die sich von Zeile zu Zeile ändern können, wie Spaltennamen oder Funktionen, die mit Nicht-Konstanten-Argumenten aufgerufen werden.

  Bei jedem Aufruf der Hauptfunktion enthält `args->args` die tatsächlichen Argumente, die für die Zeile übergeben werden, die momentan verarbeitet wird.

  Funktionen können auf ein Argument `i` wie folgt verweisen:

  - Ein Argument des Typs `STRING_RESULT` wird als ein Zeichenkettenzeiger plus einer Länge angegeben, um die Handhabung von Binärdaten oder Daten beliebiger Länge zu erlauben. Die Zeichenketten-Inhalte sind als `args->args[i]` und die Zeichenkettenlänge als `args->lengths[i]` verfügbar. Sie sollten nicht davon ausgehen, dass Zeichenketten null-terminiert sind.
  - Bei einem Argument des Typs `INT_RESULT` müssen Sie `args->args[i]` zu einem `long long`-Wert machen (cast):

```
long long int_val;
int_val = *((long long*) args->args[i]);
```

  - Bei einem Argument des Typs `REAL_RESULT` müssen Sie `args->args[i]` zu einem `double`-Wert machen (cast):

```
double    real_val;
real_val = *((double*) args->args[i]);
```

- `unsigned long *lengths`

  Bei der Initialisierungsfunktion gibt das `lengths`-Array die maximale Zeichenkettenlänge jedes Arguments an. Bei jedem Aufruf der Hauptfunktion enthält `lengths` die tatsächlichen Längen jeglicher Zeichenketten-Argumente, die für die momentan verarbeitete Zeile übergeben werden. Bei Argumenten des Typs `INT_RESULT` oder `REAL_RESULT` enthält `lengths` immer noch die maximale Länge des Arguments (wie bei der Initialisierungsfunktion).

10.1.2.3. Rückgabewerte und Fehlerbehandlung

Die Initialisierungsfunktion sollte `0` zurückgeben, wenn kein Fehler auftrat, ansonsten `1`. Wenn ein Fehler auftritt, sollte `xxx_init()` eine null-terminierte Fehlermeldung im `message`-Parameter enthalten. Die Meldung wird an den Client übergeben. Der Meldungspuffer ist `MYSQL_ERRMSG_SIZE` Zeichen lang, aber Sie sollten versuchen, die Meldung kleiner als 80 Zeichen zu halten, damit sie auf die Anzeigebreite eines Standard-Terminals passt.

Der Rückgabewert der Hauptfunktion `xxx()` ist der Funktionswert, bei `long long`- und `double`-Funktionen. Eine Zeichenkettenfunktion sollte einen Zeiger auf das Ergebnis und die Länge der Zeichenkette in den `length`-Argumenten zurückgeben.

Setzen Sie diese auf die Inhalte und Länge des Rückgabewerts. Beispiel:

```
memcpy(result, "ergebnis_zeichenkette", 13);
*length = 13;
```

Der `result`-Puffer, der an die Berechnungsfunktionen übergeben wird, ist 255 Byte Groß. Wenn Ihr Ergebnis dort hinein passt, müssen Sie sich um die Speicherzuweisung für Ergebnisse nicht kümmern.

Wenn Ihre Zeichenketten-Funktion eine Zeichenkette zurückgeben muss, die länger als 255 Bytes ist, müssen Sie den Platz dafür mit `malloc()` in Ihrer `xxx_init()`-Funktion oder Ihrer `xxx()`-Funktion zuweisen und in Ihrer `xxx_deinit()`-Funktion freigeben. Sie können den zugewiesenen Speicher im `ptr`-Slot in der `UDF_INIT`-Struktur für erneute Benutzung durch zukünftige `xxx()`-Aufrufe speichern. See Abschnitt 10.1.2.1, „UDF-Aufruf-Sequenzen“.

Um einen Rückgabewert von `NULL` in der Hauptfunktion anzuzeigen, setzen Sie `is_null` auf `1`:

```
*is_null = 1;
```

Um eine Fehlerrückgabe in der Hauptfunktion anzuzeigen, setzen Sie den `error`-Parameter auf `1`:

```
*error = 1;
```

Wenn `xxx()` `*error` für beliebige Zeilen auf `1` setzt, ist der Funktionswert der aktuellen Zeile `NULL`, was auch für nachfolgende Zeilen gilt, die von dem Statement verarbeitet werden, in dem `XXX()` aufgerufen wurde. (`xxx()` wird für nachfolgende Zeilen nicht einmal aufgerufen.) **HINWEIS:** In MySQL-Versionen vor 3.22.10 sollten Sie sowohl `*error` als auch und `*is_null` setzen:

```
*error = 1;
*is_null = 1;
```

10.1.2.4. Kompilieren und Installieren benutzerdefinierter Funktionen

Dateien, die UDFs implementieren, müssen auf dem Host kompiliert und installiert werden, auf dem der Server läuft. Dieser Prozess wird unten am Beispiel der UDF-Datei `udf_example.cc` beschrieben, die in der MySQL-Quelldistribution enthalten ist. Diese Datei enthält folgende Funktionen:

- `metaphon()` gibt eine metaphon-Zeichenkette des Zeichenkettenarguments zurück. Das ist etwas wie eine Soundex-Zeichenkette, nur etwas besser für englisch angepasst.
- `myfunc_double()` gibt die Summe der ASCII-Werte der Zeichen in ihren Argumenten zurück, geteilt durch die Summe der Längen ihrer Argumente.
- `myfunc_int()` gibt die Summe der Längen ihrer Argumente zurück.
- `sequence([const int])` gibt eine Sequenz zurück, die mit der angegebenen Zahl startet oder mit 1, wenn keine Zahl angegeben wurde.
- `lookup()` gibt die IP-Nummer für einen Hostnamen zurück.
- `reverse_lookup()` gibt den Hostnamen für eine IP-Nummer zurück. Die Funktion kann mit einer Zeichenkette `"xxx.xxx.xxx.xxx"` oder mit vier Zahlen aufgerufen werden.

Eine dynamisch ladbare Datei sollte als gemeinsam nutzbare Objektdatei kompiliert werden, etwa mit folgendem Befehl:

```
shell> gcc -shared -o udf_example.so myfunc.cc
```

Die korrekten Kompiler-Optionen für Ihr System finden Sie leicht heraus, wenn Sie diesen Befehl im `sql`-Verzeichnis Ihres MySQL-Quellbaums laufen lassen:

```
shell> make udf_example.o
```

Sie sollten einen Kompilierbefehl laufen lassen, der dem ähnelt, was `make` anzeigt, ausser dass Sie die `-c`-Option kurz vor dem Zeilenende entfernen und `-o udf_example.so` am Zeilenende hinzufügen sollten. (Auf manchen Systemen können Sie `-c` im Befehl lassen.)

Wenn Sie ein gemeinsam genutztes Objekt kompiliert haben, das UDFs enthält, müssen Sie es danach installieren und MySQL darüber informieren. Wenn Sie ein gemeinsam genutztes Objekt von `udf_example.cc` kompilieren, wird eine Datei etwa mit dem Namen `udf_example.so` erzeugt (der exakte Name variiert von Plattform zu Plattform). Kopieren Sie diese Datei in ein Verzeichnis, das von `ld` durchsucht wird, wie `/usr/lib`. Auf vielen Systemen können Sie die `LD_LIBRARY`- oder `LD_LIBRARY_PATH`-Umgebungsvariable so setzen, dass sie auf das Verzeichnis zeigt, wo Sie Ihre UDF-Funktionsdateien haben. Das `dlopen`-Handbuch sagt Ihnen, welche Variable Sie auf Ihrem System setzen sollten. Sie sollten diese auf `mysql.server` oder `safe_mysqld` setzen und `mysqld` neu starten.

Nachdem die Bibliothek installiert ist, unterrichten Sie `mysqld` über die neuen Funktionen mit diesen Befehlen:

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_double RETURNS REAL SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_int RETURNS INTEGER SONAME "udf_example.so";
mysql> CREATE FUNCTION lookup RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION reverse_lookup RETURNS STRING SONAME "udf_example.so";
mysql> CREATE AGGREGATE FUNCTION avgcost RETURNS REAL SONAME "udf_example.so";
```

Funktionen können mit `DROP FUNCTION` gelöscht werden:

```
mysql> DROP FUNCTION metaphon;
mysql> DROP FUNCTION myfunc_double;
mysql> DROP FUNCTION myfunc_int;
mysql> DROP FUNCTION lookup;
mysql> DROP FUNCTION reverse_lookup;
mysql> DROP FUNCTION avgcost;
```

Die `CREATE FUNCTION`- und `DROP FUNCTION`-Statements aktualisieren die Systemtabelle `func` in der `mysql`-Datenbank. Der Funktionsname, -typ und gemeinsam genutzte Bibliothek werden in der Tabelle gespeichert. Sie benötigen die **insert**- und **delete**-Berechtigungen für die `mysql`-Datenbank, um Funktionen zu erzeugen und zu löschen.

Sie sollten `CREATE FUNCTION` nicht benutzen, um eine Funktion hinzuzufügen, die bereits erzeugt wurde. Wenn Sie eine Funktion erneut installieren wollen, sollten Sie sie zuerst mit `DROP FUNCTION` entfernen und dann mit `CREATE FUNCTION` erneut installieren. Sie müssen so etwas zum Beispiel tun, wenn Sie eine neue Version Ihrer Funktion kompilieren, damit `mysqld` die neue Version erhält. Ansonsten würde der Server mit der alten Version weitermachen.

Aktive Funktionen werden jedes Mal neu geladen, wenn der Server startet, es sei denn, Sie starten `mysqld` mit der `--skip-grant-tables`-Option. In diesem Fall wird die UDF-Initialisierung übersprungen und UDFs sind nicht verfügbar. (Eine aktive Funktion ist eine, die mit `CREATE FUNCTION` geladen und nicht mit `DROP FUNCTION` entfernt wurde.)

10.1.3. Hinzufügen einer neuen nativen Function

Die Prozedur zum Hinzufügen einer neuen nativen Funktion wird hier beschrieben. Beachten Sie, dass Sie einer Binärdistribution keine nativen Funktionen hinzufügen können, weil die Prozedur die Änderung des MySQL-Quelltextes beinhaltet. Sie müssen MySQL selbst aus einer Quelldistribution kompilieren. Beachten Sie auch, dass Sie die Prozedur wiederholen müssen, wenn Sie auf eine andere Version von MySQL aktualisieren (beispielsweise wenn eine neue Version herauskommt).

Um eine neue native MySQL-Funktion hinzuzufügen, gehen Sie wie folgt vor:

1. Fügen Sie `lex.h` eine neue Zeile hinzu, die den Funktionsnamen im `sql_functions[]`-Array definiert.
2. Wenn der Funktionsprototyp einfach ist (nur keins, eins, zwei oder drei Argumente entgegennimmt), sollten Sie in lex.h SYM(FUNC_ARG#) angeben (wobei # die Anzahl von Argumenten ist), als zweites Argument im `sql_functions[]`-Array, und eine Funktion hinzufügen, die ein Funktionsobjekt in `item_create.cc` erzeugt. Sehen Sie sich als Beispiel hierfür `"ABS"` und `create_funcs_abs()` an.

   Wenn der Funktionsprototyp kompliziert ist (zum Beispiel eine variable Anzahl von Argumenten entgegennimmt), sollten Sie zwei Zeile zu `sql_yacc.yy` hinzufügen. Eine gibt das Präprozessorsymbol an, das `yacc` definieren soll (das sollte am Anfang der Datei stehen). Definieren Sie dann die Funktionsparameter und fügen Sie ein ``item" mit diesen Parametern zur `simple_expression`-Parsing-Regel hinzu. Sehen Sie sich als Beispiel alle Vorkommen von `ATAN` in `sql_yacc.yy` an, um zu sehen, wie das gemacht wird.
3. Deklarieren Sie in `item_func.h` eine Klasse, die von `Item_num_func` oder `Item_str_func` erbt, je nachdem, ob Ihre Funktion eine Zahl oder eine Zeichenkette zurückgibt.
4. Fügen Sie in `item_func.cc` eine der folgenden Deklarationen hinzu, je nachdem, ob Sie eine numerische oder eine Zeichenketten-Funktion definieren:

   ```
   double   Item_func_newname::val()
   longlong Item_func_newname::val_int()
   String  *Item_func_newname::Str(String *str)
   ```

Wenn Sie Ihr Objekt von irgend einem der Standard-Items erben (wie von `Item_num_func`, müssen Sie wahrscheinlich eine der oben genannten Funktionen definieren und das Elternobjekt sich um die anderen Funktionen kümmern lassen. Beispielsweise definiert die `Item_str_func`-Klasse eine `val()`-Funktion, die `atof()` auf dem Wert ausführt, der von `::str()` zurückgegeben wurde.

5. Sie müssen wahrscheinlich auch die folgende Objektfunktion definieren:

```
void Item_func_newname::fix_length_und_dec()
```

Diese Funktion sollte zumindest `max_length` basierend auf den angegebenen Argumenten berechnen. `max_length` ist die maximale Anzahl von Zeichen, die die Funktion zurückgeben kann. Diese Funktion sollte auch `maybe_null = 0` setzen, wenn die Hauptfunktion keinen `NULL`-Wert zurückgeben kann. Die Funktion kann prüfen, ob irgend eins der Funktionsargumente `NULL` zurückgeben kann, indem die Argumente der `maybe_null`-Variable geprüft werden. Sehen Sie sich als typisches Beispiel, wie das gemacht wird, `Item_func_mod::fix_length_and_dec` an.

Alle Funktionen müssen Thread-sicher sein (mit anderen Worten: Benutzen Sie keine globalen oder statischen Variablen in den Funktionen, ohne sie mit mutexes zu schützen).

Wenn Sie von `::val()`, `::val_int()` oder `::str()` `NULL` zurückgeben wollen, sollten Sie `null_value` auf 1 setzen und 0 zurückgeben.

Bei `::str()`-Objektfunktionen gibt es einige zusätzliche Überlegungen, auf die man achten sollte:

- Das `String *str`-Argument stellt einen Zeichenketten-Puffer zur Verfügung, der benutzt werden kann, um das Ergebnis zu speichern. (Weitere Informationen über den `String`-Typ finden Sie durch einen Blick in die `sql_string.h`-Datei.)
- Die `::str()`-Funktion sollte die Zeichenkette zurückgeben, die das Ergebnis enthält, oder `(char*) 0`, wenn das Ergebnis `NULL` ist.
- Alle aktuellen Zeichenketten-Funktionen versuchen, die Zuweisung jeglichen Speichers zu vermeiden, ausser wenn das absolut notwendig ist!

10.2. Hinzufügen neuer Prozeduren zu MySQL

In MySQL können Sie eine Prozedur in C++ definieren, die auf Daten in einer Anfrage zugreifen und diese ändern kann, bevor sie an den Client geschickt werden. Die Änderung kann Zeile für Zeile oder auf `GROUP BY`-Ebene geschehen.

Wir haben eine Beispiel-Prozedur in MySQL-Version 3.23 erzeugt, um zu zeigen, was getan werden kann.

Zusätzlich empfehlen wir, dass Sie einen Blick auf 'mylua' werfen, das Sie im Contrib-Verzeichnis finden. Hiermit können Sie die LUA-Sprache benutzen, um eine Prozedur zur Laufzeit in `mysqld` zu laden.

10.2.1. PROCEDURE ANALYSE

`analyse([max Elemente,[max memory]])`

Diese Prozedur ist in `sql/sql_analyse.cc` definiert. Sie untersucht das Ergebnis Ihrer Anfrage und gibt eine Analyse des Ergebnisses zurück:

- `max elements` (Vorgabe 256) ist die maximale Anzahl unterschiedlicher Werte, die `analyse` pro Spalte findet. Dieses wird von `analyse` benutzt, um zu prüfen, ob der optimale Spaltentyp vom Typ `ENUM` sein sollte.
- `max memory` (Vorgabe 8.192) ist der maximale Speicher, den `analyse` pro Spalte zuweisen sollte, wenn Sie versuchen, alle unterschiedlichen (distinct) Werte zu finden.

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

10.2.2. Eine Prozedur schreiben

Im Moment ist die einzige Dokumentation hierfür der Quelltext.

Sie finden alle Informationen über Prozeduren, wenn Sie folgende Dateien untersuchen:

- `sql/sql_analyse.cc`
- `sql/procedure.h`
- `sql/procedure.cc`
- `sql/sql_select.cc`

10.3. MySQL-Interna

Dieses Kapitel beschreibt viele Dinge, die Sie wissen müssen, wenn Sie am MySQL-Code arbeiten. Wenn Sie an der MySQL-Entwicklung mitarbeiten wollen, Zugriff auf den messerscharfen Code von Zwischenversionen haben wollen, oder einfach nur über die Entwicklung auf dem Laufenden bleiben wollen, folgen Sie den Anweisungen unter See Abschnitt 3.3, „Installation der Quelldistribution“. Wenn Sie an MySQL-Interna interessiert sind, sollten Sie auch `<internals@lists.mysql.com>` abonnieren. Das ist eine Liste mit relativ geringem Verkehr, verglichen mit `<mysql@lists.mysql.com>`.

10.3.1. MySQL-Thread

Der MySQL-Server erzeugt folgenden Thread:

- Der TCP/IP-Verbindungs-Thread erledigt alle Verbindungsanfragen und erzeugt einen neuen dedizierten Thread, um die Verarbeitung von Authentifizierung und SQL-Anfragen für jede Verbindung zu handhaben.
- Unter Windows NT gibt es einen Named-Pipe-Handler-Thread, der dasselbe tut wie der TCP/IP-Verbindungs-Thread, auf Named-Pipe-Verbindungsanforderungen.
- Der Signal-Thread handhabt alle Signale. Dieser Thread handhabt normalerweise auch Alarme und Aufrufe von `process_alarm()`, um Zeitüberschreitungen auf Verbindungen zu erzwingen, die zu lange im Leerlauf waren.
- Wenn `mysqld` mit `-DUSE_ALARM_THREAD` kompiliert wird, wird ein dedizierter Thread erzeugt, der Alarme handhabt. Das ist nur nützlich auf manchen Systemen, auf denen es Probleme mit `sigwait()` gibt, oder wenn man den `thr_alarm()`-Code in seiner Applikation ohne einen dedizierten Signal-Handhabungs-Thread benutzen will.
- Wenn man die `--flush_time=#`-Option benutzt, wird ein dedizierter Thread erzeugt, der alle Tabellen im angegebenen Intervall auf Platte zurückschreibt.
- Jede Verbindung hat ihren eigenen Thread.
- Jede unterschiedliche Tabelle, auf der man `INSERT DELAYED` benutzt, erhält ihren eigenen Thread.
- Wenn Sie `--master-host` benutzen, wird ein Slave-Replikations-Thread gestartet, der Aktualisierungen vom Master liest und anwendet.

`mysqladmin processlist` zeigt nur die Verbindungs-, `INSERT DELAYED`- und Replikations-Threads.

10.3.2. MySQL-Test-Suite

Bis vor Kurzem basierte unsere vollumfängliche Haupt-Test-Suite auf proprietären Kundendaten und war deshalb nicht öffentlich verfügbar. Der einzige öffentlich verfügbare Teil unseres Testprozesses bestand aus dem `Crash-me`-Test, einem Perl-DBI/DBD-Benchmark, der im `sql-bench`-Verzeichnis liegt, und verschiedenen Tests im `tests`-Verzeichnis. Das Fehlen einer standardisierten, öffentlich verfügbaren Test-Suite machte es unseren Benutzern und auch Entwicklern schwer, Regressionstests auf den MySQL-Code durchzuführen. Um das Problem anzugehen, haben wir ein neues Testsystem geschaffen, das ab Version 3.23.29 den Quell- und Binärdistributionen beiliegt.

Der aktuelle Satz von Testfällen testet nicht alles in MySQL, sollte aber die offensichtlichsten Bugs im SQL-Verarbeitungscode offen legen, sowie Betriebssystem- und Bibliotheks-Probleme, und er testet recht gründlich die Replikation. Unser letztliches Ziel ist es, dass die Tests 100% des Codes abdecken. Beiträge zu unserer Test-Suite sind herzlich willkommen, besonders Tests, die die Funktionalität untersuchen, die für Ihr System kritisch ist, weil das sicherstellt, dass alle zukünftigen MySQL-Releases mit Ihren Applikationen funktionieren.

10.3.2.1. Die MySQL-Test-Suite laufen lassen

Das Testsystem besteht aus einem Test-Sprachinterpreter (`mysqltest`), einem Shell-Skript, um alle Tests laufen zu lassen tests(`mysql-test-run`), den eigentlichen Testfällen, die in einer speziellen Testsprache geschrieben sind, und ihren erwarteten Ergebnissen. Um die Test-Suite nach dem Bauen auf Ihrem System laufen zu lassen, geben Sie `make test` oder `mysql-test/mysql-test-run` von der Wurzel der Quellinstallation aus ein. Wenn Sie eine Binärdistribution installiert haben,

wechseln Sie (`cd`) zur Wurzel der Installation (zum Beispiel `/usr/local/mysql`) und geben `scripts/mysql-test-run` ein. Alle Tests sollten erfolgreich durchlaufen. Wenn nicht, sollten Sie versuchen, den Grund herauszufinden, und das Problem zu berichten, wenn es ein Bug in MySQL ist. See Abschnitt 10.3.2.3, „Bugs in der MySQL-Test-Suite berichten“.

Wenn eine Kopie von `mysqld` auf Ihrer Maschine läuft, wo Sie die Test-Suite laufen lassen wollen, müssen Sie ihn nicht anhalten, solange er nicht die Ports `9306` und `9307` benutzt. Wenn einer dieser Ports belegt ist, sollten Sie `mysql-test-run` editieren und die Werte des Master- und / oder Slave-Ports auf verfügbare Ports ändern.

Sie können einen einzelnen Testfall mit `mysql-test/mysql-test-run test_name` laufen lassen.

Wenn ein Test fehlschlägt, sollten Sie versuchen, `mysql-test-run` mit der `--force`-Option laufen zu lassen, um zu prüfen, ob irgend ein weiterer Test fehlschlägt.

10.3.2.2. Die MySQL-Test-Suite erweitern

Sie können die `mysqltest`-Sprache benutzen, um Ihre eigenen Testfälle zu schreiben. Leider gibt es noch keine komplette Dokumentation dafür - das soll in Kürze aber der Fall sein. Sie können sich jedoch die aktuellen Testfälle ansehen und sie als Beispiel benutzen. Folgende Punkte sollen Ihnen beim Start helfen:

- Die Tests liegen in `mysql-test/t/*.test`
- Ein Testfall besteht aus `;`-begrenzten Statements und ist ähnlich der Eingabe in den `mysql`-Kommandozeilen-Client. Ein Statement ist vorgabemäßig eine Anfrage, die an den MySQL-Server geschickt werden soll, es sei denn, es wird als interner Befehl erkannt (zum Beispiel `sleep`).
- Alle Anfragen, die Ergebnisse produzieren, zum Beispiel `SELECT`, `SHOW`, `EXPLAIN` usw., müssen mit `@/pfad/zu/ergebnis/datei` beginnen. Die Datei muss die erwarteten Ergebnisse enthalten. Eine einfache Art, die Ergebnisdatei zu erzeugen, ist, `mysqltest -r < t/test-case-name.test` vom `mysql-test`-Verzeichnis aus laufen zu lassen und dann die erzeugten Ergebnisdateien zu editieren und sie - falls nötig - an die erwartete Ausgabe anzupassen. Seien Sie in diesem Fall sehr vorsichtig, keine unsichtbaren Zeichen hinzuzufügen oder zu löschen - stellen Sie sicher, dass Sie nur den Text ändern und / oder Zeilen löschen. Wenn Sie eine Zeile einfügen müssen, achten Sie darauf, dass die Felder mit einem harten Tabulator-Zeichen getrennt sind und dass es ein hartes Tabulator-Zeichen am Zeilenende gibt. Gegebenfalls sollten Sie `od -c` benutzen, um sich zu vergewissern, dass Ihr Texteditor beim Editieren nichts durcheinander gebracht hat. Wir hoffen natürlich, dass Sie die Ausgabe von `mysqltest -r` nie editieren müssen, weil das nur der Fall ist, wenn Sie einen Bug finden.
- Um mit unserer Einrichtung konsistent zu sein, sollten Sie Ihre Ergebnisdateien ins `mysql-test/r`-Verzeichnis stellen und sie `test_name.result` nennen. Wenn der Test mehr als ein Ergebnis erzeugt, sollten Sie `test_name.a.result`, `test_name.b.result` usw. verwenden.
- Wenn ein Statement einen Fehler zurückgibt, sollten Sie die Zeile vor dem Statement mit `--error fehler_nummer` kennzeichnen. Die Fehlernummer kann eine Auflistung möglicher Fehlerzahlen sein, getrennt durch `','`.
- Wenn Sie einen Replikations-Testfall schreiben, sollten Sie in die erste Zeile der Testdatei `source include/master-slave.inc;` schreiben. Um zwischen Master und Slave umzuschalten, benutzen Sie `connection master;` und `connection slave;`. Wenn Sie etwas auf einer abwechselnden Verbindung machen müssen, können Sie `connection master1;` für den Master und `connection slave1;` für den Slave eingeben.
- Wenn Sie etwas in einer Schleife ausführen müssen, können Sie zum Beispiel folgendes tun:

```
let $1=1000;
while ($1)
{
 # machen Sie Ihre Anfragen hier
 dec $1;
}
```

- Um zwischen Anfragen zu schlafen, benutzen Sie den `sleep`-Befehl. Er unterstützt Bruchteile von Sekunden, daher können Sie zum Beispiel `sleep 1.3;` ausführen, um 1,3 Sekunden zu schlafen.
- Um den Slave für Ihren Testfall mit zusätzlichen Optionen laufen zu lassen, geben Sie diese im Kommandozeilenformat in `mysql-test/t/test_name-slave.opt` ein. Für den Master geben Sie sie in `mysql-test/t/test_name-master.opt` ein.
- Wenn Sie eine Frage zur Test-Suite haben oder einen Testfall beisteuern wollen, schicken Sie eine E-Mail an `<Interna@lists.mysql.com>`. Weil die Liste keine Dateianhänge akzeptiert, sollten Sie alle relevanten Dateien per FTP an ftp://support.mysql.com/pub/mysql/Incoming schicken.

10.3.2.3. Bugs in der MySQL-Test-Suite berichten

Wenn Ihre MySQL-Version die Test-Suite nicht fehlerfrei durchläuft, sollten Sie folgendes tun:

- Schicken Sie keinen Bug-Bericht, bevor Sie so weit wie möglich herausgefunden haben, was schief ging! Benutzen Sie für den Bug-Bericht bitte das `mysqlbug`-Skript, so dass wir Informationen über Ihr System und die `MySQL`-Version erhalten. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.

- Stellen Sie sicher, dass die Ausgabe von `mysql-test-run` beiliegt, sowie alle Inhalte aller `.reject`-Dateien im `mysql-test/r`-Verzeichnis.

- Wenn ein Test in der Test-Suite fehlschlägt, prüfen Sie, ob der Test auch fehlschlägt, wenn er allein laufen gelassen wird:

  ```
  cd mysql-test
  mysql-test-run --local test-name
  ```

  Wenn das fehlschlägt, sollten Sie MySQL mit `--with-debug` konfigurieren und `mysql-test-run` mit der `--debug`-Option laufen lassen. Wenn auch das fehlschlägt, schicken Sie dei Trace-Datei `var/tmp/master.trace` an ftp://support.mysql.com/pub/mysql/secret, so dass wir sie untersuchen können. Denken Sie bitte daran, eine volle Beschreibung Ihres Systems beizufügen sowie die Version Ihrer mysqld-Binärdatei und wie Sie sie kompiliert haben.

- Versuchen Sie auch, `mysql-test-run` mit der `--force`-Option laufen zu lassen, um zu sehen, ob auch andere Tests fehlschlagen.

- Wenn Sie MySQL selbst kompiliert haben, sehen Sie im Handbuch nach, wie MySQL auf Ihrer Plattform kompiliert wird, oder benutzen Sie vorzugsweise eine der Binärdateien, die wir für Sie kompiliert haben und die Sie unter http://www.mysql.com/downloads/ finden. Alle unsere Standard-Binärdateien sollten die Test-Suite fehlerfrei durchlaufen!

- Wenn Sie einen Fehler wie `Result length mismatch` oder `Result content mismatch` erhalten, heißt das, dass die Ausgabe des Tests nicht genau mit der erwarteten Ausgabe übereinstimmt. Das könnte ein Bug in MySQL sein, könnte aber auch heißen, dass Ihre mysqld-Version unter bestimmten Umständen leicht abweichende Ausgaben erzeugt.

  Fehlgeschlagene Testergebnisse werden in eine Datei mit demselben Namen wie die Ergebnisdatei, mit der Endung `.reject`, gestellt. Wenn Ihr Testfall fehlschlägt, sollten Sie ein DIFF beider Dateien vornehmen. Wenn Sie nicht erkennen können, in welcher Hinsicht sie sich unterscheiden, untersuchen Sie beide mit `od -c` und prüfen Sie auch ihre Längen.

- Wenn ein Testfall völlig fehlschlägt, sollten Sie die Log-Dateien im `mysql-test/var/log`-Verzeichnis nach Hinweisen untersuchen, was schief ging.

- Wenn Sie MySQL mit Debugging kompiliert haben, können Sie versuchen, das zu debuggen, indem Sie `mysql-test-run` mit den `--gdb`- und / oder `--debug`-Optionen laufen lassen. See Abschnitt E.1.2, „Trace-Dateien erzeugen“.

  Wenn Sie MySQL nicht für Debugging kompiliert haben, sollten Sie das besser tun. Geben Sie einfach die `--with-debug`-Option für `configure` an! See Abschnitt 3.3, „Installation der Quelldistribution“.

Anhang A. Probleme und häufige Fehler

Dieses Kapitel listet einige gebräuchliche Probleme und Fehlermeldungen auf, denen Benutzer in die Arme laufen. Sie lernen herauszufinden, was das Problem ist und wie Sie es lösen. Hier finden sich auch korrekte Lösungen einiger häufiger Probleme.

A.1. Wie man feststellt, was Probleme verursacht

Wenn Sie Probleme bekommen, sollten Sie als erstes herausfinden, welches Programm oder Hardware-Teil die Probleme verursacht:

- Wenn Sie eins der folgenden Symptome beobachten, gibt es wahrscheinlich ein Hardware- (Speicher, Hauptplatine, Prozessor oder Festplatte) oder Kernel-Problem:
  - Die Tastatur funktioniert nicht. Normalerweise können Sie das durch Drücken der Feststelltaste (Caps Lock) überprüfen. Wenn sich die Anzeigeleuchte beim Drücken nicht an- und ausschaltet, müssen Sie Ihre Tastatur ersetzen. (Bevor Sie das tun, sollten Sie Ihren Computer neu starten und alle Kabelverbindungen zur Tastatur überprüfen.)
  - Der Mauszeiger bewegt sich nicht.
  - Die Maschine antwortet auf entfernte Ping-Versuche nicht.
  - Andere Programme, die mit MySQL nichts zu tun haben, funktionieren nicht korrekt.
  - Wenn Ihr System unerwartet neu startet (ein fehlerhaftes Programm auf Benutzerebene sollte NIE in der Lage sein, Ihr System zum Absturz zu bringen).

  In solchen Fällen sollten Sie zunächst alle Kabel überprüfen und Diagnoseprogramme laufen lassen, um Ihre Hardware zu untersuchen! Sie sollten auch prüfen, ob Patches, Aktualisierungen oder Service-Packs für Ihre Betriebssystem verfügbar sind, die Ihre Probleme möglicherweise lösen. Prüfen Sie auch, ob Ihre Bibliotheken (wie glibc) aktuell sind.

  Es ist immer eine gute Idee, eine Maschine mit ECC-Speicher zu benutzen, um Speicherprobleme frühzeitig zu erkennen!
- Wenn Ihre Tastatur gesperrt ist, können Sie das eventuell beheben, indem Sie sich von einer anderen Maschine aus verbinden und `kbd_mode -a` ausführen.
- Untersuchen Sie Ihre System-Log-Datei (/var/log/messages oder ähnliches) nach Gründen für Ihre Probleme. Wenn Sie glauben, dass das Problem an MySQL liegt, sollten Sie auch die Log-Dateien von MySQL überprüfen. See Abschnitt 5.9.3, „Die Update-Log-Datei“.
- Wenn Sie nicht glauben, ein Hardware-Problem zu haben, sollten Sie herausfinden, welches Programm die Probleme verursacht.

  Probieren Sie `top`, `ps`, `taskmanager` oder ein ähnliches Programm, um zu prüfen, welches Programm die gesamte Prozessorzeit konsumiert oder die Maschine blockiert.
- Prüfen Sie mit `top`, `df` oder einem ähnlichen Programm, wenn Sie keinen freien Arbeitsspeicher, Festplattenspeicher, verfügbare Datei-Handler oder eine andere kritische Ressource mehr haben.
- Wenn das Problem an einem aus dem Ruder gelaufenen Prozess liegt, können Sie versuchen, diesen zu killen. Wenn er nicht sterben will, gibt es wahrscheinlich einen Bug im Betriebssystem.

Wenn Sie alle anderen Möglichkeiten untersucht und ausgeschlossen haben und zur Schlussfolgerung gekommen sind, dass die Probleme durch den MySQL-Server oder ein MySQL-Client-Programm verursacht werden, ist es an der Zeit, einen Bug-Bericht für die Mailing-Liste oder unser Support-Team zu schreiben. Machen Sie im Bug-Bericht eine sehr detaillierte Beschreibung, wie sich Ihr System verhält und was Sie vermuten, was passiert. Sie sollten auch angeben, warum Sie denken, dass MySQL die Probleme verursacht. Ziehen Sie alle Situationen in diesem Kapitel in Betracht. Geben Sie genau an, welche Probleme wie auftauchen, wenn Sie Ihr System untersuchen. Benutzen Sie Kopieren und Einfügen, wenn Sie Ausgaben und / oder Fehlermeldungen von Programmen oder aus Log-Dateien beifügen!

Versuchen Sie detailliert zu beschreiben, welches Programm nicht funktioniert, und alle Symptome, die Sie sehen! In der Vergangenheit haben wir viele Bug-Berichte erhalten, in denen schlicht steht, dass "das System nicht funktioniert". Daraus können wir natürlich keinerlei Informationen ziehen, wie das Problem gelöst werden könnte.

Wenn ein Programm fehlschlägt, ist es immer nützlich, folgendes zu wissen:

- Hat das fragliche Programm einen Segmentation-Fehler verursacht (Core Dump)?

- Nimmt das Programm sich die gesamte Prozessorleistung? Überprüfen Sie das mit `top`. Lassen Sie das Programm eine Weile laufen, denn vielleicht evaluiert es gerade nur etwas Schwieriges.
- Wenn der `mysqld`-Server Probleme verursacht, können Sie dann `mysqladmin -u root ping` oder `mysqladmin -u root processlist` ausführen?
- Was sagt ein Client-Programm (zum Beispiel `mysql`), wenn Sie versuchen, sich mit dem MySQL-Server zu verbinden? Bricht der Client zusammen? Erhalten Sie von diesem Programm irgend welche Ausgaben?

Wenn Sie einen Bug-Bericht senden, sollten Sie immer den Angaben folgen, die in diesem Handbuch beschrieben sind. See Abschnitt 2.6.2.2, „Wie man Fragen stellt oder Bugs berichtet“.

A.2. Einige gebräuchliche Fehler bei der Benutzung von MySQL

Dieser Abschnitt listet einige Fehler auf, die Benutzer häufig erhalten. Hier finden Sie Beschreibungen dieser Fehler und wie man die Probleme löst.

A.2.1. `Access denied`-Fehler

See Abschnitt 5.2.5, „Wie das Berechtigungssystem funktioniert“, insbesondere See Abschnitt 5.2.10, „Gründe für `Access denied`-Fehler“.

A.2.2. `MySQL server has gone away`-Fehler

Dieser Abschnitt behandelt auch den verwandten `Lost connection to server during query`-Fehler.

Der häufigste Grund für den `MySQL server has gone away`-Fehler ist eine Zeitüberschreitung, nach der der Server die Verbindung schloss. Vorgabemäßig schließt der Server die Verbindung nach 8 Stunden, wenn nichts passiert ist. Sie können diesen Wert mit der `wait_timeout`-Variablen ändern, die beim Start von `mysqld` gesetzt wird.

Ein weiterer häufiger Grund für den `MySQL server has gone away`-Fehler ist das Absetzen eines ``close" auf Ihre MySQL-Verbindung mit dem anschließenden Versuch, auf der geschlossenen Verbindung eine Anfrage abzusetzen.

Sie können überprüfen, ob der MySQL-Server gestorben ist, indem Sie `mysqladmin version` ausführen und die Uptime untersuchen.

Wenn Sie ein Skript haben, müssen Sie die Anfrage lediglich noch einmal für den Client absetzen, um eine automatische Neuverbindung zu machen.

Normalerweise können Sie folgende Fehler-Codes für diesen Fall abfragen (die Betriebssystem-abhängig sind):

`CR_SERVER_GONE_ERROR`	Der Client konnte keine Anfrage an den Server schicken.
`CR_SERVER_LOST`	Der Client erhielt beim Schreiben zum Server keinen Fehler, bekam aber keine vollständige (oder überhaupt keine) Antwort.

Sie erhalten diese Fehler auch, wenn Sie eine Anfrage zum Server schicken, die falsch oder zu Groß ist. Wenn `mysqld` ein Paket erhält, das zu Groß oder nicht in Ordnung ist, nimmt er hat, dass etwas mit dem Client schief ging und schließt die Verbindung. Wenn Sie große Anfragen brauchen (beispielsweise wenn Sie mit `BLOB`-Spalten arbeiten), können Sie die Anfragebeschränkung erhöhen, indem Sie `mysqld` mit der `-O max_allowed_packet=#`-Option starten (Vorgabe 1 MB). Der zusätzliche Speicher wird bei Bedarf zugewiesen, daher benutzt `mysqld` nur dann mehr Speicher, wenn Sie eine große Anfrage ausführen oder wenn `mysqld` ein großes Ergebnis zurückgeben muss!

A.2.3. `Can't connect to [local] MySQL server`-Fehler

Ein MySQL-Client unter Unix kann sich auf zwei unterschiedliche Arten mit dem `mysqld`-Server verbinden: Unix-Sockets, die sich durch eine Datei im Dateisystem verbinden (Vorgabe `/tmp/mysqld.sock`) oder über TCP/IP, was sich über eine Portnummer verbindet. Unix-Sockets sind schneller als TCP/IP, können aber nur benutzt werden, wenn man sich zu einem Server auf demselben Computer verbindet. Unix-Sockets werden benutzt, wenn Sie keinen Hostnamen oder den speziellen Hostnamen `localhost` angeben.

Unter Windows können Sie sich nur mit TCP/IP verbinden, wenn der `mysqld`-Server unter Windows 95 / 98 läuft. Wenn er unter Windows NT läuft, können Sie sich auch mit Named Pipes verbinden. Der Name der Named Pipe ist MySQL. Wenn Sie bei der Verbindung zu `mysqld` keinen Hostnamen angeben, versucht ein MySQL-Client zuerst, sich über die Named Pipe zu verbinden. Erst wenn das fehlschlägt, versucht er, sich über den TCP/IP-Port zu verbinden. Sie können die Benutzung von Named Pipes unter Windows erzwingen, indem Sie `.` als Hostnamen benutzen.

Der Fehler (2002) `Can't connect to ...` bedeutet normalerweise, dass auf dem System kein MySQL-Server läuft oder dass Sie eine falsche Socket-Datei oder einen falschen TCP/IP-Port bei der Verbindung mit dem `mysqld`-Server benutzen.

Prüfen Sie zuerst mit `ps` oder dem Task-Manager unter Windows, ob es einen laufenden Prozess namens `mysqld` auf Ihrem Server gibt! Wenn es keinen `mysqld`-Prozess gibt, sollten Sie einen starten. See Abschnitt 3.4.2, „Probleme mit dem Start des MySQL-Servers“.

Wenn ein `mysqld`-Prozess läuft, können Sie den Server mit diesen unterschiedlichen Verbindungen überprüfen (die Portnummer und Socket-Pfadnamen können auf Ihrem System natürlich anders sein):

```
shell> mysqladmin version
shell> mysqladmin variables
shell> mysqladmin -h `hostname` version variables
shell> mysqladmin -h `hostname` --port=3306 version
shell> mysqladmin -h 'ip_ihres_hosts' version
shell> mysqladmin --socket=/tmp/mysql.sock version
```

Beachten Sie die Benutzung umgedrehter Anführungszeichen statt normaler Anführungszeichen beim `hostname`-Befehl. Diese verursachen, dass die Ausgabe durch `hostname` (das heißt des aktuellen Hostnamens) im `mysqladmin`-Befehl ersetzt wird.

Hier sind einige Gründe für das Auftreten des `Can't connect to local MySQL server`-Fehlers:

- `mysqld` läuft nicht.
- Sie fahren auf einem System, das MIT-pThread verwendet. Wenn Sie auf einem System fahren, das keine nativen Threads hat, benutzt `mysqld` das MIT-pThread-Paket. See Abschnitt 3.2.2, „Betriebssysteme, die von MySQL unterstützt werden“. Nicht alle MIT-pThread-Versionen unterstützen jedoch Unix-Sockets. Auf einem System ohne Socket-Unterstützung müssen Sie den Hostnamen immer explizit angeben, wenn Sie sich mit dem Server verbinden. Benutzen Sie diesen Befehl, um die Verbindung zum Server zu überprüfen:

  ```
  shell> mysqladmin -h `hostname` version
  ```

- Jemand hat den Unix-Socket entfernt, den `mysqld` benutzt (Vorgabe `/tmp/mysqld.sock`). Vielleicht gibt es einen `cron`-Job, der den MySQL-Socket entfernt (beispielsweise ein Job, der alte Dateien aus dem `/tmp`-Verzeichnis entfernt). Sie können `mysqladmin version` laufen lassen und überprüfen, dass der Socket, den `mysqladmin` versucht zu benutzen, tatsächlich existiert. Die Problemlösung besteht in diesem Fall darin, den `cron`-Job so zu ändern, dass er nicht `mysqld.sock` entfernt oder den Socket an andere Stelle zu platzieren. See Abschnitt A.4.5, „Wie Sie die MySQL-Socket-Datei `/tmp/mysql.sock` schützen oder ändern“.
- Sie haben den `mysqld`-Server mit der `--socket=/pfad/zu/socket`-Option gestartet. Wenn Sie den Socket-Pfadnamen zum Server ändern, müssen Sie auch die MySQL-Clients darüber unterrichten. Das können Sie tun, indem Sie den Socket-Pfad als Argument an den Client übergeben. See Abschnitt A.4.5, „Wie Sie die MySQL-Socket-Datei `/tmp/mysql.sock` schützen oder ändern“.
- Sie benutzen Linux und ein Thread ist gestorben (Core Dump). In diesem Fall müssen Sie den anderen `mysqld`-Thread killen (beispielsweise mit dem `mysql_zap`-Skript), bevor Sie einen neuen MySQL-Server starten können. See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“.
- Eventuell haben Sie keine Lese- und Schreibberechtigungen entweder für das Verzeichnis, in dem die Socket-Datei liegt, oder keine Berechtigung für die Socket-Datei selbst. In diesem Fall können Sie entweder die Berechtigung für die Datei und / oder das Verzeichnis ändern oder `mysqld` neu starten, so dass er ein Verzeichnis benutzt, auf das Sie Zugriff haben.

Wenn Sie die Fehlermeldung `Can't connect to MySQL server on ein_hostname` erhalten, können Sie folgendes probieren, um den Grund des Problems herauszufinden:

- Überprüfen Sie, ob der Server hochgefahren ist, indem Sie `telnet ihr_hostname tcp-ip-port-nummer` ausführen und einige Male die Eingabetaste (`RETURN`) drücken. Wenn es auf diesem Port einen laufenden MySQL-Server gibt, sollten Sie eine Antwort erhalten, die die Versionsnummer des Server enthält. Wenn Sie einen Fehler wie `telnet: Unable to connect to remote host: Connection refused` erhalten, gibt es auf diesem Port keinen laufenden Server.
- Versuchen Sie, sich mit dem `mysqld`-Daemon auf der lokalen Maschine zu verbinden und prüfen Sie den TCP/IP-Port, den `mysqld` laut Konfiguration benutzen soll (Variable `port`), mit `mysqladmin variables`.
- Prüfen Sie, ob Ihr `mysqld`-Server nicht gestartet wurde, indem Sie die `--skip-networking`-Option verwenden.

A.2.4. `Host '...' is blocked`-Fehler

Wenn Sie einen Fehler wie folgt erhalten:

```
Host 'hostname' is blocked because of too many connection errors.
Unblock with 'mysqladmin flush-hosts'
```

Bedeutet das, dass `mysqld` zu viele (`max_connect_errors`) Verbindungsanforderungen vom Host `'hostname'` erhalten hat, die mittendrin unterbrochen wurden. Nach `max_connect_errors` fehlgeschlagenen Anfragen nimmt `mysqld` an, dass etwas nicht stimmt (wie ein Angriff eines Crackers) und blockiert weitere Verbindungsanforderungen von der Site, bis jemand `mysqladmin flush-hosts` ausführt.

Vorgabemäßig blockiert `mysqld` einen Host nach 10 Verbindungsfehlern. Das können Sie leicht durch Starten des Servers wie folgt ändern:

```
shell> safe_mysqld -O max_connect_errors=10000 &
```

Beachten Sie, dass Sie bei dieser Fehlermeldung für einen gegebenen Host zunächst prüfen sollten, ob etwas mit den TCP/IP-Verbindungen von diesem Host aus nicht stimmt. Wenn Ihre TCP/IP-Verbindungen nicht funktionieren, nützt es Ihnen nichts, den Wert der `max_connect_errors`-Variablen heraufzusetzen!

A.2.5. `Too many connections`-Fehler

Wenn Sie beim Verbindungsversuch den Fehler `Too many connections` erhalten, heißt das, dass es bereits `max_connections` Clients gibt, die mit dem `mysqld`-Server verbunden sind.

Wenn Sie mehr Verbindungen als die Vorgabe (100) benötigen, können Sie `mysqld` mit einem größeren Wert für die `max_connections`-Variable neu starten.

Beachten Sie, dass `mysqld` tatsächlich (`max_connections`+1) Clients für Verbindungen zuläßt. Die letzte Verbindung wird für einen Benutzer mit der **process**-Berechtigung reserviert. Wenn Sie keinem normalen Benutzer diese Berechtigung geben (diese sollte sie nie benötigen), kann sich ein Administrator mit dieser Berechtigung einloggen und `SHOW PROCESSLIST` benutzen, um herauszufinden, was schief geht. See Abschnitt 5.5.5, „SHOW-Syntax“.

Die maximale Anzahl von Verbindungen ist davon abhängig, wie gut die Thread-Bibliothek auf der Plattform ist. Linux oder Solaris sollten in der Lage sein, 500 bis 1000 gleichzeitige Verbindungen zu unterstützen, abhängig davon, wie viel Arbeitsspeicher Sie haben und was Ihre Clients ausführen.

A.2.6. `Some non-transactional changed tables couldn't be rolled back`-Fehler

Wenn Sie den Fehler `Warning: Some non-transactional changed tables couldn't be rolled back` erhalten, wenn Sie ein `ROLLBACK` versuchen, bedeutet das, dass einige der bei der Transaktion benutzten Tabellen keine Transaktionen unterstützen. Diese nicht transkationalen Tabellen werden vom `ROLLBACK`-Statement nicht betroffen.

Der typischste Fall, bei dem dieser Fehler auftritt, ist, wenn Sie versucht haben, eine Tabelle von einem Typ zu erzeugen, der von Ihrer `mysqld`-Binärdatei nicht unterstützt wird. Wenn `mysqld` einen Tabellentyp nicht unterstützt (oder wenn der Tabellentyp durch die Startoption ausgeschaltet ist), wird statt dessen ein Tabellentyp erzeugt, der dem angeforderten am nächsten entspricht (wahrscheinlich `MyISAM`.

Sie können den Tabellentyp für eine Tabelle wie folgt überprüfen:

`SHOW TABLE STATUS LIKE 'tabelle'`. See Abschnitt 5.5.5.2, „SHOW TABLE STATUS“.

Sie können die Erweiterungen, die Ihre `mysqld`-Binärdatei unterstützt, wie folgt überprüfen:

`show variables like 'have_%'`. See Abschnitt 5.5.5.4, „SHOW VARIABLES“.

A.2.7. `No free memory`-Fehler

Wenn Sie eine Anfrage ausführen und etwas wie folgenden Fehler erhalten:

```
mysql: No free memory at line 42, 'malloc.c'
mysql: needed 8136 byte (8k), memory in use: 12481367 Bytes (12189k)
ERROR 2008: MySQL client ran No free memory
```

Beachten Sie, dass sich dieser Fehler auf den MySQL-Client `mysql` bezieht. Der Grund für diesen Fehler ist einfach, dass der Client nicht genug freien Speicher hat, um das gesamte Ergebnis zu speichern.

Um das Problem zu beheben, prüfen Sie zunächst, ob Ihre Anfrage korrekt ist. Sollte sie vernünftigerweise so viele Zeilen

zurückgeben? Wenn das der Fall ist, können Sie `mysql --quick` benutzen, was `mysql_use_result()` benutzt, um die Ergebnismenge abzurufen. Hierdurch wird Last vom Client auf den Server verlagert.

A.2.8. `Packet too large`-Fehler

Wenn ein MySQL-Client oder der `mysqld`-Server ein Paket erhält, das größer als `max_allowed_packet` Bytes ist, gibt er einen `Packet too large`-Fehler aus und schließt die Verbindung.

Wenn Sie den `mysql`-Client benutzen, müssen Sie einen größeren Puffer angeben, indem Sie den Client mit `mysql --set-variable=max_allowed_packet=8M` starten.

Wenn Sie andere Clients benutzen, die die Angabe der maximalen Paketgröße nicht zulassen (wie `DBI`), müssen Sie die Paketgröße beim Start des Servers setzen. Sie können eine Kommandozeilenoption für `mysqld` benutzen, um `max_allowed_packet` auf eine höhere Größe zu setzen. Wenn Sie zum Beispiel beabsichtigen, die volle Länge eines `BLOB` in eine Tabelle zu speichern, müssen Sie den Server mit der `--set-variable=max_allowed_packet=16M`-Option starten.

Sie können merkwürdige Probleme mit großen Paketen erhalten, wenn Sie große Blobs benutzen, aber `mysqld` keinen Zugriff auf genug Speicher gegeben haben, um die Anfrage zu handhaben. Wenn Sie vermuten, dass das der Fall ist, versuchen Sie, am Anfang des `safe_mysqld`-Skripts `ulimit -d 256000` hinzuzufügen, und starten Sie `mysqld` neu.

A.2.9. Kommunikationsfehler / Abgebrochene Verbindung

Ab `MySQL 3.23.40` erhalten Sie den `Aborted connection`-Fehler nur dann, wenn Sie `mysqld` mit `--warnings` starten.

Wenn Sie Fehler wie den folgenden in Ihrer Fehler-Log-Datei entdecken:

```
010301 14:38:23  Aborted connection 854 to db: 'Benutzer' user: 'josh'
```

See Abschnitt 5.9.1, „Die Fehler-Log-Datei“.

Bedeutet das, dass eins der folgenden Dinge passiert ist:

- Das Client-Programm rief vor dem Beenden nicht `mysql_close()` auf.
- Der Client schlief länger als `wait_timeout` oder `interactive_timeout`, ohne Anfragen auszuführen.

  See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“.
- Das Client-Programm wurde abrupt während einer Übertragung beendet.

Wenn das oben Genannte passiert, wird die Servervariable `Aborted_clients` heraufgezählt.

Die Servervariable `Aborted_connects` wird in folgenden Fällen heraufgezählt:

- Wenn ein Verbindungspaket nicht die richtigen Informationen enthält.
- Wenn der Benutzer keine Berechtigung hat, sich mit einer Datenbank zu verbinden.
- Wenn ein Benutzer ein falsches Passwort angegeben hat.
- Wenn es länger als `connect_timeout` Sekunden dauert, um ein Verbindungspaket zu erhalten.

Beachten Sie, dass obiges auch anzeigen könnte, dass jemand versucht, in Ihre Datenbank einzubrechen!

See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“.

Andere Gründe für Probleme mit abgebrochenen Clients / abgebrochenen Verbindungen:

- Benutzung des Duplex-Ethernet-Protokolls, sowohl Halb- als auch Voll-Duplex, unter Linux. Viele Linux-Ethernet-Treiber haben diesen Bug. Sie können auf diesen Bug überprüfen, indem Sie eine sehr große Datei via FTP zwischen diesen beiden Maschinen übertragen. Wenn ein Transfer nach dem Schema schnelle Übertragung - Pause - schnelle Übertragung - Pause läuft, haben Sie ein Linux-Duplex-Syndrom. Die einzige Lösung besteht darin, Halb- und Vollduplex auf Hubs und Switches auszuschalten.
- Probleme mit der Thread-Bibliothek, was Unterbrechungen bei Lesevorgängen verursacht.

- Schlecht konfiguriertes TCP/IP.
- Fehlerhafte Ethernets, Hubs, Switches, Kabel usw. Das kann nur durch Austausch von Hardware sauber diagnostiziert werden.
- `max_allowed_packet` ist zu klein oder Anfragen erfordern mehr Speicher, als Sie für `mysqld` zugewiesen haben.

  See Abschnitt A.2.8, „`Packet too large`-Fehler“.

A.2.10. `The table is full`-Fehler

Der Fehler tritt in älteren MySQL-Versionen auf, wenn eine Hauptspeicher-basierende temporäre Tabelle größer als `tmp_table_size` Bytes wird. Um dieses Problem zu vermeiden, können Sie die `-O tmp_table_size=#`-Option für `mysqld` benutzen, um die Größe der temporären Tabelle zu erhöhen, oder die SQL-Option `SQL_BIG_TABLES` verwenden, bevor Sie die problematische Anfrage abschicken. See Abschnitt 6.5.6, „`SET`-Syntax“.

Sie können auch `mysqld` mit der `--big-tables`-Option starten. Das ist genau dasselbe, wie `SQL_BIG_TABLES` für alle Anfragen zu benutzen.

In MySQL-Version 3.23 werden Hauptspeicher-basierende temporäre Tabellen automatisch in Festplatten-basierende `MyISAM`-Tabellen umgewandelt, wenn die Tabelle größer als `tmp_table_size` wird.

A.2.11. `Can't create/write to file`-Fehler

Wenn Sie für einige Anfragen Fehler folgenden Typs erhalten:

```
Can't create/write to file '\\sqla3fe_0.ism'.
```

Bedeutet das, dass MySQL keine temporäre Datei für die Ergebnismenge im angegebenen temporären Verzeichnis erzeugen kann. (Der obige Fehler ist eine typische Fehlermeldung unter Windows; die Unix-Fehlermeldung ist ähnlich.) Das Problem läßt sich beheben, indem Sie `mysqld` mit `--tmpdir=pfad` starten oder folgendes in Ihrer Optionsdatei ergänzen:

```
[mysqld]
tmpdir=C:/temp
```

Unter der Annahme, dass das `c:\\temp`-Verzeichnis existiert. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

Überprüfen Sie auch den Fehler-Code, den Sie bei `perror` erhalten. Ein Grund kann ein Fehler wegen fehlenden Festplattenspeichers sein:

```
shell> perror 28
Error code  28:  No space left on device
```

A.2.12. `Command out of sync`-Fehler in Client

Wenn Sie den Fehler `command out of sync; You can't run this command now` in Ihrem Client-Code erhalten, rufen Sie Client-Funktionen in der falschen Reihenfolge auf!

Das kann zum Beispiel passieren, wenn Sie `mysql_use_result()` benutzen und versuchen, eine neue Anfrage auszuführen, bevor Sie `mysql_free_result()` aufgerufen haben. Der Fehler passiert ebenfalls, wenn Sie versuchen, zwei Anfragen auszuführen, die Daten zurückgeben, ohne zwischendrin `mysql_use_result()` oder `mysql_store_result()` aufzurufen.

A.2.13. `User ignored`-Fehler

Wenn Sie folgenden Fehler erhalten:

```
Found wrong password for user: 'benutzer@ein_host'; User ignored
```

Bedeutet das, dass `mysqld` beim Start oder nach dem Neuladen der Berechtigungstabellen einen Eintrag in der `user`-Tabelle mit einem ungültigen Passwort gefunden hat. Als Ergebnis wird der Eintrag vom Berechtigungssystem einfach ignoriert.

Mögliche Gründe und Problembehebung:

- Sie lassen eine neue Version von `mysqld` mit einer alten `user`-Tabelle laufen. Das können Sie prüfen, indem Sie `mysqlshow mysql user` eingeben, um zu sehen, ob das Passwortfeld kürzer als 16 Zeichen ist. Wenn das der Fall ist, können Sie diesen Zustand beheben, indem Sie das `scripts/add_long_password`-Skript laufen lassen.

- Der Benutzer hat ein altes Passwort (8 Zeichen lang) und Sie haben `mysqld` nicht mit der `--old-protocol`-Option gestartet.
- Sie haben in der `user`-Tabelle ein Passwort eingegeben, ohne die `PASSWORD()`-Funktion zu benutzen. Benutzen Sie `mysql`, um den Benutzer in der `user`-Tabelle mit einem neuen Passwort zu aktualisieren. Stellen Sie sicher, dass Sie die `PASSWORD()`-Funktion benutzen:

```
mysql> update user set password=PASSWORD('ihr_passwort')
           where user='XXX';
```

A.2.14. `Table 'xxx' doesn't exist`-Fehler

Wenn Sie den Fehler `Table 'xxx' doesn't exist` oder `Can't find file: 'xxx' (errno: 2)` erhalten, bedeutet das, dass in der aktuellen Datenbank keine Tabelle mit dem Namen `xxx` existiert.

Beachten Sie, dass Datenbank- und Tabellennamen **abhängig von der verwendeten Groß-/Kleinschreibung** sind, weil MySQL Verzeichnisse und Dateien benutzt, um Datenbanken und Tabellen zu speichern! (Unter Windows sind Datenbank- und Tabellennamen unabhängig von der Schreibweise, aber alle Verweise auf eine gegebene Tabelle innerhalb einer Anfrage müssen dieselbe Schreibweise benutzen!)

Sie finden heraus, welche Tabellen sich in der aktuellen Datenbank befinden, indem Sie `SHOW TABLES` eingeben. See Abschnitt 5.5.5, „SHOW-Syntax“.

A.2.15. `Can't initialize charset xxx`-Fehler.

Wenn Sie folgenden Fehler erhalten:

```
MySQL Connection Failed: Can't initialize charset xxx
```

Bedeutet das eins der folgenden Dinge:

- Der Zeichensatz ist ein Multi-Byte-Zeichensatz und Ihr Client unterstützt diesen Zeichensatz nicht.

  In diesem Fall müssen Sie Ihren Client neu kompilieren und die `--with-charset=xxx`- oder die `--with-extra-charsets=xxx`-Option angeben. See Abschnitt 3.3.3, „Typische `configure`-Optionen“.

  Alle Standard-MySQL-Binärdistributionen werden mit `--with-extra-character-sets=complex` kompiliert, was die Unterstützung für alle Multi-Byte-Zeichensätze aktiviert. See Abschnitt 5.6.1, „Der für Daten und Sortieren benutzte Zeichensatz“.

- Der Zeichensatz ist ein einfacher Zeichensatz, der nicht in `mysqld` kompiliert wurde, und die Zeichensatz-Definitionsdateien sind nicht an der Stelle, an der der Client sie erwartet.

  In diesem Fall müssen Sie:

  - Den Client mit Unterstützung für den Zeichensatz neu kompilieren. See Abschnitt 3.3.3, „Typische `configure`-Optionen“.
  - Dem Client angeben, wo die Zeichensatz-Definitionsdateien sind. Bei vielen Clients können Sie das mit der `--character-sets-dir=pfad-to-charset-dir`-Option machen.
  - Die Zeichensatz-Definitionsdatei in den Pfad kopieren, wo der Client sie zu finden erwartet.

A.2.16. File Not Found

Wenn Sie den Fehler `ERROR '...' not found (errno: 23)`, `Can't open file: ... (errno: 24)` oder irgend einen anderen Fehler mit `errno 23` oder `errno 24` erhalten, bedeutet das, dass Sie MySQL nicht genug Datei-Deskriptoren zugewiesen haben. Sie können das `perror`-Dienstprogramm benutzen, um eine Beschreibung dessen zu erhalten, was die Fehlernummer bedeutet:

```
shell> perror 23
File table overflow
shell> perror 24
Too many open files
shell> perror 11
Resource temporarily unavailable
```

Das Problem hierbei ist, dass `mysqld` versucht, zu viele Dateien gleichzeitig offen zu halten. Sie können entweder `mysqld` veranlassen, nicht so viele Dateien auf einmal zu öffnen, oder die Anzahl von Datei-Deskriptoren heraufsetzen, über die `mysqld` verfügen kann.

Um `mysqld` anzuweisen, weniger Dateien zugleich offen zu halten, können Sie den Tabellen-Cache kleiner machen, indem Sie die `-O table_cache=32`-Option für `safe_mysqld` benutzen (der Vorgabewert ist 64). Wenn Sie den Wert von `max_connections` verringern, reduziert auch das die Anzahl offener Dateien (der Vorgabewert ist 90).

Um die Anzahl von Datei-Deskriptoren, die `mysqld` zur Verfügung stehen, zu ändern, können Sie die `--open-files-limit=#`-Option für `safe_mysqld` oder die `-O open-files-limit=#`-Option für `mysqld` benutzen. See Abschnitt 5.5.5.4, „`SHOW VARIABLES`“. Die einfachste Art, das zu tun, besteht darin, eine Option zu Ihrer Optionsdatei hinzuzufügen. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“. Wenn Sie eine alte `mysqld`-Version haben, die das nicht unterstützt, können Sie das `safe_mysqld`-Skript editieren. Es gibt dort eine auskommentierte Zeile `ulimit -n 256`. Entfernen Sie das `'#'`-Zeichen, um diese Zeile zu aktivieren, und ändern Sie die Anzahl 256, um die Anzahl verfügbarer Datei-Deskriptoren zu beeinflussen.

Mit `ulimit` (und `open-files-limit`) kann man die Anzahl von Datei-Deskriptoren heraufsetzen, aber nur bis zu der Grenze, die das Betriebssystem vorgibt. Darüber hinaus gibt es eine 'harte' Grenze, die nur überschrieben werden kann, wenn Sie `safe_mysqld` oder `mysqld` als Root starten (denken Sie daran, dass Sie in diesem Fall auch die `--user=..`-Option benutzen müssen). Wenn Sie die Betriebssystem-Grenze hinsichtlich der Anzahl von Datei-Deskriptoren, die für jeden Prozess verfügbar sind, heraufsetzen müssen, schauen Sie in der Dokumentation Ihres Betriebssystems nach.

Beachten Sie, dass `ulimit` nicht funktioniert, wenn Sie die `tcsh`-Shell laufen lassen! `tcsh` berichtet auch nicht korrekte Werte, wenn Sie die aktuellen Grenzen abfragen! In diesem Fall sollten Sie `safe_mysqld` mit `sh` starten!

A.3. Installationsbezogene Themen

A.3.1. Probleme beim Linken mit der MySQL-Client-Bibliothek

Wenn Sie Ihr Programm linken und Fehler für unreferenzierte Symbole erhalten, die mit `mysql_` beginnen, wie folgende:

```
/tmp/ccFKsdPa.o: In function `main':
/tmp/ccFKsdPa.o(.text+0xb): undefined reference to `mysql_init'
/tmp/ccFKsdPa.o(.text+0x31): undefined reference to `mysql_real_connect'
/tmp/ccFKsdPa.o(.text+0x57): undefined reference to `mysql_real_connect'
/tmp/ccFKsdPa.o(.text+0x69): undefined reference to `mysql_error'
/tmp/ccFKsdPa.o(.text+0x9a): undefined reference to `mysql_close'
```

Sollten Sie das durch Hinzufügen von `-Lpath-to-the-mysql-library -lmysqlclient` als **LETZTES** in Ihrer Link-Zeile beheben können.

Wenn Sie `undefined reference`-Fehler bei der `uncompress`- oder `compress`-Funktion erhalten, fügen Sie `-lz` als **LETZTES** zu Ihrer Link-Zeile hinzu und versuchen Sie es noch einmal!

Wenn Sie `undefined reference`-Fehler bei Funktionen erhalten, die es auf Ihrem System geben sollte, wie `connect`, sehen Sie in der Handbuch-Seite (ManPage) für die fragliche Funktion nach, welche Bibliotheken Sie zur Link-Zeile hinzufügen sollten!

Wenn Sie `undefined reference`-Fehler bei Funktionen erhalten, die es auf Ihrem System nicht gibt, wie folgenden:

```
mf_format.o(.text+0x201): undefined reference to `__lxstat'
```

Heißt das üblicherweise, dass Ihre Bibliothek auf einem System kompiliert wurde, das nicht 100% kompatibel zu Ihrem System ist. In diesem Fall sollten Sie die letzte MySQL-Quelldistribution herunter laden und sie selbst kompilieren. See Abschnitt 3.3, „Installation der Quelldistribution“.

Wenn Sie versuchen, ein Programm laufen zu lassen und Fehler für unreferenzierte Symbole erhalten, die mit `mysql_` anfangen, oder den Fehler, dass die `mysqlclient`-Bibliothek nicht gefunden werden kann, heißt das, dass Ihr System die gemeinsam genutzte `libmysqlclient.so`-Bibliothek nicht findet.

Das Problem beheben Sie, indem Sie Ihr System anweisen, dort nach gemeinsam genutzten Bibliotheken zu suchen, wo sich die Bibliothek befindet, mit einer der folgenden Methoden:

- Fügen Sie den Pfad zum Verzeichnis, in dem Sie `libmysqlclient.so` haben, der `LD_LIBRARY_PATH`-Umgebungsvariablen hinzu.
- Fügen Sie den Pfad zum Verzeichnis, in dem Sie `libmysqlclient.so` haben, der `LD_LIBRARY`-Umgebungsvariablen hinzu.
- Kopieren Sie `libmysqlclient.so` an eine Stelle, die von Ihrem System durchsucht wird, wie `/lib`, und aktualisieren Sie

die Informationen über gemeinsam genutzte Bibliotheken, indem Sie `ldconfig` ausführen.

Eine weitere Möglichkeit, dieses Problem zu lösen, besteht darin, Ihr Programm statisch mit `-static` zu linken oder die dynamischen MySQL-Bibliotheken zu entfernen, bevor Sie Ihren Code linken. Im letzteren Fall sollten Sie sicherstellen, dass keine anderen Programme die dynamischen Bibliotheken benutzen!

A.3.2. Wie man MySQL als normaler Benutzer laufen läßt

Der MySQL-Server `mysqld` kann von jedem beliebigen Benutzer gestartet werden und unter diesem laufen. Damit `mysqld` als Unix-Benutzer `benutzername` läuft, müssen Sie folgendes tun:

1. Halten Sie den Server an, falls er läuft (benutzen Sie `mysqladmin shutdown`).
2. Ändern Sie die Datenbankverzeichnisse und Dateien so, dass `benutzername` die Berechtigungen zum Lesen und Schreiben von Dateien darin hat (das müssen Sie eventuell als Unix-`root`-Benutzer machen):

   ```
   shell> chown -R benutzername /pfad/zu/mysql/datadir
   ```

   Wenn Verzeichnisse oder Dateien im MySQL-Daten-Verzeichnis symbolische Links sind, müssen Sie auch diesen Verknüpfungen folgen und die Verzeichnisse und Dateien, auf die sie zeigen, ändern. `chown -R` kann SymLinks für Sie folgen.
3. Starten Sie den Server als Benutzer `benutzername` oder, wenn Sie MySQL-Version 3.22 oder später benutzen, starten Sie `mysqld` als Unix-`root`-Benutzer und benutzen Sie die `--user=benutzername`-Option. `mysqld` schaltet um und läuft dann unter Unix-Benutzer `benutzername`, bevor er irgend welche Verbindungen annimmt.
4. Um den Server automatisch beim Hochfahren des Systems unter dem angegebenen Benutzernamen zu starten, fügen Sie zur `[mysqld]`-Gruppe der `/etc/my.cnf`-Optionendatei oder der `my.cnf`-Optionendatei im Datenverzeichnis des Servers eine `user`-Zeile hinzu, die den Benutzernamen angibt. Beispiel:

   ```
   [mysqld]
   user=benutzername
   ```

Nunmehr sollte Ihr `mysqld`-Prozess korrekt unter dem Unix-Benutzer `benutzername` laufen. Dennoch hat sich eins nicht geändert: Die Inhalte der Berechtigungstabellen. Vorgabemäßig (direkt nach dem Laufenlassen des Skripts `mysql_install_db`, das die Berechtigungstabellen installiert), ist der MySQL-Benutzer `root` der einzige Benutzer mit Zugriffsrechten auf die `mysql`-Datenbank. Er ist auch der einzige, der Datenbanken erzeugen und löschen kann. Wenn Sie diese Berechtigungen nicht geändert haben, sind sie noch gültig. Das sollte Sie nicht davon abhalten, auf MySQL als der MySQL-`root`-Benutzer zuzugreifen, wenn Sie als ein anderer Unix-Benutzer als `root` eingeloggt sind. Geben Sie einfach für Client-Programme die `-u root`-Option an.

Beachten Sie, dass der Zugriff auf MySQL als `root` (indem Sie `-u root` auf der Kommandozeile eingeben) *nichts* damit zu tun hat, dass Sie MySQL als Unix-`root`-Benutzer laufen lassen oder als irgend ein anderer Unix-Benutzer. Die Zugriffsberechtigungen und Benutzernamen von MySQL sind komplett unterschiedlich von den Unix-Benutzernamen. Die einzige Verbindung mit Unix-Benutzernamen besteht darin, dass ein Client versuchen wird, sich mit Ihrem Unix-Login-Namen als MySQL-Benutzernamen zu verbinden, wenn Sie beim Aufruf eines Client-Programms keine `-u`-Option angeben.

Wenn Ihre Unix-Box selbst nicht abgesichert ist, sollten Sie zumindest ein Passwort für den MySQL-`root`-Benutzer in den Berechtigungstabellen angeben. Ansonsten kann jeder beliebige Benutzer mit einem Konto auf der Maschine `mysql -u root datenbank` eingeben und tun, was er will.

A.3.3. Probleme mit Dateirechten

Wenn Sie Probleme mit Dateirechten haben, wenn `mysql` zum Beispiel folgende Fehlermeldung beim Erzeugen einer Tabelle ausgibt:

```
Error: Can't find file: 'pfad/mit/dateiname.frm' (Errcode: 13)
```

Kann es sein, dass die Umgebungsvariable `UMASK` falsch gesetzt ist, wenn `mysqld` startet. Der vorgabemäßige umask-Wert ist `0660`. Sie können dieses Verhalten ändern, indem Sie `safe_mysqld` wie folgt starten:

```
shell> UMASK=384  # = 600 in oktal
shell> export UMASK
shell> /pfad/zu/safe_mysqld &
```

Vorgabemäßig erzeugt MySQL Datenbank- und `RAID`-Verzeichnisse mit dem Berechtigungstyp 0700. Dieses Verhalten können

Sie durch Setzen der UMASK_DIR-Variablen ändern. Wenn Sie diese setzen, werden neue Verzeichnisse mit kombiniertem UMASK und UMASK_DIR erzeugt. Wenn Sie zum Beispiel Gruppenzugriff auf alle neuen Verzeichnisse geben wollen, können Sie folgendes tun:

```
shell> UMASK_DIR=504  # = 770 in oktal
shell> export UMASK_DIR
shell> /pfad/zu/safe_mysqld &
```

Ab MySQL-Version 3.23.25 nimmt MySQL an, dass die Werte für UMASK und UMASK_DIR in oktal angegeben sind, wenn sie mit einer 0 anfangen.

See Anhang F, *Umgebungsvariablen*.

A.4. Administrationsbezogene Themen

A.4.1. Was zu tun ist, wenn MySQL andauernd abstürzt

Alle MySQL-Versionen werden auf vielen Plattformen getestet, bevor sie herausgegeben werden. Das heißt nicht, dass es keinerlei Bugs in MySQL gibt, aber es heißt, dass, wenn es Bugs gibt, diese sehr wenige und schwer zu finden sind. Wenn Sie ein Problem haben, ist es immer hilfreich herauszufinden, was Ihr System zum Absturz bringt, weil Sie dann viel bessere Chancen haben, es schnell zu beheben.

Zunächst sollten Sie versuchen herauszufinden, ob das Problem darin besteht, dass Ihr mysqld-Daemon stirbt, oder ob Sie ein Problem mit Ihrem Client haben. Sie können herausfinden, seit wann Ihr mysqld-Server hochgefahren ist, indem Sie mysqladmin version ausführen. Wenn mysqld gestorben ist, finden Sie den Grund hierfür womöglich in der Datei mysql-daten-verzeichnis/`hostname`.err. See Abschnitt 5.9.1, „Die Fehler-Log-Datei“.

Viele Abstürze von MySQL werden durch beschädigte Index- oder Daten-Dateien verursacht. MySQL aktualisiert die Daten auf Platte mit dem write() Systemaufruf, nach jedem SQL-Statement und bevor der Client über das Ergebnis unterrichtet wird. (Das gilt nicht, wenn Sie mit delayed_key_writes fahren, denn in diesem Fall werden nur die Daten geschrieben.) Das bedeutet, dass die Daten sicher sind, selbst wenn mysqld abstürzt, weil das Betriebssystem sicherstellt, dass die nicht von MySQL zurückgeschriebenen Daten (flush) auf Platte zurückgeschrieben werden. Sie können MySQL zwingen, alles nach jedem SQL-Befehl auf Platte zurückzusynchronisieren, indem Sie mysqld mit --flush starten.

Das Gesagte bedeutet, dass Sie normalerweise keine beschädigten Tabellen erhalten sollten, ausser in folgenden Fällen:

- Jemand oder etwas killte mysqld oder die Maschine mitten während einer Aktualisierung.
- Sie haben einen Bug in mysqld gefunden, der dazu führte, dass er mitten während einer Aktualisierung starb.
- Jemand manipuliert die Daten- / Index-Dateien ausserhalb von **mysqld**, ohne die Tabelle korrekt zu sperren.
- Wenn Sie viele mysqld-Server auf denselben Daten auf einem System laufen lassen, das Dateisystem-Sperren nicht gut unterstützt (das wird normalerweise vom lockd-Daemon gehandhabt) oder wenn Sie mehrere Server mit --skip-locking fahren.
- Wenn Sie eine beschädigte Index- / Daten-Datei haben, die sehr falsche Daten enthält, die mysqld durcheinander brachten.
- Sie haben einen Bug im Datenspeicher-Code gefunden. Das ist nicht sehr wahrscheinlich, aber zumindest möglich. In diesem Fall können Sie versuchen, den Dateityp auf einen anderen Datenbank-Handler umzustellen, indem Sie ALTER TABLE auf eine reparierte Kopie der Tabelle anwenden!

Weil es sehr schwierig ist herauszufinden, warum etwas abstürzt, versuchen Sie zuerst herauszufinden, ob Dinge, die bei anderen funktionieren, bei Ihnen abstürzen, oder ob das nicht der Fall ist. Versuchen Sie bitte folgendes:

- Fahren Sie den mysqld-Daemon mit mysqladmin shutdown herunter und führen Sie myisamchk --silent --force */*.MYI auf alle Tabellen aus. Starten Sie den mysqld-Daemon erneut. Das stellt sicher, dass Sie von einem sauberen Ausgangszustand aus fahren. See Kapitel 5, *MySQL-Datenbankadministration*.
- Benutzen Sie mysqld --log und versuchen Sie den Informationen im Log zu entnehmen, ob eine bestimmte Anfrage den Server killt oder nicht. Etwa 95% aller Bugs beziehen sich auf eine bestimmte Anfrage! Normalerweise ist das eine der letzten Anfragen in der Log-Datei, direkt bevor MySQL neu startete. See Abschnitt 5.9.2, „Die allgemeine Anfragen-Log-Datei“. Wenn Sie MySQL wiederholt mit einer der Anfragen killen, selbst wenn Sie alle Tabellen direkt vor der Ausführung der Anfrage überprüft haben, haben Sie den Bug eingegrenzt und sollten dafür einen Bug-Bericht schreiben! See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.
- Versuchen Sie, einen Testfall herzustellen, den wir zur Reproduzierung des Problems verwenden können. See Abschnitt E.1.6, „Einen Testfall herstellen, wenn Sie Tabellenbeschädigung feststellen“.

- Versuchen Sie, die beigefügten mysql-test test und MySQL-Benchmarks laufen zu lassen. See Abschnitt 10.3.2, „MySQL-Test-Suite“. Sie können MySQL recht gut prüfen. Sie können den Benchmarks auch selbst Code hinzufügen, der Ihre Applikation simuliert! Die Benchmarks finden sich im `bench`-Verzeichnis in der Quelldistribution oder bei einer Binärdistribution im `sql-bench`-Verzeichnis unter Ihrem MySQL-Installationsverzeichnis.
- Probieren Sie `fork_test.pl` und `fork2_test.pl`.
- Wenn Sie MySQL zum Debuggen konfigurieren, ist es wesentlich einfacher, Informationen über mögliche Fehler zu erhalten, wenn etwas schief geht. Konfigurieren Sie MySQL mit der `--with-debug`-Option oder mit der `--with-debug=full`-Option für `configure` neu und kompilieren Sie neu. See Abschnitt E.1, „Einen MySQL-Server debuggen“.
- Wenn MySQL zum Debuggen konfiguriert wird, wird ein sicherer Speicher-Zuweiser (Memory Allocator) hinzugefügt, der einige Fehler finden kann. Ausserdem erfolgen etliche Ausgaben über das, was gerade geschieht.
- Haben Sie die neuesten Patches für Ihr Betriebssystem installiert?
- Benutzen Sie die `--skip-locking`-Option für `mysqld`. Auf manchen Systemen arbeitet der `lockd`-Sperrmanager nicht korrekt. Die `--skip-locking`-Option weist `mysqld` an, keine externen Sperren zu benutzen. (Das heißt, dass Sie nicht zwei `mysqld`-Server auf denselben Daten laufen lassen können und dass Sie vorsichtig sein müssen, wenn Sie `myisamchk` benutzen, aber es kann aufschlussreich sein, die Option testweise zu benutzen.)
- Haben Sie `mysqladmin -u root processlist` ausprobiert, wenn `mysqld` zu laufen scheint, aber nicht antwortet? Manchmal ist `mysqld` nicht komatös, obwohl es so aussieht. Das Problem kann darin bestehen, dass alle Verbindungen in Benutzung sind, oder es kann ein internes Sperrproblem vorliegen. `mysqladmin processlist` ist üblicherweise in der Lage, in solchen Fällen eine Verbindung aufzubauen und kann nützliche Informationen über die momentane Anzahl von Verbindungen und ihren Status liefern.
- Lassen Sie den Befehl `mysqladmin -i 5 status` oder `mysqladmin -i 5 -r status` in einem separaten Fenster laufen, um statistische Informationen auszugeben, während Sie Ihre anderen Anfragen laufen lassen.
- Versuchen Sie folgendes:

  1. Starten Sie `mysqld` von `gdb` aus (oder in einem anderen Debugger). See Abschnitt E.1.3, „mysqld unter gdb debuggen“.
  2. Lassen Sie Ihre Test-Skripts laufen.
  3. Geben Sie die Ablaufverfolgung (Backtrace) und die lokalen Variablen der untersten 3 Ebenen aus. In gdb können Sie das mit folgenden Befehle tun, wenn `mysqld` innerhalb von gdb abgestürzt ist:

     ```
     backtrace
     info local
     up
     info local
     up
     info local
     ```

     Mit gdb können Sie auch untersuchen, welchen Thread es gibt (mit `info thread` und zu einem speziellen Thread umschalten (mit `thread #`, wobei `#` die Thread-Kennung ist).
- Versuchen Sie, Ihre Applikation mit einem Perl-Skript zu simulieren, um MySQL zu zwingen, abzustürzen oder fehlerhaftes Verhalten an den Tag zu legen.
- Senden Sie einen normalen Bug-Bericht. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“. Geben Sie mehr Details an als üblich. Weil MySQL bei vielen Leuten funktioniert, kann es sein, dass der Absturz das Ergebnis von etwas ist, das nur auf Ihrem Computer existiert (beispielsweise ein Fehler, der aus Ihren besonderen Systembibliotheken resultiert).
- Wenn Sie ein Problem mit Tabellen haben, die Zeilen dynamischer Länge enthalten, und Sie nicht `BLOB/TEXT`-Spalten benutzen (sondern nur `VARCHAR`-Spalten), können Sie versuchen, alle `VARCHAR`- in `CHAR`-Spalten umzuwandeln, indem Sie `ALTER TABLE` verwenden. Das erzwingt, dass MySQL Zeilen fester Länge verwendet. Zeilen fester Länge benötigen etwas mehr Platz, sind aber fehlertoleranter gegenüber Beschädigungen!

  Der aktuelle Code mit dynamischen Zeilen ist bei MySQL AB seit mindestens drei Jahren ohne jedes Problem in Benutzung, aber naturgemäß sind Zeilen dynamischer Länge fehleranfälliger. Daher kann es eine gute Idee sein, das oben Gesagte auszuprobieren.

A.4.2. Wie ein vergessenes Passwort zurückgesetzt wird

Wenn Sie das `root`-Benutzerpasswort für MySQL vergessen haben, können Sie es mit folgender Prozedur wiederherstellen:

1. Fahren Sie den `mysqld`-Server durch Senden von `kill` (nicht `kill -9`) an den `mysqld`-Server herunter. Die Prozess-Kennung (PID) wird in einer `.pid`-Datei gespeichert, die sich normalerweise im MySQL-Datenbank-Verzeichnis befindet:

   ```
   kill `cat /mysql-daten-verzeichnis/hostname.pid`
   ```

   Hierfür müssen Sie entweder der Unix-`root`-Benutzer sein oder derselbe Benutzer, unter dem der Server läuft.

2. Starten Sie `mysqld` mit der `--skip-grant-tables`-Option neu.

3. Verbinden Sie sich mit dem `mysqld`-Server mit `mysql -h hostname mysql` und ändern Sie das Passwort mit einem `GRANT`-Befehl. See Abschnitt 5.3.1, „GRANT- und REVOKE-Syntax“. Sie können dasselbe auch mit `mysqladmin -h hostname -u benutzer password 'neues_passwort'` machen.

4. Laden Sie die Berechtigungstabellen neu mit `mysqladmin -h hostname flush-privileges` oder mit dem SQL-Befehl `FLUSH PRIVILEGES`.

Beachten Sie, dass nach dem Start von `mysqld` mit `--skip-grant-tables` jede Benutzung von `GRANT`-Befehlen zu einem `Unknown command`-Fehler führt, bis Sie `FLUSH PRIVILEGES` ausgeführt haben.

A.4.3. Wie MySQL mit vollen Festplatten umgeht

Wenn etwas hinsichtlich der Festplatte passiert, tut MySQL folgendes:

- Er prüft einmal pro Minute, um festzustellen, ob es genug Platz gibt, um die aktuelle Zeile zu schreiben oder nicht. Wenn genug Platz vorhanden ist, wird fortgefahren, als sei nichts geschehen.
- Alle 6 Minuten schreibt er einen Eintrag in die Log-Datei mit einer Warnung wegen voller Festplatte.

Um das Problem abzumildern, können Sie folgende Aktionen unternehmen:

- Um einfach weiterzumachen, müssen Sie lediglich genug Festplattenplatz freigeben, damit alle Datensätze eingefügt werden können.
- Um den Thread abzubrechen, müssen Sie `mysqladmin kill` an den Thread senden. Der Thread wird beim nächsten Mal, wenn er die Festplatte prüft (in 1 Minute) abgebrochen.
- Beachten Sie, dass eventuell ein anderer Thread auf die Tabelle wartet, die den Zustand ``Platte voll" verursachte. Wenn Sie mehrere ``gesperrte" Threads haben, kann es sein, dass Sie einen Thread killen, der wegen ``Platte voll" wartet, dass dafür aber ein anderer Thread weitermacht.

Ausnahmen zum obigen Verhalten treten bei der Benutzung von `REPAIR` oder `OPTIMIZE` auf, oder wenn die Indexe nach einem `LOAD DATA INFILE`- oder einem `ALTER TABLE`-Statement im Stapel erzeugt werden.

Alle obigen Befehle benutzen eventuell große temporäre Dateien, die - sich selbst überlassen - für den Rest des Systems große Probleme verursachen können. Wenn MySQL ein ``Platte voll" erhält, während irgend eine der obigen Operationen ausgeführt wird, entfernt er die großen temporären Dateien und markiert die Tabelle als beschädigt (ausser bei `ALTER TABLE`, wobei die alte Tabelle unverändert gelassen wird).

A.4.4. Wohin MySQL temporäre Dateien speichert

MySQL benutzt den Wert der `TMPDIR`-Umgebungsvariablen als Pfadnamen des Verzeichnisses, in dem temporäre Dateien gespeichert werden. Wenn Sie `TMPDIR` nicht gesetzt haben, benutzt MySQL die System-Vorgabe, die normalerweise `/tmp` oder `/usr/tmp` ist. Wenn das Dateisystem, das Ihr Verzeichnis für temporäre Dateien enthält, zu klein ist, sollten Sie `safe_mysqld` editieren, um `TMPDIR` so zu setzen, dass sie auf ein Verzeichnis in einem Dateisystem zeigt, wo Sie genug Platz haben! Sie können das temporäre Verzeichnis auch mit der `--tmpdir`-Option für `mysqld` setzen.

MySQL erzeugt alle temporären Dateien als versteckte Dateien. Das stellt sicher, dass die temporären Dateien entfernt werden, wenn `mysqld` beendet wird. Der Nachteil versteckter Dateien ist, dass Sie eine große temporäre Datei nicht sehen, die das Dateisystem auffüllt, in dem sich das Verzeichnis für temporäre Dateien befindet.

Zum Sortieren (`ORDER BY` oder `GROUP BY`) benutzt MySQL normalerweise ein oder zwei temporäre Dateien. Der maximal benötigte Speicherplatz ist:

```
(laenge_dessen_was_sortiert_wird + groesse_von(datenbank_zeiger)) *
anzahl_uebereinstimmender_zeilen * 2
```

`groesse_von(datenbank_zeiger)` ist üblicherweise 4, kann in Zukunft aber für wirklich große Tabellen anwachsen.

Bei einigen `SELECT`-Anfragen erzeugt MySQL zusätzliche temporäre SQL-Tabellen. Diese sind nicht versteckt und haben Namen der Form `SQL_*`.

`ALTER TABLE` erzeugt eine temporäre Tabelle im selben Verzeichnis, in dem sich die Original-Tabelle befindet.

A.4.5. Wie Sie die MySQL-Socket-Datei `/tmp/mysql.sock` schützen oder ändern

Wenn Sie Probleme damit haben, dass jeder beliebige den MySQL-Kommunikations-Socket `/tmp/mysql.sock` löschen kann, können Sie unter den meisten Versionen von Unix Ihr `/tmp`-Dateisystem schützen, indem Sie darauf das `sticky` Bit setzen. Loggen Sie sich als `root` ein und tun Sie folgendes:

```
shell> chmod +t /tmp
```

Das schützt Ihr `/tmp`-Dateisystem, so dass Dateien nur von ihren Besitzern oder dem Superuser (`root`) gelöscht werden können.

Sie können überprüfen, ob das `sticky` Bit gesetzt ist, indem Sie `ls -ld /tmp` ausführen. Wenn das letzte Berechtigungsbit `t` ist, ist das Bit gesetzt.

Sie können den Speicherort ändern, den MySQL benutzt, um die Socket-Datei Socket-Datei abzulegen, indem Sie eine der folgenden Prozeduren ausführen:

- Geben Sie den Pfad in einer globalen oder lokalen Optionsdatei an. Beispielsweise können Sie in `/etc/my.cnf` eintragen:

  ```
  [client]
  socket=pfad-fuer-socket-datei

  [mysqld]
  socket=pfad-fuer-socket-datei
  ```

  See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

- Geben Sie den Pfad auf der Kommandozeile für `safe_mysqld` und die meisten Clients mit der `--socket=pfad-fuer-socket-datei`-Option an.
- Geben Sie den Pfad zum Socket in der `MYSQL_UNIX_PORT`-Umgebungsvariablen an. variable.
- Definieren Sie den Pfad mit der `configure`-Option `--with-unix-socket-path=pfad-fuer-socket-datei`. See Abschnitt 3.3.3, „Typische `configure`-Optionen“.

Mit folgendem Befehl können Sie testen, ob der Socket funktioniert:

```
shell> mysqladmin --socket=/pfad/zu/socket version
```

A.4.6. Zeitzonen-Probleme

Wenn es Probleme damit gibt, dass `SELECT NOW()` Werte in GMT (Greenwich Mean Time) zurückgibt und nicht in Ihrer lokalen Zeit, müssen Sie die `TZ`-Umgebungsvariable auf Ihre aktuelle Zeitzone setzen. Das sollte für die Umgebung gemacht werden, in der der Server läuft, zum Beispiel in `safe_mysqld` oder `mysql.server`. See Anhang F, *Umgebungsvariablen*.

A.5. Anfragenbezogene Themen

A.5.1. Groß-/Kleinschreibung beim Suchen

Vorgabemäßig sind MySQL-Suchen unabhängig von der verwendeten Groß-/Kleinschreibung (obwohl es einige Zeichensätze gibt, die nie unabhängig von der verwendeten Groß-/Kleinschreibung sind, wie `tschechisch`). Wenn Sie daher mit `spalten_name LIKE 'a%'` suchen, erhalten Sie alle Spaltenwerte, die mit `A` oder `a` anfangen. Wenn Sie die Suche abhängig von der verwendeten Groß-/Kleinschreibung machen wollen, verwenden Sie etwas wie `INSTR(spalten_name, "A")=1`, um ein Präfix zu überprüfen, oder benutzen Sie `STRCMP(spalten_name, "A") = 0`, wenn der Spaltenwert exakt `"A"` sein muss.

Einfache Vergleichsoperationen (`>=, >, = , < , <=`, Sortieren und Gruppieren) basieren auf dem ``Sortierwert'' jedes Zeichens. Buchstaben mit demselben Sortierwert (wie E, e und é) werden als dasselbe Zeichen behandelt!

In älteren MySQL-Versionen wurden `LIKE`-Vergleiche mit dem Großschreibungswert jedes Zeichens durchgeführt (E == e, aber E

<> é). In neueren MySQL-Versionen funktioniert `LIKE` genau wie die anderen Vergleichsoperatoren.

Wenn Sie wollen, dass eine Spalte immer abhängig von der verwendeten Groß-/Kleinschreibung behandelt wird, deklarieren Sie sie als `BINARY`. See Abschnitt 7.5.3, „`CREATE TABLE`-Syntax“.

Wenn Sie chinesische Daten in der so genannten Big5-Kodierung verwenden, sollten Sie alle Zeichenspalten `BINARY` machen. Das funktioniert, weil die Sortierreihenfolge von Big5-Zeichen auf der Reihenfolge von ASCII-Codes basiert.

A.5.2. Probleme bei der Benutzung von `DATE`-Spalten

Das Format eines `DATE`-Werts ist `'YYYY-MM-DD'`. Gemäß ANSI-SQL ist kein anderes Format zulässig. Sie sollten dieses Format in `UPDATE`-Ausdrücken und in der WHERE-Klausel von `SELECT`-Statements benutzen. Beispiel:

```
mysql> SELECT * FROM tabelle WHERE date >= '1997-05-05';
```

Aus Gründen der Annehmlichkeit konvertiert MySQL automatisch ein Datum in eine Zahl, wenn das Datum in einem numerischen Zusammenhang benutzt wird (und umgekehrt). MySQL unterstützt ausserdem ein ``entspanntes" Zeichenkettenformat beim Aktualisieren und in einer `WHERE`-Klausel, die ein Datum mit einer `TIMESTAMP`-, `DATE`- oder einer `DATETIME`-Spalte vergleicht. (Entspannt heißt hierbei, dass jedes beliebige Satzzeichen als Trennzeichen zwischen Bestandteilen benutzt werden darf. Beispielsweise sind `'1998-08-15'` und `'1998#08#15'` äquivalent.) MySQL kann auch eine Zeichenkette umwandeln, die keine Trennzeichen enthält (wie `'19980815'`), vorausgesetzt, dass diese als Datum einen Sinn ergibt.

Das spezielle Datum `'0000-00-00'` kann als `'0000-00-00'` gespeichert und abgerufen werden. Wenn man ein `'0000-00-00'`-Datum über **MyODBC** benutzt, wird es automatisch in `NULL` umgewandelt (**MyODBC**-Version 2.50.12 und höher), weil ODBC diese Art von Datum nicht handhaben kann.

Weil MySQL die oben genannten Umwandlungen durchführt, funktionieren folgende Statements:

```
mysql> INSERT INTO tabelle (idate) VALUES (19970505);
mysql> INSERT INTO tabelle (idate) VALUES ('19970505');
mysql> INSERT INTO tabelle (idate) VALUES ('97-05-05');
mysql> INSERT INTO tabelle (idate) VALUES ('1997.05.05');
mysql> INSERT INTO tabelle (idate) VALUES ('1997 05 05');
mysql> INSERT INTO tabelle (idate) VALUES ('0000-00-00');

mysql> SELECT idate FROM tabelle WHERE idate >= '1997-05-05';
mysql> SELECT idate FROM tabelle WHERE idate >= 19970505;
mysql> SELECT mod(idate,100) FROM tabelle WHERE idate >= 19970505;
mysql> SELECT idate FROM tabelle WHERE idate >= '19970505';
```

Folgendes jedoch funktioniert nicht:

```
mysql> SELECT idate FROM tabelle WHERE STRCMP(idate,'19970505')=0;
```

`STRCMP()` ist eine Zeichenkettenfunktion, daher wird `idate` in eine Zeichenkette umgewandelt und ein Zeichenkettenvergleich durchgeführt. MySQL wandelt `'19970505'` nicht in ein Datum um und führt einen Datumsvergleich durch.

Beachten Sie, dass MySQL nicht prüft, ob ein Datum korrekt ist oder nicht. Wenn Sie ein falsches Datum wie `'1998-2-31'` speichern, wird das falsche Datum gespeichert. Wenn das Datum in keinen vernünftigen Wert umgewandelt werden kann, wird `0` im `DATE`-Feld gespeichert. Das ist hauptsächlich eine Sache der Geschwindigkeit, und wir sind der Meinung, dass es Sache der Applikation und nicht des Servers ist, Datumsangaben zu überprüfen.

A.5.3. Probleme mit `NULL`-Werten

Das Konzept des `NULL`-Wert ist eine häufige Quelle der Verwirrung für SQL-Anfänger. Diese denken häufig, `NULL` sei dasselbe wie eine leere Zeichenkette `''`. Das ist nicht der Fall! So sind zum Beispiel folgende Statements grundverschieden:

```
mysql> INSERT INTO meine_tabelle (Telefon) VALUES (NULL);
mysql> INSERT INTO meine_tabelle (Telefon) VALUES ("");
```

Beide Statements fügen einen Wert in die `Telefon`-Spalte ein, aber das erste fügt einen `NULL`-Wert und das zweite eine leere Zeichenkette ein. Die Bedeutung des ersten ist etwa ``Telefonnummer unbekannt" und des zweiten ``Keine Telefonnummer".

In SQL ist der `NULL`-Wert im Vergleich mit jedem anderen Wert immer UNWAHR (false), selbst im Vergleich mit `NULL`. Ein Ausdruck, der `NULL` enthält, erzeugt immer einen `NULL`-Wert, ausser wenn es in der Dokumentation der Operatoren und Funktionen, die im Ausdruck beteiligt sind, anders angegeben ist. Alle Spalten im folgenden Beispiel geben `NULL` zurück:

```
mysql> SELECT NULL,1+NULL,CONCAT('unsichtbar',NULL);
```

Wenn Sie nach Spaltenwerten suchen, die `NULL` sind, können Sie nicht `=NULL` benutzen. Folgendes Statement gibt keine Zeilen zurück, weil `ausdruck = NULL` für jeden beliebigen Ausdruck UNWAHR (false) ist:

```
mysql> SELECT * FROM meine_tabelle WHERE Telefon = NULL;
```

Um nach NULL-Werten zu suchen, müssen Sie IS NULL benutzen. Folgende Beispiele zeigen, wie Sie die NULL-Telefonnummer und die leere Telefonnummer finden:

```
mysql> SELECT * FROM meine_tabelle WHERE Telefon IS NULL;
mysql> SELECT * FROM meine_tabelle WHERE Telefon = "";
```

In MySQL können Sie - wie bei vielen anderen SQL-Servern - keine Spalten indexieren, die NULL-Werte enthalten dürfen. Sie müssen solche Spalten aus NOT NULL deklarieren. Sie dürfen in eine indexierte Spalte keine NULL-Werte einfügen.

Wenn Sie Daten mit LOAD DATA INFILE einlesen, werden leere Spalten mit '' aktualisiert. Wenn Sie einen NULL-Wert in einer Spalte haben wollen, müssen Sie in der Textdatei \N benutzen. Unter manchen Umständen kann auch das Literalwort 'NULL' benutzt werden. See Abschnitt 7.4.9, „LOAD DATA INFILE-Syntax“.

Wenn Sie ORDER BY benutzen, werden NULL-Werte zuerst angezeigt. Wenn Sie mit DESC in absteigender Reihenfolge sortieren, werden NULL-Werte zuletzt angezeigt. Wenn Sie GROUP BY benutzen, werden alle NULL-Werte als gleich betrachtet.

Um die Handhabung von NULL zu erleichtern, können Sie die IS NULL- und IS NOT NULL-Operatoren und die IFNULL()-Funktion benutzen.

Bei manchen Spaltentypen werden NULL-Werte besonders behandelt. Wenn Sie NULL in die erste TIMESTAMP-Spalte einer Tabelle einfügen, werden das aktuelle Datum und die aktuelle Zeit eingefügt. Wenn Sie NULL in eine AUTO_INCREMENT-Spalte einfügen, wird die nächste Zahl in der Zahlenfolge eingefügt.

A.5.4. Probleme mit `alias`

Sie können ein Alias verwenden, um auf eine Spalte im GROUP BY-, ORDER BY- oder HAVING-Teil zu verweisen. Aliase können auch verwendet werden, um Spalten bessere Namen zu geben:

```
SELECT SQRT(a*b) as wurzel FROM tabelle GROUP BY wurzel HAVING wurzel > 0;
SELECT id,COUNT(*) AS zaehl FROM tabelle GROUP BY id HAVING zaehl > 0;
SELECT id AS "kunden-kennung" FROM tabelle;
```

Beachten Sie, dass ANSI-SQL verbietet, in einer WHERE-Klausel auf ein Alias zu verweisen. Das liegt daran, dass der Spaltenwert möglicherweise noch nicht feststeht, wenn der WHERE-Code ausgeführt wird. Folgende Anfrage zum Beispiel ist **unzulässig**:

```
SELECT id,COUNT(*) AS zaehl FROM tabelle WHERE zaehl > 0 GROUP BY id;
```

Das WHERE-Statement wird ausgeführt, um festzulegen, welche Zeilen im GROUP BY-Teil enthalten sein sollen, während HAVING benutzt wird, um zu entscheiden, welche Zeilen der Ergebnismenge benutzt werden sollten.

A.5.5. Zeilen aus verwandten Tabellen löschen

Weil MySQL keine Sub-Selects oder die Benutzung von mehr als einer Tabelle im DELETE-Statement unterstützt, müssen Sie folgenden Ansatz wählen, um Zeilen aus zwei verwandten Tabellen zu löschen:

1. Wählen (SELECT) Sie die Zeilen auf der Grundlage einer WHERE-Bedingung in der Haupt-Tabelle aus.
2. Löschen (DELETE) Sie die Zeilen in der Haupt-Tabelle auf der Grundlage derselben Bedingung.
3. Löschen Sie die Zeilen aus der verwandten Tabelle, bei denen die verwandte Spalte in den ausgewählten Zeilen vorkommt (DELETE FROM verwandte_tabelle WHERE verwandte_spalte IN (ausgewaehlte_zeilen).

Wenn die Gesamtzahl von Zeichen in der Anfrage mit verwandte_spalte mehr als 1.048.576 beträgt (der Vorgabewert von max_allowed_packet, sollten Sie sie in kleinere Teile aufspalten und mehrfache DELETE-Statements ausführen. Wahrscheinlich geht das Löschen (DELETE) am Schnellsten, wenn Sie nur 100 bis 1000 verwandte_spalte-Kennungen pro Anfrage löschen, wenn verwandte_spalte ein Index ist. Wenn verwandte_spalte kein Index ist, ist die Geschwindigkeit unabhängig von der Anzahl von Argumenten in der IN-Klausel.

A.5.6. Probleme bei keinen übereinstimmenden Zeilen lösen

Wenn Sie eine komplizierte Anfrage haben, die viele Tabellen hat und keine Zeilen zurückgibt, sollten Sie folgende Prozedur benutzen, um herauszufinden, was bei Ihrer Anfrage falsch ist:

1. Testen Sie die Anfrage mit `EXPLAIN` und prüfen Sie, ob Sie etwas finden können, das offensichtlich falsch ist. See Abschnitt 6.2.1, „`EXPLAIN`-Syntax (Informationen über ein `SELECT` erhalten)“.

2. Wählen Sie in der `WHERE`-Klausel nur die Felder aus, die benutzt werden.

3. Entfernen Sie nacheinander Tabelle für Tabelle aus der Anfrage, bis sie Zeilen zurückgibt. Wenn die Tabellen Groß sind, ist es eine gute Idee, `LIMIT 10` bei der Anfrage zu benutzen.

4. Machen Sie ein `SELECT` für die Spalte, die mit einer Zeile hätte übereinstimmen sollen, gegen die Tabelle, die als letzte aus der Anfrage entfernt wurde.

5. Wenn Sie `FLOAT`- oder `DOUBLE`-Spalten mit Zahlen vergleichen, die Dezimalstellen haben, können Sie nicht `=` benutzen! Das Problem tritt in den meisten Computersprachen auf, weil Fließkommawerte keine exakten Werte sind:

   ```
   mysql> SELECT * FROM tabelle WHERE float_spalte=3.5;
   mysql> SELECT * FROM tabelle WHERE float_spalte between 3.45 und 3.55;
   ```

   In den meisten Fällen kann dies durch Umwandlung von `FLOAT` in `DOUBLE` behoben werden!

6. Wenn Sie immer noch nicht herausfinden können, was schief geht, erzeugen Sie einen minimalen Test, der mit `mysql test < anfrage.sql` laufen gelassen werden kann, um Ihre Probleme aufzuzeigen. Sie können eine Testdatei mit `mysqldump --quick datenbanktabellen > anfrage.sql` erzeugen. Öffnen Sie die Datei in einem Editor, entfernen Sie ein paar Einfügezeilen (wenn es davon zu viele gibt) und fügen Sie Ihr SELECT--Statement am Ende der Datei an.

   Testen Sie, ob es hiermit immer noch das Problem gibt:

   ```
   shell> mysqladmin create test2
   shell> mysql test2 < anfrage.sql
   ```

   Schicken Sie die Testdatei mittels `mysqlbug` an `<mysql@lists.mysql.com>`.

A.6. Tabellendefinitionsbezogene Themen

A.6.1. Probleme mit `ALTER TABLE`.

`ALTER TABLE` ändert eine Tabelle zum aktuellen Zeichensatz. Wenn Sie während `ALTER TABLE` einen Fehler wegen doppelter Schlüsseleinträge bekommen, liegt das entweder daran, dass die neuen Zeichensätze auf bei Schlüsseln auf dieselben Werte gemappt sind, oder dass die Tabelle beschädigt ist, wobei Sie `REPAIR TABLE` auf die Tabelle laufen lassen sollten.

Wenn `ALTER TABLE` mit einem Fehler wie folgt stirbt:

```
Error on rename of './datenbank/name.frm' to './datenbank/B-a.frm' (Errcode: 17)
```

Kann das Problem darin bestehen, dass MySQL bei einem vorhergehenden `ALTER TABLE` abgestürzt ist und es eine alte Tabelle namens `A-etwas` oder `B-etwas` gibt, die herum liegt. Gehen Sie in diesem Fall ins MySQL-Daten-Verzeichnis und löschen Sie alle Dateien, die Namen wie `A-` oder `B-` haben. (Statt löschen können Sie sie auch an eine andere Stelle verschieben.)

`ALTER TABLE` funktioniert auf folgenden Weise:

- Erzeugt eine neue Tabellen namens `A-xxx` mit den angeforderten Änderungen.
- Alle Zeilen der alten Tabelle werden nach `A-xxx` kopiert.
- Die alte Tabelle wird in `B-xxx` umbenannt.
- `A-xxx` wird in Ihren alten Tabellennamen umbenannt.
- `B-xxx` wird gelöscht.

Wenn etwas bei dieser Umbennungsoperation fehlschlägt, versucht MySQL, die Änderungen rückgängig zu machen. Wenn etwas Schwerwiegendes schief geht (was natürlich passieren kann), läßt MySQL eventuell die alte Tabelle als `B-xxx`, aber ein einfaches Umbenennen auf Systemebene sollte Ihre Daten zurückbringen.

A.6.2. Wie man die Reihenfolge der Spalten in einer Tabelle ändert

Im großen und Ganzen geht es bei SQL darum, die Applikation vom Daten-Speicherformat zu abstrahieren. Sie sollten immer die

Reihenfolge angeben, in der Sie Ihre Daten abrufen wollen. Beispiel:

```
SELECT spalten_name1, spalten_name2, spalten_name3 FROM tabelle;
```

Das gibt die Spalten in der Reihenfolge `spalten_name1`, `spalten_name2`, `spalten_name3` zurück, wohingegen:

```
SELECT spalten_name1, spalten_name3, spalten_name2 FROM tabelle;
```

die Spalten in der Reihenfolge `spalten_name1`, `spalten_name3`, `spalten_name2` zurückgibt.

Sie sollten in einer Applikation **NIE** `SELECT *` benutzen und die Spalten basierend auf ihrer Position abrufen, weil die Reihenfolge, in der Spalten zurückgegeben werden, im Zeitablauf **NICHT** garantiert werden kann. Eine einfache Änderung in Ihrer Datenbank kann dazu führen, dass Ihre Applikation dramatisch scheitert.

Wenn Sie dennoch die Spalten-Reihenfolge ändern wollen, können Sie das wie folgt tun:

1. Erzeugen Sie eine neue Tabelle mit den Spalten in der richtigen Reihenfolge.
2. Führen Sie `INSERT INTO neue_tabelle SELECT felder-in-der-reihenfolge-von-neue_tabelle FROM alte_tabelle` aus.
3. Löschen Sie `alte_tabelle` oder benennen Sie sie um.
4. Führen Sie `ALTER TABLE neue_tabelle RENAME alte_tabelle` aus.

A.6.3. TEMPORARY TABLE-Probleme

Im Folgenden eine Auflistung der Beschränkungen bei `TEMPORARY TABLES`.

- Eine temporäre Tabelle kann nur vom Typ `HEAP`, `ISAM` oder `MyISAM` sein.
- Sie können temporäre Tabellen nicht mehr als einmal in derselben Anfrage benutzen. Folgendes zum Beispiel funktioniert nicht:

  ```
  select * from temporary_table, temporary_table as t2;
  ```

  Das soll in Version 4.0 behoben werden.
- Sie können kein `RENAME` auf eine `TEMPORARY`-Tabelle benutzen. Beachten Sie, dass `ALTER TABLE alter_name RENAME neuer_name` dagegen funktioniert! Das soll in Version 4.0 behoben werden.

Anhang B. Fehlercodes und -meldungen

Dieses Kapitel listet die Fehler auf, die auftreten können, wenn Sie MySQL-Aufrufe von irgendeiner Sprache aus tätigen. Die erste Liste stellt Fehlermeldungen des Servers dar, die zweite Liste Fehlermeldungen der Client-Programme.

Fehlermeldungen des Servers befinden sich in folgenden Dateien:

- Die Fehlerwerte und die Symbole in Klammern entsprechen den Definitionen in der MySQL-Quelldatei `include/mysqld_error.h`.
- Die SQLSTATE-Werte entsprechen den Definitionen in der MySQL-Quelldatei `include/sql_state.h`.

  SQLSTATE-Fehlercodes werden erst ab MySQL-Version 4.1 angezeigt. SQLSTATE-Codes wurden aus Gründen der Kompatibilität mit X/Open, ANSI und ODBC hinzugefügt.
- Die Meldungen entsprechen den Fehlermeldungen, die in der Datei `share/errmsg.txt` aufgeführt sind. `%d` und `%s` stellen Zahlen beziehungsweise Zeichenketten dar, die in den Fehlermeldungen ersetzt werden, wenn diese angezeigt werden.

Weil sie häufig aktualisiert werden, enhalten die genannten Dateien möglicherweise zusätzliche Informationen, die hier nicht aufgeführt sind.

- Fehler: `1000` SQLSTATE: `HY000` (`ER_HASHCHK`)

  Meldung: hashchk
- Fehler: `1001` SQLSTATE: `HY000` (`ER_NISAMCHK`)

  Meldung: isamchk
- Fehler: `1002` SQLSTATE: `HY000` (`ER_NO`)

  Meldung: Nein
- Fehler: `1003` SQLSTATE: `HY000` (`ER_YES`)

  Meldung: Ja
- Fehler: `1004` SQLSTATE: `HY000` (`ER_CANT_CREATE_FILE`)

  Meldung: Kann Datei '%s' nicht erzeugen (Fehler: %d)
- Fehler: `1005` SQLSTATE: `HY000` (`ER_CANT_CREATE_TABLE`)

  Meldung: Kann Tabelle '%s' nicht erzeugen (Fehler: %d)
- Fehler: `1006` SQLSTATE: `HY000` (`ER_CANT_CREATE_DB`)

  Meldung: Kann Datenbank '%s' nicht erzeugen (Fehler: %d)
- Fehler: `1007` SQLSTATE: `HY000` (`ER_DB_CREATE_EXISTS`)

  Meldung: Kann Datenbank '%s' nicht erzeugen. Datenbank '%s' existiert bereits
- Fehler: `1008` SQLSTATE: `HY000` (`ER_DB_DROP_EXISTS`)

  Meldung: Kann Datenbank '%s' nicht löschen. Keine Datenbank '%s' vorhanden
- Fehler: `1009` SQLSTATE: `HY000` (`ER_DB_DROP_DELETE`)

  Meldung: Fehler beim Löschen der Datenbank ('%s' kann nicht gelöscht werden, Fehlernuumer: %d)
- Fehler: `1010` SQLSTATE: `HY000` (`ER_DB_DROP_RMDIR`)

  Meldung: Fehler beim Löschen der Datenbank (Verzeichnis '%s' kann nicht gelöscht werden, Fehlernummer: %d)
- Fehler: `1011` SQLSTATE: `HY000` (`ER_CANT_DELETE_FILE`)

  Meldung: Fehler beim Löschen von '%s' (Fehler: %d)

- Fehler: 1012 SQLSTATE: HY000 (ER_CANT_FIND_SYSTEM_REC)

  Meldung: Datensatz in der Systemtabelle nicht lesbar
- Fehler: 1013 SQLSTATE: HY000 (ER_CANT_GET_STAT)

  Meldung: Kann Status von '%s' nicht ermitteln (Fehler: %d)
- Fehler: 1014 SQLSTATE: HY000 (ER_CANT_GET_WD)

  Meldung: Kann Arbeitsverzeichnis nicht ermitteln (Fehler: %d)
- Fehler: 1015 SQLSTATE: HY000 (ER_CANT_LOCK)

  Meldung: Datei kann nicht gesperrt werden (Fehler: %d)
- Fehler: 1016 SQLSTATE: HY000 (ER_CANT_OPEN_FILE)

  Meldung: Datei '%s' nicht öffnen (Fehler: %d)
- Fehler: 1017 SQLSTATE: HY000 (ER_FILE_NOT_FOUND)

  Meldung: Kann Datei '%s' nicht finden (Fehler: %d)
- Fehler: 1018 SQLSTATE: HY000 (ER_CANT_READ_DIR)

  Meldung: Verzeichnis von '%s' nicht lesbar (Fehler: %d)
- Fehler: 1019 SQLSTATE: HY000 (ER_CANT_SET_WD)

  Meldung: Kann nicht in das Verzeichnis '%s' wechseln (Fehler: %d)
- Fehler: 1020 SQLSTATE: HY000 (ER_CHECKREAD)

  Meldung: Datensatz hat sich seit dem letzten Zugriff auf Tabelle '%s' geändert
- Fehler: 1021 SQLSTATE: HY000 (ER_DISK_FULL)

  Meldung: Festplatte voll (%s). Warte, bis jemand Platz schafft ...
- Fehler: 1022 SQLSTATE: 23000 (ER_DUP_KEY)

  Meldung: Kann nicht speichern, Grund: doppelter Schlüssel in Tabelle '%s'
- Fehler: 1023 SQLSTATE: HY000 (ER_ERROR_ON_CLOSE)

  Meldung: Fehler beim Schließen von '%s' (Fehler: %d)
- Fehler: 1024 SQLSTATE: HY000 (ER_ERROR_ON_READ)

  Meldung: Fehler beim Lesen der Datei '%s' (Fehler: %d)
- Fehler: 1025 SQLSTATE: HY000 (ER_ERROR_ON_RENAME)

  Meldung: Fehler beim Umbenennen von '%s' in '%s' (Fehler: %d)
- Fehler: 1026 SQLSTATE: HY000 (ER_ERROR_ON_WRITE)

  Meldung: Fehler beim Speichern der Datei '%s' (Fehler: %d)
- Fehler: 1027 SQLSTATE: HY000 (ER_FILE_USED)

  Meldung: '%s' ist für Änderungen gesperrt
- Fehler: 1028 SQLSTATE: HY000 (ER_FILSORT_ABORT)

  Meldung: Sortiervorgang abgebrochen
- Fehler: 1029 SQLSTATE: HY000 (ER_FORM_NOT_FOUND)

  Meldung: View '%s' existiert für '%s' nicht

- Fehler: 1030 SQLSTATE: HY000 (ER_GET_ERRNO)

  Meldung: Fehler %d (Tabellenhandler)

- Fehler: 1031 SQLSTATE: HY000 (ER_ILLEGAL_HA)

  Meldung: Diese Option gibt es nicht (Tabellenhandler)

- Fehler: 1032 SQLSTATE: HY000 (ER_KEY_NOT_FOUND)

  Meldung: Kann Datensatz nicht finden

- Fehler: 1033 SQLSTATE: HY000 (ER_NOT_FORM_FILE)

  Meldung: Falsche Information in Datei '%s'

- Fehler: 1034 SQLSTATE: HY000 (ER_NOT_KEYFILE)

  Meldung: Falsche Schlüssel-Datei für Tabelle '%s'. versuche zu reparieren

- Fehler: 1035 SQLSTATE: HY000 (ER_OLD_KEYFILE)

  Meldung: Alte Schlüssel-Datei für Tabelle '%s'. Bitte reparieren

- Fehler: 1036 SQLSTATE: HY000 (ER_OPEN_AS_READONLY)

  Meldung: '%s' ist nur lesbar

- Fehler: 1037 SQLSTATE: HY001 (ER_OUTOFMEMORY)

  Meldung: Kein Speicher vorhanden (%d Bytes benötigt). Bitte Server neu starten

- Fehler: 1038 SQLSTATE: HY001 (ER_OUT_OF_SORTMEMORY)

  Meldung: Kein Speicher zum Sortieren vorhanden. sort_buffer_size sollte erhöht werden

- Fehler: 1039 SQLSTATE: HY000 (ER_UNEXPECTED_EOF)

  Meldung: Unerwartetes Ende beim Lesen der Datei '%s' (Fehler: %d)

- Fehler: 1040 SQLSTATE: 08004 (ER_CON_COUNT_ERROR)

  Meldung: Zu viele Verbindungen

- Fehler: 1041 SQLSTATE: HY000 (ER_OUT_OF_RESOURCES)

  Meldung: Kein Speicher mehr vorhanden. Prüfen Sie, ob mysqld oder ein anderer Prozess allen Speicher verbraucht. Wenn nicht, sollten Sie mit 'ulimit' dafür sorgen, dass mysqld mehr Speicher benutzen darf, oder mehr Swap-Speicher einrichten

- Fehler: 1042 SQLSTATE: 08S01 (ER_BAD_HOST_ERROR)

  Meldung: Kann Hostnamen für diese Adresse nicht erhalten

- Fehler: 1043 SQLSTATE: 08S01 (ER_HANDSHAKE_ERROR)

  Meldung: Schlechter Handshake

- Fehler: 1044 SQLSTATE: 42000 (ER_DBACCESS_DENIED_ERROR)

  Meldung: Benutzer '%s'@'%s' hat keine Zugriffsberechtigung für Datenbank '%s'

- Fehler: 1045 SQLSTATE: 28000 (ER_ACCESS_DENIED_ERROR)

  Meldung: Benutzer '%s'@'%s' hat keine Zugriffsberechtigung (verwendetes Passwort: %s)

- Fehler: 1046 SQLSTATE: 3D000 (ER_NO_DB_ERROR)

  Meldung: Keine Datenbank ausgewählt

- Fehler: 1047 SQLSTATE: 08S01 (ER_UNKNOWN_COM_ERROR)

  Meldung: Unbekannter Befehl

- Fehler: 1048 SQLSTATE: 23000 (ER_BAD_NULL_ERROR)

  Meldung: Feld '%s' darf nicht NULL sein
- Fehler: 1049 SQLSTATE: 42000 (ER_BAD_DB_ERROR)

  Meldung: Unbekannte Datenbank '%s'
- Fehler: 1050 SQLSTATE: 42S01 (ER_TABLE_EXISTS_ERROR)

  Meldung: Tabelle '%s' bereits vorhanden
- Fehler: 1051 SQLSTATE: 42S02 (ER_BAD_TABLE_ERROR)

  Meldung: Unbekannte Tabelle '%s'
- Fehler: 1052 SQLSTATE: 23000 (ER_NON_UNIQ_ERROR)

  Meldung: Spalte '%s' in %s ist nicht eindeutig
- Fehler: 1053 SQLSTATE: 08S01 (ER_SERVER_SHUTDOWN)

  Meldung: Der Server wird heruntergefahren
- Fehler: 1054 SQLSTATE: 42S22 (ER_BAD_FIELD_ERROR)

  Meldung: Unbekanntes Tabellenfeld '%s' in %s
- Fehler: 1055 SQLSTATE: 42000 (ER_WRONG_FIELD_WITH_GROUP)

  Meldung: '%s' ist nicht in GROUP BY vorhanden
- Fehler: 1056 SQLSTATE: 42000 (ER_WRONG_GROUP_FIELD)

  Meldung: Gruppierung über '%s' nicht möglich
- Fehler: 1057 SQLSTATE: 42000 (ER_WRONG_SUM_SELECT)

  Meldung: Die Verwendung von Summierungsfunktionen und Spalten im selben Befehl ist nicht erlaubt
- Fehler: 1058 SQLSTATE: 21S01 (ER_WRONG_VALUE_COUNT)

  Meldung: Die Anzahl der Spalten entspricht nicht der Anzahl der Werte
- Fehler: 1059 SQLSTATE: 42000 (ER_TOO_LONG_IDENT)

  Meldung: Name des Bezeichners '%s' ist zu lang
- Fehler: 1060 SQLSTATE: 42S21 (ER_DUP_FIELDNAME)

  Meldung: Doppelter Spaltenname vorhanden: '%s'
- Fehler: 1061 SQLSTATE: 42000 (ER_DUP_KEYNAME)

  Meldung: Doppelter Name für Schlüssel (Key) vorhanden: '%s'
- Fehler: 1062 SQLSTATE: 23000 (ER_DUP_ENTRY)

  Meldung: Doppelter Eintrag '%s' für Schlüssel %d
- Fehler: 1063 SQLSTATE: 42000 (ER_WRONG_FIELD_SPEC)

  Meldung: Falsche Spaltenangaben für Spalte '%s'
- Fehler: 1064 SQLSTATE: 42000 (ER_PARSE_ERROR)

  Meldung: %s bei '%s' in Zeile %d
- Fehler: 1065 SQLSTATE: HY000 (ER_EMPTY_QUERY)

  Meldung: Leere Abfrage

- Fehler: 1066 SQLSTATE: 42000 (ER_NONUNIQ_TABLE)

  Meldung: Tabellenname/Alias '%s' nicht eindeutig
- Fehler: 1067 SQLSTATE: 42000 (ER_INVALID_DEFAULT)

  Meldung: Fehlerhafter Vorgabewert (DEFAULT): '%s'
- Fehler: 1068 SQLSTATE: 42000 (ER_MULTIPLE_PRI_KEY)

  Meldung: Mehrfacher Primärschlüssel (PRIMARY KEY) definiert
- Fehler: 1069 SQLSTATE: 42000 (ER_TOO_MANY_KEYS)

  Meldung: Zu viele Schlüssel definiert. Maximal %d Schlüssel erlaubt
- Fehler: 1070 SQLSTATE: 42000 (ER_TOO_MANY_KEY_PARTS)

  Meldung: Zu viele Teilschlüssel definiert. Maximal sind %d Teilschlüssel erlaubt
- Fehler: 1071 SQLSTATE: 42000 (ER_TOO_LONG_KEY)

  Meldung: Schlüssel ist zu lang. Die maximale Schlüssellänge beträgt %d
- Fehler: 1072 SQLSTATE: 42000 (ER_KEY_COLUMN_DOES_NOT_EXITS)

  Meldung: In der Tabelle gibt es keine Schlüsselspalte '%s'
- Fehler: 1073 SQLSTATE: 42000 (ER_BLOB_USED_AS_KEY)

  Meldung: BLOB-Feld '%s' kann beim verwendeten Tabellentyp nicht als Schlüssel verwendet werden
- Fehler: 1074 SQLSTATE: 42000 (ER_TOO_BIG_FIELDLENGTH)

  Meldung: Feldlänge für Feld '%s' zu groß (maximal %d). BLOB-Feld verwenden!
- Fehler: 1075 SQLSTATE: 42000 (ER_WRONG_AUTO_KEY)

  Meldung: Falsche Tabellendefinition. Es darf nur ein Auto-Feld geben und dieses muss als Schlüssel definiert werden
- Fehler: 1076 SQLSTATE: HY000 (ER_READY)

  Meldung: %s: Bereit für Verbindungen
- Fehler: 1077 SQLSTATE: HY000 (ER_NORMAL_SHUTDOWN)

  Meldung: %s: Normal heruntergefahren
- Fehler: 1078 SQLSTATE: HY000 (ER_GOT_SIGNAL)

  Meldung: %s: Signal %d erhalten. Abbruch!
- Fehler: 1079 SQLSTATE: HY000 (ER_SHUTDOWN_COMPLETE)

  Meldung: %s: Heruntergefahren (shutdown)
- Fehler: 1080 SQLSTATE: 08S01 (ER_FORCING_CLOSE)

  Meldung: %s: Thread %ld zwangsweise beendet. Benutzer: '%s'
- Fehler: 1081 SQLSTATE: 08S01 (ER_IPSOCK_ERROR)

  Meldung: Kann IP-Socket nicht erzeugen
- Fehler: 1082 SQLSTATE: 42S12 (ER_NO_SUCH_INDEX)

  Meldung: Tabelle '%s' besitzt keinen wie den in CREATE INDEX verwendeten Index. Index neu anlegen
- Fehler: 1083 SQLSTATE: 42000 (ER_WRONG_FIELD_TERMINATORS)

  Meldung: Feldbegrenzer-Argument ist nicht in der erwarteten Form. Bitte im Handbuch nachlesen

- Fehler: 1084 SQLSTATE: 42000 (ER_BLOBS_AND_NO_TERMINATED)

  Meldung: Eine feste Zeilenlänge kann für BLOB-Felder nicht verwendet werden. Bitte 'fields terminated by' verwenden

- Fehler: 1085 SQLSTATE: HY000 (ER_TEXTFILE_NOT_READABLE)

  Meldung: Datei '%s' muss im Datenbank-Verzeichnis vorhanden und lesbar für alle sein

- Fehler: 1086 SQLSTATE: HY000 (ER_FILE_EXISTS_ERROR)

  Meldung: Datei '%s' bereits vorhanden

- Fehler: 1087 SQLSTATE: HY000 (ER_LOAD_INFO)

  Meldung: Datensätze: %ld Gelöscht: %ld Ausgelassen: %ld Warnungen: %ld

- Fehler: 1088 SQLSTATE: HY000 (ER_ALTER_INFO)

  Meldung: Datensätze: %ld Duplikate: %ld

- Fehler: 1089 SQLSTATE: HY000 (ER_WRONG_SUB_KEY)

  Meldung: Falscher Unterteilschlüssel. Der verwendete Schlüsselteil ist entweder kein String, die verwendete Länge ist länger als der Teilschlüssel oder der Tabellenhandler unterstützt keine Unterteilschlüssel

- Fehler: 1090 SQLSTATE: 42000 (ER_CANT_REMOVE_ALL_FIELDS)

  Meldung: Mit ALTER TABLE können nicht alle Felder auf einmal gelöscht werden. Dafür DROP TABLE verwenden

- Fehler: 1091 SQLSTATE: 42000 (ER_CANT_DROP_FIELD_OR_KEY)

  Meldung: Kann '%s' nicht löschen. Existiert das Feld / der Schlüssel?

- Fehler: 1092 SQLSTATE: HY000 (ER_INSERT_INFO)

  Meldung: Datensätze: %ld Duplikate: %ld Warnungen: %ld

- Fehler: 1093 SQLSTATE: HY000 (ER_UPDATE_TABLE_USED)

  Meldung: Die Verwendung der zu aktualisierenden Zieltabelle '%s' ist in der FROM-Klausel nicht zulässig.

- Fehler: 1094 SQLSTATE: HY000 (ER_NO_SUCH_THREAD)

  Meldung: Unbekannte Thread-ID: %lu

- Fehler: 1095 SQLSTATE: HY000 (ER_KILL_DENIED_ERROR)

  Meldung: Sie sind nicht Eigentümer von Thread %lu

- Fehler: 1096 SQLSTATE: HY000 (ER_NO_TABLES_USED)

  Meldung: Keine Tabellen verwendet

- Fehler: 1097 SQLSTATE: HY000 (ER_TOO_BIG_SET)

  Meldung: Zu viele Strings für SET-Spalte %s angegeben

- Fehler: 1098 SQLSTATE: HY000 (ER_NO_UNIQUE_LOGFILE)

  Meldung: Kann keinen eindeutigen Dateinamen für die Logdatei %s erzeugen (1-999)

- Fehler: 1099 SQLSTATE: HY000 (ER_TABLE_NOT_LOCKED_FOR_WRITE)

  Meldung: Tabelle '%s' ist mit Lesesperre versehen und kann nicht aktualisiert werden

- Fehler: 1100 SQLSTATE: HY000 (ER_TABLE_NOT_LOCKED)

  Meldung: Tabelle '%s' wurde nicht mit LOCK TABLES gesperrt

- Fehler: 1101 SQLSTATE: 42000 (ER_BLOB_CANT_HAVE_DEFAULT)

  Meldung: BLOB-Feld '%s' darf keinen Vorgabewert (DEFAULT) haben

- Fehler: 1102 SQLSTATE: 42000 (ER_WRONG_DB_NAME)

  Meldung: Unerlaubter Datenbankname '%s'

- Fehler: 1103 SQLSTATE: 42000 (ER_WRONG_TABLE_NAME)

  Meldung: Unerlaubter Tabellenname '%s'

- Fehler: 1104 SQLSTATE: 42000 (ER_TOO_BIG_SELECT)

  Meldung: Die Ausführung des SELECT würde zu viele Datensätze untersuchen und wahrscheinlich sehr lange dauern. Bitte WHERE-Klausel überprüfen oder gegebenenfalls SET SQL_BIG_SELECTS=1 oder SET SQL_MAX_JOIN_SIZE=# verwenden

- Fehler: 1105 SQLSTATE: HY000 (ER_UNKNOWN_ERROR)

  Meldung: Unbekannter Fehler

- Fehler: 1106 SQLSTATE: 42000 (ER_UNKNOWN_PROCEDURE)

  Meldung: Unbekannte Prozedur '%s'

- Fehler: 1107 SQLSTATE: 42000 (ER_WRONG_PARAMCOUNT_TO_PROCEDURE)

  Meldung: Falsche Parameterzahl für Prozedur '%s'

- Fehler: 1108 SQLSTATE: HY000 (ER_WRONG_PARAMETERS_TO_PROCEDURE)

  Meldung: Falsche Parameter für Prozedur '%s'

- Fehler: 1109 SQLSTATE: 42S02 (ER_UNKNOWN_TABLE)

  Meldung: Unbekannte Tabelle '%s' in '%s'

- Fehler: 1110 SQLSTATE: 42000 (ER_FIELD_SPECIFIED_TWICE)

  Meldung: Feld '%s' wurde zweimal angegeben

- Fehler: 1111 SQLSTATE: HY000 (ER_INVALID_GROUP_FUNC_USE)

  Meldung: Falsche Verwendung einer Gruppierungsfunktion

- Fehler: 1112 SQLSTATE: 42000 (ER_UNSUPPORTED_EXTENSION)

  Meldung: Tabelle '%s' verwendet eine Extension, die in dieser MySQL-Version nicht verfügbar ist

- Fehler: 1113 SQLSTATE: 42000 (ER_TABLE_MUST_HAVE_COLUMNS)

  Meldung: Eine Tabelle muß mindestens 1 Spalte besitzen

- Fehler: 1114 SQLSTATE: HY000 (ER_RECORD_FILE_FULL)

  Meldung: Tabelle '%s' ist voll

- Fehler: 1115 SQLSTATE: 42000 (ER_UNKNOWN_CHARACTER_SET)

  Meldung: Unbekannter Zeichensatz: '%s'

- Fehler: 1116 SQLSTATE: HY000 (ER_TOO_MANY_TABLES)

  Meldung: Zu viele Tabellen. MySQL kann in einem Join maximal %d Tabellen verwenden

- Fehler: 1117 SQLSTATE: HY000 (ER_TOO_MANY_FIELDS)

  Meldung: Zu viele Spalten

- Fehler: 1118 SQLSTATE: 42000 (ER_TOO_BIG_ROWSIZE)

  Meldung: Zeilenlänge zu groß. Die maximale Spaltenlänge für den verwendeten Tabellentyp (ohne BLOB-Felder) beträgt %d. Einige Felder müssen in BLOB oder TEXT umgewandelt werden

- Fehler: 1119 SQLSTATE: HY000 (ER_STACK_OVERRUN)

Meldung: Thread-Stack-Überlauf. Benutzt: %ld von %ld Stack. 'mysqld -O thread_stack=#' verwenen, um notfalls einen größeren Stack anzulegen

- Fehler: `1120` SQLSTATE: `42000` (`ER_WRONG_OUTER_JOIN`)

  Meldung: OUTER JOIN enthält fehlerhafte Abhängigkeiten. In ON verwendete Bedingungen überprüfen

- Fehler: `1121` SQLSTATE: `42000` (`ER_NULL_COLUMN_IN_INDEX`)

  Meldung: Spalte '%s' wurde mit UNIQUE oder INDEX benutzt, ist aber nicht als NOT NULL definiert

- Fehler: `1122` SQLSTATE: `HY000` (`ER_CANT_FIND_UDF`)

  Meldung: Kann Funktion '%s' nicht laden

- Fehler: `1123` SQLSTATE: `HY000` (`ER_CANT_INITIALIZE_UDF`)

  Meldung: Kann Funktion '%s' nicht initialisieren: %s

- Fehler: `1124` SQLSTATE: `HY000` (`ER_UDF_NO_PATHS`)

  Meldung: Keine Pfade gestattet für Shared Library

- Fehler: `1125` SQLSTATE: `HY000` (`ER_UDF_EXISTS`)

  Meldung: Funktion '%s' existiert schon

- Fehler: `1126` SQLSTATE: `HY000` (`ER_CANT_OPEN_LIBRARY`)

  Meldung: Kann Shared Library '%s' nicht öffnen (Fehler: %d %s)

- Fehler: `1127` SQLSTATE: `HY000` (`ER_CANT_FIND_DL_ENTRY`)

  Meldung: Kann Funktion '%s' in der Library nicht finden

- Fehler: `1128` SQLSTATE: `HY000` (`ER_FUNCTION_NOT_DEFINED`)

  Meldung: Funktion '%s' ist nicht definiert

- Fehler: `1129` SQLSTATE: `HY000` (`ER_HOST_IS_BLOCKED`)

  Meldung: Host '%s' blockiert wegen zu vieler Verbindungsfehler. Aufheben der Blockierung mit 'mysqladmin flush-hosts'

- Fehler: `1130` SQLSTATE: `HY000` (`ER_HOST_NOT_PRIVILEGED`)

  Meldung: Host '%s' hat keine Berechtigung, sich mit diesem MySQL-Server zu verbinden

- Fehler: `1131` SQLSTATE: `42000` (`ER_PASSWORD_ANONYMOUS_USER`)

  Meldung: Sie benutzen MySQL als anonymer Benutzer und dürfen daher keine Passwörter ändern

- Fehler: `1132` SQLSTATE: `42000` (`ER_PASSWORD_NOT_ALLOWED`)

  Meldung: Sie benötigen die Berechtigung zum Aktualisieren von Tabellen in der Datenbank 'mysql', um die Passwörter anderer Benutzer ändern zu können

- Fehler: `1133` SQLSTATE: `42000` (`ER_PASSWORD_NO_MATCH`)

  Meldung: Kann keinen passenden Datensatz in Tabelle 'user' finden

- Fehler: `1134` SQLSTATE: `HY000` (`ER_UPDATE_INFO`)

  Meldung: Datensätze gefunden: %ld Geändert: %ld Warnungen: %ld

- Fehler: `1135` SQLSTATE: `HY000` (`ER_CANT_CREATE_THREAD`)

  Meldung: Kann keinen neuen Thread erzeugen (Fehler: %d). Sollte noch Speicher verfügbar sein, bitte im Handbuch wegen möglicher Fehler im Betriebssystem nachschlagen

- Fehler: `1136` SQLSTATE: `21S01` (`ER_WRONG_VALUE_COUNT_ON_ROW`)

Meldung: Anzahl der Spalten stimmt nicht mit der Anzahl der Werte in Zeile %ld überein

- Fehler: 1137 SQLSTATE: HY000 (ER_CANT_REOPEN_TABLE)

  Meldung: Kann Tabelle'%s' nicht erneut öffnen

- Fehler: 1138 SQLSTATE: 42000 (ER_INVALID_USE_OF_NULL)

  Meldung: Unerlaubte Verwendung eines NULL-Werts

- Fehler: 1139 SQLSTATE: 42000 (ER_REGEXP_ERROR)

  Meldung: regexp lieferte Fehler '%s'

- Fehler: 1140 SQLSTATE: 42000 (ER_MIX_OF_GROUP_FUNC_AND_FIELDS)

  Meldung: Das Vermischen von GROUP-Spalten (MIN(),MAX(),COUNT()...) mit Nicht-GROUP-Spalten ist nicht zulässig, wenn keine GROUP BY-Klausel vorhanden ist

- Fehler: 1141 SQLSTATE: 42000 (ER_NONEXISTING_GRANT)

  Meldung: Für Benutzer '%s' auf Host '%s' gibt es keine solche Berechtigung

- Fehler: 1142 SQLSTATE: 42000 (ER_TABLEACCESS_DENIED_ERROR)

  Meldung: %s Befehl nicht erlaubt für Benutzer '%s'@'%s' und für Tabelle '%s'

- Fehler: 1143 SQLSTATE: 42000 (ER_COLUMNACCESS_DENIED_ERROR)

  Meldung: %s Befehl nicht erlaubt für Benutzer '%s'@'%s' und Spalte '%s' in Tabelle '%s'

- Fehler: 1144 SQLSTATE: 42000 (ER_ILLEGAL_GRANT_FOR_TABLE)

  Meldung: Unzulässiger GRANT- oder REVOKE-Befehl. Verfügbare Berechtigungen sind im Handbuch aufgeführt

- Fehler: 1145 SQLSTATE: 42000 (ER_GRANT_WRONG_HOST_OR_USER)

  Meldung: Das Host- oder User-Argument für GRANT ist zu lang

- Fehler: 1146 SQLSTATE: 42S02 (ER_NO_SUCH_TABLE)

  Meldung: Tabelle '%s.%s' existiert nicht

- Fehler: 1147 SQLSTATE: 42000 (ER_NONEXISTING_TABLE_GRANT)

  Meldung: Keine solche Berechtigung für User '%s' auf Host '%s' an Tabelle '%s'

- Fehler: 1148 SQLSTATE: 42000 (ER_NOT_ALLOWED_COMMAND)

  Meldung: Der verwendete Befehl ist in dieser MySQL-Version nicht zulässig

- Fehler: 1149 SQLSTATE: 42000 (ER_SYNTAX_ERROR)

  Meldung: Fehler in der SQL-Syntax. Bitte die korrekte Syntax im Handbuch nachschlagen (diese kann für verschiedene Server-Versionen unterschiedlich sein)

- Fehler: 1150 SQLSTATE: HY000 (ER_DELAYED_CANT_CHANGE_LOCK)

  Meldung: Verzögerter (DELAYED) Einfüge-Thread konnte die angeforderte Sperre für Tabelle '%s' nicht erhalten

- Fehler: 1151 SQLSTATE: HY000 (ER_TOO_MANY_DELAYED_THREADS)

  Meldung: Zu viele verzögerte (DELAYED) Threads in Verwendung

- Fehler: 1152 SQLSTATE: 08S01 (ER_ABORTING_CONNECTION)

  Meldung: Abbruch der Verbindung %ld zur Datenbank '%s'. Benutzer: '%s' (%s)

- Fehler: 1153 SQLSTATE: 08S01 (ER_NET_PACKET_TOO_LARGE)

  Meldung: Empfangenes Paket ist größer als 'max_allowed_packet'

- Fehler: 1154 SQLSTATE: 08S01 (ER_NET_READ_ERROR_FROM_PIPE)

  Meldung: Lese-Fehler bei einer Kommunikations-Pipe
- Fehler: 1155 SQLSTATE: 08S01 (ER_NET_FCNTL_ERROR)

  Meldung: fcntl() lieferte einen Fehler
- Fehler: 1156 SQLSTATE: 08S01 (ER_NET_PACKETS_OUT_OF_ORDER)

  Meldung: Pakete nicht in der richtigen Reihenfolge empfangen
- Fehler: 1157 SQLSTATE: 08S01 (ER_NET_UNCOMPRESS_ERROR)

  Meldung: Kommunikationspaket lässt sich nicht entpacken
- Fehler: 1158 SQLSTATE: 08S01 (ER_NET_READ_ERROR)

  Meldung: Fehler beim Lesen eines Kommunikationspakets
- Fehler: 1159 SQLSTATE: 08S01 (ER_NET_READ_INTERRUPTED)

  Meldung: Zeitüberschreitung beim Lesen eines Kommunikationspakets
- Fehler: 1160 SQLSTATE: 08S01 (ER_NET_ERROR_ON_WRITE)

  Meldung: Fehler beim Schreiben eines Kommunikationspakets
- Fehler: 1161 SQLSTATE: 08S01 (ER_NET_WRITE_INTERRUPTED)

  Meldung: Zeitüberschreitung beim Schreiben eines Kommunikationspakets
- Fehler: 1162 SQLSTATE: 42000 (ER_TOO_LONG_STRING)

  Meldung: Ergebnis ist länger als 'max_allowed_packet'
- Fehler: 1163 SQLSTATE: 42000 (ER_TABLE_CANT_HANDLE_BLOB)

  Meldung: Der verwendete Tabellentyp unterstützt keine BLOB- und TEXT-Spalten
- Fehler: 1164 SQLSTATE: 42000 (ER_TABLE_CANT_HANDLE_AUTO_INCREMENT)

  Meldung: Der verwendete Tabellentyp unterstützt keine AUTO_INCREMENT-Spalten
- Fehler: 1165 SQLSTATE: HY000 (ER_DELAYED_INSERT_TABLE_LOCKED)

  Meldung: INSERT DELAYED kann nicht auf Tabelle '%s' angewendet werden, da diese mit LOCK TABLES gesperrt ist
- Fehler: 1166 SQLSTATE: 42000 (ER_WRONG_COLUMN_NAME)

  Meldung: Falscher Spaltenname '%s'
- Fehler: 1167 SQLSTATE: 42000 (ER_WRONG_KEY_COLUMN)

  Meldung: Der verwendete Tabellen-Handler kann die Spalte '%s' nicht indizieren
- Fehler: 1168 SQLSTATE: HY000 (ER_WRONG_MRG_TABLE)

  Meldung: Nicht alle Tabellen in der MERGE-Tabelle sind gleich definiert
- Fehler: 1169 SQLSTATE: 23000 (ER_DUP_UNIQUE)

  Meldung: Schreiben in Tabelle '%s' nicht möglich wegen einer eindeutigen Beschränkung (unique constraint)
- Fehler: 1170 SQLSTATE: 42000 (ER_BLOB_KEY_WITHOUT_LENGTH)

  Meldung: BLOB- oder TEXT-Spalte '%s' wird in der Schlüsseldefinition ohne Schlüssellängenangabe verwendet
- Fehler: 1171 SQLSTATE: 42000 (ER_PRIMARY_CANT_HAVE_NULL)

  Meldung: Alle Teile eines PRIMARY KEY müssen als NOT NULL definiert sein. Wenn NULL in einem Schlüssel verwendet wird, muss ein UNIQUE-Schlüssel verwendet werden

- Fehler: 1172 SQLSTATE: 42000 (ER_TOO_MANY_ROWS)

  Meldung: Ergebnis besteht aus mehr als einer Zeile

- Fehler: 1173 SQLSTATE: 42000 (ER_REQUIRES_PRIMARY_KEY)

  Meldung: Dieser Tabellentyp benötigt einen PRIMARY KEY

- Fehler: 1174 SQLSTATE: HY000 (ER_NO_RAID_COMPILED)

  Meldung: Diese MySQL-Version ist nicht mit RAID-Unterstützung kompiliert

- Fehler: 1175 SQLSTATE: HY000 (ER_UPDATE_WITHOUT_KEY_IN_SAFE_MODE)

  Meldung: MySQL läuft im sicheren Aktualisierungsmodus (safe update mode). Sie haben versucht, eine Tabelle zu aktualisieren, ohne in der WHERE-Klausel eine KEY-Spalte anzugeben

- Fehler: 1176 SQLSTATE: HY000 (ER_KEY_DOES_NOT_EXITS)

  Meldung: Schlüssel '%s' existiert in der Tabelle '%s' nicht

- Fehler: 1177 SQLSTATE: 42000 (ER_CHECK_NO_SUCH_TABLE)

  Meldung: Kann Tabelle nicht öffnen

- Fehler: 1178 SQLSTATE: 42000 (ER_CHECK_NOT_IMPLEMENTED)

  Meldung: Die Speicher-Engine für diese Tabelle unterstützt kein %s

- Fehler: 1179 SQLSTATE: 25000 (ER_CANT_DO_THIS_DURING_AN_TRANSACTION)

  Meldung: Sie dürfen diesen Befehl nicht in einer Transaktion ausführen

- Fehler: 1180 SQLSTATE: HY000 (ER_ERROR_DURING_COMMIT)

  Meldung: Fehler %d beim COMMIT

- Fehler: 1181 SQLSTATE: HY000 (ER_ERROR_DURING_ROLLBACK)

  Meldung: Fehler %d beim ROLLBACK

- Fehler: 1182 SQLSTATE: HY000 (ER_ERROR_DURING_FLUSH_LOGS)

  Meldung: Fehler %d bei FLUSH_LOGS

- Fehler: 1183 SQLSTATE: HY000 (ER_ERROR_DURING_CHECKPOINT)

  Meldung: Fehler %d bei CHECKPOINT

- Fehler: 1184 SQLSTATE: 08S01 (ER_NEW_ABORTING_CONNECTION)

  Meldung: Verbindungsabbruch %ld zur Datenbank '%s'. Benutzer: '%s', Host: `%s' (%s)

- Fehler: 1185 SQLSTATE: HY000 (ER_DUMP_NOT_IMPLEMENTED)

  Meldung: Die Speicher-Engine für die Tabelle unterstützt keinen binären Tabellen-Dump

- Fehler: 1186 SQLSTATE: HY000 (ER_FLUSH_MASTER_BINLOG_CLOSED)

  Meldung: Binlog geschlossen. Kann RESET MASTER nicht ausführen

- Fehler: 1187 SQLSTATE: HY000 (ER_INDEX_REBUILD)

  Meldung: Neuerstellung des Indizes der Dump-Tabelle '%s' fehlgeschlagen

- Fehler: 1188 SQLSTATE: HY000 (ER_MASTER)

  Meldung: Fehler vom Master: '%s'

- Fehler: 1189 SQLSTATE: 08S01 (ER_MASTER_NET_READ)

  Meldung: Netzfehler beim Lesen vom Master

- Fehler: 1190 SQLSTATE: 08S01 (ER_MASTER_NET_WRITE)

  Meldung: Netzfehler beim Schreiben zum Master

- Fehler: 1191 SQLSTATE: HY000 (ER_FT_MATCHING_KEY_NOT_FOUND)

  Meldung: Kann keinen FULLTEXT-Index finden, der der Spaltenliste entspricht

- Fehler: 1192 SQLSTATE: HY000 (ER_LOCK_OR_ACTIVE_TRANSACTION)

  Meldung: Kann den angegebenen Befehl wegen einer aktiven Tabellensperre oder einer aktiven Transaktion nicht ausführen

- Fehler: 1193 SQLSTATE: HY000 (ER_UNKNOWN_SYSTEM_VARIABLE)

  Meldung: Unbekannte Systemvariable '%s'

- Fehler: 1194 SQLSTATE: HY000 (ER_CRASHED_ON_USAGE)

  Meldung: Tabelle '%s' ist als defekt markiert und sollte repariert werden

- Fehler: 1195 SQLSTATE: HY000 (ER_CRASHED_ON_REPAIR)

  Meldung: Tabelle '%s' ist als defekt markiert und der letzte (automatische?) Reparaturversuch schlug fehl

- Fehler: 1196 SQLSTATE: HY000 (ER_WARNING_NOT_COMPLETE_ROLLBACK)

  Meldung: Änderungen an einigen nicht transaktionalen Tabellen konnten nicht zurückgerollt werden

- Fehler: 1197 SQLSTATE: HY000 (ER_TRANS_CACHE_FULL)

  Meldung: Transaktionen, die aus mehreren Befehlen bestehen, benötigen mehr als 'max_binlog_cache_size' Bytes an Speicher. Diese mysqld-Variable bitte vergrössern und erneut versuchen

- Fehler: 1198 SQLSTATE: HY000 (ER_SLAVE_MUST_STOP)

  Meldung: Diese Operation kann nicht bei einem aktiven Slave durchgeführt werden. Bitte zuerst STOP SLAVE ausführen

- Fehler: 1199 SQLSTATE: HY000 (ER_SLAVE_NOT_RUNNING)

  Meldung: Diese Operation benötigt einen aktiven Slave. Bitte Slave konfigurieren und mittels START SLAVE aktivieren

- Fehler: 1200 SQLSTATE: HY000 (ER_BAD_SLAVE)

  Meldung: Der Server ist nicht als Slave konfiguriert. Bitte in der Konfigurationsdatei oder mittels CHANGE MASTER TO beheben

- Fehler: 1201 SQLSTATE: HY000 (ER_MASTER_INFO)

  Meldung: Could not initialize master info structure, more error messages can be found in the MySQL error log

- Fehler: 1202 SQLSTATE: HY000 (ER_SLAVE_THREAD)

  Meldung: Konnte keinen Slave-Thread starten. Bitte System-Ressourcen überprüfen

- Fehler: 1203 SQLSTATE: 42000 (ER_TOO_MANY_USER_CONNECTIONS)

  Meldung: Benutzer '%s' hat mehr als max_user_connections aktive Verbindungen

- Fehler: 1204 SQLSTATE: HY000 (ER_SET_CONSTANTS_ONLY)

  Meldung: Bei SET dürfen nur konstante Ausdrücke verwendet werden

- Fehler: 1205 SQLSTATE: HY000 (ER_LOCK_WAIT_TIMEOUT)

  Meldung: Beim Warten auf eine Sperre wurde die zulässige Wartezeit überschritten. Bitte versuchen Sie, die Transaktion neu zu starten

- Fehler: 1206 SQLSTATE: HY000 (ER_LOCK_TABLE_FULL)

  Meldung: Die Gesamtzahl der Sperren überschreitet die Größe der Sperrtabelle

- Fehler: 1207 SQLSTATE: 25000 (ER_READ_ONLY_TRANSACTION)

Meldung: Während einer READ UNCOMMITED-Transaktion können keine UPDATE-Sperren angefordert werden

- Fehler: 1208 SQLSTATE: HY000 (ER_DROP_DB_WITH_READ_LOCK)

  Meldung: DROP DATABASE ist nicht erlaubt, solange der Thread eine globale Lesesperre hält

- Fehler: 1209 SQLSTATE: HY000 (ER_CREATE_DB_WITH_READ_LOCK)

  Meldung: CREATE DATABASE ist nicht erlaubt, solange der Thread eine globale Lesesperre hält

- Fehler: 1210 SQLSTATE: HY000 (ER_WRONG_ARGUMENTS)

  Meldung: Falsche Argumente für %s

- Fehler: 1211 SQLSTATE: 42000 (ER_NO_PERMISSION_TO_CREATE_USER)

  Meldung: '%s'@'%s' is nicht berechtigt, neue Benutzer hinzuzufügen

- Fehler: 1212 SQLSTATE: HY000 (ER_UNION_TABLES_IN_DIFFERENT_DIR)

  Meldung: Falsche Tabellendefinition. Alle MERGE-Tabellen müssen sich in derselben Datenbank befinden

- Fehler: 1213 SQLSTATE: 40001 (ER_LOCK_DEADLOCK)

  Meldung: Beim Versuch, eine Sperre anzufordern, ist ein Deadlock aufgetreten. Versuchen Sie, die Transaktion erneut zu starten

- Fehler: 1214 SQLSTATE: HY000 (ER_TABLE_CANT_HANDLE_FT)

  Meldung: Der verwendete Tabellentyp unterstützt keine FULLTEXT-Indizes

- Fehler: 1215 SQLSTATE: HY000 (ER_CANNOT_ADD_FOREIGN)

  Meldung: Fremdschlüssel-Beschränkung konnte nicht hinzugefügt werden

- Fehler: 1216 SQLSTATE: 23000 (ER_NO_REFERENCED_ROW)

  Meldung: Hinzufügen eines Kind-Datensatzes schlug aufgrund einer Fremdschlüssel-Beschränkung fehl

- Fehler: 1217 SQLSTATE: 23000 (ER_ROW_IS_REFERENCED)

  Meldung: Löschen eines Eltern-Datensatzes schlug aufgrund einer Fremdschlüssel-Beschränkung fehl

- Fehler: 1218 SQLSTATE: 08S01 (ER_CONNECT_TO_MASTER)

  Meldung: Fehler bei der Verbindung zum Master: %s

- Fehler: 1219 SQLSTATE: HY000 (ER_QUERY_ON_MASTER)

  Meldung: Beim Ausführen einer Abfrage auf dem Master trat ein Fehler auf: %s

- Fehler: 1220 SQLSTATE: HY000 (ER_ERROR_WHEN_EXECUTING_COMMAND)

  Meldung: Fehler beim Ausführen des Befehls %s: %s

- Fehler: 1221 SQLSTATE: HY000 (ER_WRONG_USAGE)

  Meldung: Falsche Verwendung von %s und %s

- Fehler: 1222 SQLSTATE: 21000 (ER_WRONG_NUMBER_OF_COLUMNS_IN_SELECT)

  Meldung: Die verwendeten SELECT-Befehle liefern eine unterschiedliche Anzahl von Spalten zurück

- Fehler: 1223 SQLSTATE: HY000 (ER_CANT_UPDATE_WITH_READLOCK)

  Meldung: Augrund eines READ LOCK-Konflikts kann die Abfrage nicht ausgeführt werden

- Fehler: 1224 SQLSTATE: HY000 (ER_MIXING_NOT_ALLOWED)

  Meldung: Die gleichzeitige Verwendung von Tabellen mit und ohne Transaktionsunterstützung ist deaktiviert

- Fehler: 1225 SQLSTATE: HY000 (ER_DUP_ARGUMENT)

  Meldung: Option '%s' wird im Befehl zweimal verwendet
- Fehler: 1226 SQLSTATE: 42000 (ER_USER_LIMIT_REACHED)

  Meldung: Benutzer '%s' hat die Ressourcenbeschränkung '%s' überschritten (aktueller Wert: %ld)
- Fehler: 1227 SQLSTATE: HY000 (ER_SPECIFIC_ACCESS_DENIED_ERROR)

  Meldung: Befehl nicht zulässig. Hierfür wird die Berechtigung %s benötigt
- Fehler: 1228 SQLSTATE: HY000 (ER_LOCAL_VARIABLE)

  Meldung: Variable '%s' ist eine lokale Variable und kann nicht mit SET GLOBAL verändert werden
- Fehler: 1229 SQLSTATE: HY000 (ER_GLOBAL_VARIABLE)

  Meldung: Variable '%s' ist eine globale Variable und muss mit SET GLOBAL verändert werden
- Fehler: 1230 SQLSTATE: 42000 (ER_NO_DEFAULT)

  Meldung: Variable '%s' hat keinen Vorgabewert
- Fehler: 1231 SQLSTATE: 42000 (ER_WRONG_VALUE_FOR_VAR)

  Meldung: Variable '%s' kann nicht auf '%s' gesetzt werden
- Fehler: 1232 SQLSTATE: 42000 (ER_WRONG_TYPE_FOR_VAR)

  Meldung: Falscher Argumenttyp für Variable '%s'
- Fehler: 1233 SQLSTATE: HY000 (ER_VAR_CANT_BE_READ)

  Meldung: Variable '%s' kann nur verändert, nicht gelesen werden
- Fehler: 1234 SQLSTATE: 42000 (ER_CANT_USE_OPTION_HERE)

  Meldung: Falsche Verwendung oder Platzierung von '%s'
- Fehler: 1235 SQLSTATE: 42000 (ER_NOT_SUPPORTED_YET)

  Meldung: Diese MySQL-Version unterstützt '%s' nicht
- Fehler: 1236 SQLSTATE: HY000 (ER_MASTER_FATAL_ERROR_READING_BINLOG)

  Meldung: Schwerer Fehler %d: '%s vom Master beim Lesen des binären Logs aufgetreten
- Fehler: 1237 SQLSTATE: HY000 (ER_SLAVE_IGNORED_TABLE)

  Meldung: Slave-SQL-Thread hat die Abfrage aufgrund von replicate-*-table-Regeln ignoriert
- Fehler: 1238 SQLSTATE: HY000 (ER_INCORRECT_GLOBAL_LOCAL_VAR)

  Meldung: Variable '%s' is a %s variable
- Fehler: 1239 SQLSTATE: 42000 (ER_WRONG_FK_DEF)

  Meldung: Falsche Fremdschlüssel-Definition für '%s': %s
- Fehler: 1240 SQLSTATE: HY000 (ER_KEY_REF_DO_NOT_MATCH_TABLE_REF)

  Meldung: Schlüssel- und Tabellenverweis passen nicht zusammen
- Fehler: 1241 SQLSTATE: 21000 (ER_OPERAND_COLUMNS)

  Meldung: Operand solle %d Spalte(n) enthalten
- Fehler: 1242 SQLSTATE: 21000 (ER_SUBQUERY_NO_1_ROW)

  Meldung: Unterabfrage lieferte mehr als einen Datensatz zurück

- Fehler: 1243 SQLSTATE: HY000 (ER_UNKNOWN_STMT_HANDLER)

  Meldung: Unbekannter Prepared-Statement-Handler (%.*s) für %s angegeben

- Fehler: 1244 SQLSTATE: HY000 (ER_CORRUPT_HELP_DB)

  Meldung: Die Hilfe-Datenbank ist beschädigt oder existiert nicht

- Fehler: 1245 SQLSTATE: HY000 (ER_CYCLIC_REFERENCE)

  Meldung: Zyklischer Verweis in Unterabfragen

- Fehler: 1246 SQLSTATE: HY000 (ER_AUTO_CONVERT)

  Meldung: Spalte '%s' wird von %s nach %s umgewandelt

- Fehler: 1247 SQLSTATE: 42S22 (ER_ILLEGAL_REFERENCE)

  Meldung: Verweis '%s' wird nicht unterstützt (%s)

- Fehler: 1248 SQLSTATE: 42000 (ER_DERIVED_MUST_HAVE_ALIAS)

  Meldung: Für jede abgeleitete Tabelle muss ein eigener Alias angegeben werden

- Fehler: 1249 SQLSTATE: 01000 (ER_SELECT_REDUCED)

  Meldung: Select %u wurde während der Optimierung reduziert

- Fehler: 1250 SQLSTATE: 42000 (ER_TABLENAME_NOT_ALLOWED_HERE)

  Meldung: Tabelle '%s', die in einem der SELECT-Befehle verwendet wurde, kann nicht in %s verwendet werden

- Fehler: 1251 SQLSTATE: 08004 (ER_NOT_SUPPORTED_AUTH_MODE)

  Meldung: Client unterstützt das vom Server erwartete Authentifizierungsprotokoll nicht. Bitte aktualisieren Sie Ihren MySQL-Client

- Fehler: 1252 SQLSTATE: 42000 (ER_SPATIAL_CANT_HAVE_NULL)

  Meldung: Alle Teile eines SPATIAL KEY müssen als NOT NULL deklariert sein

- Fehler: 1253 SQLSTATE: 42000 (ER_COLLATION_CHARSET_MISMATCH)

  Meldung: COLLATION '%s' ist für CHARACTER SET '%s' ungültig

- Fehler: 1254 SQLSTATE: HY000 (ER_SLAVE_WAS_RUNNING)

  Meldung: Slave läuft bereits

- Fehler: 1255 SQLSTATE: HY000 (ER_SLAVE_WAS_NOT_RUNNING)

  Meldung: Slave wurde bereits angehalten

- Fehler: 1256 SQLSTATE: HY000 (ER_TOO_BIG_FOR_UNCOMPRESS)

  Meldung: Unkomprimierte Daten sind zu groß. Die maximale Größe beträgt %d

- Fehler: 1257 SQLSTATE: HY000 (ER_ZLIB_Z_MEM_ERROR)

  Meldung: ZLIB: Steht nicht genug Speicher zur Verfügung

- Fehler: 1258 SQLSTATE: HY000 (ER_ZLIB_Z_BUF_ERROR)

  Meldung: ZLIB: Im Ausgabepuffer ist nicht genug Platz vorhanden (wahrscheinlich wurde die Länge der unkomprimierten Daten beschädigt)

- Fehler: 1259 SQLSTATE: HY000 (ER_ZLIB_Z_DATA_ERROR)

  Meldung: ZLIB: Eingabedaten beschädigt

- Fehler: 1260 SQLSTATE: HY000 (ER_CUT_VALUE_GROUP_CONCAT)

Meldung: %d Zeile(n) durch GROUP_CONCAT() abgeschnitten

- Fehler: 1261 SQLSTATE: 01000 (ER_WARN_TOO_FEW_RECORDS)

  Meldung: Anzahl der Datensätze in Zeile %ld geringer als Anzahl der Spalten

- Fehler: 1262 SQLSTATE: 01000 (ER_WARN_TOO_MANY_RECORDS)

  Meldung: Anzahl der Datensätze in Zeile %ld größer als Anzahl der Spalten

- Fehler: 1263 SQLSTATE: 01000 (ER_WARN_NULL_TO_NOTNULL)

  Meldung: Daten abgeschnitten, NULL für NOT NULL-Spalte '%s' in Zeile %ld angegeben

- Fehler: 1264 SQLSTATE: 01000 (ER_WARN_DATA_OUT_OF_RANGE)

  Meldung: Daten abgeschnitten, außerhalb des Wertebereichs für Spalte '%s' in Zeile %ld

- Fehler: 1265 SQLSTATE: 01000 (ER_WARN_DATA_TRUNCATED)

  Meldung: Daten abgeschnitten für Spalte '%s' in Zeile %ld

- Fehler: 1266 SQLSTATE: HY000 (ER_WARN_USING_OTHER_HANDLER)

  Meldung: Für Tabelle '%s' wird Speicher-Engine %s benutzt

- Fehler: 1267 SQLSTATE: HY000 (ER_CANT_AGGREGATE_2COLLATIONS)

  Meldung: Unerlaubte Vermischung der Kollationen (%s,%s) und (%s,%s) für die Operation '%s'

- Fehler: 1268 SQLSTATE: HY000 (ER_DROP_USER)

  Meldung: Kann einen oder mehrere der angegebenen Benutzer nicht löschen

- Fehler: 1269 SQLSTATE: HY000 (ER_REVOKE_GRANTS)

  Meldung: Kann nicht alle Berechtigungen widerrufen, grant for one or more of the requested users

- Fehler: 1270 SQLSTATE: HY000 (ER_CANT_AGGREGATE_3COLLATIONS)

  Meldung: Unerlaubte Vermischung der Kollationen (%s,%s), (%s,%s), (%s,%s) für die Operation '%s'

- Fehler: 1271 SQLSTATE: HY000 (ER_CANT_AGGREGATE_NCOLLATIONS)

  Meldung: Unerlaubte Vermischung der Kollationen für die Operation '%s'

- Fehler: 1272 SQLSTATE: HY000 (ER_VARIABLE_IS_NOT_STRUCT)

  Meldung: Variable '%s' ist keine Variablen-Komponenten (kann nicht als XXXX.variablen_name verwendet werden)

- Fehler: 1273 SQLSTATE: HY000 (ER_UNKNOWN_COLLATION)

  Meldung: Unbekannte Kollation: '%s'

- Fehler: 1274 SQLSTATE: HY000 (ER_SLAVE_IGNORED_SSL_PARAMS)

  Meldung: SSL-Parameter in CHANGE MASTER werden ignoriert, weil dieser MySQL-Slave ohne SSL-Unterstützung kompiliert wurde. Sie können aber später verwendet werden, wenn der MySQL-Slave mit SSL gestartet wird

- Fehler: 1275 SQLSTATE: HY000 (ER_SERVER_IS_IN_SECURE_AUTH_MODE)

  Meldung: Server läuft im Modus --secure-auth, aber '%s'@'%s' hat ein Passwort im alten Format. Bitte Passwort ins neue Format ändern

- Fehler: 1276 SQLSTATE: HY000 (ER_WARN_FIELD_RESOLVED)

  Meldung: Feld oder Verweis '%s%s%s%s%s' im SELECT-Befehl Nr. %d wurde im SELECT-Befehl Nr. %d aufgelöst

- Fehler: 1277 SQLSTATE: HY000 (ER_BAD_SLAVE_UNTIL_COND)

  Meldung: Falscher Parameter oder falsche Kombination von Parametern für START SLAVE UNTIL

- Fehler: 1278 SQLSTATE: HY000 (ER_MISSING_SKIP_SLAVE)

  Meldung: Es wird empfohlen, mit --skip-slave-start zu starten, wenn mit START SLAVE UNTIL eine Schritt-für-Schritt-Replikation ausgeführt wird. Ansonsten gibt es Probleme, wenn der Slave-Server unerwartet neu startet

- Fehler: 1279 SQLSTATE: HY000 (ER_UNTIL_COND_IGNORED)

  Meldung: SQL-Thread soll nicht gestartet werden. Daher werden UNTIL-Optionen ignoriert

- Fehler: 1280 SQLSTATE: 42000 (ER_WRONG_NAME_FOR_INDEX)

  Meldung: Incorrect index name '%s'

- Fehler: 1281 SQLSTATE: 42000 (ER_WRONG_NAME_FOR_CATALOG)

  Meldung: Incorrect catalog name '%s'

- Fehler: 1282 SQLSTATE: HY000 (ER_WARN_QC_RESIZE)

  Meldung: Query cache failed to set size %lu, new query cache size is %lu

- Fehler: 1283 SQLSTATE: HY000 (ER_BAD_FT_COLUMN)

  Meldung: Column '%s' cannot be part of FULLTEXT index

- Fehler: 1284 SQLSTATE: HY000 (ER_UNKNOWN_KEY_CACHE)

  Meldung: Unknown key cache '%s'

- Fehler: 1285 SQLSTATE: HY000 (ER_WARN_HOSTNAME_WONT_WORK)

  Meldung: MySQL is started in --skip-name-resolve mode. You need to restart it without this switch for this grant to work

- Fehler: 1286 SQLSTATE: 42000 (ER_UNKNOWN_STORAGE_ENGINE)

  Meldung: Unknown table engine '%s'

- Fehler: 1287 SQLSTATE: HY000 (ER_WARN_DEPRECATED_SYNTAX)

  Meldung: '%s' is deprecated, use '%s' instead

- Fehler: 1288 SQLSTATE: HY000 (ER_NON_UPDATABLE_TABLE)

  Meldung: The target table %s of the %s is not updateable

- Fehler: 1289 SQLSTATE: HY000 (ER_FEATURE_DISABLED)

  Meldung: The '%s' feature was disabled; you need MySQL built with '%s' to have it working

- Fehler: 1290 SQLSTATE: HY000 (ER_OPTION_PREVENTS_STATEMENT)

  Meldung: The MySQL server is running with the %s option so it cannot execute this statement

- Fehler: 1291 SQLSTATE: HY000 (ER_DUPLICATED_VALUE_IN_TYPE)

  Meldung: Column '%s' has duplicated value '%s' in %s

- Fehler: 1292 SQLSTATE: HY000 (ER_TRUNCATED_WRONG_VALUE)

  Meldung: Truncated wrong %s value: '%s'

- Fehler: 1293 SQLSTATE: HY000 (ER_TOO_MUCH_AUTO_TIMESTAMP_COLS)

  Meldung: Incorrect table definition; there can be only one TIMESTAMP column with CURRENT_TIMESTAMP in DEFAULT or ON UPDATE clause

- Fehler: 1294 SQLSTATE: HY000 (ER_INVALID_ON_UPDATE)

  Meldung: Invalid ON UPDATE clause for '%s' column

- Fehler: 1295 SQLSTATE: HY000 (ER_UNSUPPORTED_PS)

Meldung: This command is not supported in the prepared statement protocol yet

- Fehler: `1296` SQLSTATE: `HY000` (`ER_GET_ERRMSG`)

  Meldung: Got error %d '%s' from %s

- Fehler: `1297` SQLSTATE: `HY000` (`ER_GET_TEMPORARY_ERRMSG`)

  Meldung: Got temporary error %d '%s' from %s

- Fehler: `1298` SQLSTATE: `HY000` (`ER_UNKNOWN_TIME_ZONE`)

  Meldung: Unknown or incorrect time zone: '%s'

- Fehler: `1299` SQLSTATE: `HY000` (`ER_WARN_INVALID_TIMESTAMP`)

  Meldung: Invalid TIMESTAMP value in column '%s' at row %ld

- Fehler: `1300` SQLSTATE: `HY000` (`ER_INVALID_CHARACTER_STRING`)

  Meldung: Invalid %s character string: '%s'

- Fehler: `1301` SQLSTATE: `HY000` (`ER_WARN_ALLOWED_PACKET_OVERFLOWED`)

  Meldung: Result of %s() was larger than max_allowed_packet (%ld) - truncated

- Fehler: `1302` SQLSTATE: `HY000` (`ER_CONFLICTING_DECLARATIONS`)

  Meldung: Conflicting declarations: '%s%s' and '%s%s'

Fehlerinformationen der Clients stammen aus folgenden Dateien:

- Die Fehlerwerte und die Symbole in Klammern entsprechen den Definitionen in der MySQL-Quelldatei `include/errmsg.h`.
- Die Meldungen entsprechen den Fehlermeldungen, die in der Datei `libmysql/errmsg.c` aufgeführt sind. `%d` und `%s` stellen Zahlen beziehungsweise Zeichenketten dar, die in den Fehlermeldungen ersetzt werden, wenn diese angezeigt werden.

Weil sie häufig aktualisiert werden, enhalten die genannten Dateien möglicherweise zusätzliche Informationen, die hier nicht aufgeführt sind.

- Fehler: `2000` (`CR_UNKNOWN_ERROR`)

  Meldung: Unknown MySQL error

- Fehler: `2001` (`CR_SOCKET_CREATE_ERROR`)

  Meldung: Can't create UNIX socket (%d)

- Fehler: `2002` (`CR_CONNECTION_ERROR`)

  Meldung: Can't connect to local MySQL server through socket '%s' (%d)

- Fehler: `2003` (`CR_CONN_HOST_ERROR`)

  Meldung: Can't connect to MySQL server on '%s' (%d)

- Fehler: `2004` (`CR_IPSOCK_ERROR`)

  Meldung: Can't create TCP/IP socket (%d)

- Fehler: `2005` (`CR_UNKNOWN_HOST`)

  Meldung: Unknown MySQL server host '%s' (%d)

- Fehler: `2006` (`CR_SERVER_GONE_ERROR`)

Meldung: MySQL server has gone away

- Fehler: `2007` (`CR_VERSION_ERROR`)

  Meldung: Protocol mismatch; server version = %d, client version = %d
- Fehler: `2008` (`CR_OUT_OF_MEMORY`)

  Meldung: MySQL client ran out of memory
- Fehler: `2009` (`CR_WRONG_HOST_INFO`)

  Meldung: Wrong host info
- Fehler: `2010` (`CR_LOCALHOST_CONNECTION`)

  Meldung: Localhost via UNIX socket
- Fehler: `2011` (`CR_TCP_CONNECTION`)

  Meldung: %s via TCP/IP
- Fehler: `2012` (`CR_SERVER_HANDSHAKE_ERR`)

  Meldung: Error in server handshake
- Fehler: `2013` (`CR_SERVER_LOST`)

  Meldung: Lost connection to MySQL server during query
- Fehler: `2014` (`CR_COMMANDS_OUT_OF_SYNC`)

  Meldung: Commands out of sync; you can't run this command now
- Fehler: `2015` (`CR_NAMEDPIPE_CONNECTION`)

  Meldung: Named pipe: %s
- Fehler: `2016` (`CR_NAMEDPIPEWAIT_ERROR`)

  Meldung: Can't wait for named pipe to host: %s pipe: %s (%lu)
- Fehler: `2017` (`CR_NAMEDPIPEOPEN_ERROR`)

  Meldung: Can't open named pipe to host: %s pipe: %s (%lu)
- Fehler: `2018` (`CR_NAMEDPIPESETSTATE_ERROR`)

  Meldung: Can't set state of named pipe to host: %s pipe: %s (%lu)
- Fehler: `2019` (`CR_CANT_READ_CHARSET`)

  Meldung: Can't initialize character set %s (path: %s)
- Fehler: `2020` (`CR_NET_PACKET_TOO_LARGE`)

  Meldung: Got packet bigger than 'max_allowed_packet' bytes
- Fehler: `2021` (`CR_EMBEDDED_CONNECTION`)

  Meldung: Embedded server
- Fehler: `2022` (`CR_PROBE_SLAVE_STATUS`)

  Meldung: Error on SHOW SLAVE STATUS:
- Fehler: `2023` (`CR_PROBE_SLAVE_HOSTS`)

  Meldung: Error on SHOW SLAVE HOSTS:
- Fehler: `2024` (`CR_PROBE_SLAVE_CONNECT`)

Meldung: Error connecting to slave:

- Fehler: `2025` (`CR_PROBE_MASTER_CONNECT`)

  Meldung: Error connecting to master:

- Fehler: `2026` (`CR_SSL_CONNECTION_ERROR`)

  Meldung: SSL connection error

- Fehler: `2027` (`CR_MALFORMED_PACKET`)

  Meldung: Malformed packet

- Fehler: `2028` (`CR_WRONG_LICENSE`)

  Meldung: This client library is licensed only for use with MySQL servers having '%s' license

- Fehler: `2029` (`CR_NULL_POINTER`)

  Meldung: Invalid use of null pointer

- Fehler: `2030` (`CR_NO_PREPARE_STMT`)

  Meldung: Statement not prepared

- Fehler: `2031` (`CR_PARAMS_NOT_BOUND`)

  Meldung: No data supplied for parameters in prepared statement

- Fehler: `2032` (`CR_DATA_TRUNCATED`)

  Meldung: Data truncated

- Fehler: `2033` (`CR_NO_PARAMETERS_EXISTS`)

  Meldung: No parameters exist in the statement

- Fehler: `2034` (`CR_INVALID_PARAMETER_NO`)

  Meldung: Invalid parameter number

- Fehler: `2035` (`CR_INVALID_BUFFER_USE`)

  Meldung: Can't send long data for non-string/non-binary data types (parameter: %d)

- Fehler: `2036` (`CR_UNSUPPORTED_PARAM_TYPE`)

  Meldung: Using unsupported buffer type: %d (parameter: %d)

- Fehler: `2037` (`CR_SHARED_MEMORY_CONNECTION`)

  Meldung: Shared memory: %s

- Fehler: `2038` (`CR_SHARED_MEMORY_CONNECT_REQUEST_ERROR`)

  Meldung: Can't open shared memory; client could not create request event (%lu)

- Fehler: `2039` (`CR_SHARED_MEMORY_CONNECT_ANSWER_ERROR`)

  Meldung: Can't open shared memory; no answer event received from server (%lu)

- Fehler: `2040` (`CR_SHARED_MEMORY_CONNECT_FILE_MAP_ERROR`)

  Meldung: Can't open shared memory; server could not allocate file mapping (%lu)

- Fehler: `2041` (`CR_SHARED_MEMORY_CONNECT_MAP_ERROR`)

  Meldung: Can't open shared memory; server could not get pointer to file mapping (%lu)

- Fehler: `2042` (`CR_SHARED_MEMORY_FILE_MAP_ERROR`)

Meldung: Can't open shared memory; client could not allocate file mapping (%lu)

- Fehler: `2043` (`CR_SHARED_MEMORY_MAP_ERROR`)

  Meldung: Can't open shared memory; client could not get pointer to file mapping (%lu)

- Fehler: `2044` (`CR_SHARED_MEMORY_EVENT_ERROR`)

  Meldung: Can't open shared memory; client could not create %s event (%lu)

- Fehler: `2045` (`CR_SHARED_MEMORY_CONNECT_ABANDONED_ERROR`)

  Meldung: Can't open shared memory; no answer from server (%lu)

- Fehler: `2046` (`CR_SHARED_MEMORY_CONNECT_SET_ERROR`)

  Meldung: Can't open shared memory; cannot send request event to server (%lu)

- Fehler: `2047` (`CR_CONN_UNKNOW_PROTOCOL`)

  Meldung: Wrong or unknown protocol

- Fehler: `2048` (`CR_INVALID_CONN_HANDLE`)

  Meldung: Invalid connection handle

- Fehler: `2049` (`CR_SECURE_AUTH`)

  Meldung: Connection using old (pre-4.1.1) authentication protocol refused (client option 'secure_auth' enabled)

- Fehler: `2050` (`CR_FETCH_CANCELED`)

  Meldung: Row retrieval was canceled by mysql_stmt_close() call

- Fehler: `2051` (`CR_NO_DATA`)

  Meldung: Attempt to read column without prior row fetch

- Fehler: `2052` (`CR_NO_STMT_METADATA`)

  Meldung: Prepared statement contains no metadata

Anhang C. Danksagungen

Dieser Anhang listet die Entwickler, Kontributoren und Unterstützer auf, die mitgeholfen haben, dass MySQL das wird, was es heute ist.

C.1. Entwickler bei MySQL AB

Hier sind die Entwickler, die von `MySQL AB` angestellt wurden, um an `MySQL` zu arbeiten, ungefähr in der Reihenfolge ihres Eintritts. Neben dem Namen wird aufgelistet, für welche Teilaufgaben der Entwickler verantwortlich ist oder welche Leistungen er erbracht hat.

- Michael (Monty) Widenius

  Schrieb folgende Bestandteile von MySQL:

  - Den gesamten Haupt-Code in `mysqld`.
  - Neue Funktionen für die Zeichenketten-Bibliothek.
  - Das meiste der `mysys`-Bibliothek.
  - Die `ISAM`- und `MyISAM`-Bibliotheken (B-Baum-Index-Datei-Handler mit Index-Komprimierung und verschiedenen Datensatzformaten).
  - Die `HEAP`-Bibliothek. Ein Speicher-Tabellensystem mit unserem überragenden komplett dynamischen Hashing. In Gebrauch seit 1981 und veröffentlicht um 1984.
  - Das `replace`-Programm (ansehen, es ist COOL!).
  - **MyODBC**, den ODBC-Treiber für Windows95.
  - Behob Bugs in MIT-pThread, um sie für MySQL zum Laufen zu bringen, sowie Unireg, ein curses-basierendes Applikationswerkzeug vielen Utilities.
  - Portierung von `mSQL`-Werkzeugen wie `msqlperl`, `DBD/DBI` und `db2mysql`.
  - Das meiste von Crash-me und die Grundlage für die MySQL-Benchmarks.
- David Axmark
  - Koordinator und ursprünglicher Haupt-Schreiber des **Referenzhandbuchs**, inklusive Verbesserungen von `texi2HTML`.
  - Automatische Website-Aktualisierung des Handbuchs.
  - Ursprüngliche Autoconf-, Automake- und Libtool-Unterstützung.
  - Den Lizensierungs-Kram.
  - Teile all der Textdateien. (Heutzutage ist nur noch die `README` übrig. Der Rest befindet sich im Handbuch.)
  - Viel Testen neuer Features.
  - Unser ``kostenloser'' Inhouse-Software-Anwalt.
  - Derjenige, der die Mailing-Liste wartet (und nie die Zeit hatte, es richtig zu machen ...)
  - Unser Original-Portabilitätscode (jetzt mehr als 10 Jahre alt). Heutzutage sind nur noch Teile von `mysys` übrig.
  - Jemand, den Monty mitten in der Nacht anrufen kann, wenn er gerade das neue Feature zum Laufen gebracht hat.
- Jani Tolonen
  - `mysqlimport`
  - Etliche Erweiterungen zum `mysql`-Client.
  - `PROCEDURE ANALYSE()`
- Sinisa Milivojevic

  - Kompression (mit `zlib`) im Client-Server-Protokoll.
  - Perfektes Hashing für die lexikalische Analyse-Phase.
  - Den MySQLGUI-Client.
  - Derjenige, der mysql++ wartet.
- Tonu Samuel
  - Unser Sicherheitsexperte.
  - Vio-Schnittstelle (die Grundlage für das verschlüsselte Client-Server-Protokoll).
  - MySQL-Dateisystem (eine Art, MySQL-Datenbanken als Dateien und Verzeichnisse zu benutzen).
  - Den CASE-Ausdruck.
  - Die MD5()- und COALESCE()-Funktionen.
  - `RAID`-Unterstützung für `MyISAM`-Tabellen.
- Sasha Pachev
  - Replikation.
  - `SHOW CREATE TABLE`.
  - mod_mysql_include
  - cgi++
  - mysql-bench
- Matt Wagner
  - MySQL-Test-Suite.
  - Unser Webmaster.
- Miguel Solorzano
  - Winmysqladmin.
- Timothy Smith
  - Dynamische Zeichen-Unterstützung.
  - Verantwortlich für MySQL-configure.
- Sergei Golubchik
  - Volltextsuche.
  - Fügte Schlüssel zur `MERGE`-Bibliothek hinzu.
- Jeremy Cole
  - Korrekturlesen und Editieren dieses netten Handbuchs.
  - `ALTER TABLE ... ORDER BY ...`.
  - `UPDATE ... ORDER BY ...`.
  - `DELETE ... ORDER BY ...`.
- John Dean
  - Den MySQL-GUI-Client.
- Indrek Siitan

- Designer / Programmierer unserer Web-Schnittstelle.

Folgende Nicht-Entwickler arbeiten ebenfalls bei oder zusammen mit MySQL AB:

- Hans Kierkegaard - verantwortlich für die MySQL-Lizenz-Handhabung.
- Antti Halonen - Vertriebsleiter.
- Jonas Norrman - Beantwortet Lizensierungsfrage, die an `<info@mysql.com>` geschickt werden.
- Erik Granberg - bedient MySQL-Partner (und eine Menge sonstiger Kram).
- Allan Larsson (der BOSS für TCX DataKonsult AB).

C.2. Kontributoren zu MySQL

Während `MySQL AB` das gesammte Copyright für den `MySQL Server` und das `MySQL manual` besitzt, möchten wir hier diejenigen Menschen nennen, die das Ein oder Andere zur `MySQL Distribution` beigetragen haben. Die Kontributoren sind in eher zufälliger Reihenfolge aufgeführt:

- Paul DuBois

  Hilft mit, das Referenzhandbuch korrekt und verständlich zu machen. Das beinhaltet, Montys und Davids Englischversuche in das Englisch zu übertragen, das andere Leute kennen.

- Gianmassimo Vigazzola `<qwerg@mbox.vol.it>` oder `<qwerg@tin.it>`

  Die ursprüngliche Portierung auf Win32/NT.

- Kim Aldale

  Half, Montys und Davids frühe Englischversuche ins Englische umzuschreiben.

- Per Eric Olsson

  Mehr oder weniger konstruktive Kritik und Testen des dynamischen Datensatzformats.

- Irena Pancirov `<irena@mail.yacc.it>`

  Win32-Portierung mit dem Borland-Compiler. `mysqlshutdown.exe` und `mysqlwatch.exe`

- David J. Hughes

  Er bemühte sich, eine Shareware-SQL-Datenbank herzustellen. Wir bei TcX fingen mit `mSQL` an, fanden aber, dass es unsere Bedürfnisse nicht befriedigen könne, daher schrieben wir stattdessen eine SQL-Schnittstelle zu unserem Applikation-Builder Unireg. `mysqladmin` und `mysql` sind Programme, die stark von ihren `mSQL`-Pendants beeinflusst sind. Wir haben uns große Mühe gegeben, die MySQL-Syntax zu einer Obermenge von `mSQL` zu machen. Viele API-Ideen sind von `mSQL` entliehen, damit es einfach ist, kostenlose `mSQL`-Programme nach MySQL zu portieren. MySQL enthält keinen Code von `mSQL`. Zwei Dateien in der Distribution (`client/insert_test.c` und `client/select_test.c`) basieren auf den entsprechenden (keinem Copyright unterliegenden) Dateien in der `mSQL`-Distribution, sind aber als Beispiele abgeändert, die die notwendigen Änderungen aufzeigen, wenn man Code von `mSQL` nach MySQL konvertiert. (`mSQL` unterliegt dem Copyright von David J. Hughes.)

- Fred Fish

  Seine exzellente C-Debugging- und Trace-Bibliothek. Monty hat eine Reihe kleinerer Verbesserungen an der Bibliothek vorgenommen (Geschwindigkeit und zusätzliche Optionen).

- Richard A. O'Keefe

  Seine Public-Domain-Zeichenketten-Bibliothek.

- Henry Spencer

  Seine Regex-Bibliothek, benutzt bei `WHERE spalte REGEXP regexp`.

- Free Software Foundation

  Von ihnen haben wir einen exzellenten Compiler (`gcc`), die `libc`-Bibliothek (aus der wir `strto.c` entliehen haben, damit einiger Code unter Linux funktioniert), und die `readline`-Bibliothek (für den `mysql`-Client).

- Free Software Foundation und das XEmacs-Entwicklungsteam

  Ihr großartiger Editor, der für fast jeden Artikeltext bei TcX/MySQL AB/detron benutzt wird.

- Patrick Lynch

  Für seine Hilfe bei `http://www.mysql.com/`.

- Fred Lindberg

  Er half, qmail aufzusetzen, um die MySQL Mailing-Liste zu handhaben, und für seine unglaubliche Unterstützung bei der Verwaltung der MySQL Mailing-Listen.

- Igor Romanenko `<igor@frog.kiev.ua>`

  `mysqldump` (vormals `msqldump`, aber portiert und verbessert von Monty).

- Yuri Dario

  Er unterhält die MySQL-OS/2-Portierung und baut sie aus.

- Tim Bunce, Alligator Descartes

  Für die `DBD`-(Perl)-Schnittstelle.

- Tim Bunce

  Autor von `mysqlhotcopy`.

- Andreas Koenig `<a.koenig@mind.de>`

  Für die Perl-Schnittstelle zu MySQL.

- Eugene Chan `<eugene@acenet.com.sg>`

  Für den Port von PHP zu MySQL.

- Michael J. Miller Jr. `<mke@terrapin.turbolift.com>`

  Er schrieb das erste MySQL-Handbuch, und nahm etliche Bereinigungen der Rechtschreibung / Sprache für die FAQ vor (aus dieser entstand vor langer Zeit das MySQL-Handbuch).

- Yan Cailin

  Erster Übersetzer des MySQL-Referenzhandbuch in vereinfachtes Chinesisch, Anfang 2000, auf der die Big5- und HK-kodierten Versionen (mysql.hitstar.com) basieren. Private Homepage bei linuxdb.yeah.net.

- Giovanni Maruzzelli `<maruzz@matrice.it>`

  Für die Portierung von iODBC (Unix ODBC).

- Chris Provenzano

  Portierbarer Benutzerebene-pThread. Aus dem Copyright: Dieses Produkt beinhaltet Software, die von Chris Provenzano, University of California, Berkeley und Kontributoren entwickelt wurde. Momentan benutzen wir Version 1_60_beta6, die von Monty gepatcht wurde (siehe `with-pThread/Changes-mysql`).

- Xavier Leroy `<Xavier.Leroy@inria.fr>`

  Der Autor von LinuxThread (benutzt von MySQL unter Linux).

- Zarko Mocnik `<zarko.mocnik@dem.si>`

  Sortieren für slowenische Sprache und die `cset.tar.gz`-Module, die es vereinfachen, andere Zeichensätze hinzuzufügen.

- "TAMITO" `<tommy@valley.ne.jp>`

Die `_MB`-Zeichensatz-Makros und die ujis- und sjis-Zeichensätze.

- Joshua Chamas `<joshua@chamas.com>`

  Grundlage für gleichzeitige Einfügeoperationen, erweiterte Datums-Syntax, Debuggen unter NT und Antworten in der MySQL-Mailing-Liste.

- Yves Carlier `<Yves.Carlier@rug.ac.be>`

  `mysqlaccess`, ein Programm, das die Zugriffsrechte für einen Benutzer anzeigt.

- Rhys Stefan `<rhys@wales.com>` (und GWE Technologies Limited)

  Für JDBC, ein Modul, um Daten aus MySQL mit einem Java-Client zu extrahieren.

- Dr. Xiaokun Kelvin ZHU `<X.Zhu@brad.ac.uk>`

  Weiterentwicklung der JDBC-Treiber und anderer MySQL-bezogener Java-Werkzeuge.

- James Cooper `<pixel@organic.com>`

  Aufsetzen eines durchsuchbaren Mailing-Listen-Archivs auf seiner Site.

- Rick Mehalick `<Rick_Mehalick@i-o.com>`

  Für `xmysql`, einen grafischen X-Client für MySQL.

- Doug Sisk `<sisk@wix.com>`

  Er stellt RPM-Pakete von MySQL für RedHat Linux bereit.

- Diemund Alexunder V. `<axeld@vial.ethz.ch>`

  Er stelle RPM-Pakete von MySQL für RedHat Linux-Alpha bereit.

- Antoni Pamies Olive `<toni@readysoft.es>`

  Er stellt RPM-Versionen vieler MySQL-Clients für Intel und SPARC bereit.

- Jay Bloodworth `<jay@pathways.sde.state.sc.us>`

  Er stellte RPM-Versionen für MySQL-Version 3.21 bereit.

- Jochen Wiedmann `<wiedmann@neckar-alb.de>`

  Für die Wartung der Perl-`DBD::mysql`-Module.

- Therrien Gilbert `<gilbert@ican.net>`, Jean-Marc Pouyot `<jmp@scalaire.fr>`

  Französische Fehlermeldungen.

- Petr snajdr, `<snajdr@pvt.net>`

  Tschechische Fehlermeldungen.

- Jaroslaw Lewundowski `<jotel@itnet.com.pl>`

  Polnische Fehlermeldungen.

- Miguel Angel Fernundez Roiz

  Spanische Fehlermeldungen.

- Roy-Magne Mo `<rmo@www.hivolda.no>`

  Norwegische Fehlermeldungen und Testen von Version 3.21.#.

- Timur I. Bakeyev `<root@timur.tatarstan.ru>`

  Russische Fehlermeldungen.

- <brenno@dewinter.com> && Filippo Grassilli <phil@hyppo.com>

  Italienische Fehlermeldungen.

- Dirk Munzinger <dirk@trinity.saar.de>

  Deutsche Fehlermeldungen.

- Billik Stefan <billik@sun.uniag.sk>

  Slowakische Fehlermeldungen.

- Stefan Saroiu <tzoompy@cs.washington.edu>

  Rumänische Fehlermeldungen.

- Peter Feher

  Ungarische Fehlermeldungen.

- Roberto M. Serqueira

  Portugiesische Fehlermeldungen.

- Carsten H. Pedersen

  Dänische Fehlermeldungen

- David Sacerdote <davids@secnet.com>

  Knowhow für die Sicherheitsprüfung von DNS-Hostnamen.

- Wei-Jou Chen <jou@nematic.ieo.nctu.edu.tw>

  Unterstützung für chinesisch(BIG5)-Zeichen.

- Wei He <hewei@mail.ied.ac.cn>

  Viel Funktionalität für den chinesischen (GBK-) Zeichensatz.

- Zeev Suraski <bourbon@netvision.net.il>

  FROM_UNIXTIME()-Zeitformatierung, ENCRYPT()-Funktionen und bison-Ratgeber. Aktives Mitglied der Mailing-Liste.

- Luuk de Boer <luuk@wxs.nl>

  Portierte (und erweiterte) die Benchmark-Suite für DBI/DBD. War eine große Hilfe bei Crash-me und beim Laufenlassen von Benchmarks. Einige neue Datumsfunktionen. Das mysql_setpermissions-Skript.

- Jay Flaherty <fty@mediapulse.com>

  Große Teile des Perl-DBI/DBD-Abschnitts im Handbuch.

- Paul Southworth <pauls@etext.org>, Ray Loyzaga <yar@cs.su.oz.au>

  Korrekturlesen des Referenzhandbuchs.

- Alexis Mikhailov <root@medinf.chuvashia.su>

  Benutzerdefinierte Funktionen (UDFs); CREATE FUNCTION und DROP FUNCTION.

- Andreas F. Bobak <bobak@relog.ch>

  Die AGGREGATE-Erweiterung für UDF-Funktionen.

- Ross Wakelin <R.Wakelin@march.co.uk>

  Half, InstallShield für MySQL-Win32 aufzusetzen.

- Jethro Wright III <jetman@li.net>

  Die libmysql.dll-Bibliothek.

- James Pereria <jpereira@iafrica.com>

  Mysqlmanager, ein grafisches Win32-Werkzeug für die Administration von MySQL.

- Curt Sampson <cjs@portal.ca>

  Portierung von MIT-pThread auf NetBSD/Alpha und NetBSD 1.3/i386.

- Antony T. Curtis <antony.curtis@olcs.net>

  Portiierung von MySQL auf OS/2.

- Martin Ramsch <m.ramsch@computer.org>

  Beispiele im MySQL-Tutorial.

- Steve Harvey

  Er machte `mysqlaccess` sicherer.

- Konark IA-64 Centre of Persistent Systems Private Limited

  http://www.pspl.co.in/konark/. Hilfe bei der Win64-Portierung des MySQL-Servers.

- Albert Chin-A-Young.

  Configure-Aktualisierungen für Tru64, Unterstützung großer Dateien und verbesserte Unterstützung von TCP-Wrappern.

- John Birrell

  Emulation von pthread_mutex() für OS/2.

- Benjamin Pflugmann

  Erweiterte `MERGE`-Tabellen, so dass sie `INSERTS` handhaben. Aktives Mitglied der MySQL-Mailing-Listen.

Andere Kontributoren, Bug-Finder und Tester: James H. Thompson, Maurizio Menghini, Wojciech Tryc, Luca Berra, Zarko Mocnik, Wim Bonis, Elmar Haneke, <jehamby@lightside>, <psmith@BayNetworks.com>, <duane@connect.com.au>, Ted Deppner <ted@psyber.com>, Mike Simons, Jaakko Hyvatti.

Und viele Bug-Berichte und Patches von den Leuten auf der Mailing-Liste.

Große Anerkennung zollen wir denjenigen, die uns halfen, Fragen auf der `mysql@lists.mysql.com`-Mailing-Liste zu beantworten:

- Daniel Koch <dkoch@amcity.com>

  Irix-Setup.

- Luuk de Boer <luuk@wxs.nl>

  Benchmark-Fragen.

- Tim Sailer <tps@Benutzer.buoy.com>

  `DBD-mysql`-Fragen.

- Boyd Lynn Gerber <gerberb@zenez.com>

  SCO-bezogene Fragen.

- Richard Mehalick <RM186061@shellus.com>

  `xmysql`-bezogene Fragen und grundsätzliche Installationsfragen.

- Zeev Suraski <bourbon@netvision.net.il>

  Fragen zur Apache-Modul-Konfiguration (log & auth), PHP-bezogene Fragen, SQL-Syntax-bezogene Fragen und andere allgemeine Fragen.

- France Guasch <frankie@citel.upc.es>

  Allgemeine Fragen.

- Jonathan J Smith <jsmith@wtp.net>

  Fragen zu Betriebssystem-spezifschen Dingen bei Linux, SQL-Syntax- und andere Dinge, die etwas Überarbeitung bedürfen.

- David Sklar <sklar@student.net>

  MySQL von PHP und Perl aus benutzen.

- Alistair MacDonald <A.MacDonald@uel.ac.uk>

  Noch nicht festgelegt, aber er ist flexibel und kann Linux und vielleicht HP-UX handhaben. Wird versuchen, Benutzer dazu zu bringen, `mysqlbug` zu benutzen.

- John Lyon <jlyon@imag.net>

  Fragen zur Installation von MySQL auf Linux-Systemen, entweder mit `.rpm`-Dateien oder durch Kompilieren vom Quelltext.

- Lorvid Ltd. <lorvid@WOLFENET.com>

  Einfache Fragen zu Rechnung / Lizenz / Support / Copyright.

- Patrick Sherrill <patrick@coconet.com>

  Fragen zur ODBC- und VisualC++-Schnittstelle.

- Rundy Harmon <rjharmon@uptimecomputers.com>

  `DBD`, Linux, und einige SQL-Syntax-Fragen.

C.3. Unterstützer von MySQL

Während `MySQL AB` das gesammte Copyright für den `MySQL Server` und das `MySQL manual` besitzt, möchten wir hier diejenigen Unternehmen nennen, die die Entwicklung des `MySQL Servers` unterstützt haben. Sie haben geholfen, indem sie uns für die Entwicklung eines neuen Features bezahlten, indem sie MySQL-Features selbst entwickelten oder indem sie uns Hardware für die MySQL-Entwicklung gaben.

- VA Linux / Andover.net

  Stiftete Replikation.

- NuSphere

  Editieren des MySQL-Referenzhandbuchs.

- Stork Design studio

  Die MySQL-Website zwischen 1998 und 2000.

- Intel

  Trugen zur Entwicklung auf Windows- und Linux-Plattformen bei.

- Compaq

  Trugen zur Entwicklung auf Linux/Alpha bei.

- SWSoft

  Entwicklung der eingebetteten `mysqld`-Version.

- FutureQuest

  `--skip-show-variables`

Anhang D. MySQL-Änderungsverlauf (Change History)

Dieser Anhang listet die Änderungen von Version zu Version im MySQL-Quellcode auf.

Beachten Sie, dass wir versuchen, das Handbuch zeitgleich mit den Änderungen an MySQL zu aktualisieren. Wenn Sie unten eine Version aufgelistet sehen, die Sie auf der MySQL-Download-Seite nicht finden können, heißt das, dass die Version noch nicht veröffentlicht wurde!

D.1. Änderungen in Release 4.0.x (Entwicklung; Alpha)

Wir arbeiten mittlerweile aktiv an MySQL 4.0 und werden nur noch kritische Bug-Bereinigungen für MySQL 3.23 herausgeben. Wir aktualisieren diesen Abschnitt, wenn wir neue Features hinzufügen, so dass andere unserer Entwicklung folgen können.

Unser TODO-Abschnitt enthält, was wir für 4.0 planen. See Abschnitt 2.8, „MySQL und die Zukunft (das TODO)“.

D.1.1. Änderungen in Release 4.0.2

- Bug in `FLUSH QUERY CACHE` behoben.
- `CAST()`- und `CONVERT()`-Funktionen hinzugefügt.
- Reihenfolge geändert, wie Schlüssel in Tabellen erzeugt werden.
- Neue Spalten `Null` und `Index_type` zu `SHOW INDEX` hinzugefügt.

D.1.2. Änderungen in Release 4.0.1

- Bug behoben, wenn `HANDLER` mit einem nicht unterstützten Tabellentyp verwendet wurde.
- `mysqldump` schreibt jetzt `ALTER TABLE tabelle DISABLE KEYS` und `ALTER TABLE tabelle DISABLE KEYS` in den SQL-Dump.
- `mysql_fix_extensions`-Skript hinzugefügt.
- Stack-Überlaufproblem `LOAD DATA FROM MASTER` auf OSF1 behoben.
- Herunterfahr-Problem auf HPUX behoben.
- Funktionen `des_encrypt()` und `des_decrypt()` hinzugefügt.
- Statement FLUSH DES_KEY_FILE hinzugefügt.
- mysqld-Option `--des-key-file` hinzugefügt.
- `HEX(string)` gibt jetzt die Buchstaben in der Zeichenkette konvertiert in hexadezimal zurück.
- Problem mit `GRANT` bei der Benutzung von `lower_case_tables == 1` behoben.
- `SELECT ... IN SHARE MODE` in `SELECT .. LOCK IN SHARE MODE` (wie in MySQL 3.23) geändert.
- Ein neuer Anfragen-Cache, der Ergebnisse identischer `SELECT`-Anfragen zwischenspeichert.
- Coredump-Bug auf 64-Bit-Maschinen beim Erhalt eines falschen Kommunikationspakets behoben.
- `MATCH ... AGAINST(... IN BOOLEAN MODE)` funktioniert jetzt auch ohne `FULLTEXT`-Index.
- Slave, der vom 3.23-Master repliziert, in Ordnung gebracht.
- Diverse Replikationsprobleme behoben / bereinigt.
- Herunterfahren funktioniert jetzt auf Mac OS X.
- `myisam/ft_dump`-Werkzeug zur Low-Level-Inspektion von `FULLTEXT`-Indexen hinzugefügt.
- Bug in `DELETE ... WHERE ... MATCH ...` behoben.

- Unterstützung für `MATCH ... AGAINST(... IN BOOLEAN MODE)` hinzugefügt. **Hinweis: Sie müssen Ihre Tabellen mit `ALTER TABLE tabelle TYPE=MyISAM` neu aufbauen, um Boole'sche Volltextsuche benutzen zu können.**
- `LOCATE()` und `INSTR()` sind abhängig von der verwendeten Groß-/Kleinschreibung, wenn keins der Argumente eine binäre Zeichenkette ist.
- `RND()`-Initialisierung geändert, so dass `RND(N)` und `RND(N+1)` verschiedener sind.
- Coredump-Bug in `UPDATE ... ORDER BY` behoben.
- `INSERT INTO .. SELECT` geändert, damit es bei Fehlern vorgabemäßig anhält.
- `DATA DIRECTORY`- und `INDEX DIRECTORY`-Anweisungen werden unter Windows ignoriert.
- Boole'sche Volltextsuche hinzugefügt. Diese sollte als frühe Alphaversion betrachtet werden.
- `MODIFY` und `CHANGE` in `ALTER TABLE` erweitert, damit sie das `AFTER`-Schlüsselwort akzeptieren.
- Index wird jetzt in `ORDER BY` von einer ganzen InnoDB-Tabelle verwendet.

D.1.3. Änderungen in Release 4.0.0

- Variablen `ft_min_word_len`, `ft_max_word_len` und `ft_max_word_len_for_sort` hinzugefügt.
- Dokumentation für `libmysqld`, die eingebettete MySQL-Server-Bibliothek, hinzugefügt. Beispielprogramme (ein `mysql`-Client und `mysqltest`-Testprogramm) hinzugefügt, die `libmysqld` benutzen.
- `my_thread_init()` und `my_thread_end()` aus mysql_com.h entfernt und `mysql_thread_init()` und `mysql_thread_end()` zu mysql.h hinzugefügt.
- Vorzeichenlose `BIGINT`-Konstanten funktionieren jetzt. `MIN()` und `MAX()` handhabt vorzeichenbehaftete und vorzeichenlose `BIGINT`-Zahlen korrekt.
- Neuer Zeichensatz `latin_de`, der korrektes deutsches Sortieren ermöglicht.
- `TRUNCATE TABLE` und `DELETE FROM tabelle` sind jetzt separate Funktionen. Ein Vorteil davon ist, dass `DELETE FROM tabelle` jetzt die Anzahl gelöschter Zeilen zurückgibt.
- `DROP DATABASE` führt jetzt ein `DROP TABLE` auf alle Tabellen in der Datenbank aus, was ein Problem mit InnoDB-Tabellen behebt.
- Unterstützung für `UNION` hinzugefügt.
- Eine neue `HANDLER`-Schnittstelle zu `MyISAM`-Tabellen.
- Unterstützung für `INSERT` auf `MERGE`-Tabellen hinzugefügt. Patch von Benjamin Pflugmann.
- `WEEK(#,0)` dem Kalender in den USA angepasst.
- `COUNT(DISTINCT)` ist etwa 30% schneller.
- Alle internen Listen-Handlings in der Geschwindigkeit verbessert.
- Das Erzeugen von Volltext-Indexen ist jetzt viel schneller.
- Baum-ähnlicher Cache, um Massen-Einfügevorgänge und die `myisam_bulk_insert_tree_size`-Variable zu beschleunigen.
- Suchen auf komprimierten (`CHAR`/`VARCHAR`)-Schlüsseln ist jetzt viel schneller.
- Anfragen folgenden Typs optimiert: `SELECT DISTINCT * from tabelle ORDER by schluessel_teil1 LIMIT #`
- `SHOW CREATE TABLE` zeigt jetzt alle Tabellenattribute.
- `ORDER BY ... DESC` kann jetzt Schlüssel benutzen.
- `LOAD DATA FROM MASTER` setzt jetzt "auto-magisch" einen Slave auf.

- `safe_mysqld` in `mysqld_safe` umbenannt.
- Unterstützung für symbolische Links auf `MyISAM`-Tabellen hinzugefügt. Symlink-Handhabung ist jetzt vorgabemäßig für Windows aktiviert.
- `LOAD DATA FROM MASTER` setzt "auto-magisch" einen Slave auf.
- `SQL_CALC_FOUND_ROWS` und `FOUND_ROWS()` hinzugefügt. Das ermöglicht es herauszufinden, wie viele Zeilen eine Anfrage ohne eine `LIMIT`-Klausel zurückgegeben hätte.
- Ausgabeformat von `SHOW OPEN TABLES` geändert.
- `SELECT ausdruck LIMIT ...` wird zugelassen.
- `IDENTITY` als Synonym für `AUTO_INCREMENT` hinzugefügt (wie Sybase).
- `ORDER BY`-Syntax zu `UPDATE` und `DELETE` hinzugefügt.
- `SHOW INDEXES` ist jetzt ein Synonym für `SHOW INDEX`.
- `ALTER TABLE tabelle DISABLE KEYS`- und `ALTER TABLE tabelle ENABLE KEYS`-Befehle hinzugefügt.
- `IN` kann anstelle von `FROM` in `SHOW`-Befehlen benutzt werden.
- ANSI-SQL-Syntax `X'Hexadezimalzahl'` wird zugelassen.
- Globale Sperr-Handhabung für `FLUSH TABLES with READ LOCK` aufgeräumt.
- Problem mit `DATETIME = constant` in `WHERE`-Optimierungen behoben.

D.2. Änderungen in Release 3.23.x (Stabil)

Das 3.23-Release hat etliche wichtige Features, die in früheren Versionen nicht vorhanden sind. Es wurden drei neue Tabellentypen hinzugefügt:

- **MyISAM**

  Eine neue ISAM-Bibliothek, die auf SQL und Unterstützung großer Dateien abgestimmt ist.

- **BerkeleyDB** oder **BDB**

  Benutzt die Berkeley-DB-Bibliothek von Sleepycat Software, um transaktionssichere Tabellen zu implementieren.

- **InnoDB**

  Ein transaktionssicherer Tabellen-Handler, der Sperren auf Zeilenebene und viele Oracle-ähnliche Features unterstützt.

Beachten Sie, dass nur MyISAM in der Standard-Binärdistribution verfügbar ist.

Das 3.23-Release beinhaltet ausserdem Unterstützung für Datenbank-Replikation zwischen einem Master und vielen Slaves, Volltext-Indexierung und vieles mehr.

Alle neuen Features werden in der 4.0-Version weiter entwickelt. Nur Bug-Behebungen und kleinere Verbesserungen bestehender Features werden zu 3.23 hinzugefügt.

Der Replikationscode und der BerkeleyDB-Code sind noch nicht so gut getestet wie der Rest des Codes, daher wird es wahrscheinlich zukünftig noch einige Releases von 3.23 mit kleineren Behebungen für diesen Teil des Codes geben. Solange Sie diese Features nicht benutzen, sollten Sie mit MySQL 3.23 auf der sicheren Seite liegen!

Beachten Sie, dass das Gesagte nicht heißt, dass Replikation oder Berkeley DB nicht funktionieren. Wir haben den gesamten Code ausgiebig getestet, inklusive Replikation und BDB, ohne irgend welche Probleme zu finden. Es heißt nur, dass nicht so viele Benutzer diesen Code verwenden wie den Rest des Codes, weshalb wir noch nicht 100% auf diesen Teil des Codes vertrauen.

D.2.1. Änderungen in Release 3.23.43

- Bug behoben, der mit sehr geringer Wahrscheinlichkeit auftritt, der keine übereinstimmenden Zeilen zurückgab bei SELECT

mit vielen Tabellen, mehrspaltigen Indexen und 'Bereichs-'-Typen.

- Coredump-Bug behoben, der mit sehr geringer Wahrscheinlichkeit auftritt, wenn man `EXPLAIN SELECT` mit vielen Tabellen und `ORDER BY` ausführt.
- Bug in `LOAD DATA FROM MASTER` bei der Benutzung einer Tabelle mit `CHECKSUM=1` behoben.
- Eindeutige Fehlermeldung hinzugefügt, die man bei einer Blockierung (Deadlock) während einer Transaktion mit BDB-Tabellen erhält.
- Problem mit `BDB`-Tabellen und `UNIQUE`-Spalten, die als `NULL` definiert wurden, behoben.
- Problem mit `myisampack` bei der Benutzung von CHAR-Spalten, die Leerzeichen aufgefüllt wurden, behoben.
- Patch von Yuri Dario für OS2 angewandt.
- Bug in `--safe-user-create` behoben.

D.2.2. Änderungen in Release 3.23.42

- Problem bei der Benutzung von `LOCK TABLES` und `BDB`-Tabellen behoben.
- Problem mit `REPAIR TABLE` auf MyISAM-Tabellen mit Zeilenlängen zwischen 65517 und 65520 Bytes behoben.
- Seltenen Hänger bei `mysqladmin shutdown` behoben, wenn es viel Aktivität auf einem anderen Thread gab.
- Problem mit `INSERT DELAYED` behoben, bei dem ein verzögerter Thread auf `Upgrade locks` ohne ersichtlichen Grund hängen konnte.
- Problem mit `myisampack` und `BLOB` behoben.
- Problem beim Editieren von `.MRG`-Tabellen von Hand behoben. (Patch von Benjamin Pflugmann).
- Es wird erzwungen, dass alle Tabellen in einer `MERGE`-Tabelle von derselben Datenbank kommen.
- Bug mit `LOAD DATA INFILE` und transaktionalen Tabellen behoben.
- Bug bei der Benutzung von `INSERT DELAYED` mit falschen Spaltendefinitionen behoben.
- Coredump während `REPAIR` besonders beschädigter Tabellen behoben.
- Bug in `InnoDB`- und `AUTO_INCREMENT`-Spalten behoben.
- Bug in `InnoDB` und `RENAME TABLE`-Spalten behoben.
- Kritischer Bug in `InnoDB`- und `BLOB`-Spalten behoben. Wenn man `BLOB`-Spalten größer als 8000 Bytes in einer `InnoDB`-Tabelle benutzte, musste man die Tabelle mit `mysqldump` sichern, löschen und aus dem Dump neu aufbauen.
- Großen Patch für OS/2 von Yuri Dario angewandt.
- Problem mit `InnoDB` behoben, bei dem man den Fehler `Can't execute the given command...` bekommen konnte, selbst wenn man keine aktive Transaktion hatte.
- Einige kleine Probleme behoben, die Gemini betrafen.
- Echte arithmetische Operationen werden selbst dann in einem Ganzzahl-Zusammenhang benutzt, wenn nicht alle Argumente Ganzzahlen sind. (Behebt einen wenig häufigen Bug in einigen Ganzzahl-Kontexten.)
- Unter Windows wird nicht alles in Kleinschreibung erzwungen (um ein Problem mit Windows und `ALTER TABLE` zu beheben). `--lower_case_names` funktioniert jetzt auch unter Unix.
- Automatisches Rollback behoben, das ausgeführt wurde, wenn das Beenden eines Threads keinen anderen Thread blockiert.

D.2.3. Änderungen in Release 3.23.41

- Option `--sql-mode=option[,option[,option]]` hinzugefügt. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

- Mögliches Problem mit `shutdown` auf Solaris behoben, wobei die `.pid`-Datei nicht gelöscht wurde.
- InnoDB unterstützt jetzt Zeilen < 4 GB. Die vorherige Beschränkung war 8.000 Bytes.
- Die `doublewrite`-Datei-Flush-Methode wird in InnoDB benutzt. Sie reduziert die Notwendigkeit von Unix-fsync-Aufrufen auf einen Bruchteil und verbessert die Performance auf den meisten Unix-Varianten.
- Sie können jetzt den InnoDB-Monitor benutzen, um etliche Informationen über den InnoDB-Status auf die Standardausgabe auszugeben, inklusive Sperren. Nützlich zum Tunen der Performance.
- Mehrere Bugs, die in InnoDB Hänger verursachen konnten, behoben.
- `record_buffer` in `record_buffer` und `record_rnd_buffer` aufgeteilt. Um zu vorherigen MySQL-Versionen kompatibel zu bleiben, wird `record_rnd_buffer` auf den Wert von `record_buffer` gesetzt, wenn es nicht explizit gesetzt wird.
- Optimierungs-Bug in `ORDER BY` behoben, bei dem einige `ORDER BY`-Teile fälschlicherweise entfernt wurden.
- Overflow-Bug bei `ALTER TABLE` und `MERGE`-Tabellen behoben.
- Prototypen für `my_thread_init()` und `my_thread_end()` zu `mysql_com.h` hinzugefügt.
- Option `--safe-user-create` to `mysqld` hinzugefügt.
- Bug in `SELECT DISTINCT ... HAVING` behoben, der die Fehlermeldung `Can't find record in '#...` verursachte.

D.2.4. Änderungen in Release 3.23.40

- Problem mit `--low-priority-updates` und `INSERT`'s behoben.
- Bug im Slave-Thread beseitigt, bei dem dieser in seltenen Fällen um 22 Byte vor den Offset im Master kommen konnte.
- `slave_wait_timeout` für Replikation hinzugefügt.
- Problem mit `UPDATE` und `BDB`-Tabellen behoben.
- Problematischen Bug in `BDB`-Tabellen behoben, der bei der Benutzung von Schlüsselteilen auftrat.
- Problem bei der Benutzung von `GRANT FILE ON datenbank.* ...` behoben. Vorher wurde die `DROP`-Berechtigung für die Datenbank hinzugefügt.
- Bug bei `DELETE FROM tabelle ... LIMIT 0` und `UPDATE FROM tabelle ... LIMIT 0` behoben, die sich vorher so verhielten, als gäbe es keine `LIMIT`-Klausel (sie löschten oder aktualisierten alle ausgewählten Zeilen).
- `CHECK TABLE` prüft jetzt, ob eine `AUTO_INCREMENT`-Spalte den Wert 0 enthält.
- Wenn man `SIGHUP` an `mysqld` schickt, werden jetzt nur die Logs auf Platte zurückgeschrieben (flush), nicht die Replikation zurückgesetzt.
- Parser in Ordnung gebracht, so dass er jetzt Fließkommazahlen des Typs `1.0e1` (kein Vorzeichen nach `e`) zuläßt.
- Option `--force` für `myisamchk` aktualisiert jetzt auch Zustände (Status).
- Option `--warnings` für `mysqld` hinzugefügt. `mysqld` gibt jetzt nur den Fehler `Aborted connection` aus, wenn diese Option benutzt wird.
- Problem mit `SHOW CREATE TABLE` behoben, wenn man keinen `PRIMARY KEY` hatte.
- Saubere Behebung der Umbenennung von `innodb_unix_file_flush_method` in `innodb_flush_method`.
- Bug beim Umwandeln von `UNSIGNED BIGINT` in `DOUBLE` behoben. Dieser verursachte bei Vergleichen mit `BIGINT`-Werten ausserhalb des vorzeichenbehafteten Bereichs ein Problem.
- Bug in `BDB`-Tabellen behoben, wenn man leere Tabellen abfragte.
- Bug bei der Benutzung von `COUNT(DISTINCT)` mit `LEFT JOIN` behoben, wenn es keine übereinstimmenden Zeilen gab.
- Alle Dokumentation bezüglich `GEMINI`-Tabellen entfernt. `GEMINI` wird nicht unter einer Open-Source-Lizenz

herausgegeben.

D.2.5. Änderungen in Release 3.23.39

- Die `AUTO_INCREMENT`-Zahlenfolge wurde beim Löschen und Hinzufügen einer `AUTO_INCREMENT`-Spalte nicht zurückgesetzt.
- `CREATE ... SELECT` erzeugt jetzt nicht eindeutige Indexe verzögert.
- Problem behoben, bei dem `LOCK TABLES tabelle READ` gefolgt von `FLUSH TABLES` eine exklusive Sperre auf die Tabelle setzte.
- `REAL`-@Variablen wurden mit 2 Ziffern dargestellt, wenn sie in Zeichenketten umgewandelt wurden.
- Problem mit hängendem Client behoben, wenn `LOAD TABLE FROM MASTER` fehlschlug.
- `myisamchk --fast --force` repariert jetzt keine Tabellen mehr, bei denen nur der Öffnen-Zähler falsch ist.
- Funktionen zur Handhabung von symbolischen Links hinzugefügt, um sich das Leben in Version 4.0 zu erleichtern.
- Wir benutzen jetzt die `-lcma`-Thread-Bibliothek unter HP-UX 10.20, so dass MySQL auf HP-UX stabiler läuft.
- Problem mit `IF()` und Anzahl von Dezimalstellen im Ergebnis behoben.
- Funktionen zum Extrahieren von Datumsanteilen in Ordnung gebracht, so dass sie jetzt mit Datumsangaben funktionieren, bei denen Tag und / oder Monat 0 sind.
- Argumentlänge in Optionsdateien von 256 auf 512 Zeichen erweitert.
- Problem bei Herunterfahren, wenn `INSERT DELAYED` auf ein `LOCK TABLE` wartete, behoben.
- Coredump-Bug in InnoDB behoben, wenn der Tabellenplatz (Tablespace) voll war.
- Problem mit `MERGE`-Tabellen und großen Tabellen (> 4 GB) und der Benutzung von `ORDER BY` behoben.

D.2.6. Änderungen in Release 3.23.38

- Bug behoben, bei dem `SELECT` von `MERGE`-Tabellen manchmal zu falsch sortierten Zeilen führte.
- Bug in `REPLACE()` bei der Benutzung des ujis-Zeichensatzes behoben.
- Sleepycat-BDB-Patches 3.2.9.1 und 3.2.9.2 angewandt.
- Option `--skip-stack-trace` zu `mysqld` hinzugefügt.
- `CREATE TEMPORARY` funktioniert jetzt mit `InnoDB`-Tabellen.
- `InnoDB` zieht jetzt Teile von Schlüsseln (Sub-Keys) ganzen Schlüsseln vor.
- Option `CONCURRENT` für `LOAD DATA` hinzugefügt.
- Bessere Fehlermeldung, wenn die Slave-`max_allowed_packet`-Variable zu niedrig ist, um ein sehr langes Log-Ereignis vom Master zu lesen.
- Bug behoben, wenn zu viele Zeilen bei der Benutzung von `SELECT DISTINCT ... HAVING` entfernt wurden.
- `SHOW CREATE TABLE` gibt jetzt `TEMPORARY` für temporäre Tabellen zurück.
- `Rows_examined` für Langsame-Anfragen-Log-Datei hinzugefügt.
- Probleme mit Funktion behoben, die eine leere Zeichenkette zurückgab, wenn sie zusammen mit einer Gruppenfunktion und einem `WHERE` benutzt wurde, das keine Zeilenübereinstimmung ergab.
- Neues Programm `mysqlcheck`.
- Datenbankname zur Ausgabe für administrative Befehle wie `CHECK`, `REPAIR`, `OPTIMIZE` hinzugefügt.

- Viele Portabilitätsbehebungen für InnoDB.
- Optimierer geändert, so dass Anfragen wie `SELECT * FROM tabelle,tabelle2 ... ORDER BY schluessel_teil1 LIMIT #` den Index auf `schluessel_teil1` anstelle von `filesort` benutzen.
- Bug bei der Ausführung von `LOCK TABLE to_table WRITE,...; INSERT INTO to_table... SELECT ...` behoben, wenn `to_table` leer war.
- Bug mit `LOCK TABLE` und BDB-Tabellen behoben.

D.2.7. Änderungen in Release 3.23.37

- Bug bei der Benutzung von `MATCH` in `HAVING`-Klausel.
- Bug bei der Benutzung von `HEAP`-Tabellen mit `LIKE` behoben.
- `--mysql-version` für `safe_mysqld` hinzugefügt.
- `INNOBASE` in `InnoDB` geändert (weil der `INNOBASE`-Name bereits benutzt wurde). Alle `configure`-Optionen und `mysqld`-Startoptionen benutzen jetzt `innodb` anstelle von `innobase`. Das heißt, dass Sie jegliche Konfigurationsdateien, in denen Sie `innobase`-Optionen benutzt haben, ändern müssen, bevor Sie auf diese Version aktualisieren!
- Bug bei der Benutzung von Indexen auf `CHAR(255) NULL`-Spalten behoben.
- Slave-Thread wird jetzt auch dann gestartet, wenn `master-host` nicht gesetzt ist, so lange `server-id` gesetzt wird und es eine gültige `master.info` gibt.
- Teilweise Aktualisierungen (beendet mit kill) werden jetzt mit einem speziellen Fehler-Code in die Binär-Log-Datei geschrieben. Der Slave weigert sich, sie auszuführen, wenn der Fehler-Code anzeigt, dass die Aktualisierung abnorm beendet wurde, und muss mit `SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START` wieder dazu veranlasst werden, nachdem eine manuelle Überprüfung / Korrektur der Datenintegrität durchgeführt wurde.
- Bug behoben, der das Löschen einer internen temporären Tabelle beim Beenden des Threads irrtümlicherweise in die Binär-Log-Datei schrieb. Dieser Bug betraf Replikation.
- Bug in `REGEXP()` auf 64-Bit-Maschinen.
- `UPDATE` und `DELETE` mit `WHERE eindeutiger_schluessel_teil IS NULL` aktualisierte / löschte nicht alle Zeilen.
- `INSERT DELAYED` für Tabellen abgeschaltet, die Transaktionen unterstützen.
- Bug bei der Benutzung von DATE-Funktionen auf `TEXT`/`BLOB`-Spalten mit falschem Datumsformat behoben.
- UDFs (benutzerdefinierte Funktionen) funktionieren jetzt auch unter Windows (Patch von Ralph Masona).
- Bug in `ALTER TABLE` und `LOAD DATA INFILE` behoben, der das Sortieren von Schlüsseln deaktivierte. Diese Befehle sollten jetzt in den meisten Fällen schneller sein.
- Performance-Bug beim erneuten Öffnen von Tabellen behoben (Tabellen, die auf ein `FLUSH` oder `REPAIR` warteten), die für die nächste Anfrage keine Indexe benutzten.
- Problem mit `ALTER TABLE` für Innobase-Tabellen auf FreeBSD behoben.
- `mysqld`-Variablen `myisam_max_sort_file_size` und `myisam_max_extra_sort_file_size` hinzugefügt.
- Signale werden frühzeitig initialisiert, um Problem mit Signalen in Innobase zu vermeiden.
- Patch für den `tis620`-Zeichensatz hinzugefügt, um Vergleiche unabhängig von der verwendeten Groß-/Kleinschreibung zu machen und einen Bug in `LIKE` für diesen Zeichensatz zu beheben. **HINWEIS**: Alle Tabellen, die den `tis620`-Zeichensatz benutzen, müssen mit `myisamchk -r` oder `REPAIR TABLE` in Ordnung gebracht werden!
- `--skip-safemalloc`-Option zu `mysqld` hinzugefügt.

D.2.8. Änderungen in Release 3.23.36

- Bug behoben, der Datenbanknamen mit einem '`.`'-Zeichen zuließ. Das behebt ein schwerwiegendes Sicherheitsproblem, wenn

man `mysqld` unter dem Benutzer root laufen läßt.

- Bug behoben, wenn die Erzeugung eines Threads fehlschlägt (das konnte bei der Herstellung SEHR vieler Verbindungen in kurzer Zeit passieren).
- Einige Probleme mit `FLUSH TABLES` und `TEMPORARY`-Tabellen behoben. (Problem mit dem Freisetzen des Schlüssel-Cache und Fehler `Can't reopen table...`).
- Problem in Innobase mit anderen Zeichensätze als `latin1` und ein anderes Problem bei der Benutzung von vielen Spalten behoben.
- Bug behoben, der einen Coredump bei der Benutzung einer sehr komplexen Anfrage mit `DISTINCT` und Summenfunktionen verursachte.
- `SET TRANSACTION ISOLATION LEVEL ...` hinzugefügt.
- `SELECT ... FOR UPDATE` hinzugefügt.
- Bug behoben, bei dem die Anzahl von betroffenen Zeilen nicht zurückgegeben wurde, wenn `MySQL` ohne Transaktionsunterstützung kompiliert wurde.
- Bug in `UPDATE` behoben, bei dem nicht immer Schlüssel benutzt wurden, um die zu aktualisierenden Zeilen zu finden.
- Bug in `CONCAT_WS()` behoben, bei dem diese Funktion falsche Ergebnisse zurückgab.
- `CREATE ... INSERT` und `INSERT ... SELECT` geändert, so dass diese noch kleine gleichzeitigen Einfügevorgänge zulassen, weil das dazu führen könnte, dass die Binär-Log-Datei schwer zu wiederholen ist. (Gleichzeitige Einfügevorgänge sind aktiviert, wenn Sie nicht die Binär- oder Update-Log-Datei verwenden.)
- Einige Makros geändert, so dass schnelles mutex mit glibc 2.2 verwendet werden kann.

D.2.9. Änderungen in Release 3.23.35

- Neu eingeführter Bug in `ORDER BY` behoben.
- Falsches Definieren von `CLIENT_TRANSACTIONS` behoben.
- Bug in `SHOW VARIABLES` bei der Benutzung von `INNOBASE`-Tabellen behoben.
- Das Setzen und Benutzen von Benutzer-Variablen in `SELECT DISTINCT` funktionierte nicht.
- `SHOW ANALYZE` für kleine Tabellen verbessert.
- Handhabung von Argumenten im Benchmark-Skript `run-all-tests` behoben.

D.2.10. Änderungen in Release 3.23.34a

- Zusätzliche Dateien zur Distribution hinzugefügt, die es ermöglichen, mit `INNOBASE`-Unterstützung zu kompilieren.

D.2.11. Änderungen in Release 3.23.34

- `INNOBASE`-Tabellen-Handler und `BDB`-Tabellen-Handler zur MySQL-Quelldistribution hinzugefügt.
- Die Dokumentation zu `GEMINI`-Tabellen aktualisiert.
- Bug in `INSERT DELAYED` behoben, der den Thread zum Hängen brachte, wenn `NULL` in eine `AUTO_INCREMENT`-Spalte eingefügt wurde.
- Bug in `CHECK TABLE` / `REPAIR TABLE` behoben, der einen Thread zum Hängen bringen konnte.
- `REPLACE` ersetzt keine Zeile mehr, die mit einem durch `auto_increment` erzeugten Schlüssel in Konflikt steht.
- `mysqld` setzt jetzt nur `CLIENT_TRANSACTIONS` in `mysql->server_capabilities`, wenn der Server einen transaktionssicheren Handler unterstützt.

- `LOAD DATA INFILE` läßt jetzt das Einfügen numerischer Werte in `ENUM`- und `SET`-Spalten zu.
- Fehlerdiagnose beim Slave-Thread-Exit verbessert.
- Bug in `ALTER TABLE ... ORDER BY` behoben.
- Option `max_user_connections` für `mysqld` hinzugefügt.
- Anfragelänge für Replikation auf `max_allowed_packet` begrenzt, nicht auf die willkürliche Länge von 4 MB.
- Leerzeichen um `=` herum im Argument zu `--set-variable` werden zugelassen.
- Problem beim automatischen Reparieren behoben, der einen Thread im Zustand `Waiting for table` lassen konnte.
- `SHOW CREATE TABLE` gibt jetzt das `UNION()` für `MERGE`-Tabellen aus.
- `ALTER TABLE` merkt sich jetzt die alte `UNION()`-Definition.
- Bug beim Replizieren von Timestamps behoben.
- Bug bei der bidirektionalen Replikation behoben.
- Bug im `BDB`-Tabellen-Handler behoben, der bei der Benutzung eines Indexes auf mehrteilige Schlüssel vorkam, wenn ein Schlüsselteil `NULL` sein konnte.
- `MAX()`-Optimierung für Schlüsselteile (Sub-Keys) für `BDB`-Tabellen verbessert.
- Problem behoben, bei dem 'Müll'-Ergebnisse bei der Benutzung von `BDB`-Tabellen und `BLOB`- oder `TEXT`-Feldern beim Verknüpfen (Join) vieler Tabellen auftraten.
- Problem mit `BDB`-Tabellen und `TEXT`-Spalten behoben.
- Bug bei der Benutzung eines `BLOB`-Schlüssels behoben, wenn eine Konstanten-Zeile nicht gefunden wurde.
- Problem behoben, dass `mysqlbinlog` den Timestamp-Wert für jede Anfrage schreibt. Das stellt sicher, dass man dieselben Werte bei Datumsfunktionen wie `NOW()` bei der Benutzung von `mysqlbinlog` erhält, um die Anfragen zu einem anderen Server durchzureichen (pipe).
- Es wird zugelassen, dass `--skip-gemini`, `--skip-bdb` und `--skip-innobase` für `mysqld` angegeben werden, selbst wenn diese Datenbanken nicht in `mysqld` einkompiliert sind.
- Man kann jetzt `GROUP BY ... DESC` ausführen.
- Blockierung im `SET`-Code behoben, wenn man `SET @foo=bar` ausführte, wobei `bar` ein Spaltenverweis ist. Hier wurde die Fehlermeldung nicht korrekt erzeugt.

D.2.12. Änderungen in Release 3.23.33

- DNS-Lookups benutzen jetzt nicht mehr denselben mutex wie der Hostnamen-Cache. Das gestattet, dass bekannte Hosts schnell aufgelöst werden können, selbst wenn ein DNS-Lookup lange Zeit braucht.
- `--character-sets-dir` für `myisampack` hinzugefügt.
- Warnungen beim Laufenlassen von `REPAIR TABLE ... EXTENDED` entfernt.
- Bug, der einen Coredump bei der Benutzung von `GROUP BY` auf ein Alias verursachte, wobei der Alias dasselbe wie ein existierender Spaltenname war, entfernt.
- `SEQUENCE()` als Beispiel-UDF-Funktion hinzugefügt.
- `mysql_install_db` geändert, so dass es `BINARY` für `CHAR`-Spalten in den Berechtigungstabellen benutzt.
- `TRUNCATE tabelle` zu `TRUNCATE TABLE tabelle` geändert, um dieselbe Syntax wie Oracle zu verwenden. Bis Version 4.0 lassen wir weiterhin `TRUNCATE tabelle` zu, um alten Code nicht zum Absturz zu bringen.
- 'no found rows'-Bug in `MyISAM`-Tabellen behoben, wenn ein `BLOB` erster Teil eines mehrteiligen Schlüssels war.
- Bug behoben, bei dem `CASE` mit `GROUP BY` nicht funktionierte.

- Option `--sort-recover` für `myisamchk` hinzugefügt.
- `myisamchk -S` und `OPTIMIZE TABLE` funktionieren jetzt unter Windows.
- Bug bei der Benutzung von `DISTINCT` auf Ergebnisse von Funktionen behoben, die sich auf eine Gruppenfunktion bezogen, wie:

```
SELECT a, DISTINCT SEC_TO_TIME(sum(a)) from tabelle GROUP BY a, b;
```

- Puffer-Überlauf in `libmysqlclient`-Bibliothek behoben. Bug bei der Handhabung des `STOP`-Ereignisses nach `ROTATE`-Ereignis bei Replikation.
- Einen weiteren Puffer-Überlauf in `DROP DATABASE` behoben.
- `Table_locks_immediate`- und `Table_locks_waited`-Status-Variablen hinzugefügt.
- Bug in Replikation behoben, der den Slave-Server-Start bei existierendem `master.info` unterbrach. Das behebt einen Bug, der in Version 3.23.32 eingeführt wurde.
- `SET SQL_SLAVE_SKIP_COUNTER=n`-Befehl hinzugefügt, um nach Replikationsstörungen ohne volle Datenbankkopie wiederherzustellen.
- `max_binlog_size`-Variable hinzugefügt; die Binär-Log-Datei wird automatisch rotiert, wenn die Größe die Grenze überschreitet.
- `Last_error`, `Last_errno` und `Slave_skip_counter` für `SHOW SLAVE STATUS` hinzugefügt.
- Bug in `MASTER_POS_WAIT()`-Funktion behoben.
- Coredump-Handler auf `SIGILL` und `SIGBUS` zusätzlich zu `SIGSEGV`.
- Auf x86-Linux wird im Coredump-Handler die aktuelle Anfrage und die Thread- (Verbindungs-) Kennung, falls verfügbar, angegeben.
- Mehrere Timing-Bugs in der Test-Suite behoben.
- `mysqltest` erweitert, so dass es sich um Probleme mit dem Timing in der Test-Suite kümmert.
- `ALTER TABLE` kann jetzt auch benutzt werden, um die Definition einer `MERGE`-Tabelle zu ändern.
- Erzeugung von `MERGE`-Tabellen unter Windows in Ordnung gebracht.
- Portabilitätsbehebungen für OpenBSD und OS/2.
- `--temp-pool`-Option zu `mysqld` hinzugefügt. Das Benutzen dieser Option führt dazu, dass temporäre Dateien nur einen kleinen Satz von Namen benutzen, statt eines eindeutigen Namens für jede neue Datei. Das ist ein Workaround um ein Problem im Linux-Kernel beim Erzeugen einer großen Menge neuer Dateien mit unterschiedlichen Namen. Beim alten Verhalten scheint es so, als ein Linux ein Speicher-'Loch' hätte, weil zum Verzeichniseintrags-Cache statt zum Festplatten-Cache zugewiesen (alloziert) wird.

D.2.13. Änderungen in Release 3.23.32

- Code geändert, um um einen Compiler-Bug in Compaq C++ auf OSF1 herumzuarbeiten, der `BACKUP`, `RESTORE`, `CHECK`, `REPAIR` und `ANALYZE TABLE` beschädigte.
- Option `FULL` für `SHOW COLUMNS` hinzugefügt. Jetzt wird die Berechtigungsliste für die Spalten nur angezeigt, wenn diese Option angegeben wird.
- Bug in `SHOW LOGS` behoben, wenn es keine BDB-Logs gab.
- Timing-Problem in Replikation behoben, dass das Abschicken einer Aktualisierung an den Client verzögern konnte, bis eine weitere Aktualisierung durchgeführt wurde.
- Feldnamen bei der Benutzung von `mysql_list_fields()` nicht umwandeln. Damit bleibt dieser Code kompatibel mit `SHOW FIELDS`.
- `MERGE`-Tabellen funktionierten nicht unter Windows.
- Problem mit `SET PASSWORD=...` unter Windows.

- Fehlende `my_config.h` für RPM-Distribution hinzugefügt.
- `TRIM("foo" von "foo")` gab keine leere Zeichenkette zurück.
- `--with-version-suffix` für `configure` hinzugefügt.
- Coredump behoben, wenn der Client eine Verbindung ohne `mysql_close()` abbrach.
- Bug in `RESTORE TABLE` behoben, wenn man versuchte, aus einem nicht vorhandenen Verzeichnis wiederherzustellen.
- Bug behoben, der einen Coredump auf dem Slave bei der Replikation von `SET PASSWORD` verursachte.
- `MASTER_POS_WAIT()` hinzugefügt.

D.2.14. Änderungen in Release 3.23.31

- Die Test-Suite testet jetzt jeden erreichbaren BDB-Schnittstellen-Code. Während der Tests fanden und behoben wir viele Fehler im Schnittstelle-Code.
- Die Benutzung von `HAVING` auf eine leere Tabelle konnte eine Ergebniszeile ergeben, ohne dass es das sollte.
- Problem behoben, so dass das MySQL-RPM nicht mehr von Perl5 abhängt.
- Einige Probleme mit `HEAP`-Tabellen unter Windows behoben.
- `SHOW TABLE STATUS` zeigte nicht die korrekte durchschnittliche Zeilenlänge bei Tabellen größer als 4 GB.
- `CHECK TABLE ... EXTENDED` prüften keine Zeilen-Links für Tabellen fester Größe.
- Option `MEDIUM` für `CHECK TABLE` hinzugefügt.
- Problem bei der Benutzung von `DECIMAL()`-Schlüsseln auf negative Zahlen behoben.
- `HOUR()` (und einige andere `TIME`-Funktionen) auf einer `CHAR`-Spalte gaben immer `NULL` zurück.
- Sicherheits-Bug in etwas behoben (bitte aktualisieren Sie, wenn Sie eine frühere MySQL-3.23-Version benutzen).
- Bug mit Puffer-Überlauf behoben, wenn eine bestimmte Fehlermeldung ausgegeben wurde.
- Benutzung von `setrlimit()` unter Linux hinzugefügt, damit `-O --open-files-limit=#` unter Linux läuft.
- Neue `mysqld`-Variable `bdb_version` hinzugefügt.
- Bug bei der Benutzung von Ausdrücken folgenden Typs behoben:

  ```
  SELECT ... FROM t1 LEFT JOIN t2 ON (t1.a=t2.a) WHERE t1.a=t2.a
  ```

  In diesem Fall wurde der Test in der `WHERE`-Klausel fälschlicherweise weg optimiert.
- Bug in `MyISAM` beim Löschen von Schlüsseln mit möglichen `NULL`-Werten behoben, wenn die erste Schlüsselspalte keine Präfix-komprimierte Text-Spalte war.
- `mysql.server` repariert, so dass es den `mysql.server`-Optionsabschnitt anstelle von `mysql_server` liest.
- `safe_mysqld` und `mysql.server` repariert, so dass sie den `server`-Optionsabschnitt lesen.
- `thread_created`-Status-Variable zu `mysqld` hinzugefügt.

D.2.15. Änderungen in Release 3.23.30

- `SHOW OPEN TABLES`-Befehl hinzugefügt.
- `myisamdump` funktioniert jetzt mit alten `mysqld`-Servern.
- `myisamchk -k#` funktioniert jetzt wieder.
- Problem mit Replikation behoben, wenn die Binär-Log-Datei-Datei auf 32-Bit-Systemen größer als 2 GB wurde.

- `LOCK TABLES` startet jetzt automatisch eine neue Transaktion.
- `BDB`-Tabellen so geändert, dass sie interne Sub-Transaktionen benutzen und offene Dateien wiederholt benutzen, um mehr Geschwindigkeit zu erzielen.
- Option `--mysqld=#` für `safe_mysqld` hinzugefügt.
- Hexadezimale Konstanten in `--fields-*-by`- und `--lines-terminated-by`-Optionen für `mysqldump` und `mysqlimport` werden jetzt zugelassen. Von Paul DuBois.
- Option `--safe-show-Datenbank` für `mysqld` hinzugefügt.
- `have_bdb`, `have_gemini`, `have_innobase`, `have_raid` und `have_openssl` für `SHOW VARIABLES` hinzugefügt, um das Testen auf unterstützte Erweiterungen zu erleichtern.
- Option `--open-files-limit` für `mysqld` hinzugefügt.
- Option `--open-files` zu `--open-files-limit` in `safe_mysqld` geändert.
- Bug behoben, bei dem einige Zeilen nicht gefunden wurden, wenn `HEAP`-Tabellen viele Schlüssel hatten.
- `--bdb-no-sync` funktioniert jetzt.
- `--bdb-recover` in `--bdb-no-recover` geändert, weil recover vorgabemäßig angeschaltet sein sollte.
- Die vorgabemäßige Anzahl von BDB-Sperren auf 10.000 geändert.
- Bug aus Version 3.23.29 behoben, beim Zuweisen der gemeinsam genutzten Struktur, die für BDB-Tabellen benötigt wird.
- `mysqld_multi.sh` für benutzerkonfigurierbare Variablen geändert. Patch von Christopher McCrory.
- Include-Dateien für Solaris 2.8 in Ordnung gebracht.
- Bug mit `--skip-networking` auf Debian Linux behoben.
- Problem behoben, dass einige temporäre Dateien den Namen `UNOPENED` in Fehlermeldungen hatten.
- Bug beim Laufenlassen von zwei gleichzeitigen `SHOW LOGS`-Anfragen behoben.

D.2.16. Änderungen in Release 3.23.29

- Configure-Aktualisierungen für Tru64, Unterstützung großer Dateien und besser TCP-Wrapper-Unterstützung. Von Albert Chin-A-Young.
- Bug in `<=>`-Operator behoben.
- Bug in `REPLACE` mit BDB-Tabellen behoben.
- `LPAD()` und `RPAD()` kürzen jetzt die Ergebnis-Zeichenkette, wenn sie länger als das Längenargument ist.
- `SHOW LOGS`-Befehl hinzugefügt.
- Unbenutzte BDB-Logs werden beim Herunterfahren entfernt.
- Beim Erzeugen einer Tabelle werden `PRIMARY`-Schlüsseln zuerst gesetzt, gefolgt von `UNIQUE`-Schlüsseln.
- Bug in `UPDATE` behoben, wenn mehrteilige Schlüssel benutzt wurden, bei denen alle Schlüsselteile sowohl in der Aktualisierung als auch im `WHERE`-Teil angegeben wurden. In diesem Fall könnte MySQL versuchen, einen Datensatz zu aktualisieren, der nicht dem gesamten `WHERE`-Teil entspricht.
- Löschen von Tabellen so geändert, dass zunächst die Tabelle und dann die `.frm`-Datei gelöscht wird.
- Bug im Hostnamen-Cache behoben, der dazu führte, dass `mysqld` den Hostnamen als `''` in manchen Fehlermeldungen berichtete.
- Bug mit `HEAP`-Tabellen behoben; die Variable `max_heap_table_size` wurde nicht benutzt. Jetzt kann entweder `MAX_ROWS` oder `max_heap_table_size` benutzt werden, um die Größe einer `HEAP`-Tabelle zu beschränken.
- Die vorgabemäßige Server-Kennung auf 1 für Master-Server und 2 für Slaves geändert, um die Benutzung der Binär-Log-Datei

zu erleichtern.

- Variable `bdb_lock_max` in `bdb_max_lock` umbenannt.
- Unterstützung für `auto_increment` auf Unter-Felder (Sub-Fields) für BDB-Tabellen hinzugefügt.
- `ANALYZE` von BDB-Tabellen hinzugefügt.
- In BDB-Tabellen wird jetzt die Anzahl von Zeilen gespeichert. Das hilft, Anfragen zu optimieren, wenn dafür die ungefähre Anzahl von Zeilen benötigt wird.
- Wenn es einen Fehler in einem mehrzeiligen Statement gibt, wird jetzt nur das letzte Statement zurückgerollt, nicht die gesamte Transaktion.
- Wenn man `ROLLBACK` nach der Aktualisierung einer nicht transaktionalen Tabelle ausführt, erhält man als Warnung einen Fehler.
- Option `--bdb-shared-data` für `mysqld` hinzugefügt.
- Status-Variable `Slave_open_temp_tables` hinzugefügt.
- Variablen `binlog_cache_size` und `max_binlog_cache_size` für `mysqld` hinzugefügt.
- `DROP TABLE`, `RENAME TABLE`, `CREATE INDEX` und `DROP INDEX` sind jetzt Transaktions-Endpunkte.
- Wenn Sie ein `DROP DATABASE` auf eine symbolisch verknüpfte Datenbank ausführen, werden sowohl der Link als auch die Original-Datenbank gelöscht.
- `DROP DATABASE` funktioniert jetzt auf OS/2.
- Bug bei der Ausführung von `SELECT DISTINCT ... tabelle1 LEFT JOIN tabelle2 ...` behoben, wenn tabelle2 leer war.
- `--abort-slave-event-count`- und `--disconnect-slave-event-count`-Optionen für `mysqld` zum Debuggen und Testen der Replikation hinzugefügt.
- Replikation temporärer Tabellen in Ordnung gebracht. Handhabt alles ausser dem Neustart von Slaves.
- `SHOW KEYS` zeigt jetzt, ob ein Schlüssel `FULLTEXT` ist oder nicht.
- Neues Skript `mysqld_multi`. See Abschnitt 5.7.3, „mysqld_multi, Programm zur Verwaltung mehrerer MySQL-Server“.
- Neues Skript `mysql-multi.server.sh` hinzugefügt. Vielen Dank an Tim Bunce `<Tim.Bunce@ig.co.uk>` für die Modifizierung von `mysql.server`, um auf einfache Weise Hosts zu handhaben, die viele `mysqld`-Prozesse laufen lassen.
- `safe_mysqld`, `mysql.server` und `mysql_install_db` wurden so abgeändert, dass sie `mysql_print_defaults` anstelle verschiedener Hacks benutzen, um `my.cnf`-Dateien zu lesen. Zusätzlich wurde die Handhabung verschiedener Pfade konsistenter gemacht, in Bezug auf wie `mysqld` vorgabemäßig handhabt.
- Berkeley-DB-Transaktions-Logs, die nicht mehr in Benutzung sind, werden automatisch entfernt.
- Bug bei mehreren `FULLTEXT`-Indexen in einer Tabelle behoben.
- Warnung hinzugefügt, wenn sich die von Zeilen bei `REPAIR`/`OPTIMIZE` ändert.
- Patches für OS/2 von `Yuri Dario` angewandt.
- `FLUSH TABLES tabelle` schrieb den Index-Baum nicht immer korrekt auf die Festplatte zurück.
- `--bootstrap` läuft jetzt in einem separaten Thread. Das behebt ein Problem, das bei `mysql_install_db` einen Coredump auf einigen Linux-Maschinen verursachte.
- `mi_create()` abgeändert, so dass es weniger Stack-Platz benötigt.
- Bug beim Optimierer, wenn er versucht, `MATCH`, mit `UNIQUE`-Schlüsseln benutzt, zu überoptimieren.
- `Crash-me` und die MySQL-Benchmarks funktionieren jetzt auch mit FrontBase.
- `RESTRICT` und `CASCADE` werden nach einem `DROP TABLE` zugelassen, um die Portierung einfacher zu machen.
- Status-Variable zurückgesetzt, die Probleme hervorrufen konnte, wenn man `--slow-log` benutzte.

- Variable `connect_timeout` für `mysql` und `mysqladmin` hinzugefügt.
- `connect_timeout` als Alias für `timeout` für Optionsdateien, die von `mysql_options()` gelesen werden, hinzugefügt.

D.2.17. Änderungen in Release 3.23.28

- Neue Optionen `--pager[=...]`, `--no-pager`, `--tee=...` und `--no-tee` für den `mysql`-Client hinzugefügt. Die entsprechenden neuen interaktiven Befehle heißen `pager`, `nopager`, `tee` und `notee`. Siehe See Abschnitt 5.8.2, „Das Kommandozeilen-Werkzeug“, `mysql --help` und die interaktive Hilfe wegen weiterer Informationen.
- Absturz behoben, der beim Fehlschlagen der Reparatur von `MyISAM`-Tabellen auftrat.
- Größerer Performance-Bug im Tabellensperren-Code behoben, wenn man permanent VIELE `SELECT`-, `UPDATE`- und `INSERT`-Statements laufen hatte. Das Symptom zeigte sich darin, dass die `UPDATE`- und `INSERT`-Anfragen lange gesperrt waren, während neue `SELECT`-Statements vor den Aktualisierungen ausgeführt wurden.
- Beim Lesen von `options_files` mit `mysql_options()` wurde die `return-found-rows`-Option ignoriert.
- Man kann jetzt `interactive-timeout` in der Optionsdatei angeben, die von `mysql_options()` gelesen wird. Das ermöglicht es, Programme, die lange laufen (wie `mysqlhotcopy`), zu zwingen, `interactive_timeout` anstelle von `wait_timeout` zu benutzen.
- Zur Langsame-Anfragen-Log-Datei Zeit und Benutzernamen für jede geloggte Anfrage hinzugefügt. Wenn Sie `--log-long-format` benutzen, werden auch Anfragen, die keinen Index benutzen, geloggt, selbst wenn die Anfrage weniger als `long_query_time` Sekunden benötigt.
- Problem in `LEFT JOIN` behoben, was dazu führte, dass alle Spalten in einer Verweistabelle `NULL` waren.
- Problem bei der Benutzung von `NATURAL JOIN` ohne Schlüssel behoben.
- Bug bei der Benutzung eines mehrteiligen Schlüssels behoben, bei dem der erste Teil vom Typ `TEXT` oder `BLOB` war.
- `DROP` von temporären Tabellen wurde nicht in der Update-/Binär-Log-Datei gespeichert.
- Bug behoben, der bei `SELECT DISTINCT * ... LIMIT #` nur eine Zeile zurückgab.
- Bug im Assembler-Code in `strstr()` für sparc behoben und `global.h`-Header-Datei aufgeräumt, um ein Problem mit schlechtem Aliasing des Compilers zu vermeiden, der bei RedHat 7.0 beiliegt (berichtet von Trond Eivind Glomsrød).
- Die Option `--skip-networking` funktioniert jetzt sauber unter Windows NT.
- Lang ausstehender Bug in den `ISAM`-Tabellen behoben, wenn eine Zeile mit einer Länge von mehr als 65 KB um ein einzelnes Byte gekürzt wurde.
- Bug in `MyISAM` beim Laufenlassen mehrfacher Aktualisierungsprozesse auf dieselbe Tabelle behoben.
- Es wird zugelassen, dass `FLUSH TABLE tabelle` benutzt wird.
- `--replicate-ignore-table`, `--replicate-do-table`, `--replicate-wild-ignore-table` und `--replicate-wild-do-table` hinzugefügt.
- Alle Log-Dateien so geändert, dass sie unseren eigenen `IO_CACHE`-Mechanismus anstelle von `FILE` benutzen, um Betriebssystemprobleme zu vermeiden, wenn zu viele Dateien offen sind.
- Optionen `--open-files` und `--timezone` für `safe_mysqld` hinzugefügt.
- Schweren Bug in `CREATE TEMPORARY TABLE ... SELECT ...` behoben.
- Problem mit `CREATE TABLE ... SELECT NULL` behoben.
- Variablen `large_file_support`, `net_read_timeout`, `net_write_timeout` und `query_buffer_size` für `SHOW VARIABLES` hinzugefügt.
- Status-Variablen `created_tmp_files` und `sort_merge_passes` für `SHOW STATUS` hinzugefügt.
- Bug behoben, bei dem kein Index-Name nach der `FOREIGN KEY`-Definition zugelassen wurde.
- `TRUNCATE tabelle` als ein Synonym für `DELETE FROM tabelle` hinzugefügt.

- Bug in einer BDB-Schlüsselvergleichsfunktion beim Vergleich von Schlüsselteilen behoben.
- Variable `bdb_lock_max` für `mysqld` hinzugefügt.
- Weitere Tests zur Benchmark-Suite hinzugefügt.
- Überlauf-Bug im Client-Code bei der Benutzung von überlangen Datenbanknamen behoben.
- `mysql_connect()` bricht jetzt unter Linux ab, wenn der Server nicht in `timeout` Sekunden antwortet.
- `SLAVE START` funktionierte nicht, wenn Sie mit `--skip-slave-start` starteten und vorher nicht explizit `CHANGE MASTER TO` laufen ließen.
- Die Ausgabe von `SHOW MASTER STATUS` in Ordnung gebracht, damit sie konsistent mit `SHOW SLAVE STATUS` ist. (Sie hat jetzt kein Verzeichnis im Log-Namen.)
- `PURGE MASTER LOGS TO` hinzugefügt.
- `SHOW MASTER LOGS` hinzugefügt.
- `--safemalloc-mem-limit`-Option für `mysqld` hinzugefügt, um Speichermangel zu simulieren, wenn mit `--with-debug=full` kompiliert wurde.
- Mehrere Coredumps unter Bedingungen, in denen Arbeitsspeicher fehlt, behoben.
- `SHOW SLAVE STATUS` benutzte einen nicht initialisierten mutex, wenn der Slave noch nicht gestartet wurde.
- Bug in `ELT()` und `MAKE_SET()` behoben, wenn die Anfrage eine temporäre Tabelle benutzte.
- `CHANGE MASTER TO` ohne Angabe von `MASTER_LOG_POS` setzte es auf 0 statt auf 4 und erreichte die magische Zahl im binären Master-Log.
- `ALTER TABLE ... ORDER BY ...`-Syntax hinzugefügt. Das erzeugt die Tabelle mit Zeilen in einer festgelegten Reihenfolge.

D.2.18. Änderungen in Release 3.23.27

- Bug behoben, bei dem das automatische Reparieren von MyISAM-Tabellen manchmal fehlschlug, wenn die Daten-Datei beschädigt war.
- Bug in `SHOW CREATE` bei der Benutzung von `AUTO_INCREMENT`-Spalten behoben.
- BDB-Tabellen so geändert, dass sie die neue Vergleichsfunktion in Berkeley DB 3.2.3 benutzen.
- Sie können jetzt Unix-Sockets bei `mit-pThread` benutzen.
- Neuer latin5- (türkischer) Zeichensatz.
- Kleinere Portabilitätsbehebungen.

D.2.19. Änderungen in Release 3.23.26

- `<>` funktioniert jetzt sauber mit `NULL`.
- Problem mit `SUBSTRING_INDEX()` und `REPLACE()` behoben (Patch von Alexunder Igonitchev).
- `CREATE TEMPORARY TABLE IF NOT EXISTS` gab keinen Fehler, wenn die Tabelle existierte.
- Wenn Sie keinen `PRIMARY KEY` in einer BDB-Tabelle erzeugen, wird ein versteckter `PRIMARY KEY` erzeugt.
- Nur-Lese-Schlüssel-Optimierung to BDB-Tabellen hinzugefügt.
- `LEFT JOIN` bevorzugte in manchen Fällen einen vollen Tabellen-Scan, wenn es keine `WHERE`-Klausel gab.
- Bei der Benutzung von `--log-slow-query` die Wartezeit auf eine Sperre nicht zählen.
- Bug im Sperr-Code unter Windows behoben, der dazu führte, dass der Schlüssel-Cache berichtete, dass die Schlüssel-Datei

beschädigt sei, obwohl sie in Ordnung war.

- Automatische Reparatur von `MyISAM`-Tabellen, wenn Sie `mysqld` mit `--myisam-recover` starten, hinzugefügt.
- Das `TYPE=`-Schlüsselwort wurde von `CHECK` und `REPAIR` entfernt. Es wird zugelassen, dass `CHECK`-Optionen kombiniert werden. (Sie können immer noch `TYPE=` benutzen, aber die Benutzung wird nicht empfohlen.)
- Mutex-Bug im binären Replikations-Log behoben - lange Aktualisierungsanfragen konnten vom Slave nur teilweise gelesen werden, wenn er das zur falschen Zeit machte, was nicht schwerwiegend ist, aber zu einem Performance-verschlechternden erneuten Verbinden führte, sowie zu einer beunruhigenden Nachricht in der Fehler-Log-Datei.
- Das Format der Binär-Log-Datei wurde geändert - hinzugefügt wurden magische Zahl, Serverversion, Binlog-Version, Server-Kennung und Anfragen-Fehlercode für jedes Anfrage-Ereignis.
- Replikations-Thread vom Slave killt jetzt alle darnieder liegenden Threads vom selben Server.
- Lange Replikations-Benutzernamen wurden bislang nicht korrekt gehandhabt.
- `--replicate-rewrite-db`-Option zu `mysqld` hinzugefügt.
- `--skip-slave-start`-Option to `mysqld` hinzugefügt.
- Aktualisierungen, die einen Fehlercode erzeugten (wie `INSERT INTO foo(schluessel) values (1),(1)`) beendeten bislang irrtümlich den Slave-Thread.
- Optimierung von Anfragen, bei denen `DISTINCT` nur auf Spalten aus denselben Tabellen benutzt wird, hinzugefügt.
- Fließkommazahlen ohne Vorzeichen nach dem Exponent (wie 1e1) werden zugelassen.
- `SHOW GRANTS` zeigte nicht immer alle Spaltenberechtigungen.
- `--default-extra-file=#` für alle MySQL-Clients hinzugefügt.
- Spalten, auf die in `INSERT`-Statements verwiesen wird, werden nun sauber initialisiert.
- `UPDATE` funktioniert nicht immer, wenn es mit einem Bereich auf einem Timestamp benutzt wurde, der Teil des Schlüssels war, der benutzt wurde, um Zeilen zu finden.
- Bug in `FULLTEXT`-Index beim Einfügen einer `NULL`-Spalte behoben.
- `mkstemp()` wird jetzt anstelle von `tempnam()` benutzt. Basiert auf einem Patch von John Jones.

D.2.20. Änderungen in Release 3.23.25

- `database` funktioniert als zweites Argument für `mysqlhotcopy`.
- `UMASK` und `UMASK_DIR` können jetzt oktal angegeben werden.
- `RIGHT JOIN`. Hierdurch wird `RIGHT` zu einem reservierten Wort.
- `@@IDENTITY` als ein Synonym für `LAST_INSERT_ID()` hinzugefügt, aus Gründen der Visual-Basic-Kompatibilität.)
- Bug in `myisamchk` und `REPAIR` bei der Benutzung von `FULLTEXT`-Indexen behoben.
- `LOAD DATA INFILE` funktioniert jetzt mit FIFOs (Patch von Toni L. Harbaugh-Blackford).
- `FLUSH LOGS` brach die Replikation ab, wenn Sie einen Log-Namen mit einer expliziten Erweiterung als Wert der `log-bin`-Option angaben.
- Bug in `MyISAM` mit komprimierten mehrteiligen Schlüsseln behoben.
- Absturz bei der Benutzung von `CHECK TABLE` unter Windows behoben.
- Bug, bei dem `FULLTEXT`-Index immer den koi8_ukr-Zeichensatz benutzten, behoben.
- Berechtigungsüberprüfung für `CHECK TABLE` in Ordnung gebracht.
- Der `MyISAM`-Reparatur-/Reindexierungs-Code benutzte nicht die `--tempdir`-Option für seine temporären Dateien.

- `BACKUP TABLE/RESTORE TABLE` hinzugefügt.
- Coredump auf `CHANGE MASTER TO` behoben, wenn der Slave keinen Master hatte, mit dem er startet.
- Falsche `time` in der Prozessliste für `Connect` des Slave-Threads in Ordnung gebracht.
- Der Slave loggt jetzt, wann er sich mit dem Master verbindet.
- Coredump-Bug beim Ausführen von `FLUSH MASTER` behoben, wenn man kein Dateinamens-Argument für `--log-bin` angab.
- Fehlende `ha_berkeley.x`-Dateien zu MySQL unter Windows hinzugefügt.
- Einige mutex-Bugs im Log-Code behoben, die zu Thread-Blockierungen führen konnten, wenn neue Log-Dateien nicht erzeugt werden konnten.
- Sperrzeit und Anzahl von ausgewählten bearbeiteten Zeilen zur Langsame-Anfragen-Log-Datei hinzugefügt.
- `--memlock`-Option für `mysqld`, um `mysqld` im Arbeitsspeicher auf Systemen mit dem `mlockall()`-Aufruf (wie in Solaris) zu sperren, hinzugefügt.
- `HEAP`-Tabellen benutzten Schlüssel nicht korrekt (Bug aus Version 3.23.23).
- Bessere Unterstützung für `MERGE`-Tabellen (Schlüssel, Mapping, Erzeugung, Dokumentation und mehr) hinzugefügt. See Abschnitt 8.2, „MERGE-Tabellen“.
- Bug in `mysqldump` aus Version 3.23 behoben, der dazu führte, dass einige `CHAR`-Spalten nicht in Anführungsstrichen standen.
- `analyze`, `check`, `optimize` und Reparatur-Code zusammengefasst.
- `OPTIMIZE TABLE` wird jetzt auf `REPAIR` mit Statistiken und Sortieren des Index-Baums gemappt. Das heißt, das es momentan nur auf `MyISAM`-Tabellen funktioniert.
- Einen pre-alloced Block zu root_malloc hinzugefügt, um weniger mallocs zu erhalten.
- Viele neue Statistik-Variablen hinzugefügt.
- `ORDER BY`-Bug bei BDB-Tabellen behoben.
- Warnungen entfernt, dass `mysqld` die `.pid`-Datei unter Windows nicht entfernen konnte.
- `--log-isam` zum Loggen von **MyISAM**-Tabellen anstelle von isam-Tabellen abgeändert.
- `CHECK TABLE` funktioniert jetzt auch unter Windows.
- Datei-mutexes hinzugefügt, um `pwrite()` unter Windows sicher zu machen.

D.2.21. Änderungen in Release 3.23.24

- `mysqld`-Variable `created_tmp_disk_tables` hinzugefügt.
- Um das verlässliche Dumpen und Wiederherstellen von Tabellen mit `TIMESTAMP(X)`-Spalten zu ermöglichen, berichtet MySQL jetzt Spalten mit `X` anders als 14 oder 8 als Zeichenketten.
- Sortierreihenfolge für latin1 abgeändert im Vergleich zu MySQL-Version vor 3.23.23. Jede Tabelle mit `CHAR`-Spalten, die Zeichen mit ASCII-Werten größer als 128 enthalten darf, die vor Version 3.23.22 erzeugt oder geändert wurde, muss repariert werden!
- Kleines Speicherleck behoben, das in Version 3.23.22 beim Einfügen einer temporären Tabelle eingeführt wurde.
- Problem mit BDB-Tabellen und Lesen auf einem eindeutigen (nicht primären) Schlüssel behoben.
- Der win1251-Zeichensatz wurde wiederhergestellt (er ist jetzt nur als nicht empfohlen gekennzeichnet).

D.2.22. Änderungen in Release 3.23.23

- Geänderte Sortierreihenfolge für 'deutsch'; Alle Tabellen mit 'deutscher' Sortierreihenfolge müssen mit `REPAIR TABLE` oder `myisamchk` repariert werden, bevor sie benutzt werden können!
- Option `--core-file` für `mysqld` hinzugefügt, um eine Core-Datei unter Linux zu erhalten, wenn `mysqld` durch das SIGSEGV-Signal stirbt.
- MySQL-Client `mysql` startet jetzt vorgabemäßig mit `--no-named-commands` (`-g`). Diese Option kann mit `--enable-named-commands` (`-G`) abgeschaltet werden. Das kann in manchen Fällen Inkompatibilitätsprobleme hervorrufen, zum Beispiel in SQL-Skripten, die benannte Befehle ohne Semikolon benutzen! Langformat-Befehle funktionieren immer noch von der ersten Zeile.
- Problem bei der Benutzung vieler anhängiger `DROP TABLE`-Statements zugleich behoben.
- Der Optimierer verwendete Schlüssel nicht korrekt bei der Benutzung von `LEFT JOIN` auf eine leere Tabelle.
- Kürzerer Hilfetext beim Aufruf von `mysqld` mit falschen Optionen.
- Nicht schwerwiegender `free()`-Bug in `mysqlimport` behoben.
- Bug in der `MyISAM`-Index-Handhabung von `DECIMAL`-/`NUMERIC`-Schlüsseln behoben.
- Bug beim gleichzeitigen Einfügen in `MyISAM`-Tabellen behoben; in manchen Zusammenhängen gab die Benutzung von `MIN(schluessel_teil)` oder `MAX(schluessel_teil)` eine leere Ergebnismenge zurück.
- `mysqlhotcopy` für die Benutzung der neuen `FLUSH TABLES tabellen_liste`-Syntax aktualisiert. Nur Tabellen, die gesichert werden, werden jetzt auf Platte zurückgeschrieben (flush).
- Verhalten von `--enable-thread-safe-client` so geändert, dass sowohl nicht gethreadete (`-lmysqlclient`) als auch gethreadete (`-lmysqlclient_r`) Bibliotheken eingebaut werden. Benutzer, die gegen ein gethreadetes `-lmysqlclient` linkten, müssen jetzt gegen `libmysqlclient_r` linken.
- Atomischer `RENAME`-Befehl hinzugefügt.
- Einträge mit `NULL` werden in `COUNT(DISTINCT ...)` nicht gezählt.
- `ALTER TABLE`, `LOAD DATA INFILE` auf leere Tabellen und `INSERT ... SELECT ...` auf leere Tabellen so geändert, dass nicht eindeutige Indexe in einem separaten Stapellauf mit Sortieren erzeugt werden. Das macht die genannten Aufrufe viel schneller, wenn Sie viele Indexe haben.
- `ALTER TABLE` loggt jetzt die zuerst benutzte insert_id korrekt.
- Absturz beim Hinzufügen eines Vorgabewerts zu einer `BLOB`-Spalte behoben.
- Bug bei `DATE_ADD/DATE_SUB` behoben, der eine DATETIME anstelle eines DATE zurückgab.
- Problem mit dem Thread-Cache behoben, der dazu führte, dass einige Threads als `***DEAD***` in `SHOW PROCESSLIST` erschienen.
- Eine Sperre in unserem thr_rwlock-Code beseitigt, die dazu führen konnte, dass SELECTs, die zur selben Zeit laufen wie gleichzeitige Einfügevorgänge, abstürzen. Das betrifft nur Systeme, die nicht den `pthread_rwlock_rdlock`-Code haben.
- Beim Löschen von Zeilen mit einem nicht eindeutigen Schlüssel in einer HEAP-Tabelle wurden nicht immer alle Zeilen gelöscht.
- Bug im Bereichsoptimierer für HEAP-Tabellen bei Suchen auf einem Teil-Index behoben.
- `SELECT` auf Teilschlüsseln funktioniert jetzt bei BDB-Tabellen.
- `INSERT INTO bdb_tabelle ... SELECT` funktioniert jetzt bei BDB-Tabellen.
- `CHECK TABLE` aktualisiert jetzt Schlüsselstatistiken für die Tabelle.
- `ANALYZE TABLE` aktualisiert jetzt nur Tabellen, die seit dem letzten `ANALYZE` geändert wurden. Beachten Sie, dass das ein neues Feature ist, und dass Tabellen nicht als analysiert gekennzeichnet werden, bis sie auf irgend eine Weise mit Version 3.23.23 oder neuer aktualisiert wurden. Bei älteren Tabellen müssen Sie `CHECK TABLE` ausführen, um die Schlüsselverteilung zu aktualisieren.
- Einige kleinere Berechtigungsprobleme bei `CHECK`, `ANALYZE`, `REPAIR` und `SHOW CREATE` behoben.
- `CHANGE MASTER TO`-Befehl hinzugefügt.

- `FAST`-, `QUICK`- `EXTENDED`-Überprüfungsarten zu `CHECK TABLES` hinzugefügt.
- `myisamchk` abgeändert, so dass `--fast` und `--check-changed-tables` auch bei `--sort-index` und `--analyze` berücksichtigt werden.
- Schwerwiegenden Bug in `LOAD TABLE FROM MASTER` behoben, bei dem die Tabelle während des Neuaufbaus des Indexes nicht gesperrt wurde.
- `LOAD DATA INFILE` brach die Replikation ab, wenn die Datenbank aus der Replikation ausgeschlossen war.
- Mehr Variablen zu `SHOW SLAVE STATUS` und `SHOW MASTER STATUS` hinzugefügt.
- `SLAVE STOP` gibt jetzt solange nichts zurück, bis der Thread tatsächlich beendet ist.
- Volltextsuche mit der `MATCH`-Funktion und `FULLTEXT`-Indextyp hinzugefügt (für MyISAM-Dateien). Das macht `FULLTEXT` zu einem reservierten Wort.

D.2.23. Änderungen in Release 3.23.22

- `lex_hash.h` wird jetzt für jede MySQL-Distribution korrekt erzeugt.
- `MASTER` und `COLLECTION` sind jetzt reservierte Wörter.
- Die Log-Datei, die von `--slow-query-log` erzeugt wird, enthielt nicht die gesamten Anfragen.
- Offene Transaktionen in BDB-Tabellen werden jetzt nicht mehr zurückgerollt, wenn die Verbindung unerwartet geschlossen wird.
- Workaround für einen Bug in `gcc` 2.96 (intel) und `gcc` 2.9 (Ia64) in `gen_lex_hash.c` hinzugefügt.
- Speicherleck in der Client-Bibliothek bei der Benutzung von `host=` in der `my.cnf`-Datei behoben.
- Funktionen optimiert, die Stunden/Minuten/Sekunden bearbeiten.
- Bug beim Vergleich des Ergebnisses von `DATE_ADD()`/`DATE_SUB()` mit einer Zahl behoben.
- Bedeutung von `-F, --fast` für `myisamchk` geändert. Option `-C, --check-only-changed` für `myisamchk` hinzugefügt.
- `ANALYZE tabelle` zum Aktualisieren von Schlüsselstatistiken für Tabellen hinzugefügt.
- Binäreinheiten `0x...` abgeändert, so dass sie vorgabemäßig als Ganzzahlen betrachtet werden.
- Fehlerbehebung für SCO und `SHOW PROCESSLIST`.
- `auto-rehash` beim erneuten Verbinden für den `mysql`-Client hinzugefügt.
- Neu eingeführten Bug in `MyISAM` behoben, bei dem die Index-Datei nicht größer als 64 MB werden durfte.
- `SHOW MASTER STATUS` und `SHOW SLAVE STATUS` hinzugefügt.

D.2.24. Änderungen in Release 3.23.21

- `mysql_character_set_name(MYSQL *mysql)`-Funktion zur MySQL-C-API hinzugefügt.
- Update-Log-Datei `ASCII 0`-sicher gemacht.
- `mysql_config`-Skript hinzugefügt.
- Problem bei der Benutzung von `<` oder `>` mit einer CHAR-Spalte, die nur teilweise indexiert war, behoben.
- Man erhielt einen Coredump, wenn die Log-Datei nicht vom MySQL-Benutzer lesbar war.
- `mysqladmin` so geändert, dass es die `CREATE DATABASE`/`DROP DATABASE`-Befehle anstelle der alten, nicht empfohlenen API-Aufrufe benutzt.
- `chown`-Warnung in `safe_mysqld` in Ordnung gebracht.

- Bug in `ORDER BY` behoben, der in Version 3.23.19 eingeführt wurde.
- `DELETE FROM tabelle` wird nur dann optimiert, ein Löschen und Neuerzeugen der Tabelle auszuführen, wenn man sich im `AUTOCOMMIT`-Modus befindet (benötigt für BDB-Tabellen).
- Zusätzliche Prüfungen hinzugefügt, um Index-Beschädigung zu vermeiden, wenn die `ISAM`/`MyISAM`-Index-Dateien während eines `INSERT`/`UPDATE` voll werden.
- `myisamchk` aktualisierte die Zeilenprüfsumme nicht korrekt, wenn es mit `-ro` benutzt wurde (sondern gab nur bei nachfolgenden Läufen eine Warnung aus).
- Bug in `REPAIR TABLE` behoben, so dass es bei Tabellen ohne Indexe funktioniert.
- Puffer-Überlauf in `DROP DATABASE` behoben.
- `LOAD TABLE FROM MASTER` ist ausreichend ohne Bugs, um es als Feature vorstellen zu können.
- `MATCH` und `AGAINST` sind jetzt reservierte Wörter.

D.2.25. Änderungen in Release 3.23.20

- Bug in Version 3.23.19 behoben; `DELETE FROM tabelle` entfernte die .frm-Datei.
- `SHOW CREATE TABLE`.

D.2.26. Änderungen in Release 3.23.19

- Copyright für alle Dateien zu GPL für den Server-Code und die Dienstprogramme und LGPL für die Client-Bibliotheken geändert.
- Bug behoben, bei dem nicht alle übereinstimmenden Zeilen bei einer `MyISAM`-Tabelle aktualisiert wurden, wenn man eine Aktualisierung basierend auf einem Schlüssel auf eine Tabelle mit vielen Schlüsseln durchführte, und sich einige Schlüsselwerte änderten.
- Die Linux-MySQL-RPMs und -Binärdateien werden jetzt bei einer Linux-Thread-Version statisch gelinkt, die schnellere mutex-Handhabung bei der Benutzung mit MySQL hat.
- `ORDER BY` kann jetzt `REF`-Schlüssel benutzen, um eine Untermenge von Zeilen zu finden, die sortiert werden müssen.
- Der Name von `print_defaults` wurde in `my_print_defaults` geändert, um Namenskonflikte zu vermeiden.
- `NULLIF()` funktioniert jetzt gemäß ANSI-SQL99.
- `net_read_timeout` und `net_write_timeout` als Startparameter für `mysqld` hinzugefügt.
- Bug behoben, der den Index bei der Ausführung von `myisamchk --sort-records` auf eine Tabelle mit Präfix-komprimiertem Index zerstörte.
- pack_isam und myisampack zur Standard-MySQL-Distribution hinzugefügt.
- Die Syntax `BEGIN WORK` hinzugefügt (dasselbe wie `BEGIN`).
- Coredump-Bug bei der Benutzung von `ORDER BY` auf `CONV()`-Ausdruck behoben.
- `LOAD TABLE FROM MASTER` hinzugefügt.
- `FLUSH MASTER` und `FLUSH SLAVE` hinzugefügt.
- Großes/kleines 'endian'-Problem in der Replikation behoben.

D.2.27. Änderungen in Release 3.23.18

- Problem aus Version 3.23.17 bei der Auswahl eines Zeichensatzes auf der Client-Seite behoben.

- `FLUSH TABLES with READ LOCK` geändert, so dass es eine globale Sperre macht, die für das Herstellen einer Kopie der MySQL-Daten-Dateien geeignet ist.
- `CREATE TABLE ... SELECT ... PROCEDURE` funktioniert jetzt.
- Interne temporäre Tabellen benutzen jetzt einen komprimierten Index bei der Benutzung von `GROUP BY` auf `VARCHAR/CHAR`-Spalten.
- Problem behoben beim Sperren derselbe Tabelle mit einer `READ`- und einer `WRITE`-Sperre.
- Problem mit myisamchk und `RAID`-Tabellen behoben.

D.2.28. Änderungen in Release 3.23.17

- Bug in `find_in_set()` behoben, wenn das erste Argument `NULL` war.
- Tabellensperren für Berkeley-DB hinzugefügt.
- Bug bei `LEFT JOIN` und `ORDER BY` behoben, bei dem die erste Tabelle nur eine übereinstimmende Zeile hatte.
- 4 `my.cnf`-Beispiel-Dateien im `Support-files`-Verzeichnis hinzugefügt.
- `duplicated key`-Problem bei der Ausführung großer `GROUP BY`s behoben. (Dieser Bug wurde wahrscheinlich in Version 3.23.15 eingeführt).
- Syntax für `INNER JOIN` geändert, um ANSI-SQL zu entsprechen.
- `NATURAL JOIN`-Syntax hinzugefügt.
- Viele Korrekturen in der `BDB`-Schnittstelle.
- Handhabung von `--no-defaults` und `--defaults-file` für `safe_mysqld.sh` und `mysql_install_db.sh` hinzugefügt.
- Bug beim Lesen komprimierter Tabellen mit vielen Threads behoben.
- `USE INDEX` funktioniert jetzt mit `PRIMARY`-Schlüsseln.
- `BEGIN`-Statement geändert, so dass es eine Transaktion im `AUTOCOMMIT`-Modus startet.
- Symbolische-Links-Unterstützung für Windows.
- Protokoll geändert, so dass der Client weiß, ob der Server im AUTOCOMMIT- Modus ist und ob es eine anhängige Transaktion gibt. Wenn das der Fall ist, gibt die Client-Bibliothek einen Fehler aus, bevor sie sich wieder mit dem Server verbindet, damit der Client weiß, dass der Server ein Rollback durchgeführt hat. Das Protokoll ist noch abwärtskompatibel mit den alten Clients.
- `KILL` funktioniert jetzt auf einem Thread, der durch ein 'Schreiben' auf einen toten Client gesperrt ist.
- Speicherleck im Replikations-Slave-Thread behoben.
- Neue Option `log-slave-updates` hinzugefügt, die das Hintereinanderhängen im Kreis (Daisy-Chaining, 'Ringelrei') von Slaves erlaubt.
- Compile-Fehler auf FreeBSD und anderen Systemen behoben, auf denen `pthread_t` nicht dasselbe wie `int` ist.
- Herunterfahren des Masters bricht den Slave-Thread nicht mehr ab.
- Race-Bedingung im `INSERT DELAYED`-Code beim Ausführen von `ALTER TABLE` behoben.
- Blockierungsüberprüfung für `INSERT DELAYED` hinzugefügt.

D.2.29. Änderungen in Release 3.23.16

- Option `TYPE=QUICK` für `CHECK` und `REPAIR` hinzugefügt.
- Bug in `REPAIR TABLE` behoben, wenn die Tabelle durch einen anderen Thread in Benutzung war.

- Einen Thread-Cache hinzugefügt, um zu ermöglichen, MySQL mit `gdb` zu debuggen, wenn man viele erneute Verbindungen durchführt. Das verbessert auch Systeme, auf denen man keine persistenten Verbindungen benutzen kann.
- Viele Korrekturen in der Berkeley-DB-Schnittstelle.
- `UPDATE IGNORE` bricht jetzt nicht mehr ab, wenn eine Aktualisierung zu einem `DUPLICATE_KEY`-Fehler führt.
- `CREATE TEMPORARY TABLE`-Befehle werden in die Update-Log-Datei geschrieben.
- Bug bei der Handhabung von maskierten IP-Nummern in den Berechtigungstabellen behoben.
- Bug mit `delayed_key_writes`-Tabellen und `CHECK TABLE` behoben.
- `replicate-do-db` und `replicate-ignore-db`-Optionen hinzugefügt, um auf Datenbanken zu beschränken, die repliziert werden.
- `SQL_LOG_BIN`-Option hinzugefügt.

D.2.30. Änderungen in Release 3.23.15

- Um `mysqld` als `root` zu starten, müssen Sie jetzt die `-- user=root`-Option benutzen.
- Schnittstelle zu Berkeley-DB hinzugefügt. (Diese funktioniert noch nicht richtig. Spielen Sie mit ihr auf eigenes Risiko herum!)
- Replikation zwischen Master und Slaves hinzugefügt.
- Bug behoben, bei dem ein anderer Thread eine Sperre stehlen konnte, wenn ein Thread eine Sperre auf eine Tabelle hatte und einen `FLUSH TABLES`-Befehl ausführte.
- Die `slow_launch_time`-Variable und die `slow_launch_thread`-Status-Variable zu `mysqld` hinzugefügt. Diese können mit `mysqladmin variables` und `mysqladmin extended-status` betrachtet werden.
- Funktionen `INET_NTOA()` und `INET_ATON()` hinzugefügt.
- Der vorgabemäßige Typ von `IF()` hängt jetzt vom zweiten und dritten Argument ab und nicht nur vom zweiten.
- Fall behoben, bei dem `myisamchk` beim Versuch, eine Tabelle zu reparieren, in eine Schleife geraten konnte.
- `INSERT DELAYED` nicht in die Update-Log-Datei schreiben, wenn `SQL_LOG_UPDATE=0`.
- Problem mit `REPLACE` auf `HEAP`-Tabellen behoben.
- Mögliche Zeichensätze und Zeitzone zu `SHOW VARIABLES` hinzugefügt.
- Bug im Sperr-Code behoben, der zu Sperrproblemen bei gleichzeitigen Einfügevorgängen unter hoher Last führen konnte.
- Problem bei `DELETE` vieler Zeilen auf eine Tabelle mit komprimierten Schlüsseln behoben, bei dem MySQL den Index scannte, um Zeilen zu finden.
- Problem mit `CHECK` auf Tabelle mit gelöschten Schlüsselblöcken behoben.
- Bug beim Neuverbinden (auf der Client-Seite) behoben, bei dem in manchen Situationen Speicher nicht freigegeben wurde.
- Probleme in der Update-Log-Datei bei der Benutzung von `LAST_INSERT_ID()` zum Aktualisieren einer Tabelle mit einem auto_increment-Schlüssel behoben.
- Funktion `NULLIF()` hinzugefügt.
- Bug bei der Benutzung von `LOAD DATA INFILE` auf eine Tabelle mit `BLOB/TEXT`-Spalten behoben.
- MyISAM optimiert, um es beim Einfügen von Schlüsseln in sortierter Reihenfolge schneller zu machen.
- `EXPLAIN SELECT ...` gibt jetzt auch aus, ob MySQL eine temporäre Tabelle oder Dateisortieren verwendet, wenn das `SELECT` aufgelöst wird.
- Optimierung hinzugefügt, um `ORDER BY`-Teile zu überspringen, bei denen der Teil ein konstanter Ausdruck im `WHERE`-Teil ist. Indexe können jetzt benutzt werden, selbst wenn das `ORDER BY` nicht genau mit dem Index übereinstimmt, solange alle nicht benutzten Index-Teile und alle zusätzlichen `ORDER BY`-Spalten Konstanten in der `WHERE`-Klausel sind. See Abschnitt 6.4.3, „Wie MySQL Indexe benutzt“.

- `UPDATE` und `DELETE` auf einen gesamten eindeutigen Schlüssel im `WHERE`-Teil ist jetzt schneller als vorher.
- `RAID_CHUNKSIZE` so geändert, dass es in 1024 Bytes inkrementiert.
- Coredump in LOAD_FILE(NULL) behoben.

D.2.31. Änderungen in Release 3.23.14

- Bug in `CONCAT()` behoben, bei dem eins der Argumente eine Funktion war, die ein verändertes Argument zurückgab.
- Kritischen Bug in `myisamchk` behoben, wobei es den Header in der Index-Datei aktualisierte, wenn man die Tabelle nur prüfte. Das brachte den `mysqld`-Daemon durcheinander, wenn er dieselbe Tabelle zur gleichen Zeit aktualisierte. Jetzt wird der Status in der Index-Datei nur dann aktualisiert, wenn man `--update-state` benutzt. Bei älteren `myisamchk`-Versionen sollten Sie `--read-only` benutzen, wenn Sie Tabellen nur prüfen, wenn es auch nur die geringste Chance gibt, dass der `mysqld`-Server zur gleichen Zeit auf der Tabelle arbeitet!
- `DROP TABLE` wird nicht mehr in der Update-Log-Datei geloggt.
- Problem beim Suchen auf `DECIMAL()`-Schlüsselfeld behoben, wenn die Spalte Daten mit führenden Nullen enthielt.
- Bug in `myisamchk` behoben, wenn auto_increment nicht der erste Schlüssel ist.
- `DATETIME` wird im ISO-8601-Format zugelassen: 2000-03-12T12:00:00
- Dynamische Zeichensätze hinzugefügt. Eine `mysqld`-Binärdatei kann jetzt viele unterschiedliche Zeichensätze handhaben (welche, können Sie beim Start von `mysqld` angeben).
- Befehl `REPAIR TABLE` hinzugefügt.
- C-API-Funktion `mysql_thread_safe()` hinzugefügt.
- `UMASK_DIR`-Umgebungsvariable hinzugefügt.
- Funktion `CONNECTION_ID()` hinzugefügt.
- Bei der Benutzung von `=` auf `BLOB`- oder `VARCHAR BINARY`-Schlüsseln, bei denen nur ein Teil der Spalte indexiert war, wurde nicht die gesamte Spalte der Ergebniszeile verglichen.
- Problembehebung für sjis-Zeichensatz und `ORDER BY`.
- Beim Laufenlassen im ANSI-Modus wird nicht mehr zugelassen, dass Spalten benutzt werden, die nicht im `GROUP BY`-Teil angegeben wurden.

D.2.32. Änderungen in Release 3.23.13

- Problem behoben bei der Ausführung von Sperren auf dieselbe Tabelle mehr als zweimal im selben `LOCK TABLE`-Befehl. Dadurch wurde das Problem behoben, das man bekam, wenn man test-ATIS test mit `--fast` oder `--check-only-changed` laufen ließ.
- Option `SQL_Puffer_RESULT` für `SELECT` hinzugefügt.
- Leerzeichen am Ende von Double-/Float-Zahlen in Ergebnissen aus temporären Tabellen entfernt. `CHECK TABLE`-Befehl hinzugefügt.
- Änderungen für MyISAM in Version 3.23.12 hinzugefügt, die wegen CVS-Problemen nicht in die Quelldistribution gelangten.
- Bug behoben, so dass `mysqladmin shutdown` darauf wartet, dass der lokale Server herunter fährt.
- Mögliche Endlosschleife bei der Zeitstempel-Berechnung repariert.
- `print_defaults` für die `.rpm`-Dateien hinzugefügt. `mysqlbug` aus der Client-`.rpm`-Datei entfernt.

D.2.33. Änderungen in Release 3.23.12

- Bug in `MyISAM` behoben, bei dem `REPLACE ... SELECT ...` eine beschädigte Tabelle ergeben konnte.
- Bug in `myisamchk` behoben, bei dem der auto_increment-Wert falsch zurückgesetzt wurde.
- VIELE Patches für Linux Alpha. MySQL scheint mittlerweile auf Linux Alpha relativ stabil zu laufen.
- `DISTINCT` auf `HEAP` temporäre Tabellen so geändert, dass gehashte Schlüssel verwendet werden, um doppelte Zeilen (Duplikate) schnell zu finden. Das betrifft meistens Anfragen des Typs `SELECT DISTINCT ... GROUP BY ...`. Das behebt ein Problem, bei dem nicht alle Duplikate in Anfragen des genannten Typs entfernt wurden. Zusätzlich ist der neue Code VIEL schneller.
- Patches hinzugefügt, damit MySQL auf Mac OS X kompiliert.
- Option `IF NOT EXISTS` für `CREATE DATABASE` hinzugefügt.
- Optionen `--all-databases` und `--databases` für `mysqldump` hinzugefügt, um das Dumpen vieler Datenbanken zugleich zu ermöglichen.
- Bug im komprimierten `DECIMAL()`-Index in `MyISAM`-Tabellen behoben.
- Bug beim Speichern von 0 in ein Timestamp-Feld behoben.
- Beim Ausführen von `mysqladmin shutdown` auf eine lokale Verbindung wartet `mysqladmin` jetzt, bis die PID-Datei entfernt ist, bevor es sich beendet.
- Coredump bei einigen `COUNT(DISTINCT ...)`-Anfragen behoben.
- `myisamchk` funktioniert jetzt sauber bei RAID-Tabellen.
- Problem bei `LEFT JOIN` und `schluessel_feld IS NULL` behoben.
- Bug in `net_clear()` behoben, der den Fehler `Aborted connection` in MySQL-Clients ausgeben konnte.
- Optionen `USE INDEX (schluessel_liste)` und `IGNORE INDEX (schluessel_liste)` als Join-Parameter in `SELECT` hinzugefügt.
- `DELETE` und `RENAME` sollten jetzt auf `RAID`-Tabellen funktionieren.

D.2.34. Änderungen in Release 3.23.11

- `ALTER TABLE tabelle ADD (feld_liste)`-Syntax wird zugelassen.
- Problem mit dem Optimierer behoben, der manchmal falsche Schlüssel benutzte.
- `GRANT/REVOKE ALL PRIVILEGES` betrifft jetzt nicht mehr `GRANT OPTION`.
- Zusätzliche Klammer (`()`) aus der Ausgabe von `SHOW GRANTS` entfernt.
- Problem beim Speichern von Zahlen in Timestamps behoben.
- Problem mit Zeitzonen behoben, die einen Halbstunden-Offset haben.
- Syntax `UNIQUE INDEX` in `CREATE`-Statements wird jetzt zugelassen.
- `mysqlhotcopy` hinzugefügt. Das ist ein schnelles Online-Datensicherungsdienstprogramm für lokale MySQL-Datenbanken. Von Tim Bunce.
- Neues, sichereres `mysqlaccess` hinzugefügt. Dank an Steve Harvey hierfür.
- Optionen `--i-am-a-dummy` und `--safe-updates` für `mysql` hinzugefügt.
- Variablen `select_limit` und `max_join_size` für `mysql` hinzugefügt.
- SQL-Variablen `SQL_MAX_JOIN_SIZE` und `SQL_SAFE_UPDATES` hinzugefügt.
- `READ LOCAL`-Sperre hinzugefügt, die die Tabelle nicht für gleichzeitige Einfügevorgänge sperrt (das wird von `mysqldump` benutzt).
- `LOCK TABLES ... READ` läßt keine gleichzeitigen Einfügevorgänge mehr zu.

- Option `--skip-delay-key-write` für `mysqld` hinzugefügt.
- Sicherheitsproblem im Protokoll betreffend Passwortüberprüfung behoben.
- `_rowid` kann jetzt als Alias für eine eindeutig indexierte Spalte vom Typ Ganzzahl benutzt werden.
- Zurück-Blockieren (Back Blocking) für `SIGPIPE` beim Kompilieren mit `--thread-safe-clients` hinzugefügt, um Dinge für alte Clients sicher zu machen.

D.2.35. Änderungen in Release 3.23.10

- Bug in Version 3.23.9 behoben, bei dem Speicher nicht korrekt freigegeben wurde, wenn man `LOCK TABLES` ausführte.

D.2.36. Änderungen in Release 3.23.9

- Problem behoben, dass betroffene Anfragen Berechnungen auf Gruppenfunktionen durchführten.
- Problem mit timestamps und `INSERT DELAYED` behoben.
- `datum_spalte BETWEEN konstanten_datum AND konstanten_datum` funktioniert.
- Problem behoben, wenn man nur eine 0 zu `NULL` in einer Tabelle mit `BLOB/TEXT`-Spalten änderte.
- Bug im Bereichsoptimierer bei der Benutzung von vielen Schlüsselteilen und / oder den mittleren Schlüsselteilen behoben: `WHERE K1=1 and K3=2 and (K2=2 and K4=4 or K2=3 and K4=5)`
- Befehl `source` für `mysql` hinzugefügt, um Lesen von Stapeldateien innerhalb des `mysql`-Clients zu ermöglichen. Original-Patch von Matthew Vanecek.
- Kritisches Problem mit der `WITH GRANT OPTION`-Option behoben.
- Keinen unnötigen `GRANT`-Fehler bei der Benutzung von Tabellen von vielen Datenbanken in derselben Anfrage ausgeben.
- VIO-Wrapper (benötigt für SSL-Unterstützung) hinzugefügt. Von Andrei Errapart und Tõnu Samuel).
- Optimiererproblem bei `SELECT` bei der Benutzung von vielen überlappenden Indexen behoben. MySQL sollte jetzt in der Lage sein, Schlüssel noch besser auszusuchen, wenn es viele Schlüssel zur Auswahl gibt.
- Optimierer so geändert, dass er einen Bereichsschlüssel anstelle eines Verweisschlüssels bevorzugt, wenn der Bereichsschlüssel mehr Spalten als der Verweisschlüssel benutzen kann (der nur Spalten mit = verwenden kann). Folgender Anfragentyp beispielsweise sollte jetzt schneller sein: `SELECT * from schluessel_teil_1=konstante und schluessel_teil_2 > konstante2`
- Bug behoben, bei dem eine Änderung aller `VARCHAR`-Spalten in `CHAR`-Spalten den Spaltentyp nicht von dynamisch auf fest änderte.
- Fließkomma-Ausnahmefehler für FreeBSD abgeschaltet, um Coredump beim Ausführen von `SELECT floor(pow(2,63))` zu vermeiden.
- `mysqld`-Startoption `--delay-key-write` in `--delay-key-write-for-all-tables` geändert.
- `read-next-on-key` für `HEAP`-Tabellen hinzugefügt. Das sollte alle Probleme mit `HEAP`-Tabellen bei der Benutzung von Nicht-`UNIQUE`-Schlüsseln beheben.
- Optionen für die Ausgabe vorgabemäßiger Argumente für alle Clients hinzugefügt.
- `--log-slow-queries` für `mysqld` hinzugefügt, um alle Anfragen in einer separate Log-Datei zu loggen, die lange dauerten, mit einer Zeitangabe, wie lange die Anfrage benötigte.
- Coredump bei der Ausführung von `WHERE schluessel_spalte=RAND(...)` behoben.
- Optimierungs-Bug in `SELECT ... LEFT JOIN ... schluessel_spalte IS NULL` behoben, wenn `schluessel_spalte` `NULL`-Werte enthalten konnte.
- Problem mit 8-Bit-Zeichen als Trennzeichen in `LOAD DATA INFILE` behoben.

D.2.37. Änderungen in Release 3.23.8

- Problem bei der Handhabung von Index-Dateien größer als 8 GB behoben.
- neueste Patches für mit-pThread für NetBSD angewandt.
- Probleme mit Zeitzonen < GMT - 11 behoben.
- Bug beim Löschen komprimierter Schlüssel in `MyISAM` behoben.
- Problem mit `ISAM` bei der Ausführung einiger `ORDER BY ... DESC`-Anfragen behoben.
- Bug bei der Ausführung eines Joins auf einen Text-Schlüssel behoben, der nicht den gesamten Schlüssel abdeckte.
- Option `--delay-key-write` schaltete verzögertes Schlüssel-Schreiben nicht an.
- Aktualisierung von `TEXT`-Spalten, die nur Änderungen der Groß-/Kleinschreibung beinhalteten, in Ordnung gebracht.
- `INSERT DELAYED` aktualisiert jetzt Timestamps, die angegeben sind.
- Funktion `YEARWEEK()` und Optionen `x`, `X`, `v` und `V` für `DATE_FORMAT()` hinzugefügt.
- Problem mit `MAX(indexierte_spalte)` und HEAP-Tabellen behoben.
- Problem mit `BLOB NULL`-Schlüsseln und `LIKE` "praefix%" behoben.
- Problem mit `MyISAM` und Zeilen fester Länge < 5 Bytes behoben.
- Problem behoben, bei dem es vorkommen konnte, dass MySQL auf freigegebenen Speicher zugriff, wenn er sehr komplizierte `GROUP BY`-Anfragen ausführte.
- Coredump behoben, wenn man eine beschädigte Tabelle erhielt, in der ein `ENUM`-Feldwert zu Groß war.

D.2.38. Änderungen in Release 3.23.7

- Workaround unter Linux in Ordnung gebracht, um Probleme mit `pthread_mutex_timedwait`, was bei `INSERT DELAYED` benutzt wird, zu vermeiden. See Abschnitt 3.6.1, „Linux (alle Linux-Versionen)“.
- Man erhält jetzt einen 'disk full'-Fehler, wenn die Festplatten beim Sortieren voll wird (statt darauf zu warten, bis mehr Plattenplatz verfügbar ist).
- Bug in `MyISAM` mit Schlüsseln > 250 Zeichen behoben.
- In `MyISAM` kann man jetzt ein `INSERT` zur selben Zeit durchführen, in der andere Threads aus der Tabelle lesen.
- Variable `max_write_lock_count` für `mysqld` hinzugefügt, um eine `READ`-Sperre nach einer bestimmten Anzahl von `WRITE`-Sperren zu erzwingen.
- Flag `delayed_key_write` bei `show variables` invertiert.
- Variable `concurrency` in `thread_concurrency` umbenannt.
- Folgende Funktionen sind jetzt Multi-Byte-sicher: `LOCATE(teilzeichenfolge,zeichenkette)`, `POSITION(teilzeichenfolge IN zeichenkette)`, `LOCATE(teilzeichenfolge,zeichenkette,position)`, `INSTR(zeichenkette,teilzeichenfolge)`, `LEFT(zeichenkette,laenge)`, `RIGHT(zeichenkette,laenge)`, `SUBSTRING(zeichenkette,pos,laenge)`, `SUBSTRING(zeichenkette FROM position FOR laenge)`, `MID(zeichenkette,position,laenge)`, `SUBSTRING(zeichenkette,position)`, `SUBSTRING(zeichenkette FROM pos)`, `SUBSTRING_INDEX(zeichenkette,begrenzer,zaehler)`, `RTRIM(zeichenkette)`, `TRIM([[BOTH | TRAILING] [entfernzeichenkette] FROM] zeichenkette)`, `REPLACE(zeichenkette,from_zeichenkette,to_zeichenkette)`, `REVERSE(zeichenkette)`, `INSERT(zeichenkette,pos,laenge,newstr)`, `LCASE(zeichenkette)`, `LOWER(zeichenkette)`, `UCASE(zeichenkette)` und `UPPER(zeichenkette)`. Patch von Wei He.
- Coredump beim Aufheben einer Sperre von einer nicht existierenden Tabelle behoben.
- Sperren auf Tabellen werden jetzt entfernt, bevor Duplikate entfernt werden.

- Option `FULL` für `SHOW PROCESSLIST` hinzugefügt.
- Option `--verbose` für `mysqladmin` hinzugefügt.
- Problem beim automatischen Umwandeln von HEAP in MyISAM behoben.
- Bug in HEAP-Tabellen behoben, wenn man INSERT + DELETE + INSERT + Scannen der Tabelle ausführt.
- Bugs auf Alpha mit `REPLACE()` und `LOAD DATA INFILE` behoben.
- `mysqld`-Variable `interactive_timeout` hinzugefügt.
- Argument für `mysql_data_seek()` von `ulong` zu `ulonglong` geändert.

D.2.39. Änderungen in Release 3.23.6

- `mysqld`-Option `-O lower_case_tables={0|1}` hinzugefügt, damit Benutzer Tabellennamen to Kleinschreibung erzwingen können.
- `SELECT ... INTO DUMPFILE` hinzugefügt.
- `mysqld`-Option `--ansi` hinzugefügt, um einige Funktionen `ANSI-SQL`-kompatibler zu machen.
- Temporäre Tabellen fangen jetzt mit `#sql` an.
- Quoten von Bezeichnern mit ` (" im `--ansi`-Modus).
- Jetzt wird snprintf() bei der Ausgabe von Fließkommazahlen benutzt, um einige Puffer-Überläufe unter FreeBSD zu vermeiden.
- `[floor()` überlaufsicher unter FreeBSD gemacht.
- Option `--quote-names` für `mysqldump` hinzugefügt.
- Bug behoben, dass man einen Teil eines `PRIMARY KEY NOT NULL` machen konnte.
- `encrypt()` in Ordnung gebracht, um Thread-sicher zu sein und Puffer nicht erneut zu benutzen.
- `mysql_odbc_escape_string()`-Funktion zur Unterstützung von big5-Zeichen in MyODBC hinzugefügt.
- Die Tabellen-Handler wurden umgeschrieben und benutzen jetzt Klassen. Hierdurch wird viel neuer Code eingeführt, aber die Tabellenhandhabung wird schneller und besser.
- Patch von Sasha für benutzerdefinierte Variablen angewandt.
- `FLOAT` und `DOUBLE` (ohne jeden Längen-Modifikator) sind jetzt keine festen Dezimalpunkt-Zahlen mehr.
- Die Bedeutung von `FLOAT(X)` wurde geändert: Jetzt ist das dasselbe wie `FLOAT`, wenn X <= 24, und `DOUBLE`, wenn 24 < X <= 53.
- `DECIMAL(X)` ist jetzt ein Alias für `DECIMAL(X,0)`, und `DECIMAL` ist jetzt ein Alias für `DECIMAL(10,0)`. Dasselbe gilt für `NUMERIC`.
- Option `ROW_FORMAT={default | dynamic | static | compressed}` für `CREATE_TABLE` hinzugefügt.
- `DELETE FROM tabelle` funktionierte nicht auf temporären Tabellen.
- Funktion `CHAR_LENGTH()` geändert, so dass sie Multi-Byte-Zeichen-sicher ist.
- Funktion `ORD(zeichenkette)` hinzugefügt.

D.2.40. Änderungen in Release 3.23.5

- Einige Jahr-2000-Probleme in der neuen Daten-Handhabung in Version 3.23 behoben.
- Problem mit `SELECT DISTINCT ... ORDER BY RAND()` behoben.
- Patches von Sergei A. Golubchik für Textsuche auf MyISAM-Ebene angewandt.

- Cache-Überlaufproblem bei der Benutzung von Full Joins ohne Schlüssel behoben.
- Einige configure-Probleme bereinigt.
- Einige kleine Änderungen, um das Parsen schneller zu machen.
- `ALTER TABLE` + Hinzufügen einer Spalte nach dem letzten Feld funktionierte nicht.
- Problem bei der Benutzung einer auto_increment-Spalte in zwei Schlüsseln behoben.
- Bei MyISAM kann man jetzt den auto_increment-Teil als Untermenge haben: `CREATE TABLE foo (a int not null auto_increment, b char(5), primary key (b,a))`
- Bug in MyISAM mit komprimierten CHAR-Schlüsseln, die `NULL` sein konnten, behoben.
- `AS` auf Feldname mit `CREATE TABLE tabelle SELECT ...` funktionierte nicht.
- Benutzung von `NATIONAL` und `NCHAR` bei der Definition von Zeichenspalten wird zugelassen. Das ist dasselbe, als wenn man `BINARY` nicht benutzt.
- Keine `NULL`-Spalten in einem `PRIMARY KEY` zulassen (nur in `UNIQUE`-Schlüsseln).
- `LAST_INSERT_ID` wird gelöscht (clear), wenn man diese in ODBC benutzt: `WHERE auto_increment_spalte IS NULL`. Das scheint einige Probleme mit Access zu beheben.
- `SET SQL_AUTO_IS_NULL=0|1` schaltet jetzt die Handhabung von Suchen nach der letzten eingefügten Zeile bei `WHERE auto_increment_spalte IS NULL` aus / an.
- Neue `mysqld`-Variable `concurrency` für Solaris hinzugefügt.
- Option `--relative` für `mysqladmin` hinzugefügt, um mit `extended-status` eine bessere Beobachtung von Änderungen zu erzielen.
- Bug bei der Benutzung von `COUNT(DISTINCT ...)` auf eine leere Tabelle behoben.
- Unterstützung für den chinesischen Zeichensatz GBK hinzugefügt.
- Problem mit `LOAD DATA INFILE` und `BLOB`-Spalten behoben.
- Bit-Operator ~ (Negation) hinzugefügt.
- Problem mit `UDF`-Funktionen behoben.

D.2.41. Änderungen in Release 3.23.4

- Einfügen eines `DATETIME`-Werts in eine `TIME`-Spalte versucht jetzt nicht mehr, darin 'Tage' zu speichern.
- Problem mit der Speicherung von Float / Double auf kleinen Endian-Maschinen behoben (das betraf `SUM()`).
- Verbindungs-Zeitüberschreitung (Timeout) auf TCP/IP-Verbindungen hinzugefügt.
- Problem mit `LIKE` "%" auf einem Index, der `NULL`-Werte enthalten darf, behoben.
- `REVOKE ALL PRIVILEGES` widerrief nicht alle Berechtigungen.
- Erzeugung temporärer Tabellen mit demselben Namen wie die Original-Tabelle wird zugelassen.
- Wenn man einem Benutzer eine Berechtigungsoption (Grant Option) für eine Datenbank gewährte, konnte er die Berechtigungen nicht an andere Benutzer weitergeben.
- Neuer Befehl `SHOW GRANTS FOR benutzer` hinzugefügt (von Sinisa).
- Neue `date_add`-Syntax `date/datetime + INTERVAL # intervall_typ` hinzugefügt. Von Joshua Chamas.
- Berechtigungsüberprüfung für `LOAD DATA REPLACE` in Ordnung gebracht.
- Automatische Reparatur beschädigter Include-Dateien auf Solaris 2.7 hinzugefügt.
- Einige configure-Probleme behoben, um Probleme bei der Erkennung großer Dateisysteme zu beheben.

- `REGEXP` ist jetzt unabhängig von der verwendeten Groß-/Kleinschreibung, wenn Sie nicht binäre Zeichenketten verwenden.

D.2.42. Änderungen in Release 3.23.3

- Patches für MIT-pThread auf NetBSD angewandt.
- Bereichs-Bug in MyISAM behoben.
- `ASC` ist jetzt wieder Vorgabe für `ORDER BY`.
- `LIMIT` für `UPDATE` hinzugefügt.
- Neue Client-Funktion `mysql_change_user()` hinzugefügt.
- Zeichensatz zu `SHOW VARIABLES` hinzugefügt.
- Unterstützung von `--[leerraum]`-Kommentaren hinzugefügt.
- `INSERT into tabelle VALUES ()` wird zugelassen. Das heißt, Sie können jetzt eine leere Wertliste angeben, die in eine Zeile eingefügt wird, und in der jede Spalte auf ihren Vorgabewert gesetzt wird.
- `SUBSTRING(text FROM position)` geändert, um ANSI-SQL-kompatibel zu sein. (Vorher gab dieses Konstrukt das rechteste 'position'-Zeichen zurück.)
- `SUM()` mit `GROUP BY` gab auf manchen Systemen 0 zurück.
- Ausgabe bei `SHOW TABLE STATUS` geändert.
- `DELAY_KEY_WRITE`-Option für `CREATE TABLE` hinzugefügt.
- `AUTO_INCREMENT` wird für jeden beliebigen Schlüsselteil zugelassen.
- Problem mit `YEAR(NOW())` und `YEAR(CURDATE())` behoben.
- `CASE`-Konstrukt hinzugefügt.
- Neue Funktion `COALESCE()` hinzugefügt.

D.2.43. Änderungen in Release 3.23.2

- Bereichsoptimierer-Bug behoben: `SELECT * FROM tabelle WHERE schluessel_teil1 >= konstante AND (schluessel_teil2 = konstante OR schluessel_teil2 = konstante)`. Der Bug bestand darin, dass manche Zeilen im Ergebnis doppelt auftauchen konnten.
- Das Laufenlassen von `myisamchk` ohne `-a` aktualisierte die Index-Verteilung falsch.
- `SET SQL_LOW_PRIORITY_UPDATES=1` gab vorher einen Parser-Fehler.
- Sie können jetzt Spalten indexieren, die in der `WHERE`-Klausel benutzt werden. `UPDATE tabelle SET KEY=KEY+1 WHERE KEY > 100`
- Datums-Handhabung sollte jetzt etwas schneller sein.
- Handhabung von 'fuzzy' Datumsangaben möglich (Datumsangaben, bei denen der Tag oder der Monat 0 sind, wie 1999-01-00).
- Optimierung von `SELECT ... WHERE schluessel_teil1=konstante1 AND schluessel_teil_2=konstante2 AND schluessel_teil1=konstante4 AND schluessel_teil2=konstante4` in Ordnung gebracht. Indextyp sollte `range` anstelle von `ref` sein.
- `egcs`-1.1.2-Optimierer-Bug behoben (bei der Benutzung von `BLOB`s) auf Linux Alpha.
- Problem mit `LOCK TABLES` in Kombination mit `DELETE FROM tabelle` behoben.
- MyISAM-Tabellen lassen jetzt Schlüssel auf `NULL` und `BLOB/TEXT`-Spalten zu.
- Folgender Join ist jetzt viel schneller: `SELECT ... FROM t1 LEFT JOIN t2 ON ... WHERE t2.nicht_null_spalte IS NULL`.

- `ORDER BY` und `GROUP BY` können jetzt auf Funktionen angewendet werden.
- Handhabung von 'konstante' geändert, um Handhabung von `ORDER BY RAND()` zu gestatten.
- Indexe werden jetzt für `WHERE schluessel_spalte = funktion` benutzt.
- Indexe werden jetzt für `WHERE schluessel_spalte = spalten_name` benutzt, selbst wenn die Spalten nicht identisch komprimiert sind.
- Indexe werden jetzt für `WHERE spalten_name IS NULL` benutzt.
- HEAP-Tabellen so geändert, dass in der Reihenfolge niedriges Byte zuerst gespeichert wird (um es zu erleichtern, MyISAM-Tabellen zu konvertieren).
- Automatische Änderung temporärer HEAP-Tabellen in MyISAM-Tabellen im Falle von 'table is full'-Fehlern.
- Option `--init-file=datei` für `mysqld` hinzugefügt.
- `COUNT(DISTINCT wert, [wert, ...])` hinzugefügt.
- `CREATE TEMPORARY TABLE` erzeugt jetzt eine temporäre Tabelle in ihrem eigenen Namensraum, die automatisch gelöscht wird, wenn die Verbindung beendet wird.
- Neue reservierte Wörter (erforderlich für `CASE`): `CASE, THEN, WHEN, ELSE und END`.
- Neue Funktionen `EXPORT_SET()` und `MD5()` hinzugefügt.
- Unterstützung für den GB2312 chinesischen Zeichensatz hinzugefügt.

D.2.44. Änderungen in Release 3.23.1

- Einige Kompilierungsprobleme behoben.

D.2.45. Änderungen in Release 3.23.0

- Eine neue Tabellen-Handler-Bibliothek (`MyISAM`) mit vielen neuen Features hinzugefügt. See Abschnitt 8.1, „MyISAM-Tabellen“.
- Sie können `HEAP`-Tabellen im Hauptspeicher erzeugen, die zum Nachschlagen extrem schnell sind.
- Unterstützung für große Dateien (63-Bit) auf Systemen, die große Dateien unterstützen, hinzugefügt.
- Neue Funktion `LOAD_FILE(datei)` hinzugefügt, um die Inhalte einer Datei als Zeichenkettenwert zu erhalten.
- Neuer Operator `<=>` hinzugefügt, der wie `=` funktioniert, aber WAHR (true) zurückgibt, wenn beide Argumente `NULL` sind. Das ist nützlich, um Änderungen zwischen Tabellen zu vergleichen.
- ODBC-3.0-`EXTRACT(intervall FROM datetime)`-Funktion hinzugefügt.
- Spalten, die als `FLOAT(X)` definiert sind, werden beim Speichern nicht gerundet und dürfen beim Abruf in wissenschaftlicher Notation sein (1.0 E+10).
- `REPLACE` ist jetzt schneller als vorher.
- `LIKE`-Zeichenvergleiche geändert, so dass sie sich wie `=` verhalten. Das heißt, dass `'e' LIKE 'é'` jetzt WAHR (true) ist (falls hier etwas nicht richtig angezeigt wird: Das letztgenannte 'e' ist das französische 'e' mit Akzent).
- `SHOW TABLE STATUS` gibt eine Menge an Informationen über die Tabellen zurück.
- `LIKE` für den `SHOW STATUS`-Befehl hinzugefügt.
- Berechtigungsspalte zu `SHOW COLUMNS` hinzugefügt.
- Spalten `packed` und `comment` für `SHOW INDEX` hinzugefügt.
- Kommentare zu Tabellen (mit `CREATE TABLE ... COMMENT "kommentar"`) hinzugefügt.

- `UNIQUE`, wie bei `CREATE TABLE tabelle (spalte int not null UNIQUE)`, hinzugefügt.
- Neue CREATE-Syntax: `CREATE TABLE tabelle SELECT ...`
- Neue CREATE-Syntax: `CREATE TABLE IF NOT EXISTS ...`
- Die Erzeugung von `CHAR(0)`-Spalten wird zugelassen.
- `DATE_FORMAT()` erfordert jetzt '`%`' vor jeglichem Formatierungszeichen.
- `DELAYED` ist jetzt ein reserviertes Wort (tut uns leid :( ).
- Eine Beispiel-Prozedur wurde hinzugefügt: `analyse`, Datei: `sql_analyse.c`. Diese beschreibt die Daten in Ihrer Anfrage. Probieren Sie folgendes:

  ```
  SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elemente,[max speicher]])
  ```

  Diese Prozedur ist extrem nützlich, wenn Sie die Daten in Ihrer Tabelle prüfen wollen!
- `BINARY`-Cast, um zu erzwingen, dass eine Zeichenkette abhängig von der verwendeten Groß-/Kleinschreibung verglichen wird.
- Option `--skip-show-database` für `mysqld` hinzugefügt.
- Das Prüfen, ob sich eine Zeile bei einem `UPDATE` geändert hat, funktioniert jetzt auch bei `BLOB`-/`TEXT`-Spalten.
- Die `INNER`-Join-Syntax wurde hinzugefügt. **HINWEIS**: Hierdurch wurde `INNER` zu einem reservierten Wort!
- Unterstützung für Netmasks zum Hostname in den MySQL-Tabellen hinzugefügt. Sie können eine Netmask mit der `IP/NETMASK`-Syntax angeben.
- Wenn Sie eine `NOT NULL DATE/DATETIME`-Spalte mit `IS NULL` vergleichen, wird das zu einem Vergleich auf `0` geändert, um einige ODBC-Applikationen zufrieden zu stellen (von `<shreeve@uci.edu>`).
- `NULL IN (...)` gibt jetzt `NULL` anstelle von `0` zurück. Das stellt sicher, dass `null_spalte NOT IN (...)` nicht mit `NULL`-Werten übereinstimmt.
- Speicherung von Fließkommawerten in `TIME`-Spalten in Ordnung gebracht.
- Das Parsen von `TIME`-Zeichenketten geändert, so dass es strenger ist. Jetzt wird der Bruchteil-Sekunden-Teil erkannt (und momentan noch übergangen). Folgende Formate werden unterstützt:
  - `[[DAYS] [H]H:]MM:]SS[.bruchteil]`, `[[[[[H]H]H]H]MM]SS[.bruchteil]`
- Erkennen (und Ignorieren) des zweiten Bruchteil-Anteils von `DATETIME` hinzugefügt.
- `LOW_PRIORITY`-Attribut für `LOAD DATA INFILE` hinzugefügt.
- Der vorgabemäßige Index-Name benutzt jetzt dieselbe Groß-/Kleinschreibung wie der benutzte Spaltenname.
- Vorgabemäßige Anzahl von Verbindungen auf 100 geändert.
- Bei der Benutzung von `LOAD DATA INFILE` werden größere Puffer verwendet.
- `DECIMAL(x,y)` funktioniert jetzt gemäß ANSI-SQL.
- Aggregat-UDF-Funktionen. Dank an Andreas F. Bobak `<bobak@relog.ch>` hierfür!
- `LAST_INSERT_ID()` wird jetzt bei `INSERT INTO ... SELECT` aktualisiert.
- Einige kleinere Änderungen am Join-Tabellenoptimierer, um einige Joins schneller zu machen.
- `SELECT DISTINCT` ist viel schneller. Es benutzt die neue `UNIQUE`-Funktionalität in `MyISAM`. Ein Unterschied im Vergleich zur MySQL-Version 3.22 besteht darin, dass die Ausgabe von `DISTINCT` nicht mehr sortiert wird.
- Alle C-Client-API-Makros sind jetzt Funktionen, um die gemeinsam genutzten (shared) Bibliotheken verlässlicher zu machen. Deswegen können Sie nicht mehr `mysql_num_fields()` auf ein `MYSQL`-Objekt aufrufen, sondern müssen statt dessen `mysql_field_count()` benutzen.
- Benutzung von `LIBEWRAP`; Patch von Henning P. Schmiedehausen.

- `AUTO_INCREMENT` wird nur noch für numerische Spalten zugelassen.
- Durch die Verwendung von `AUTO_INCREMENT` wird die Spalte automatisch `NOT NULL`.
- `NULL` wird als Vorgabewert für AUTO_INCREMENT-Spalten angezeigt.
- `SQL_BIG_RESULT`; `SQL_SMALL_RESULT` ist jetzt Vorgabe.
- Ein gemeinsam genutztes (shared) Bibliothek-RPM hinzugefügt. Diese Verbesserung wurde von David Fox (dsfox@cogsci.ucsd.edu) beigesteuert.
- Ein `--enable-large-files/--disable-large-files`-Schalter zu `configure` hinzugefügt. Siehe `configure.in` wegen mancher Systeme, auf denen dies wegen nicht funktionierender Implementation automatisch abgeschaltet ist.
- `readline` für Version 4.0 aktualisiert.
- Neue `CREATE TABLE`-Optionen: `PACK_KEYS` und `CHECKSUM`.
- `mysqld`-Option `--default-table-type` hinzugefügt.

Anhang E. Anmerkungen zur Portierung auf andere Systeme

Für den Server wird eine funktionierende Posix-Thread-Bibliothek benötigt. Auf Solaris 2.5 benutzen wir Sun PThread (die native Thread-Unterstützung in Version 2.4 und früher ist nicht gut genug). Auf Linux benutzen wir LinuxThread von Xavier Leroy, <Xavier.Leroy@inria.fr>.

Der schwierige Teil der Portierung auf eine neue Unix-Variante ohne gute native Thread-Unterstützung ist wahrscheinlich, MIT-pThread zu portieren. Siehe with-pThread/README und POSIX-Thread programmieren.

Die MySQL-Distribution enthält eine gepatchte Version von Provenzanos PThread von MIT (siehe MIT-PThread-Website). Diese kann für einige Betriebssysteme benutzt werden, die kein POSIX-Thread haben.

Es ist ebenfalls möglich, ein anderes Thread-Paket auf Benutzerebene namens FSU-PThread zu benutzen (siehe FSU-PThread-Homepage). Diese Implementation wird für die SCO-Portierung benutzt.

In den thr_lock.c- und thr_alarm.c-Programmen im mysys-Verzeichnis finden Sie einige Tests / Beispiele dieser Probleme.

Sowohl Server als auch Client benötigen einen funktionierenden C++-Kompiler (wir benutzen gcc und haben SparcWorks ausprobiert). Ein anderer bekanntermaßen funktionierender Compiler ist Irix cc.

Um nur den Client zu kompilieren, benutzen Sie ./configure --without-server.

Es gibt momentan keine Unterstützung, um nur den Server zu kompilieren, noch ist es wahrscheinlich, dass eine solche hinzugefügt wird, falls nicht jemand einen guten Grund dafür findet.

Wenn Sie irgend welche Makefile oder das configure-Skript ändern wollen / müssen, müssen Sie sich Automake und Autoconf holen. Wir haben die automake-1.2- und autoconf-2.12-Distributionen benutzt.

Alle Schritte, die notwendig sind, um alles aus den grundlegendsten Dateien neu zu machen (make):

```
/bin/rm */.deps/*.P
/bin/rm -f config.cache
aclocal
autoheader
aclocal
automake
autoconf
./configure --with-debug=full --prefix='ihr_installationsverzeichnis'

# Die oben erzeugten makefiles benötigen GNU-make 3.75 oder neuer.
# (unten gmake genannt)
gmake clean all install init-db
```

Wenn Sie bei einer neuen Portierung Probleme bekommen, kann es sein, dass Sie MySQL etwas debuggen müssen! See Abschnitt E.1, „Einen MySQL-Server debuggen“.

HINWEIS: Bevor Sie mit dem Debuggen von mysqld anfangen, bringen Sie sich zuerst die Testprogramme mysys/thr_alarm und mysys/thr_lock zum Laufen. Das stellt sicher, dass Ihre Thread-Installation zumindest überhaupt eine Chance hat, zu funktionieren!

E.1. Einen MySQL-Server debuggen

Wenn Sie Funktionalität benutzen, die in MySQL sehr neu ist, können Sie versuchen, mysqld mit der --skip-new-Option laufen zu lassen (die alle sehr neue, potenziell unsichere Funktionalität abschaltet) oder mit --safe-mode, was viel an Optimierung abschaltet, die möglicherweise Probleme verursacht. See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“.

Wenn mysqld nicht starten will, sollten Sie prüfen, ob Sie irgend welche my.cnf-Dateien haben, die mit Ihrer Konfiguration in Konflikt kommen! Sie können Ihre my.cnf-Argumente mit mysqld --print-defaults prüfen und sie vermeiden, indem Sie mit mysqld --no-defaults ... starten.

Wenn mysqld anfängt, Prozessorleistung oder Speicher zu fressen, oder wenn er ``hängt", können Sie mysqladmin processlist status benutzen, um herauszufinden, ob irgend etwas eine Anfrage ausführt, die sehr lange dauert. Es ist eine gute Idee, mysqladmin -i10 processlist status in irgend einem Fenster laufen zu haben, wenn Sie Performance-Probleme oder Probleme damit haben, dass sich neue Clients nicht verbinden können.

Der Befehl mysqladmin debug dumpt Informationen über Sperren, die in Gebrauch sind, den benutzten Speicher und den Anfragengebrauch in die mysql-Log-Datei aus. Das kann helfen, einige Probleme zu lösen. Dieser Befehl stellt auch nützliche Informationen zur Verfügung, selbst wenn Sie MySQL nicht zum Debuggen kompiliert haben!

Wenn das Problem darin besteht, dass einige Tabellen langsamer und langsamer werden, sollten Sie versuchen, die Tabelle mit

`OPTIMIZE TABLE` der `myisamchk` zu optimieren. See Kapitel 5, *MySQL-Datenbankadministration*. Sie sollten langsame Anfragen darüber hinaus mit `EXPLAIN` überprüfen.

Ebenfalls sollten Sie den Abschnitt über betriebssystemspezifische Dinge in diesem Handbuch lesen, weil Sie Probleme haben könnten, die einzigartig für Ihre Umgebung sind. See Abschnitt 3.6, „Betriebssystem-spezifische Anmerkungen“.

E.1.1. MySQL zum Debuggen kompilieren

Wenn Sie sehr spezielle Probleme haben, können Sie immer versuchen, MySQL zu debuggen. Dafür müssen Sie MySQL mit der `--with-debug`- oder der `--with-debug=full`-Option kompilieren. Sie können prüfen, ob MySQL mit Debuggen kompiliert wurde oder nicht, wenn Sie `mysqld --help` ausführen. Wenn das `--debug`-Flag in den Optionen aufgeführt ist, haben Sie Debuggen eingeschaltet. `mysqladmin ver` gibt die `mysqld`-Version in diesem Fall ebenfalls als `mysql ... --debug` aus.

Wenn Sie gcc oder egcs benutzen, ist die empfohlene configure-Zeile:

```
CC=gcc CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors -fno-exceptions -fno-rtti" ./configure --prefix=/usr/lo
```

Das vermeidet Probleme mit der `libstdc++`-Bibliothek und mit C++-Ausnahmen (viele Compiler haben Probleme mit C++-Ausnahmen in threaded Code) und kompiliert eine MySQL-Version mit Unterstützung für alle Zeichensätze.

Wenn Sie einen Speicherüberlauffehler vermuten, können Sie MySQL mit `--with-debug=full` kompilieren, was zusätzlich einen (`SAFEMALLOC`)-Prüfer für die Speicherzuweisung installiert. Das Laufenlassen mit `SAFEMALLOC` ist jedoch recht langsam. Wenn Sie daher Performance-Probleme bekommen, sollten Sie `mysqld` mit der `--skip-safemalloc`-Option starten. Das schaltet die Speicherüberlaufprüfung für jeden Aufruf von `malloc` und `free` ab.

Wenn `mysqld` nicht mehr abstürzt, wenn Sie ihn mit `--with-debug` kompilieren, haben Sie wahrscheinlich einen Compiler-Bug oder einen Timing-Bug innerhalb von MySQL gefunden. In diesem Fall können Sie versuchen, `-g` für die `CFLAGS`- und `CXXFLAGS`-Variablen oben hinzuzufügen und nicht mehr `--with-debug` zu benutzen. Wenn `mysqld` jetzt stirbt, können Sie wenigstens mit `gdb` mit ihm verbinden oder `gdb` auf die Core-Datei benutzen, um herauszufinden, was passiert ist.

Wenn Sie MySQL zum Debuggen konfigurieren, können Sie viele zusätzliche Sicherheitprüffunktionen hinzufügen, die die Gesundheit von `mysqld` beobachten. Wenn Sie etwas ``Unerwartetes" finden, wird ein Eintrag nach `stderr` geschrieben, den `safe_mysqld` in die Fehler-Log-Datei leitet! Das heißt auch, dass Sie bei unerwarteten Problemen mit MySQL und der Benutzung einer Quelldistribution als erstes MySQL zum Debuggen konfigurieren sollten! (Die zweite Sache wäre natürlich, eine E-Mail an `<mysql@lists.mysql.com>` zu schicken und um Hilfe zu bitten. Bitte benutzen Sie das `mysqlbug`-Skript für alle Bug-Berichte oder Fragen hinsichtlich der MySQL-Version, die Sie benutzen!

In der Windows-MySQL-Distribution wird `mysqld.exe` vorgabemäßig mit Unterstützung für Trace-Dateien kompiliert.

E.1.2. Trace-Dateien erzeugen

Wenn der `mysqld`-Server nicht startet oder wenn Sie den `mysqld`-Server schnell zum Absturz bringen können, können Sie versuchen, eine Trace-Datei zu erzeugen, um das Problem zu finden.

Hierfür brauchen Sie einen `mysqld`, der zum Debuggen kompiliert ist. Sie können das mit `mysqld -V` prüfen. Wenn die Versionsnummer mit `-debug` endet, ist Unterstützung für Trace-Dateien einkompiliert.

Starten Sie den `mysqld`-Server mit einem Trace-Log in `/tmp/mysqld.trace` (oder `C:\mysqld.trace` unter Windows):

```
mysqld --debug
```

Unter Windows sollten Sie auch den `--standalone`-Flag benutzen, um `mysqld` nicht als Systemdienst zu starten.

Machen Sie folgendes in einem DOS-Fenster:

```
mysqld --debug --standalone
```

Danach können Sie das `mysql.exe`-Kommandozeilenwerkzeug in einem zweiten DOS-Fenster benutzen, um das Problem zu reproduzieren. Sie können den obigen `mysqld`-Server mit `mysqladmin shutdown` herunter fahren.

Beachten Sie, dass die Trace-Datei sehr *Groß* wird! Wenn Sie eine kleinere Trace-Datei haben wollen, können Sie etwa folgendes tun:

```
mysqld --debug=d,info,error,query,general,where:O,/tmp/mysqld.trace
```

Das gibt nur Informationen für die interessantesten Dinge in `/tmp/mysqld.trace` aus.

Wenn Sie hierüber einen Bug-Bericht erstellen, schicken Sie bitte nur die Zeilen aus der Trace-Datei an die entsprechende Mailing-Liste, in denen etwas schief zu gehen scheint! Wenn Sie diese Stelle nicht finden können, können Sie die Trace-Datei per FTP einschicken, zusammen mit einem kompletten Bug-Bericht, an ftp://Support.mysql.com/pub/mysql/secret, so dass ein MySQL-

Entwickler sich das ansehen kann.

Die Trace-Datei wird mit dem **DBUG**-Paket von Fred Fish hergestellt. See Abschnitt E.3, „Das DBUG-Paket“.

E.1.3. mysqld unter gdb debuggen

Auf den meisten Systemen können Sie `mysqld` von `gdb` starten, um mehr Informationen zu erhalten, wenn `mysqld` abstürzt.

Bei einigen älteren `gdb`-Versionen unter Linux müssen Sie `run --one-thread` benutzen, um den `mysqld`-Thread debuggen zu können. In diesem Fall können Sie zur gleichen Zeit nur einen Thread aktiv haben.

Wenn Sie `mysqld` unter gdb laufen lassen, sollten Sie den Stack-Trace mit `--skip-stack-trace` abschalten, um Segmentation-Fehler innerhalb gdb abfangen zu können.

Es ist sehr schwierig, MySQL unter `gdb` zu debuggen, wenn Sie permanent viele neue Verbindungen aufbauen, weil `gdb` den Speicher für den alten Thread nicht freigibt. Sie können dieses Problem vermeiden, indem Sie `mysqld` mit `-O thread_cache_size= 'maximale_verbindungen +1'` starten. In den meisten Fällen hilft bereits schon die Benutzung von `-O thread_cache_size=5'` recht viel!

Wenn Sie einen Coredump unter Linux erhalten wollen, wenn `mysqld` mit einem SIGSEGV-Signal stirbt, können Sie `mysqld` mit der `--core-file`-Option starten. Diese Core-Datei kann benutzt werden, um eine Zurückverfolgung (Backtrace) zu machen, die Ihnen helfen kann herauszufinden, warum `mysqld` starb:

```
shell> gdb mysqld core
gdb>   backtrace full
gdb>   exit
```

See Abschnitt A.4.1, „Was zu tun ist, wenn MySQL andauernd abstürzt“.

Wenn Sie gdb 4.17.x oder höher unter Linux benutzen, sollten Sie eine `.gdb`-Datei mit folgenden Informationen in Ihrem aktuellen Verzeichnis installieren:

```
set print sevenbit off
handle SIGUSR1 nostop noprint
handle SIGUSR2 nostop noprint
handle SIGWAITING nostop noprint
handle SIGLWP nostop noprint
handle SIGPIPE nostop
handle SIGALRM nostop
handle SIGHUP nostop
handle SIGTERM nostop noprint
```

Wenn Sie Probleme haben, den Thread mit gdb zu debuggen, sollten Sie gdb 5.x herunter laden und diesen statt dessen benutzen. Die neue gdb-Version hat eine stark verbesserte Thread-Handhabung!

Hier ist ein Beispiel, wie man mysqld debuggt:

```
shell> gdb /usr/local/libexec/mysqld
gdb> run
...
backtrace full # Tun Sie das, wenn mysqld abstürzt
```

Schließen Sie die obige Ausgabe in eine Mail ein, die mit `mysqlbug` erzeugt wurde und schicken Sie sie an `mysql@lists.mysql.com`.

Wenn `mysqld` hängen bleibt, können Sie versuchen, einige Systemwerkzeuge wie `strace` oder `/usr/proc/bin/pstack` zu benutzen, um herauszufinden, was `mysqld` zum Hängen brachte.

```
strace /tmp/log libexec/mysqld
```

Wenn Sie die Perl-`DBI`-Schnittstelle benutzen, können Sie Debug-Informationen anschalten, indem Sie die `trace`-Methode benutzen oder die `DBI_TRACE`-Umgebungsvariable setzen. See Abschnitt 9.2.2, „Die `DBI`-Schnittstelle“.

E.1.4. Einen Stack-Trace benutzen

Auf manchen Betriebssystemen enthält die Fehler-Log-Datei einen Stack-Trace, wenn `mysqld` unerwartet stirbt. Diese können Sie benutzen, um herauszufinden, wo (und vielleicht warum) `mysqld` starb. See Abschnitt 5.9.1, „Die Fehler-Log-Datei“. Um einen Stack-Trace zu erhalten, sollten Sie `mysqld` NICHT mit der `-fomit-frame-pointer`-Option für gcc kompilieren. See Abschnitt E.1.1, „MySQL zum Debuggen kompilieren“.

Wenn die Fehlerdatei etwas wie folgendes enthält:

```
mysqld got signal 11;
```

```
The manual section 'debugging a MySQL server' tells you how to use a
stack trace and/or the core file to produce a readable backtrace that may
help in finding out why mysqld died
Attemping backtrace. You can use the following information to find out
where mysqld died.  Wenn you see no messages after this, something went
terribly wrong
stack range sanity check, ok, backtrace follows
0x40077552
0x81281a0
0x8128f47
0x8127be0
0x8127995
0x8104947
0x80ff28f
0x810131b
0x80ee4bc
0x80c3c91
0x80c6b43
0x80c1fd9
0x80c1686
```

Können Sie herausfinden, wo `mysqld` starb, indem Sie folgendes tun:

1. Kopieren Sie die obigen Zahlen in eine Datei, zum Beispiel `mysqld.stack`.

2. Machen Sie eine symbolische Datei für den `mysqld`-Server:

   ```
   nm -n libexec/mysqld > /tmp/mysqld.sym
   ```

   Beachten Sie, dass viele MySQL-Binärdistributionen die obige Datei namens `mysqld.sym.gz` enthalten. In diesem Fall müssen Sie sie wie folgt entpacken:

   ```
   gunzip < bin/mysqld.sym.gz > /tmp/mysqld.sym
   ```

3. Führen Sie `resolve_stack_dump -s /tmp/mysqld.sym -n mysqld.stack` aus.

   Das gibt aus, wo `mysqld` starb. Wenn Ihnen das nicht hilft, herauszufinden, warum `mysqld` starb, sollten Sie einen Bug-Bericht machen und die Ausgabe des obigen Befehls in diesen Bericht einschließen.

   Beachten Sie aber, dass es uns in den meisten Fällen nicht weiterhilft, nur einen Stack-Trace zu haben, um die Ursache des Problems herauszufinden. Um den Bug feststellen oder einen Workaround zur Verfügung stellen zu können, müssen wir in den meisten Fällen die Anfrage kennen, die `mysqld` tötete, und am besten einen Testfall, so dass wir das Problem wiederholen können! See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.

E.1.5. Log-Dateien benutzen, um Gründe für Fehler in mysqld zu finden

Beachten Sie, dass Sie vor dem Start von `mysqld` mit `--log` alle Ihre Tabellen mit `myisamchk` prüfen sollten. See Kapitel 5, *MySQL-Datenbankadministration*.

Wenn `mysqld` stirbt oder hängenbleibt, sollten Sie ihn mit `--log` starten. Wenn `mysqld` wieder stirbt, können Sie das Ende der Log-Datei nach der Anfrage durchsuchen, die `mysqld` tötete.

Wenn Sie `--log` ohne einen Dateinamen verwenden, wird das Log im Datenbank-Verzeichnis als 'hostname'.log gespeichert. In den meisten Fällen ist es die letzte Anfrage in der Log-Datei, die `mysqld` tötete, aber das sollten Sie falls möglich sicherstellen, indem Sie `mysqld` neu starten und dieselbe Anfrage mit dem `mysql`-Kommandozeilenwerkzeug wiederholen. Wenn das funktioniert, sollten Sie ebenfalls alle komplizierten Anfragen testen, die nicht beendet wurden.

Sie können auch den Befehl `EXPLAIN` auf alle `SELECT`-Statements ausprobieren, die lange Zeit benötigen, um sicherzustellen, dass `mysqld` Indexe korrekt benutzt. See Abschnitt 6.2.1, „`EXPLAIN`-Syntax (Informationen über ein `SELECT` erhalten)“.

Sie finden Anfragen, die zur Ausführung lange Zeit benötigen, indem Sie `mysqld` mit `--log-slow-queries` starten. See Abschnitt 5.9.5, „Die Anfragen-Log-Datei für langsame Anfragen“.

Wenn Sie den Text `mysqld restarted` in der Fehler-Log-Datei-Datei (normalerweise namens `hostname.err`) finden, haben Sie wahrscheinlich eine Anfrage gefunden, die `mysqld` zum Absturz brachte. Wenn das passiert, sollten Sie alle Ihre Tabellen mit `myisamchk` prüfen (see Kapitel 5, *MySQL-Datenbankadministration*) und die Anfragen in den MySQL-Log-Dateien untersuchen, um herauszufinden, ob eine nicht funktioniert. Wenn Sie eine solche Anfrage finden, versuchen Sie zunächst, auf die neueste MySQL-Version zu aktualisieren. Wenn das nicht hilft und Sie nichts im `mysql`-Mailarchiv finden können, sollten Sie den Bug an `<mysql@lists.mysql.com>` berichten. Links zu Mailarchiven finden Sie online auf der MySQL-Dokumentationsseite.

Wenn Sie `mysqld` mit `--with-myisam-recover` gestartet haben, prüft MySQL automatisch `MyISAM`-Tabellen und versucht

sie zu reparieren, wenn sie als 'nicht korrekt geschlossen' oder 'beschädigt' gekennzeichnet sind. Wenn das passiert, schreibt MySQL einen Eintrag in die `hostname.err`-Datei `'Warning: Checking table ...'`, der von `Warning: Repairing table` gefolgt wird, wenn die Tabelle repariert werden muss. Wenn Sie viele solcher Fehler erhalten, ohne dass `mysqld` direkt davor unerwartet gestorben ist, stimmt etwas nicht und muss weiter untersucht werden. See Abschnitt 5.1.1, „mysqld-Kommandozeilenoptionen“.

Natürlich ist es kein gutes Zeichen, wenn `mysqld` unerwartet stirbt, doch in diesem Fall sollte man nicht die `Checking table...`-Meldungen untersuchen, sondern statt dessen versuchen herauszufinden, warum `mysqld` starb.

E.1.6. Einen Testfall herstellen, wenn Sie Tabellenbeschädigung feststellen

Wenn Sie beschädigte Tabellen erhalten oder wenn `mysqld` immer nach irgend einem Aktualisierungsbefehl fehlschlägt, können Sie mit folgendem überprüfen, ob der Bug reproduzierbar ist:

- Fahren Sie den MySQL-Daemon herunter (mit `mysqladmin shutdown`).
- Machen Sie eine Datensicherung der Tabellen (um dem sehr unwahrscheinlichen Fall vorzubeugen, dass die Reparatur etwas Schlechtes macht).
- Prüfen Sie alle Tabellen mit `myisamchk -s Datenbank/*.MYI`. Reparieren Sie jegliche beschädigten Tabellen mit `myisamchk -r datenbank/tabelle.MYI`.
- Machen Sie eine Datensicherung der Tabellen.
- Entfernen (oder verschieben) Sie jegliche alten Log-Dateien aus dem MySQL-Daten-Verzeichnis, wenn Sie mehr Platz brauchen.
- Starten Sie `mysqld` mit `--log-binary`. See Abschnitt 5.9.4, „Die binäre Update-Log-Datei“. Wenn Sie eine Anfrage finden wollen, die `mysqld` zum Absturz brachte, sollten Sie `--log --log-binary` benutzen.
- Wenn Sie eine beschädigte Tabelle erhalten, halten Sie `mysqld` an.
- Stellen Sie die Datensicherung wieder her.
- Starten Sie den `mysqld`-Server neu, **ohne** `--log-binary`.
- Führen Sie die Befehle mit `mysqlbinlog update-log-file | mysql` erneut aus. Die Update-Log-Datei wird im MySQL-Datenbank-Verzeichnis unter dem Namen `hostname-bin.#` gespeichert.
- Wenn die Tabellen wieder beschädigt werden oder Sie `mysqld` wieder dazu bringen können zu sterben, haben Sie einen reproduzierbaren Bug gefunden, der sich leicht beheben lassen sollte! Schicken Sie die Tabellen und die Binär-Log-Datei an ftp://support.mysql.com/pub/mysql/secret und schicken Sie eine E-Mail an `<bugs@lists.mysql.com>` oder (wenn Sie ein Support-Kunde sind) an `<Support@mysql.com>`, und das MySQL-Team wird den Bug so schnell wie möglich beheben.

Sie können auch das Skript `mysql_find_rows` benutzen, um einfach einige der Aktualisierungs-Statements auszuführen, wenn Sie das Problem eingrenzen wollen.

E.2. Einen MySQL-Client debuggen

Um einen MySQL-Client mit dem integrierten Debug-Paket debuggen zu können, sollten Sie MySQL mit `--with-debug` oder `--with-debug=full` kompilieren. See Abschnitt 3.3.3, „Typische `configure`-Optionen“.

Bevor Sie einen Client laufen lassen, sollten Sie die `MYSQL_DEBUG`-Umgebungsvariable setzen:

```
shell> MYSQL_DEBUG=d:t:O,/tmp/client.trace
shell> export MYSQL_DEBUG
```

Das bringt Clients dazu, eine Trace-Datei in `/tmp/client.trace` zu erzeugen.

Wenn Sie Probleme mit Ihrem eigenen Client-Code haben, sollten Sie versuchen, sich mit dem Server zu verbinden und Ihre Anfragen mit einem Client laufen zu lassen, der bekanntermaßen funktioniert. Lassen Sie dabei `mysql` im Debug-Modus laufen (unter der Annahme, dass Sie MySQL mit angeschaltetem Debuggen kompiliert haben):

```
shell> mysql --debug=d:t:O,/tmp/client.trace
```

Das stellt nützliche Informationen für den Fall bereit, dass Sie einen Bug-Bericht schicken. See Abschnitt 2.6.2.3, „Wie man Bugs oder Probleme berichtet“.

Wenn Ihr Client bei irgend einem 'zulässigen' Sperr-Code abstürzt, sollten Sie sicherstellen, dass Ihre `mysql.h`-Include-Datei mit Ihrer MySQL-Bibliotheksdatei zusammenpasst. Es ist ein häufiger Fehler, eine alte `mysql.h`-Datei aus einer alten MySQL-Installation mit einer neuen MySQL-Bibliothek zu benutzen.

E.3. Das DBUG-Paket

Der MySQL-Server und die meisten MySQL-Clients werden mit dem DBUG-Paket kompiliert, das ursprünglich von Fred Fish stammt. Wenn man MySQL zum Debuggen kompiliert hat, ermöglicht es dieses Paket, eine Trace-Datei davon zu erhalten, was das Programm debuggt. See Abschnitt E.1.2, „Trace-Dateien erzeugen“.

Man benutzt das Debug-Paket durch Aufruf des Programms mit der `--debug="..."`- oder der `-#...`-Option.

Die meisten MySQL-Programme haben eine vorgabemäßige Debug-Zeichenkette, die benutzt wird, wenn Sie keine Option für `--debug` angeben. Die vorgabemäßige Trace-Datei ist üblicherweise `/tmp/programm_name.trace` unter Unix und `\programm_name.trace` unter Windows.

Die Debug-Steuerungs-Zeichenkette ist eine Folge durch Doppelpunkte getrennter Felder, wie folgt:

```
<feld_1>:<feld_2>:...:<feld_N>
```

Jedes Feld besteht aus einem zwingend erforderlichen Flag-Zeichen, gefolgt durch ein optionales Komma (",") und eine durch Kommas getrennte Auflistung von Modifikatoren:

```
flag[,modifikator,modifikator,...,modifikator]
```

Aktuell werden folgende Flag-Zeichen erkannt:

d	Ausgabe von DBUG_<N>-Makros des aktuellen Status ermöglichen. Gegebenenfalls gefolgt von einer Auflistung von Schlüsselwörtern, die Ausgaben nur für die DBUG-Makros mit diesem Schlüsselwort auswählt. Eine leere Auflistung von Schlüsselwörtern bedeutet Ausgabe für alle Makros.
D	Nach jeder Debugger-Ausgabezeile verzögern. Das Argument ist die Anzahl von Zehntelsekunden der Verzögerung, abhängig von den Fähigkeiten der Maschine. `-#D,20` bedeutet als eine Verzögerung von 2 Sekunden.
f	Debuggen und / oder Tracen und Profilen auf die in der Auflistung genannten Funktionen beschränken. Beachten Sie, dass eine leere Liste alle Funktionen abschaltet. Die entsprechenden "d"- oder "t"-Flags müssen immer noch angegeben werden; dieser Flag beschränkt nur ihre Aktionen, wenn Sie angeschaltet sind.
F	Den Quell-Dateinamen für jede Zeile der Debug- oder Trace-Ausgabe festlegen.
i	Den Prozess mit der PID- oder Thread-Kennung für jede Ziele der Debug- oder Trace-Ausgabe festlegen.
g	Profiling anschalten. Es wird eine Datei namens 'dbugmon.out' erzeugt, die Informationen enthält, die benutzt werden können, um das Programm zu profilen. Wir gegebenenfalls von einer Auflistung von Schlüsselwörter gefolgt, die Profiling nur für die Funktionen in dieser Liste auswählen. Eine leere Liste bedeutet, dass alle Funktionen in Betracht gezogen werden.
L	Die Quell-Datei-Zeilennummer für jede Zeile der Debug- oder Trace-Ausgabe festlegen.
n	Die aktuelle Funktionsverschachtelungstiefe für jede Zeile der Debug- oder Trace-Ausgabe ausgeben.
N	Jede Zeile der dbug-Ausgabe nummerieren.
o	Die Debugger-Ausgabe in die angegebene Datei umlenken. Die vorgabemäßige Ausgabe ist stderr.
O	Wie `O`, aber die Datei wird nach jedem Schreiben auf die Platte zurückgeschrieben (flush). Wenn nötig, wird die Datei geschlossen und wieder geöffnet.
p	Debugger-Aktionen auf die angegebenen Prozesse beschränken. Ein Prozess muss mit dem DBUG_PROCESS-Makro gekennzeichnet sein und mit einer der Aktionen in der Liste übereinstimmen, damit Debugger-Aktionen durchgeführt werden.
P	Den aktuellen Prozessnamen für jede Zeile der Debug- oder Trace-Ausgabe ausgeben.
r	Wenn ein neuer Zustand gepusht wird, nicht die Funktionsverschachtelungsebene des alten Zustands übernehmen (erben). Nützlich, wenn die Ausgabe am linken Rand anfangen soll.
S	Funktion _sanity(_datei_,_zeile_) bei jeder debuggten Funktion ausführen, bis _sanity() etwas anderes als 0 zurückgibt. (Wird meist zusammen mit safemalloc benutzt, um Speicherlecks zu finden.)
t	Trace-Zeile für Funktionsaufrufen / Funktionsende anschalten. Wird gegebenenfalls gefolgt von einer Liste (die nur einen Modifikator enthält), in der numerisch eine maximale Trace-Ebene angegeben wird, nach der keine Ausgaben mehr erfolgen, weder für Debuggen noch für das Tracen von Makros. Die Vorgabe ist eine Kompilierzeit-Option.

Einige Beispiele von Debug-Steuerungs-Zeichenketten, die auf einer Shell-Kommandozeile erscheinen können (das "-#" wird

typischerweise benutzt, um eine Steuerungs-Zeichenkette für ein Applikationsprogramm einzuführen):

```
-#d:t
-#d:f,main,subr1:F:L:t,20
-#d,input,output,files:n
-#d:t:i:O,\\mysqld.trace
```

In MySQL werden gebräuchlicherweise (mit der `d`-Option) folgende Tags ausgegeben: `enter`, `exit`, `error`, `warning`, `info` und `loop`.

E.4. Sperrmethoden

Momentan unterstützt MySQL Tabellensperren nur für `ISAM`- / `MyISAM`- und `HEAP`-Tabellen und Sperren auf Seitenebene nur für `BDB`-Tabellen. See Abschnitt 6.3.1, „Wie MySQL Tabellen sperrt“. Bei `MyISAM`-Tabellen können Sie `INSERT` und `SELECT` ohne Sperren frei vermischen. (`Versionierung`).

Ab Version 3.23.33 können Sie die Tabellensperr-Konkurrenz auf Ihrem System durch Prüfen der `Table_locks_waited`- und `Table_locks_immediate`-Umgebungsvariablen analysieren.

Einige Datenbankbenutzer behaupten, dass MySQL keine große Anzahl gleichzeitiger Benutzer unterstützen kann, weil es kein Sperren auf Zeilenebene hat. Das mag bei einigen speziellen Applikationen zutreffen, aber nicht allgemein. Wie immer hängt das völlig davon ab, was Ihre Applikation macht, und davon, wie das Zugriffs-/Aktualisierungs-Muster der Daten aussieht.

Vorteile für Zeilensperren:

- Weniger Sperrkonflikte beim Zugriff auf unterschiedliche Zeilen in vielen Threads.
- Weniger Änderungen bei Rollbacks.
- Macht es möglich, eine einzelne Zeile lange zu sperren.

Nachteile:

- Benötigt mehr Speicher als Sperren auf Seiten- oder Tabellenebene.
- Ist langsamer als Sperren auf Seiten- oder Tabellenebene, wenn es einen großen Teil der Tabelle betrifft, weil man viel mehr Sperren durchführen muss.
- Ist definitiv viel schlechter als andere Sperren, wenn Sie oft `GROUP BY` auf einen großen Teil der Daten ausführen oder wenn man die gesamte Tabelle oft scannen muss.
- Bei Sperren auf höherer Ebene kann man einfacher Sperren unterschiedlichen Typs unterstützen, um die Applikation zu optimieren, weil der Sperr-Overhead sich weniger als bei Sperren auf Zeilenebene bemerkbar macht.

Tabellensperren sind Seiten- oder Zeilensperren in folgenden Fällen überlegen:

- Wenn man meist liest.
- Wenn Lese- und Aktualisierungsoperationen auf strengen Schlüsseln erfolgen. Das ist dann der Fall, wenn man eine Zeile aktualisiert oder löscht, die mit einem Schlüssel-Lesen geholt werden kann:

  ```
  UPDATE tabelle SET spalte=wert WHERE eindeutige_schluessel_nummer
  DELETE FROM tabelle WHERE eindeutiger_schluessel=#
  ```

- `SELECT` in Kombination mit `INSERT` (und sehr wenigen `UPDATE`'s und `DELETE`'s).
- Viele Scans / `GROUP BY` auf die gesamte Tabelle ohne irgend welche Schreibvorgänge.

Andere Optionen als Sperren auf Zeilen- / Seiten-Ebene:

Versionierung (wie die, die wir bei MySQL für gleichzeitige Einfügevorgänge nutzen), bei der man gleichzeitig einen Schreibvorgang haben kann, während viele Lesevorgänge stattfinden. Das heißt, dass die Datenbank / Tabelle verschiedene Sichten der Daten unterstützt, abhängig davon, wann man anfing, darauf zuzugreifen. Andere Namen hierfür sind Zeitreisen, Kopieren beim Schreiben (Copy on Write) oder Kopieren bei Bedarf (Copy on Demand).

Kopieren bei Bedarf ist in vielen Fällen viel besser als Sperren auf Seiten- oder Zeilenebene. Im schlimmsten Fall wird jedoch viel mehr Speicher verbraucht als bei der Benutzung normaler Sperren.

Anstelle von Zeilen-Sperren kann man Sperren auf Applikationsebene benutzen (wie get_lock/release_lock in MySQL). Das funktioniert natürlich nur bei 'wohl erzogenen' Applikationen.

In vielen Fällen kann man auf fortgeschrittene Art raten, welcher Sperrtyp der beste für die Applikation ist, aber allgemein ist es sehr schwer zu sagen, dass ein bestimmter Sperrtyp besser ist als ein anderer. Alles hängt von der Applikation ab, und verschiedene Teile der Applikation können nach unterschiedlichen Sperrtypen verlangen.

Hier sind einige Tipps zu Sperren in MySQL:

Bei Web-Applikation führen die meisten Applikationen viele SELECTs aus, sehr wenige DELETEs, UPDATEs hauptsächlich auf Schlüssel und INSERTs in einigen bestimmten Tabellen. Die grundlegende Einrichtung von MySQL ist hierfür BESTENS optimiert.

Gleichzeitige Benutzer sind kein Problem, solange man UPDATEs und SELECTs nicht vermischt, die beide gleichzeitig viele Zeilen in derselben Tabelle untersuchen müssen.

Wenn man INSERTs und DELETEs auf dieselbe Tabelle mischt, kann `INSERT DELAYED` eine große Hilfe sein.

Man kann auch `LOCK TABLES` benutzen, um Dinge zu beschleunigen (viele UPDATEs innerhalb einer einzelnen Sperre sind viel schneller als UPDATEs ohne Sperren). Daten in unterschiedliche Tabellen aufteilen hilft hierbei auch.

Wenn Sie Geschwindigkeitsprobleme mit den Tabellensperren in MySQL bekommen, können Sie diese eventuell dadurch lösen, dass Sie Ihre Tabellen in `BDB`-Tabellen umwandeln. See Abschnitt 8.6, „BDB- oder Berkeley_db-Tabellen“.

Der Optimierungsabschnitt dieses Handbuchs behandelt viele verschiedene Aspekte dessen, wie man seine Applikationen optimieren kann. See Abschnitt 6.2.11, „Weitere Optimierungstipps“.

E.5. Anmerkungen zu RTS-Thread

Ich habe versucht, die RTS-Thread-Pakete bei MySQL zu benutzen, bin aber über folgende Probleme gestolpert:

Sie benutzen die alte Version vieler POSIX-Aufrufe und es ist sehr mühsam, Wrapper für alle Funktionen zu schreiben. Ich neige dazu zu denken, dass es leichter ist, die Thread-Bibliotheken auf die neueste POSIX-Spezifikation zu ändern.

Einige Wrapper sind bereits geschrieben. Siehe `mysys/my_pThread.c` wegen weiterer Informationen.

Zumindest folgendes sollte geändert werden:

`pthread_get_specific` sollte ein Argument benutzen. `sigwait` sollte zwei Argumente entgegennehmen. Viele Funktionen (zumindest `pthread_cond_wait` und `pthread_cond_timedwait`) sollten bei einem Fehler den Fehler-Code zurückgeben. Momentan geben sie -1 zurück und setzen `errno`.

Ein weiteres Problem ist, dass Threads auf Benutzerebene das `ALRM`-Signal benutzen und dass dieses viele Funktionen abbricht (`read`, `write`, `open`, ...). MySQL sollte versuchen, nach der Unterbrechung all dieser Funktionen weiterzumachen, aber das ist nicht einfach zu verifizieren.

Das größte ungelöste Problem ist folgendes:

Um Alarme auf Thread-Ebene zu erhalten, änderte ich `mysys/thr_alarm.c` in der Art, dass es zwischen Alarmen wartet, mit `pthread_cond_timedwait()`, aber das bricht mit Fehler `EINTR` ab. Ich versuchte, die Thread-Bibliothek zu debuggen, um den Grund herauszufinden, konnte aber keine einfache Lösung finden.

Wenn jemand MySQL mit RTS-Thread ausprobieren möchte, schlage ich folgendes vor:

- Funktionen, die MySQL benutzt, von der Thread-Bibliothek zu POSIX ändern. Das sollte nicht lange dauern.
- Alle Bibliotheken mit `-DHAVE_rts_thread` kompilieren.
- `thr_alarm` kompilieren.
- Wenn es kleine Unterschiede in der Implementation gibt, können diese behoben werden, indem man `my_pThread.h` und `my_pThread.c` ändert.
- `thr_alarm` laufen lassen. Wenn es ohne irgend welche ``warning''-, ``error''- oder ``aborted''-Meldungen läuft, sind Sie auf dem richtigen Weg. Hier ist ein erfolgreiches Laufenlassen unter Solaris:

```
Main Thread: 1
```

```
Thread 0 (5) started
Thread: 5  Waiting
process_alarm
Thread 1 (6) started
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 1 (1) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 2 (2) sec
Thread: 6  Simulation of no alarm needed
Thread: 6  Slept for 0 (3) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 4 (4) sec
Thread: 6  Waiting
process_alarm
thread_alarm
Thread: 5  Slept for 10 (10) sec
Thread: 5  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 5 (5) sec
Thread: 6  Waiting
process_alarm
process_alarm

...
thread_alarm
Thread: 5  Slept for 0 (1) sec
end
```

E.6. Unterschiede zwischen verschiedenen Thread-Paketen

MySQL ist sehr abhängig vom verwendeten Thread-Paket. Wenn Sie daher eine gute Plattform für MySQL auswählen, ist das Thread-Paket sehr wichtig.

Es gibt mindestens drei Typen von Thread-Paketen:

- Benutzer-Thread in einem einzelnen Prozess. Das Thread-Umschalten wird mit Alarmen gemacht und die Thread-Bibliothek verwaltet alle nicht Thread-sicheren Funktionen mit Sperren. Lese-, Schreib- und Auswahl-Operationen werden üblicherweise mit einer Thread-spezifischen Auswahl verwaltet, die auf einen anderen Thread umschaltet, wenn der laufende Thread auf Daten warten muss. Wenn die Benutzer-Thread-Pakete in die Standard-Bibliotheken integriert sind (FreeBSD- und BSDI-Thread), erfordert das Thread-Paket weniger Overhead als Thread-Pakete, die alle unsicheren Aufrufen mappen müssen (MIT-pThread, FSU-PThread und RTS-Thread). In einigen Umgebungen (beispielsweise SCO) sind alle Systemaufrufe Thread-sicher, weshalb das Mapping sehr leicht durchgeführt werden kann (FSU-PThread unter SCO). Nachteil: Alle gemappten Aufrufe benötigen etwas Zeit und es ist sehr verzwickt, alle Situationen handhaben zu können. Üblicherweise gibt es auch einige Systemaufrufe, die vom Thread-Paket nicht gehandhabt werden (wie MIT-pThread und Sockets). Thread-Scheduling ist nicht immer optimal.
- Benutzer-Thread in separaten Prozessen. Das Thread-Umschalten wird vom Kernel durchgeführt und alle Daten werden zwischen den Threads geteilt. Das Thread-Paket verwaltet die Standard-Thread-Aufrufe, so dass diese Daten zwischen Threads teilen können. LinuxThread benutzt diese Methode. Nachteil: viele Prozesse. Die Erzeugung von Threads ist langsam. Wenn ein Thread stirbt, bleiben die übrigen üblicherweise hängen, und Sie müssen alle töten, bevor Sie neu starten können. Man kann sagen, dass die Thread-Umschaltung ziemlich viel kostet.
- Kernel-Thread. Das Thread-Umschalten wird von der Thread-Bibliothek oder dem Kernel durchgeführt und ist sehr schnell. Alles wird in einem Prozess gemacht, aber auch manchen Systemen zeigt `ps` die verschiedenen Threads. Wenn ein Thread abbricht, bricht der gesamte Prozess ab. Die meisten Systemaufrufe sind Thread-sicher und sollten sehr wenig Overhead beanspruchen. Solaris, HP-UX, AIX und OSF1 haben Kernel-Thread.

Auf manchen Systemen wird Kernel-Thread gehandhabt, indem Benutzerebenen-Thread in die Systembibliotheken integriert wird. In solchen Fällen kann das Umschalten nur von der Thread-Bibliothek durchgeführt werden und der Kernel ist sich nicht wirklich ``der Threads bewusst''.

Anhang F. Umgebungsvariablen

Hier ist eine Auflistung aller Umgebungsvariablen, die direkt oder indirekt von MySQL benutzt werden. Die meisten von ihnen finden sich auch an anderen Stellen dieses Handbuchs.

Beachten Sie, dass jegliche Optionen auf der Kommandozeile vorrangig vor Werten, die in Konfigurationsdateien und Umgebungsvariablen angegeben sind, und Werte in Konfigurationsdateien vorrangig vor Werten in Umgebungsvariablen sind.

In vielen Fällen ist es vorzuziehen, eine configure-Datei anstelle von Umgebungsvariablen zu verwenden, um das Verhalten von MySQL zu beeinflussen. See Abschnitt 5.1.2, „my.cnf-Optionsdateien“.

`CCX`	Setzen Sie diese für Ihren C++-Kompiler, wenn Sie configure laufen lassen.
`CC`	Setzen Sie diese für Ihren C-Kompiler, wenn Sie configure laufen lassen.
`CFLAGS`	Flags für Ihren C-Kompiler, wenn Sie configure laufen lassen.
`CXXFLAGS`	Flags für Ihren C++-Kompiler wenn Sie configure laufen lassen.
`DBI_USER`	Der vorgabemäßige Benutzername für Perl-DBI.
`DBI_TRACE`	Beim Tracen in Perl-DBI benutzt.
`HOME`	Der vorgabemäßige Pfad für die `mysql`-History-Datei ist `$HOME/.mysql_history`.
`LD_RUN_PATH`	Wird benutzt um anzugeben, wo Ihr `libmysqlclient.so` ist.
`MYSQL_DEBUG`	Debug-Trace-Optionen beim Debuggen.
`MYSQL_HISTFILE`	Der Pfad zur `mysql`-History-Datei.
`MYSQL_HOST`	Vorgabemäßiger Hostname, der von der `mysql`-Befehlszeilenaufforderung benutzt wird.
`MYSQL_PWD`	Das vorgabemäßige Passwort bei der Verbindung mit `mysqld`. Beachten Sie, dass die Benutzung dieser Umgebungsvariablen unsicher ist!
`MYSQL_TCP_PORT`	Der vorgabemäßige TCP/IP-Port.
`MYSQL_UNIX_PORT`	Der vorgabemäßige Socket; benutzt für Verbindungen nach `localhost`.
`PATH`	Wird von der Shell benutzt, um die MySQL-Programme zu finden.
`TMPDIR`	Das Verzeichnis, in dem temporäre Tabellen / Dateien erzeugt werden.
`TZ`	Diese Variable sollte auf Ihre lokale Zeitzone gesetzt sein. See Abschnitt A.4.6, „Zeitzonen-Probleme“.
`UMASK_DIR`	Die Erzeugungsmaske (Creation Mask) des Benutzer-Verzeichnisses, wenn Verzeichnisse angelegt werden. Beachten Sie, dass diese mit `UMASK` mit einem logischen UND verknüpft wird!
`UMASK`	Die Erzeugungsmaske (Creation Mask) bei der Erzeugung von Dateien.
`USER`	Der vorgabemäßige Benutzer unter Windows, der beim Verbinden mit `mysqld` benutzt wird.

Anhang G. Beschreibung der MySQL-Syntax für reguläre Ausdrücke

Ein regulärer Ausdruck (regex) ist eine mächtige Möglichkeit, eine komplexe Suche zu formulieren.

MySQL benutzt Henry Spencers Implementation regulärer Ausdrücke, die anstrebt, POSIX-1003.2-konform zu sein. MySQL benutzt die erweiterte Version.

Die vorliegende vereinfachte Referenz überspringt die Details. Um genauere Informationen zu erhalten, sehen Sie sich Henry Spencers `regex(7)`-Handbuchseite an, die in der Quelldistribution enthalten ist. See Anhang C, *Danksagungen*.

Ein regulärer Ausdruck beschreibt einen Satz von Zeichenketten. Der einfachste regexp ist einer, der keine Sonderzeichen enthält. Der regexp `hello` beispielsweise stimmt mit `hello` und sonst nichts überein.

Nicht triviale reguläre Ausdrücke benutzen bestimmte spezielle Konstrukte, so dass sie mit mehr als einer Zeichenkette übereinstimmen können. Der regexp `hallo|stefan` beispielsweise stimmt entweder mit der Zeichenkette `hallo` oder der Zeichenkette `stefan` überein.

Um ein komplexeres Beispiel zu geben, stimmt der regexp `B[an]*s` mit jeder der Zeichenketten `Bananas`, `Baaaaas`, `Bs` und jeder anderen Zeichenkette überein, die mit einem `B` anfängt, mit einem `s` aufhört und jede beliebige Anzahl von `a`- oder `n`-Zeichen dazwischen enthält.

Ein regulärer Ausdruck kann jedes der folgenden Sonderzeichen bzw. Konstrukte benutzen (0 = keine Übereinstimmung):

- `^`

  Stimmt mit dem Anfang einer Zeichenkette überein.

  ```
  mysql> select "fo\nfo" REGEXP "^fo$";            -> 0
  mysql> select "fofo" REGEXP "^fo";               -> 1
  ```

- `$`

  Stimmt mit dem Ende einer Zeichenkette überein.

  ```
  mysql> select "fo\no" REGEXP "^fo\no$";          -> 1
  mysql> select "fo\no" REGEXP "^fo$";             -> 0
  ```

- `.`

  Stimmt mit jedem Zeichen überein (inklusive neue Zeile).

  ```
  mysql> select "fofo" REGEXP "^f.*";              -> 1
  mysql> select "fo\nfo" REGEXP "^f.*";            -> 1
  ```

- `a*`

  Stimmt mit jeder Folge von 0 oder mehr `a`-Zeichen überein.

  ```
  mysql> select "Ban" REGEXP "^Ba*n";              -> 1
  mysql> select "Baaan" REGEXP "^Ba*n";            -> 1
  mysql> select "Bn" REGEXP "^Ba*n";               -> 1
  ```

- `a+`

  Stimmt mit jeder Folge von einem oder mehr `a`-Zeichen überein.

  ```
  mysql> select "Ban" REGEXP "^Ba+n";              -> 1
  mysql> select "Bn" REGEXP "^Ba+n";               -> 0
  ```

- `a?`

  Stimmt mit 0 oder einem `a`-Zeichen überein.

  ```
  mysql> select "Bn" REGEXP "^Ba?n";               -> 1
  mysql> select "Ban" REGEXP "^Ba?n";              -> 1
  mysql> select "Baan" REGEXP "^Ba?n";             -> 0
  ```

- `de|abc`

  Stimmt mit den Zeichenfolgen `de` oder `abc` überein.

```
mysql> select "pi" REGEXP "pi|apa";            -> 1
mysql> select "axe" REGEXP "pi|apa";           -> 0
mysql> select "apa" REGEXP "pi|apa";           -> 1
mysql> select "apa" REGEXP "^(pi|apa)$";       -> 1
mysql> select "pi" REGEXP "^(pi|apa)$";        -> 1
mysql> select "pix" REGEXP "^(pi|apa)$";       -> 0
```

- `(abc)*`

  Stimmt mit 0 oder mehr Instanzen der Folge `abc` überein.

```
mysql> select "pi" REGEXP "^(pi)*$";           -> 1
mysql> select "pip" REGEXP "^(pi)*$";          -> 0
mysql> select "pipi" REGEXP "^(pi)*$";         -> 1
```

- `{1}`, `{2,3}`

  Es gibt eine allgemeinere Schreibweise für regexps, die mit vielen Vorkommen des vorherigen Atoms übereinstimmen.

  - `a*`

    Kann als `a{0,}` geschrieben werden.

  - `a+`

    Kann als `a{1,}` geschrieben werden.

  - `a?`

    Kann als `a{0,1}` geschrieben werden.

  Um genauer zu sein, stimmt ein Atom, gefolgt von einer Begrenzung, die eine Ganzzahl `i` und keine Kommas enthält, mit einer Folge von genau `i` Übereinstimmungen des Atoms überein. Ein Atom gefolgt von einer Begrenzung, die eine Ganzzahl `i` und ein Komma enthält, stimmt mit einer Folge von `i` oder mehr Übereinstimmungen des Atoms überein. Ein Atom, gefolgt von einer Begrenzung, die zwei Ganzzahlen `i` und `j` Übereinstimmungen enthält, stimmt mit einer Folge von `i` bis `j` (inklusive) Übereinstimmungen des Atoms überein.

  Beide Argumente müssen im Bereich von `0` bis `RE_DUP_MAX` (Vorgabe 255) inklusive sein. Wenn es zwei Argumente gibt, muss das zweite größer oder gleich dem ersten sein.

- `[a-dX]`, `[^a-dX]`

  Stimmt mit jedem Zeichen überein, was entweder `a`, `b`, `c`, `d` oder `X` ist (oder nicht ist, wenn ^ benutzt wird). Um ein literales `]`-Zeichen einzuschließen, muss es unmittelbar der öffnenden Klammer `[` folgen. Um ein literales `-`-Zeichen einzuschließen, muss es zuerst oder zuletzt geschrieben werden. Daher stimmt `[0-9]` mit jeder Dezimalziffer überein. Alle Zeichen, die innerhalb eines `[]`-Paars keine definierte Bedeutung haben, haben keine spezielle Bedeutung und stimmen nur mit sich selbst überein.

```
mysql> select "aXbc" REGEXP "[a-dXYZ]";        -> 1
mysql> select "aXbc" REGEXP "^[a-dXYZ]$";      -> 0
mysql> select "aXbc" REGEXP "^[a-dXYZ]+$";     -> 1
mysql> select "aXbc" REGEXP "^[^a-dXYZ]+$";    -> 0
mysql> select "gheis" REGEXP "^[^a-dXYZ]+$";   -> 1
mysql> select "gheisa" REGEXP "^[^a-dXYZ]+$";  -> 0
```

- `[[.zeichen.]]`

  Die Zeichenfolge des vereinigten Elements. Die Folge ist ein einzelnes Element der Ausdrucksliste in der Klammer. Ein Klammerausdruck, der ein Mehrzeichen-Vereinigungselement enthält, kann daher mit mehr als einem Zeichen übereinstimmen. Wenn die Vereinigungsfolge zum Beispiel ein `ch`-Vereinigungselement enthält, stimmt der reguläre Ausdruck `[[.ch.]]*c` mit den ersten fünf Zeichen von `chchcc` überein.

- `[=zeichen_klasse=]`

  Eine Äquivalenzklasse, die für Zeichenfolgen aller Vereinigungselemente dieser steht, inklusive sich selbst.

  Wenn zum Beispiel `o` und `(+)` die Mitglieder einer Äquivalenzklasse sind, sind `[[=o=]]`, `[[=(+)=]]` und `[o(+)]` allesamt Synonyme. Eine Äquivalenzklasse darf kein Endpunkt eines Bereichs sein.

- `[:zeichen_klasse:]`

  Innerhalb eines Klammerausdrucks steht der Name einer Zeichenklasse, die in `[:` und `:]` eingeschlossen ist, für die Auflistung aller Zeichen, die zu dieser Klasse gehören. Standard-Zeichenklassennamen sind:

alnum	digit	punct
alpha	graph	space
empty	lower	upper
cntrl	print	xdigit

  Diese stehen für die Zeichenklassen, die auf der `ctype(3)`-Handbuchseite definiert sind. Ein Locale darf andere zur Verfügung stellen. Eine Zeichenklasse darf nicht als Endpunkt eines Bereichs benutzt werden.

```
mysql> select "justalnums" REGEXP "[[:alnum:]]+";       -> 1
mysql> select "!!" REGEXP "[[:alnum:]]+";               -> 0
```

- `[[:<:]]`, `[[:>:]]`

  Diese stimmen mit der Null-Zeichenkette am Anfang bzw. am Ende eines Worts überein. Ein Wort ist definiert als Folge von Wort-Zeichen, dem weder Wortzeichen vorangestellt sind noch darauf folgen. Ein Wortzeichen ist ein alnum-Zeichen (wie in `ctype(3)` definiert) oder ein Unterstrich (_).

```
mysql> select "a word a" REGEXP "[[:<:]]word[[:>:]]";      -> 1
mysql> select "a xword a" REGEXP "[[:<:]]word[[:>:]]";     -> 0
```

```
mysql> select "weeknights" REGEXP "^(wee|week)(knights|nights)$"; -> 1
```

Anhang H. GNU GENERAL PUBLIC LICENSE

Version 2, Juni 1991

Vorwort

Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Software weiterzugeben und zu verändern. Im Gegensatz dazu soll Ihnen die GNU General Public License, die allgemeine öffentliche GNU-Lizenz, ebendiese Freiheit garantieren. Sie soll sicherstellen, dass die Software für alle Benutzer frei ist. Diese Lizenz gilt für den Großteil der von der Free Software Foundation herausgegebenen Software und für alle anderen Programme, deren Autoren ihr Werk dieser Lizenz unterstellt haben. Auch Sie können diese Möglichkeit der Lizenzierung für Ihre Programme anwenden. (Ein anderer Teil der Software der Free Software Foundation unterliegt stattdessen der GNU Library General Public License, der allgemeinen öffentlichen GNU-Lizenz für Bibliotheken.)

Die Bezeichnung "freie" Software bezieht sich auf Freiheit, nicht auf den Preis. Unsere Lizenzen sollen Ihnen die Freiheit garantieren, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), die Möglichkeit, die Software im Quelltext zu erhalten oder den Quelltext auf Wunsch zu bekommen. Die Lizenzen sollen garantieren, dass Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen - und dass Sie wissen, dass Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir Einschränkungen machen, die es jedem verbieten, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesen Einschränkungen folgen bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Software verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die Sie selbst haben, wenn Sie - kostenlos oder gegen Bezahlung - Kopien eines solchen Programms verbreiten. Sie müssen sicherstellen, dass auch sie den Quelltext erhalten bzw. erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie Ihre Rechte kennen.

Wir schützen Ihre Rechte in zwei Schritten: (1) Wir stellen die Software unter ein Urheberrecht (Copyright), und (2) wir bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Autoren und uns zu schützen, wollen wir darüberhinaus sicherstellen, dass jeder erfährt, dass für diese freie Software keinerlei Garantie besteht. Wenn die Software von jemand anderem modifiziert und weitergegeben wird, möchten wir, dass die Empfänger wissen, dass sie nicht das Original erhalten haben, damit von anderen verursachte Probleme nicht den Ruf des ursprünglichen Autors schädigen.

Schließlich und endlich ist jedes freie Programm permanent durch Software-Patente bedroht. Wir möchten die Gefahr ausschließen, dass Distributoren eines freien Programms individuell Patente lizensieren - mit dem Ergebnis, dass das Programm proprietär würde. Um dies zu verhindern, haben wir klargestellt, dass jedes Patent entweder für freie Benutzung durch jedermann lizenziert werden muss oder überhaupt nicht lizenziert werden darf.

Es folgen die genauen Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung:

GNU GENERAL PUBLIC LICENSE Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung

1. Diese Lizenz gilt für jedes Programm und jedes andere Werk, in dem ein entsprechender Vermerk des Copyright-Inhabers darauf hinweist, dass das Werk unter den Bestimmungen dieser General Public License verbreitet werden darf. Im folgenden wird jedes derartige Programm oder Werk als "das Programm" bezeichnet; die Formulierung "auf dem Programm basierendes Werk" bezeichnet das Programm sowie jegliche Bearbeitung des Programms im urheberrechtlichen Sinne, also ein Werk, welches das Programm, auch auszugsweise, sei es unverändert oder verändert und/oder in eine andere Sprache übersetzt, enthält. (Im folgenden wird die Übersetzung ohne Einschränkung als "Bearbeitung" eingestuft.) Jeder Lizenznehmer wird im folgenden als "Sie" angesprochen.

   Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von dieser Lizenz nicht berührt; sie fallen nicht in Ihren Anwendungsbereich. Der Vorgang der Ausführung des Programms wird nicht eingeschränkt, und die Ausgaben des Programms unterliegen dieser Lizenz nur, wenn der Inhalt ein auf dem Programm basierendes Werk darstellt (unabhängig davon, dass die Ausgabe durch die Ausführung des Programmes erfolgte). Ob dies zutrifft, hängt von den Funktionen des Programms ab.

2. Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, dass Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluss veröffentlichen, alle Vermerke, die sich auf diese Lizenz und das Fehlen einer Garantie beziehen,

unverändert lassen und desweiteren allen anderen Empfängern des Programms zusammen mit dem Programm eine Kopie dieser Lizenz zukommen lassen.

Sie dürfen für den eigentlichen Kopiervorgang eine Gebühr verlangen. Wenn Sie es wünschen, dürfen Sie auch gegen Entgelt eine Garantie für das Programm anbieten.

3. Sie dürfen Ihre Kopie(n) des Programms oder eines Teils davon verändern, wodurch ein auf dem Programm basierendes Werk entsteht; Sie dürfen derartige Bearbeitungen unter den Bestimmungen von Paragraph 1 vervielfältigen und verbreiten, vorausgesetzt, dass zusätzlich alle folgenden Bedingungen erfüllt werden:

   a. Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung hinweist.

   b. Sie müssen dafür sorgen, dass jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von dem Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird.

   c. Wenn das veränderte Programm normalerweise bei der Ausführung interaktiv Kommandos einliest, müssen Sie dafür sorgen, dass es, wenn es auf dem üblichsten Wege für solche interaktive Nutzung gestartet wird, eine Meldung ausgibt oder ausdruckt, die einen geeigneten Copyright-Vermerk enthält sowie einen Hinweis, dass es keine Gewährleistung gibt (oder anderenfalls, dass Sie Garantie leisten), und dass die Benutzer das Programm unter diesen Bedingungen weiter verbreiten dürfen. Auch muss der Benutzer darauf hingewiesen werden, wie er eine Kopie dieser Lizenz ansehen kann. (Ausnahme: Wenn das Programm selbst interaktiv arbeitet, aber normalerweise keine derartige Meldung ausgibt, muss Ihr auf dem Programm basierendes Werk auch keine solche Meldung ausgeben).

   Diese Anforderungen betreffen das veränderte Werk als Ganzes. Wenn identifizierbare Abschnitte des Werkes nicht von dem Programm abgeleitet sind und vernünftigerweise selbst als unabhängige und eigenständige Werke betrachtet werden können, dann erstrecken sich diese Lizenz und Ihre Bedingungen nicht auf diese Abschnitte, wenn sie als eigenständige Werke verbreitet werden. Wenn Sie jedoch dieselben Abschnitte als Teil eines Ganzen verbreiten, dass ein auf dem Programm basierendes Werk darstellt, dann muss die Verbreitung des Ganzen nach den Bedingungen dieser Lizenz erfolgen, deren Bedingungen für weitere Lizenznehmer somit auf die Gesamtheit ausgedehnt werden - und damit auf jeden einzelnen Teil, unabhängig vom jeweiligen Autor.

   Somit ist es nicht die Absicht dieses Abschnittes, Rechte für Werke in Anspruch zu nehmen oder zu beschneiden, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Werken, die auf dem Programm basieren oder unter seiner auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

   Ferner bringt ein einfaches Zusammenstellen eines anderen Werkes, das nicht auf dem Programm basiert, zusammen mit dem Programm oder einem auf dem Programm basierenden Werk auf ein- und demselben Speicher- oder Vertriebsmedium das andere Werk nicht in den Anwendungsbereich dieser Lizenz.

4. Sie dürfen das Programm (oder ein darauf basierendes Werk gemäß Paragraph 2) als Objectcode oder in ausführbarer Form unter den Bedingungen von Paragraph 1 und 2 vervielfältigen und verbreiten - vorausgesetzt, dass Sie außerdem eine der folgenden Leistungen erbringen:

   a. Liefern Sie das Programm zusammen mit dem vollständigen zugehörigen maschinenlesbaren Quelltext auf einem für den Datenaustausch üblichen Medium aus, wobei die Verteilung unter den Bedingungen der Paragraphen 1 und 2 erfolgen muß. Oder:

   b. Liefern Sie das Programm zusammen mit einem mindestens drei Jahre lang gültigen schriftlichen Angebot aus, jedem Dritten eine vollständige maschinenlesbare Kopie des Quelltextes zur Verfügung zu stellen - zu nicht höheren Kosten als denen, die durch den physikalischen Kopiervorgang anfallen -, wobei der Quelltext unter den Bedingungen der Paragraphen 1 und 2 auf einem für den Datenaustausch üblichen Medium weitergegeben wird. Oder:

   c. Liefern Sie das Programm zusammen mit dem schriftlichen An- gebot der Zurverfügungstellung des Quelltextes aus, das Sie selbst erhalten haben. (Diese Alternative ist nur für nicht-kommerzielle Verbreitung zulässig und nur, wenn Sie das Programm als Objectcode oder in ausführbarer Form mit einem entsprechenden Angebot gemäß Absatz b erhalten haben.)

   Unter dem Quelltext eines Werkes wird diejenige Form des Werkes verstanden, die für Bearbeitungen vorzugsweise verwendet wird. Für ein ausführbares Programm bedeutet "der komplette Quell- text": Der Quelltext aller im Programm enthaltenen Module ein- schließlich aller zugehörigen Modulschnittstellen-Definitions- dateien sowie der zur Compilation und Installation verwendeten Skripte. Als besondere Ausnahme jedoch braucht der verteilte Quelltext nichts von dem zu enthalten, was üblicherweise (entweder als Quelltext oder in binärer Form) zusammen mit den Hauptkomponenten des Betriebssystems (Kernel, Compiler usw.) geliefert wird, unter dem das Programm läuft - es sei denn, diese Komponente selbst gehört zum ausführbaren Programm.

   Wenn die Verbreitung eines ausführbaren Programms oder des Objectcodes dadurch erfolgt, dass der Kopierzugriff auf eine dafür vorgesehene Stelle gewährt wird, so gilt die Gewährung eines gleichwertigen Zugriffs auf den Quelltext als Verbreitung des Quelltextes, auch wenn Dritte nicht dazu gezwungen sind, den Quelltext zusammen mit dem Objectcode zu kopieren.

5. Sie dürfen das Programm nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz. Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.

6. Sie sind nicht verpflichtet, diese Lizenz anzunehmen, da Sie sie nicht unterzeichnet haben. Jedoch gibt Ihnen nichts anderes die Erlaubnis, das Programm oder von ihm abgeleitete Werke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie diese Lizenz nicht anerkennen. Indem Sie das Programm (oder ein darauf basierendes Werk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit dieser Lizenz und mit allen Ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung des Programms oder eines darauf basierenden Werkes.

7. Jedesmal, wenn Sie das Programm (oder ein auf dem Programm basierendes Werk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Programm entsprechend den hier festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern. Sie dürfen keine weiteren Einschränkungen der Durchsetzung der hierin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

8. Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patent fragen begrenzt) Bedingungen (durch Gerichtsbeschluß, Vergleich oder anderweitig) auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, das Programm unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu ver- breiten, dann dürfen Sie als Folge das Programm überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die gebührenfreie Weiterverbreitung des Programms durch diejenigen erlaubt, die das Programm direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, sowohl das Patentrecht als auch diese Lizenz zu befolgen, darin, ganz auf die Verbreitung des Programms zu verzichten.

   Sollte sich ein Teil dieses Paragraphen als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Paragraph seinem Sinne nach angewandt werden; im übrigen soll dieser Paragraph als Ganzes gelten.

   Zweck dieses Paragraphen ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Paragraph hat einzig den Zweck, die Integrität des Verbreitungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zu dem großen Angebot der mit diesem System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet; es liegt am Autor/Geber, zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer hat auf diese Entscheidung keinen Einfluss.

   Dieser Paragraph ist dazu gedacht, deutlich klarzustellen, was als Konsequenz aus dem Rest dieser Lizenz betrachtet wird.

9. Wenn die Verbreitung und/oder die Benutzung des Programms in be- stimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnittstellen eingeschränkt ist, kann der Urheberrechts- inhaber, der das Programm unter diese Lizenz gestellt hat, eine explizite geographische Beschränkung der Verbreitung angeben, in der diese Staaten ausgeschlossen werden, so dass die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht ausge- schlossen sind. In einem solchen Fall beinhaltet diese Lizenz die Beschränkung, als wäre sie in diesem Text niedergeschrieben.

10. Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der General Public License veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

    Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, dass es dieser Lizenz in einer bestimmten Versionsnummer oder "jeder späteren Version" ("any later version") unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der Free Software Foundation veröffentlicht wurde.

11. Wenn Sie den Wunsch haben, Teile des Programms in anderen freien Programmen zu verwenden, deren Bedingungen für die Verbreitung anders sind, schreiben Sie an den Autor, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der Free Software Foundation steht, schreiben Sie an die Free Software Foundation; wir machen zu diesem Zweck gelegentlich Ausnahmen. Unsere Entscheidung wird von den beiden Zielen geleitet werden, zum einen den freien Status aller von unserer freien Software abgeleiteten Werke zu erhalten und zum anderen das gemeinschaftliche Nutzen und Wiederverwenden von Software im allgemeinen zu fördern

    Keine Gewährleistung

12. Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte das Programm so zur Verfügung, "wie es ist", ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich - aber nicht begrenzt auf - Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

13. In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich - aber nicht beschränkt auf - Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Copyright-Inhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

Ende der Bedingungen

Anhang: Wie Sie diese Bedingungen auf Ihre neuen Programme anwendbar machen

Wenn Sie ein neues Programm entwickeln und wollen, dass es von größtmöglichem Nutzen für die Allgemeinheit ist, dann erreichen Sie das am besten, indem Sie es zu freier Software machen, die jeder unter diesen Bestimmungen weiterverbreiten und verändern kann.

Um dies zu erreichen, fügen Sie die folgenden Anmerkungen zu Ihrem Programm hinzu. Am sichersten ist es, sie an den Anfang einer jeden Quelldatei zu stellen, um den Gewährleistungsausschluß möglichst deutlich darzustellen; außerdem sollte jede Datei mindestens eine "Copyright"-Zeile besitzen sowie einen kurzen Hinweis darauf, wo die vollständige Lizenz gefunden werden kann.

```
eine Zeile mit dem Programmnamen und einer kurzen Beschreibung
Copyright (C) yyyy  Name des Autors

This Programm ist free Software; you can redistribute it und / oder modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License oder
(at your option) any later version.

This Programm ist distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; ohne even the implied warranty of
MERCHANTABILITY oder FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License für mehr details.

You should have received a copy of the GNU General Public License
along mit this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

Also add information on how to contact you von electronic und paper mail.

```
eine Zeile mit dem Programmnamen und einer kurzen Beschreibung
Copyright (C) yyyy  Name des Autors

    Dieses Programm ist freie Software. Sie können es unter
    den Bedingungen der GNU General Public License, wie von der
    Free Software Foundation herausgegeben, weitergeben und/oder
    modifizieren, entweder unter Version 2 der Lizenz oder (wenn
    Sie es wünschen) jeder späteren Version.

    Die Veröffentlichung dieses Programms erfolgt in der
    Hoffnung, dass es Ihnen von Nutzen sein wird, aber OHNE JEDE
    GEWÄHRLEISTUNG - sogar ohne die implizite Gewährleistung
    der MARKTREIFE oder der EIGNUNG FÜR EINEN BESTIMMTEN ZWECK.
    Details finden Sie in der GNU General Public License.

    Sie sollten eine Kopie der GNU General Public License zusammen
    mit diesem Programm erhalten haben. Falls nicht, schreiben Sie
    an die Free Software Foundation, Inc., 675 Mass Ave, Cambridge,
    MA 02139, USA.
```

Wenn Ihr Programm interaktiv ist, sorgen Sie dafür, dass es nach dem Start einen kurzen Vermerk ausgibt:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes mit ABSOLUTELY NO WARRANTY; für details type `show w'.
Das ist free Software und you are welcome to redistribute it
under certain conditions; type `show c' für details.
```

```
    Gnomovision Version 69, Copyright (C) 19[jj]  [Name des Autors]
    Für Gnomovision besteht KEINERLEI GARANTIE; geben Sie `show w'
    für Details ein. Gnomovision ist freie Software, die Sie unter
    bestimmten Bedingungen weitergeben dürfen; geben Sie `show c'
    für Details ein.
```

Die hypothetischen Kommandos `show w' und `show c' sollten die entsprechenden Teile der GNU-GPL anzeigen. Natürlich können die von Ihnen verwendeten Kommandos anders heißen als `show w' und `show c'; es könnten auch Mausklicks oder Menüpunkte sein - was immer am besten in Ihr Programm passt.

Soweit vorhanden, sollten Sie auch Ihren Arbeitgeber (wenn Sie als Programmierer arbeiten) oder Ihre Schule einen Copyright-Verzicht für das Programm unterschreiben lassen. Hier ein Beispiel; ändern Sie bitte die Namen:

```
Yoyodyne, Inc., herefrom disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

```
Die Yoyodyne GmbH erhebt keinerlei urheberrechtlichen Anspruch auf das
Programm "Gnomovision" (einem Schrittmacher für Compiler),
geschrieben von James Hacker.

Unterschrift von Ty Coon, 1 April 1989
Ty Coon, Vizepräsident
```

Diese General Public License gestattet nicht die Einbindung des Programms in proprietäre Programme. Ist Ihr Programm eine Funktionsbibliothek, so kann es sinnvoller sein, das Linken proprietärer Programme mit dieser Bibliothek zu gestatten. Wenn Sie dies tun wollen, sollten Sie die GNU Library General Public License anstelle dieser Lizenz verwenden.

```
 Erstellt im Auftrag der S.u.S.E. GmbH [suse@suse.de]
von Katja Lachmann Übersetzungen [na194@fim.uni-erlangen.de],
überarbeitet von Peter Gerwinski [peter.gerwinski@uni-essen.de] (31. Oktober 1996)
Diese Übersetzung wird mit der Absicht angeboten, das Verständnis der
GNU General Public License (GNU-GPL) zu erleichtern. Es handelt sich jedoch
nicht um eine offizielle oder im rechtlichen Sinne anerkannte Übersetzung.
Die Free Software Foundation (FSF) ist nicht der Herausgeber dieser Übersetzung,
und sie hat diese Übersetzung auch nicht als rechtskräftigen Ersatz für die
Original-GNU-GPL anerkannt. Da die Übersetzung nicht sorgfältig von Anwälten
überprüft wurde, können die Übersetzer nicht garantieren, dass die Übersetzung
die rechtlichen Aussagen der GNU-GPL exakt wiedergibt. Wenn Sie sichergehen
wollen, dass von Ihnen geplante Aktivitäten im Sinne der GNU-GPL gestattet sind,
halten Sie sich bitte an die englischsprachige Originalversion.
Die Free Software Foundation möchte Sie darum bitten, diese Übersetzung
nicht als offizielle Lizenzbedingungen für von Ihnen geschriebene Programme
zu verwenden. Bitte benutzen Sie hierfür stattdessen die von der
Free Software Foundation herausgegebene englischsprachige Originalversion.
```

Anhang I. GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, Februar 1999

Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige Originalversion!

Vorwort

Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Software weiterzugeben und zu verändern. Im Gegensatz dazu sollen Ihnen die GNU General Public Licenses, die Allgemeinen Öffentlichen GNU-Lizenzen, ebendiese Freiheit des Weitergebens und Veränderns garantieren und somit sicherstellen, dass diese Software für alle Benutzer frei ist.

Diese Lizenz, die Kleine Allgemeine Öffentliche Lizenz (Lesser General Public License), gilt für einige besonders bezeichnete Software-Pakete - typischerweise Programmbibliotheken - von der Free Software Foundation und anderen Autoren, die beschließen, diese Lizenz zu verwenden. Auch Sie können sie verwenden; wir empfehlen aber, vorher gründlich darüber nachzudenken, ob diese Lizenz (LGPL) oder aber die gewöhnliche Allgemeine Öffentliche Lizenz (GPL) die bessere Strategie zur Anwendung im jeweiligen speziellen Fall ist. Dabei bieten Ihnen die untenstehenden Erläuterungen eine Grundlage für Ihre Entscheidung.

Die Bezeichnung ,,freie`` Software bezieht sich auf Freiheit der Nutzung, nicht auf den Preis. Unsere Allgemeinen Öffentlichen Lizenzen sollen sicherstellen, dass Sie die Freiheit haben, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), dass Sie die Software im Quelltext erhalten oder den Quelltext auf Wunsch bekommen können, dass Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen, und dass Sie darüber informiert sind, dass Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir Einschränkungen machen, die es jedem, der die Software weitergibt, verbieten, Ihnen diese Rechte zu verweigern oder Sie zum Verzicht auf diese Rechte aufzufordern. Aus diesen Einschränkungen ergeben sich bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Bibliothek verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die wir Ihnen eingeräumt haben, wenn Sie - kostenlos oder gegen Bezahlung - Kopien der Bibliothek verbreiten. Sie müssen sicherstellen, dass auch die Empfänger den Quelltext erhalten bzw. erhalten können. Wenn Sie einen anderen Code mit der Bibliothek linken, müssen Sie den Empfängern die vollständigen Objekt-Dateien zukommen lassen, so dass sie selbst diesen Code mit der Bibliothek neu linken können, auch nachdem sie Veränderungen an der Bibliothek vorgenommen und sie neu compiliert haben. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie Ihre Rechte kennen.

Wir schützen Ihre Rechte in zwei Schritten: (1) Wir stellen die Bibliothek unter ein Urheberrecht (Copyright), und (2) wir bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Bibliothek zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um jeden, der die Software weitergibt, zu schützen, wollen wir darüber hinaus vollkommen klarstellen, dass für diese freie Bibliothek keinerlei Garantie besteht. Auch sollten, falls die Software von jemand anderem modifiziert und weitergegeben wird, die Empfänger wissen, dass sie nicht das Original erhalten haben, damit irgendwelche von anderen verursachte Probleme nicht den Ruf des ursprünglichen Autors schädigen.

Schließlich und endlich stellen Software-Patente für die Existenz jedes freien Programms eine ständige Bedrohung dar. Wir möchten sicherstellen, dass keine Firma den Benutzern eines freien Programms Einschränkungen auferlegen kann, indem sie von einem Patentinhaber eine die freie Nutzung einschränkende Lizenz erwirbt. Deshalb bestehen wir darauf, dass jegliche für eine Version der Bibliothek erworbene Patentlizenz mit der in dieser Lizenz (also der LGPL) im einzelnen angegebenen Nutzungsfreiheit voll vereinbar sein muß.

Die meiste GNU-Software einschließlich einiger Bibliotheken fällt unter die gewöhnliche Allgemeine Öffentliche GNU-Lizenz (GNU-GPL). Die vorliegende Lizenz, also die GNU-LGPL, gilt für gewisse näher bezeichnete Bibliotheken. Sie unterscheidet sich wesentlich von der gewöhnlichen Allgemeinen Öffentlichen Lizenz (GNU-GPL). Wir benutzen diese Lizenz für gewisse Bibliotheken, um das Linken (d.h. die Verknüpfung von Bibliotheken und anderen Programmteilen zu einem lauffähigen Programm - Anmerkung der Übersetzer) von Programmen, die nicht frei sind, mit diesen Bibliotheken zu gestatten.

Wenn ein Programm mit einer Bibliothek gelinkt wurde, sei es nun statisch oder dynamisch, so ist die Kombination der beiden, rechtlich gesehen, ein ,,kombiniertes Datenwerk``, also eine abgeleitete Version der Orginal-Bibliothek. Die gewöhnliche GPL erlaubt ein solches Linken nur dann, wenn die ganze Kombination die Kriterien für freie Software erfüllt. Die LGPL erlaubt

dagegen weniger strenge Kriterien für das Linken von irgendeiner anderen Software mit der Bibliothek.

Wir nennen diese Lizenz die "Kleine" Allgemeine Öffentliche Lizenz **Lesser** General Public License weil sie weniger *Less* dazu beiträgt, die Freiheit des Benutzers zu schützen, als die gewöhnliche Allgemeine Öffentliche Lizenz (GPL). Sie verschafft auch anderen Entwicklern freier Software ein "Weniger" an Vorteil gegenüber konkurrierenden nichtfreien Programmen. Diese Nachteile sind ein Grund dafür, dass wir die gewöhnliche GPL für viele Bibliotheken benutzen. Die "kleine" Lizenz (LGPL) bietet aber unter bestimmten besonderen Umständen doch Vorteile.

So kann, wenn auch nur bei seltenen Gelegenheiten, eine besondere Notwendigkeit bestehen, einen Anreiz zur möglichst weitgehenden Benutzung einer bestimmten Bibliothek zu schaffen, so dass diese dann ein De-facto-Standard wird. Um dies zu erreichen, müssen nichtfreie Programme die Bibliothek benutzen dürfen. Ein häufigerer Fall ist der, dass eine freie Bibliothek dasselbe leistet wie weithin benutzte nichtfreie Bibliotheken. In diesem Falle bringt es wenig Nutzen, die freie Bibliothek allein auf freie Software zu beschränken, und dann benutzen wir eben die LGPL.

In anderen Fällen ermöglicht die Erlaubnis zur Benutzung einer speziellen Bibliothek in nichtfreien Programmen viel mehr Leuten, eine umfangreiche Sammlung freier Software zu nutzen. So ermöglicht z.B. die Erlaubnis zur Benutzung der GNU-C-Bibliothek in nichtfreien Programmen einer viel größeren Zahl von Leuten, das ganze GNU-Betriebssystem ebenso wie seine Variante, das Betriebssystem GNU/Linux, zu benutzen.

Obwohl die LGPL die Freiheit des Benutzers weniger schützt, stellt sie doch sicher, dass der Benutzer eines Programms, das mit der Bibliothek gelinkt wurde, die Freiheit und die erforderlichen Mittel hat, das Programm unter Benutzung einer abgeänderten Version der Bibliothek zu betreiben.

Die genauen Bedingungen für das Kopieren, Weitergeben und Abändern finden Sie im nachstehenden Kapitel. Achten Sie genau auf den Unterschied zwischen "work Basiert auf the library", d.h. "Datenwerk, das auf der Bibliothek basiert" und "work that uses the library" d.h. "Datenwerk, das die Bibliothek benutzt". Ersteres enthält Code, der von der Bibliothek abgeleitet ist, während letzteres lediglich mit der Bibliothek kombiniert werden muß, um betriebsfähig zu sein.

GNU LESSER GENERAL PUBLIC LICENSE Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung

1. Diese Lizenz gilt für jedes Programm und jedes andere Datenwerk, in dem ein entsprechender Vermerk des Copyright-Inhabers oder eines anderen dazu Befugten darauf hinweist, dass das Datenwerk unter den Bestimmungen dieser Lesser General Public License (im weiteren auch als "diese Lizenz" bezeichnet) verbreitet werden darf. Jeder Lizenznehmer wird hierin einfach als "Sie" angesprochen.

   Eine "Bibliothek" bedeutet eine Zusammenstellung von Software-Funktionen und/oder Daten, die so vorbereitet ist, dass sie sich bequem mit Anwendungsprogrammen (welche einige dieser Funktionen und Daten benutzen) zum Bilden von ausführbaren Programmen linken (d.h. verbinden, kombinieren) läßt.

   Der Begriff "Bibliothek" bezieht sich im weiteren immer nur auf solche Software-Bibliotheken und solche Datenwerke, die unter diesen Bedingungen der Lesser-GPL-Lizenz verbreitet worden sind. Ein "auf der Bibliothek basierendes Datenwerk" bezeichnet die betreffende Bibliothek selbst sowie jegliche davon abgeleitete Bearbeitung im urheberrechtlichen Sinne, also ein Datenwerk, welches die Bibliothek oder einen Teil davon, sei es unverändert oder verändert und/oder direkt in eine andere Sprache übersetzt, enthält. (Im folgenden wird die Übersetzung ohne Einschränkung als "Bearbeitung" eingestuft.)

   Unter dem "Quelltext" eines Datenwerks ist seine für das Vornehmen von Veränderungen bevorzugte Form zu verstehen. Für eine Bibliothek bedeutet "vollständiger Quelltext" den gesamten Quelltext für alle in ihr enthaltenen Bestandteile, für jegliche zu ihr gehörenden Dateien zur Definition von Schnittstellen und schließlich auch für die Skripte, die zur Steuerung der Compilation und Installation der Bibliothek benutzt werden.

   Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von dieser Lizenz nicht berührt; sie fallen nicht in Ihren Anwendungsbereich. Das Ausführen eines Programms unter Benutzung der Bibliothek wird nicht eingeschränkt, und die Ausgaben des Programms unterliegen dieser Lizenz nur dann, wenn der Inhalt ein auf der Bibliothek basierendes Datenwerk darstellt (unabhängig davon, dass die Bibliothek in einem Werkzeug zum Schreiben dieses Programms benutzt wurde). Ob dies zutrifft, hängt davon ab, was die Bibliothek bewirkt und was das Programm, das die Bibliothek nutzt, bewirkt.

2. Sie dürfen auf beliebigen Medien unveränderte Kopien des vollständigen Quelltextes des Programms so, wie sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, dass Sie mit jeder Kopie deutlich erkennbar und in angemessener Form einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluss veröffentlichen, alle Vermerke, die sich auf diese Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen und zusammen mit der Bibliothek jeweils eine Kopie dieser Lizenz weitergeben.

   Sie dürfen für den eigentlichen Kopier- und Versandvorgang eine Gebühr verlangen. Wenn Sie es wünschen, dürfen Sie auch gegen Entgelt eine Garantie anbieten.

3. Sie dürfen Ihre Kopie(n) der Bibliothek oder irgendeines Teils davon verändern, wodurch ein auf der Bibliothek basierendes Datenwerk entsteht, und Sie dürfen derartige Bearbeitungen unter den Bestimmungen von Paragraph 1 vervielfältigen und verbreiten, vorausgesetzt, dass zusätzlich alle im folgenden genannten Bedingungen erfüllt werden:

a. Das Bearbeitungsergebnis muss selbst wieder eine Software-Bibliothek sein.

b. Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung der Dateien hinweist und das Datum jeder Änderung angibt.

c. Sie müssen dafür sorgen, dass das Datenwerk als Ganzes Dritten unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird.

d. Wenn sich eine Funktionseinheit der bearbeiteten Bibliothek auf eine Funktion oder Datentabelle stützt, die von einem die Funktionseinheit nutzenden Anwendungsprogramm bereitgestellt werden muß, ohne dass sie als Argument übergeben werden muß, wenn die Funktionseinheit angesprochen wird, dann müssen Sie sich nach bestem Wissen und Gewissen bemühen, sicherzustellen, dass die betreffende Funktionseinheit auch dann noch funktioniert, wenn die Anwendung eine solche Funktion oder Datentabelle nicht bietet, und dass sie den sinnvoll bleibenden Teil Ihres Bestimmungszwecks noch ausführt.

   (So hat z.B. eine Funktion zum Berechnen von Quadratwurzeln einen von der Anwendung unabhängigen genau definierten Zweck. Deshalb verlangt §2 Absatz d, dass jede von der Anwendung bereitgestellte Funktion oder von dieser Funktion benutzte Tabelle optional sein muß: Auch wenn die Anwendung sie nicht bereitstellt, muss die Quadratwurzelfunktion trotzdem noch Quadratwurzeln berechnen).

Diese Anforderungen gelten für das bearbeitete Datenwerk als Ganzes. Wenn identifizierbare Teile davon nicht von der Bibliothek stammen und vernünftigerweise als unabhängige und gesonderte Datenwerke für sich selbst zu betrachten sind, dann gelten diese Lizenz und Ihre Bedingungen nicht für die betreffenden Teile, wenn Sie diese als gesonderte Datenwerke weitergeben. Wenn Sie jedoch dieselben Teile als Teil eines Ganzen weitergeben, dass ein auf der Bibliothek basierendes Datenwerk darstellt, dann muss die Weitergabe dieses Ganzen nach den Bedingungen dieser Lizenz erfolgen, deren Bedingungen für weitere Lizenznehmer somit auf das gesamte Ganze ausgedehnt werden - und somit auf jeden einzelnen Teil, unabhängig vom jeweiligen Autor.

Somit ist es nicht die Absicht dieses Abschnittes, Rechte für Datenwerke in Anspruch zu nehmen oder Ihnen Rechte für Datenwerke streitig zu machen, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Datenwerken, die auf der Bibliothek basieren oder unter Ihrer auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

Ferner bringt auch dass einfache Zusammenlegen eines anderen Datenwerkes, das nicht auf der Bibliothek basiert, mit der Bibliothek oder mit einem auf der Bibliothek basierenden Datenwerk auf ein- und demselben Speicher- oder Vertriebsmedium dieses andere Datenwerk nicht in den Anwendungsbereich dieser Lizenz.

4. Sie können sich für die Anwendung der Bedingungen der gewöhnlichen Allgemeinen Öffentlichen GNU-Lizenz (GNU-GPL) statt dieser Lizenz auf eine gegebene Kopie der Bibliothek entscheiden. Um dies zu tun, müssen Sie alle Eintragungen, die sich auf diese Lizenz beziehen, ändern, so dass sie nun für die gewöhnliche GNU-GPL, Version 2, statt für diese Lizenz (LGPL) gelten. (Wenn eine neuere Version als Version 2 der gewöhnlichen GNU-GPL erschienen ist, können Sie diese angeben, wenn Sie das wünschen.) Nehmen Sie keine anderen Veränderungen in diesen Eintragungen vor.

   Wenn diese Veränderung in einer gegebenen Kopie einmal vorgenommen ist, dann ist sie für diese Kopie nicht mehr zurücknehmbar, und somit gilt dann die gewöhnliche GNU-GPL für alle nachfolgenden Kopien und abgeleiteten Datenwerke, die von dieser Kopie gemacht worden sind.

   Diese Option ist nützlich, wenn Sie einen Teil des Codes der Bibliothek in ein Programm kopieren wollen, das keine Bibliothek ist.

5. Sie können die Bibliothek (oder einen Teil oder eine Ableitung von ihr, gemäß Paragraph 2) in Objektcode-Form oder in ausführbarer Form unter den Bedingungen der obigen Paragraphen 1 und 2 kopieren und weitergeben, sofern Sie den vollständigen entsprechenden maschinenlesbaren Quelltext beifügen, der unter den Bedingungen der obigen Paragraphen 1 und 2 auf einem Medium weitergegeben werden muß, das üblicherweise zum Austausch von Software benutzt wird.

   Wenn die Weitergabe von Objektcode durch das Angebot eines Zugangs zum Kopienabruf von einem angegebenen Ort erfolgt, dann erfüllt das Angebot eines gleichwertigen Zugangs zum Kopieren des Quelltextes von demselben Ort die Anforderung, auch den Quelltext weiterzugeben, obwohl Dritte nicht verplichtet sind, den Quelltext zusammen mit dem Objektcode zu kopieren.

6. Ein Programm, das nichts von irgendeinem Teil der Bibliothek Abgeleitetes enthält, aber darauf ausgelegt ist, mit der Bibliothek zusammenzuarbeiten, indem es mit ihr compiliert oder gelinkt wird, nennt man ein "Datenwerk, das die Bibliothek nutzt". Solch ein Datenwerk, für sich allein genommen, ist kein von der Bibliothek abgelei tes Datenwerk und fällt daher nicht unter diese Lizenz.

   Wird jedoch ein "Datenwerk, das die Bibliothek nutzt", mit der Bibliothek gelinkt, so entsteht ein ausführbares Programm, dass ein von der Bibliothek abgeleitetes Datenwerk (weil es Teile der Bibliothek enthält) und kein "Datenwerk, das die Bibliothek nutzt" ist. Das ausführbare Programm fällt daher unter diese Lizenz. Paragraph 6 gibt die Bedingungen für die Weitergabe solcher ausführbarer Programme an.

Wenn ein "Datenwerk, das die Bibliothek nutzt", Material aus einer Header-Datei verwendet, die Teil der Bibliothek ist, dann kann der Objektcode für das Datenwerk ein von der Bibliothek abgeleitetes Datenwerk sein, selbst wenn der Quelltext dies nicht ist. Ob dies jeweils zutrifft, ist besonders dann von Bedeutung, wenn das Datenwerk ohne die Bibliothek gelinkt werden kann oder wenn das Datenwerk selbst eine Bibliothek ist. Die genaue Grenze, von der an dies zutrifft, ist rechtlich nicht genau definiert.

Wenn solch eine Objektdatei nur numerische Parameter, Daten- struktur-Layouts und Zugriffsfunktionen sowie kleine Makros und kleine Inlinefunktionen (zehn Zeilen lang oder kürzer) benutzt, dann unterliegt die Benutzung der Objektdatei keinen Beschränkungen, ohne Rücksicht darauf, ob es rechtlich gesehen ein abgeleitetes Datenwerk ist. (Ausführbare Programme, welche diesen Objektcode plus Teile der Bibliothek enthalten, fallen jedoch weiterhin unter die Bestimmungen von Paragraph 6).

Ansonsten können Sie, wenn das Datenwerk ein von der Bibliothek abgeleitetes ist, den Objektcode für das Datenwerk unter den Bedingungen von Paragraph 6 weitergeben. Alle ausführbaren Programme, welche dieses Datenwerk enthalten, fallen ebenfalls unter Paragraph 6, gleichgültig, ob sie direkt mit der Bibliothek selbst gelinkt sind oder nicht.

7. Als Ausnahme von den Bestimmungen der vorstehenden fünf Paragraphen dürfen Sie auch ein "Datenwerk, das die Bibliothek nutzt", mit der Bibliothek kombinieren oder linken, um ein Datenwerk zu erzeugen, das Teile der Bibliothek enthält, und dieses unter Bedingungen Ihrer eigenen Wahl weitergeben, sofern diese Bedingungen Bearbeitungen für den eigenen Gebrauch des Empfängers und ein Rückbilden ("reverse engineering") zum Beheben von Mängeln solcher Bearbeitungen gestatten.

   Sie müssen bei jeder Kopie des Datenwerks deutlich erkennbar angeben, dass die Bibliothek darin genutzt wird und dass die Bibliothek und Ihre Benutzung durch die Lizenz abgedeckt sind. Sie müssen eine Kopie dieser Lizenz mitgeben. Wenn das Datenwerk bei seiner Ausführung Copyright-Vermerke anzeigt, müssen Sie den Copyright-Vermerk für die Bibliothek mit anzeigen lassen und dem Benutzer einen Hinweis geben, der ihn zu einer Kopie dieser Lizenz führt. Ferner müssen Sie eines der nachfolgend genannten fünf Dinge tun:

   a. Liefern Sie das Datenwerk zusammen mit dem vollständigen zugehörigen maschinenlesbaren Quelltext der Bibliothek aus, und zwar einschließlich jeglicher in dem Datenwerk angewandter Änderungen (wobei dessen Weitergabe gemäß den Bedingungen der Paragraphen 1 und 2 erfolgen muß); und wenn das Datenwerk ein ausführbares, mit der Bibliothek gelinktes Programm ist, dann liefern Sie es zusammen mit dem vollständigen maschinenlesbaren "Datenwerk, das die Bibliothek nutzt, in Form von Objektcode und/oder Quelltext, so dass der Benutzer die Bibliothek verändern und dann erneut linken kann, um ein verändertes ausführbares Programm zu erzeugen, das die veränderte Bibliothek enthält. (Es versteht sich, dass der Benutzer, der die Inhalte von Definitionsdateien in der veränderten Bibliothek verändert, nicht notwendigerweise in der Lage sein wird, die Anwendung neu zu compilieren, um die veränderten Definitionen zu benutzen.)

   b. Benutzen Sie einen geeigneten ,,shared-library-Mechanismus`` zum Linken mit der Bibliothek. Geeignet ist ein solcher Mechanismus, der erstens während der Laufzeit eine im Computersystem des Benutzers bereits vorhandene Kopie der Bibliothek benutzt, anstatt Bibliotheksfunktionen in das ausführbare Programm zu kopieren, und der zweitens auch mit einer veränderten Version der Bibliothek, wenn der Benutzer eine solche installiert, richtig funktioniert, solange die veränderte Version schnittstellenkompatibel mit der Version ist, mit der das Datenwerk erstellt wurde.

   c. Liefern Sie das Datenwerk zusammen mit einem mindestens drei Jahre lang gültigen schriftlichen Angebot, demselben Benutzer die oben in Paragraph 6, Absatz (a) genannten Materialien zu Kosten, welche die reinen Weitergabekosten nicht übersteigen, zur Verfügung zu stellen.

   d. Wenn die Weitergabe des Datenwerks dadurch erfolgt, dass die Möglichkeit des Abrufens einer Kopie von einem bestimmten Ort angeboten wird, bieten Sie gleichwertigen Zugang zum Kopieren der oben angegebenen Materialien von dem gleichen Ort an.

   e. Sie vergewissern sich, dass der Benutzer bereits eine Kopie dieser Materialien erhalten hat oder dass Sie diesem Benutzer bereits eine Kopie geschickt haben.

   Für ein ausführbares Programm muss die verlangte Form des "Datenwerks, das die Bibliothek nutzt" alle Daten und Hilfsprogramme mit einschließen, die man braucht, um daraus das ausführbare Programm zu reproduzieren. Doch gilt eine spezielle Ausnahme: Die weiterzugebenden Materialien brauchen nicht alles das zu enthalten, was normalerweise (in Quelltext-Form oder in binärer Form) mit den Hauptbestandteilen (Compiler, Kern usw.) des Betriebssystems, auf denen das ausführbare Programm läuft, weitergegeben wird, es sei denn, das ausführbare Programm gehört selbst zu diesem Hauptbestandteil.

   Es kann vorkommen, dass diese Anforderung im Widerspruch zu Lizenzbeschränkungen anderer, proprietärer Bibliotheken steht, die normalerweise nicht zum Betriebssystem gehören. Ein solcher Widerspruch bedeutet, dass Sie nicht gleichzeitig jene proprietären Bibliotheken und die vorliegende Bibliothek zusammen in einem ausführbaren Programm, das Sie weitergeben, verwenden dürfen.

8. Sie dürfen Bibliotheks-Funktionseinheiten, die ein auf der Bibliothek basierendes Datenwerk darstellen, zusammen mit anderen, nicht unter diese Lizenz fallenden Funktionseinheiten in eine einzelne Bibliothek einbauen und eine solche

kombinierte Bibliothek weitergeben, vorausgesetzt, dass die gesonderte Weitergabe des auf der Bibliothek basierenden Datenwerks einerseits und der anderen Funktionseinheiten andererseits ansonsten gestattet ist, und vorausgesetzt, dass Sie folgende zwei Dinge tun:

a. Geben Sie zusammen mit der kombinierten Bibliothek auch eine Kopie desselben auf der Bibliothek basierenden Datenwerks mit, die nicht mit irgendwelchen anderen Funktionseinheiten kombiniert ist. Dieses Datenwerk muss unter den Bedingungen der obigen Paragraphen weitergegeben werden.

b. Weisen Sie bei der kombinierten Bibliothek an prominenter Stelle auf die Tatsache hin, dass ein Teil davon ein auf der Bibliothek basierendes Datenwerk ist, und erklären Sie, wo man die mitgegebene unkombinierte Form desselben Datenwerks finden kann.

9. Sie dürfen die Bibliothek nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten oder mit ihr linken, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung sowie des Linkens mit der Bibliothek ist unzulässig und beendet automatisch Ihre Rechte unter dieser Lizenz. Doch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese Dritten die Lizenz voll anerkennen und befolgen.

10. Sie sind nicht verpflichtet, diese Lizenz anzunehmen, da Sie diese nicht unterzeichnet haben. Doch gibt Ihnen sonst nichts die Erlaubnis, die Bibliothek oder von ihr abgeleitete Datenwerke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie diese Lizenz nicht annehmen. Indem Sie die Bibliothek (oder ein darauf basierendes Datenwerk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit dieser Lizenz, die Ihnen das erlaubt, mit allen Ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung der Bibliothek oder eines darauf basierenden Datenwerks.

11. Jedesmal, wenn Sie die Bibliothek (oder irgendein auf der Bibliothek basierendes Datenwerk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, die Bibliothek entsprechend den hier festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern und mit ihr zu linken. Sie dürfen keine weiteren Einschränkungen der Ausübung der hierin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

12. Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patentfragen begrenzt) Bedingungen (durch Gerichtsbeschluss, Vergleich oder anderweitig) auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien diese Umstände Sie nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, die Bibliothek unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu verbreiten, dann dürfen Sie als Folge davon die Bibliothek überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die gebührenfreie Weiterverbreitung der Bibliothek durch diejenigen erlaubt, welche die Bibliothek direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, sowohl dem Patentrecht als auch dieser Lizenz zu genügen, darin, ganz auf die Verbreitung der Bibliothek zu verzichten.

    Sollte sich ein Teil dieses Paragraphen als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Paragraph seinem Sinne nach angewandt werden; im übrigen soll dieser Paragraph als Ganzes gelten.

    Zweck dieses Paragraphen ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Paragraph hat vielmehr einzig den Zweck, die Integrität des Verbreitungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zu dem weitreichenden Angebot der durch dieses System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet; es obliegt dem Autor bzw. Geber, zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer jedoch darf darüber nicht entscheiden.

    Dieser Paragraph ist dazu gedacht, deutlich klarzustellen, was als Konsequenz aus den übrigen Bestimmungen dieser Lizenz zu betrachten ist.

13. Wenn die Verbreitung und/oder die Benutzung der Bibliothek in bestimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnittstellen eingeschränkt ist, kann der Urheberrechtsinhaber, der die Bibliothek unter diese Lizenz gestellt hat, eine explizite geographische Beschränkung der Verbreitung angeben, in der diese Staaten ausgeschlossen werden, so dass die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht demgemäß ausgeschlossen sind. In einem solchen Fall beinhaltet diese Lizenz die Beschränkung, als wäre sie in diesem Text niedergeschrieben.

14. Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der Lesser General Public License veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

    Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, dass es dieser Lizenz in einer bestimmten Versionsnummer oder "jeder späteren Version" ("any later version") unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn die Bibliothek keine Lizenz-Versionsnummer angibt, können Sie eine beliebige Version wählen, die jemals von der Free Software Foundation veröffentlicht wurde.

15. Wenn Sie den Wunsch haben, Teile der Bibliothek in anderen freien Programmen zu verwenden, deren Bedingungen für die

Verbreitung anders sind, schreiben Sie an den Autor der Bibliothek, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der Free Software Foundation steht, schreiben Sie an die Free Software Foundation; wir machen zu diesem Zweck gelegentlich Ausnahmen. Unsere Entscheidung wird von den beiden Zielen geleitet werden, zum einen den freien Status aller von unserer freien Software abgeleiteten Datenwerke zu erhalten und zum anderen das gemeinschaftliche Nutzen und Wiederverwenden von Software im allgemeinen zu fördern.

Keine Gewährleistung

16. Da die Bibliothek ohne jegliche Gebühren lizenziert wird, besteht keinerlei Gewährleistung für die Bibliothek, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte die Bibliothek "so, wie sie ist" zur Verfügung, ohne Gewährleistung irgendeiner Art, weder ausdrücklich noch implizit. Dieser Garantieausschluss gilt auch - ohne darauf beschränkt zu sein - für Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungs- fähigkeit der Bibliothek liegt bei Ihnen. Sollte sich die Bibliothek als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur sämtlich bei Ihnen.

17. In keinem Fall, außer wenn dies durch geltendes Recht gefordert wird oder schriftlich zugesichert wurde, ist irgendein Copyright-Inhaber oder irgendein Dritter, der die Bibliothek wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar. Dies gilt auch für jegliche allgemeine oder spezielle Schäden, für Schäden durch Nebenwirkungen oder Folgeschäden, die sich aus der Benutzung oder der Unbenutzbarkeit der Bibliothek ergeben (das gilt insbesondere - ohne darauf beschränkt zu sein - für Datenverluste, das Hineinbringen von Ungenauigkeiten in irgendwelche Daten, für Verluste, die Sie oder Dritte erlitten haben oder für ein Unvermögen der Bibliothek, mit irgendeiner anderen Software zusammenzuarbeiten), und zwar auch dann, wenn ein Copyright-Inhaber oder ein Dritter über die Möglichkeit solcher Schäden informiert worden ist.

Ende der Bedingungen

Anhang: Wie Sie diese Bedingungen auf Ihre eigenen, neuen Bibliotheken anwenden können

Wenn Sie eine neue Bibliothek entwickeln und wünschen, dass sie von größtmöglichem Nutzen für die Allgemeinheit ist, dann empfehlen wir Ihnen, sie zu einer freien Software zu machen, die jedermann weiterverteilen und verändern kann. Dies können sie tun, indem Sie eine Weiterverteilung unter den Bedingungen dieser Lizenz, also der Lesser GPL erlauben (oder - als Alternative - unter den Bedingungen der gewöhnlichen Allgemeinen Öffentlichen GNU-Lizenz, der GPL).

Zur Anwendung dieser Bedingungen fügen Sie zu der Bibliothek die unten angegebenen Vermerke hinzu. Es ist am sichersten, sie an den Start jeder Quelldatei anzufügen, um so am wirksamsten den Garantieausschluß bekannt zu machen; zumindest aber sollte jede Datei die Copyright-Zeile und eine Angabe enthalten, wo die vollständigen Vermerke zu finden sind.

```
eine Zeile mit dem Namen der Bibliothek und einer kurzen  Beschreibung Ihres Zwecks
Copyright (C) yyyy  Name des Autors

This library is free software; you can redistribute it and/or
modify it only under the terms of the GNU Lesser General Public License
as published by the Free Software Foundation; either version 2.1
of the License oder (at your option) any later version.

This library is distributed in the hope that it will be useful,
but ohne ANY WARRANTY; ohne even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General
Public License along with this library; if not, write to the

Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Auf Deutsch:

Bibliothek ist freie Software; Sie dürfen sie unter den Bedingungen
der GNU Lesser General Public License, wie von der Free Software
Foundation veröffentlicht, weiterverteilen und/oder modifizieren;
entweder gemäß Version 2.1 der Lizenz oder (nach Ihrer Option)
jeder späteren Version.

Diese Bibliothek wird in der Hoffnung weiterverbreitet, dass sie
nützlich sein wird, jedoch OHNE IRGENDEINE GARANTIE, auch ohne
die implizierte Garantie der MARKTREIFE oder der VERWENDBARKEIT
FÜR EINEN BESTIMMTEN ZWECK. Mehr Details finden Sie in der
GNU Lesser General Public License.

Sie sollten eine Kopie der GNU Lesser General Public License
zusammen mit dieser Bibliothek erhalten haben; falls nicht,
schreiben Sie an die
Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
```

Fügen Sie auch einen kurzen Hinweis hinzu, wie Sie elektronisch und per Brief erreichbar sind.

```
Soweit vorhanden, sollten Sie auch Ihren Arbeitgeber
(wenn Sie als Programmierer arbeiten) oder Ihre Schule einen
```

```
Copyright-Verzicht für die Bibliothek unterschreiben lassen.
Hier ein Beispiel. Die Namen müssen Sie natürlich ändern.

Yoyodyne, Inc., hereby disclaims all copyright interest
in the library "Frob" (a library for tweaking knobs)
written by James Random Hacker.


Unterschrift von Ty Coon , 1 April 1990
Ty Coon, President of Vice

Auf Deutsch:

DIe Yoyodyne GmbH erhebt keinen urheberechtlichen Anspruch
auf die von James Random Hacker geschriebene Bibliothek "Frob"
(eine Bibliothek für das Zwicken von Knöpfen).


Unterschrift von Ty Coon, 1. April 1990
Ty Coon, Vizepräsident
```

Das war schon alles!

tichwortverzeichnis

Symbole

A

B

C

D

E

F

G

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z