MySQL - Datenbanken

Autor: Alexander Fischer (*selflinux@tbanus.org*)
Layout: Torsten Hemm (*T.Hemm@gmx.de*)

Inhaltsverzeichnis

1 Einleitung

Vorweg ein "Herzliches Willkommen" bei der Dokumentation von *MySQL*. Da Sie die Einführung lesen, gehe ich davon aus, daß Sie ein allgemeines Interesse an relationalen Datenbanken und insbesondere einen gesteigerten Informationsbedarf an *MySQL* haben.

Die erste Datenbank entstand schon weit früher, als es die erste Rechenmaschine gab. Erstaunt? Nun ja, zugegeben, es waren an sich keine Datenbanken, wie wir sie heute kennen, aber immerhin hatten sie nahezu die gleichen Aufgaben. Primäres Ziel ist heute wie früher, Datenbestände aufzubewahren, sie zu speichern, sie in eine bestimmte Form bringen und sie bei Bedarf jederzeit wieder abrufen und nutzen zu können.

Da wir ja schon bei geschichtlichen Abläufen sind, ist es nicht vermessen, wenn ich behaupte, dass *MySQL* zu den bedeutenden relationalen Datenbanksystemen gezählt werden kann. Es ist trotz aller Einschränkungen, sehr leistungsfähig, ausreichend sicher und stabil und führt sozusagen einen Quasi-Standard bei der Verwendung als Web-Datenbanksystem ein.

2 Folgende Voraussetzung sollten Sie erfüllen

* Sie besitzen einen Computer
* Sie wissen was Linux ist, und können das System starten
* Sie wissen auch, dass WWW keine Automarke ist und -surfen- so ab und zu durch die Weiten des Web's.
* Sie sind eventuell Inhaber einer Website und möchten diese mit einer Datenbank ein wenig aufpäppeln.
* Sie sind eventuell Programmierer und möchten Ihr Programm um eine Datenbanklösung erweitern
* Sie gehören nicht zu den Enthusiasten, dessen größtes Glück auf Erden darin besteht, daß Sie Tag und Nacht vor einem eckigen Kasten verbringen, seltsam verzückt auf Plastikknöpfchen herumtippen, gerade die 26. Programmiersprache gelernt und nebenbei Fehler in Datenbanksystemen aufgespürt haben.

3 Die Geschichte der Datenbanken

In dieser Geschichte müssen wir ein wenig ausschweifen, da der Grundsatz der Datenbanken schon sehr viel früher entstand als Sie zu glauben wagen. Vor vielen tausenden von Jahren entwickelte der Mensch ein regelrechtes Zahlenbewusstsein. Dieses Bewusstsein ist die Grundlage für die Rechenvorschriften und die daraus später entstehenden ersten Rechenhilfen. Zum Beispiel der Abakus war eine davon. Ein Rechengerät, das nahezu perfekt ist. Bis heute hat es sich nicht groß verändert, da es keinen Bedarf für eine Verbesserung gab, und bis heute wird es noch verwendet - immerhin geschah das bereits vor der Geburt Christi. Der Professor *Willhelm Schickart* erfand 1623 die Rechenuhr (es war die erste Rechenmaschine), die nachweislich die 4 Grundrechenarten beherrschte. Nach einer kurzen Wartezeit, befinden wir uns im Jahre 1833. Der Engländer *Charles Babbage* erfand eine dampfgetriebene Differenziermaschine und später ein Modell, das unter dem Namen *analytical Engine* bekannt wurde. Beide funktionierten nicht so richtig, aber stellten ein grundlegendes Konzept für die heutige Computertechnologie dar. Auch eine Dame war an der Entwicklung beteiligt. Die Gräfin *Augusta Ada Lovelace* (1815-1852), galt als erste Programmiererin in der Datenverarbeitung. Sie verstand die Programmierung bereist 100 Jahre bevor es die ersten technischen Möglichkeiten dazu gab. 1890 gab die amerikanische Regierung die Volkszählung in Auftrag. *Hermann Hollerith* (1860-1929) verwendete dafür eine Lochkarte zur Auswertung. Die geschätzte Dauer von zehn Jahren für die Auswertung, bewerkstelligte er innerhalb von nahezu sechs Wochen. Dieser Erfindung zu Grunde, wurde 1896 die Firma *Tabulating Machine Company* gegründet. Aus ihr ging 1924 die Firma *IBM* hervor. Zum Schluss bleibt noch Professor *Dr. Konrad Zuse* zu nennen. Er ist der Erfinder und Konstrukteur des ersten funktionstüchtigen, frei programmierbaren Rechners der Welt - der *ZUSE Z 3*. 1941 wurde dieser fertig gestellt, und arbeitete bereits mit Dualzahlen und Gleitkommadarstellung.

Dies war ein "kurzer" Auszug aus der Geschichte. Sicherlich hat es noch vieles mehr gegeben, aber dies soll ja ein Dokument zu *MySQL* und nicht zur Computer-Geschichte werden.

Nähere Informationen finden Sie unter:
http://www.konrad-zuse-computermuseum.de
http://www.computergeschichte.de

4 Die Geschichte von MySQL

Die Entstehung vom *MySQL* befand sich ähnlich wie bei *Linus Torvalds* (der finnische Erfinder von Linux) auch in den nördlichen Breiten. *Michael Widenius* entwickelte 1979 für die schwedische Firma (nein nicht *IKEA*) *TcX* ein Datenbankwerkzeug mit dem Namen *UNIREG*. Die zahlreichen Weiterentwicklungen von *UNIREG* führten allerdings nicht zu den Bekanntheitsgrad wie *ORACLE*, *SYBASE* oder *INFORMIX*.Dieses allerdings von *TcX* auch nicht beabsichtigt war.

Das offizielle Logo von MySQL

Als einige Jahre später durch das WWW (World Wide Web) der Bedarf an dynamischen, datenbankgestützten Websites und Web-Applikationen aufkam, erkannte die Firma *TcX* sehr schnell, dass in den Erfahrungen, die bei der Entwicklung und Weiterentwicklung von *UNIREG* gesammelt wurden, erhebliches Potential lag, um ein System für die Anforderungen im *WWW* zu kreieren.Gleichzeitig arbeitete *David Hughes* an seinem Mini SQL (mSQL), das insbesondere durch die freie und kostenlose Nutzung große Verbreitung erlangen konnte und in Sachen Geschwindigkeit so etwas wie eine Messlatte setzte. *TcX* entschied, dass *MySQL* mindestens so schnell werden müsste wie *mSQL*, zusätzlich aber noch einige Funktionen mehr bieten sollte, die der Konkurrent nicht abdeckte.

5 Bezugsquellen

MySQL ist normalerweise bei jeder Linux-Distribution dabei. Die Installation erfolgt dabei mit dem mitgelieferten Installationswerkzeug meist problemlos (und einfacher). Dennoch möchte ich Ihnen zeigen, wo Sie die Programme beziehen können, da nicht alle eine Distribution haben, bzw. die Programme nicht bei der Linux-Zusammenstellung dabei waren. Wenn Sie Ihre Datenbanksoftware mal auf dem neuesten Stand haben wollen, dann brauchen Sie die neueste Version von *MySQL*, um das bestehende Updaten zu können.

Die erste Anlaufstelle ist die Website von *MySQL AB Company*. Durch dieses Unternehmen wird *MySQL* weltweit vermarktet und vertrieben. Sie erhalten auf deren Homepage jeden erdenklichen Support bei Fragen rund um *MySQL*, die Lizenzsierung oder können sogar bei dem Projekt mitarbeiten. Die Präsenz von *MySQL AB Company* ist unter http://www.mysql.com erreichbar.

Durch die auf der Seite entsprechenden Links kommen Sie recht schnell auf den zutreffenden Downloadbereich. Spätestens jetzt müssen Sie sich für die Versionsnummer entscheiden, die Sie einsetzen möchten. Fast wie bei jedem Programm (od. Projekt) gibt es auch hier so genannte Versionsnummern und eine Attribute. Sie sollten unter normalen Umständen die Attribute "stable" (stabil / für den produktiven Einsatz) benutzen. Unter der stable Version handelt sich um eine Version die keine weiteren, neuen Funktionen mehr bekommt. Es wird zwar weiterhin an der Beseitigung einzelner Fehler gearbeitet, aber die Grundstruktur wird nicht mehr verändert. Neue Versionen erhalten der Reihe nach die Attribute alpha, beta, gamma und abschließend dann stable.

Beim Downloaden sollten Sie dabei beachten, einen Mirror in Ihrer nähe zu wählen, da ansonsten der FTP-Server von *MySQL* sehr schnell überlastet wäre und der Download nur unnötig lange dauern würde.

6 Installation

Je Distribution unterschiedlich stehen mehrere Programmpakete zur Auswahl. Nachfolgend liste ich die wichtigsten auf. Je nach Distribution stehen noch mehrere zur Verfügung.

`kmysql`	ein grafischer Client für MySQL
`mysql`	Das Datenbanksystem MySQL
`mysql-client`	Die Client-Applikation für MySQL
`mysql-bench`	So genannte Benchmarktools zur Überprüfung der Performance des Datenbanksystems
`mysql-devel`	Das Developer-Paket inklusive aller Include-Dateien und Bibliotheken für Programmierer
`mysql-shared`	Fehlermeldungen in verschiedenen Sprachen und weitere Libraries

Wenn Sie die *MySQL* von Ihrem Paketmanager installieren können, tun Sie es. Für alle anderen erkläre ich kurz die "zu-Fuß-Installation". Genauere Hinweise und Erklärung zu Installation entnehmen Sie bitte dem Buch (selbstverständlich auch bei *SelfLinux*) "Installation"

Für das kompilieren brauchen Sie zu Anfang einige Variable. Diese legen Sie mit dem Tool `configure` fest. Als Beispiel verwende ich folgende Variable:

```
MySQL-Verzeichnis = /usr/local/mysql
Daten-Verzeichnis = /var/local/mysql
```

Ich möchte ich nochmal darauf Hinweisen, dass sämtliche Verzeichnisangaben als Beispiel zu sehen sind! Bitte passen Sie die Parameter auf Ihre Bedürfnisse an. - Erspart Ihnen eine Menge Ärger...

Aus diesen Variablen entsteht folgendes configure-Kommando, daß Sie in dem Verzeichnis ausführen, wo die Quell-Dateien liegen (Standardmäßig wäre es das Verzeichnis: **/usr/scr/mysql-4.0.0** von diesem Beispiel gehe ich in folgenden Abschnitten aus.):

```
user@linux ~/ # /configure --prefix=/usr/local/mysql
--localstatedir=/var/local/mysql
```

Nach dem betätigen der Taste "Enter" fängt auch schon die Festplatte an zu rattern. Je nach Systemleistung kann sich der Prozess von ein paar Sekunden bis zu etlichen Minuten hinziehen.

Nachdem `configure` seine Arbeit erfolgreich zu Ende gebracht hat, können Sie mit dem kompilieren beginnen. Viel Spaß...

Was? Sie wissen nicht wie das geht? Nagut ich will nun nicht so sein: Die Kompilierung verläuft eigentlich wie bei jeder anderen Installation auch:

```
user@linux ~/ # make
user@linux ~/ # make install
```

Auch diese beiden Befehle sollten Sie - da ja ein Ergebnis erwünscht ist - mit "Enter" abschließen. Nun ist es an der Zeit, einen Kaffee zu trinken, denn die Aufgabe die Sie Ihrem grauen Blechkasten aufgetragen haben wird sich ein wenig hinziehen.

Der Kompiliervorgang hat sein Werk fertiggestellt? Gut. *MySQL* legt automatisch eine MySQL-Start-Datei an, mit der Sie den MySQL-Dämon starten können, und auch nun tun werden. Wechseln Sie hierzu in das Verzeichnis: **usr/src/mysql-4.0.0/support-files/**(kann bei Ihnen Abweichen)

Um den *MySQL*-Dämonen zu starten müssen Sie die Start-Datei in Ihr Verzeichnis mit des Auszuführenden Dateien

kopieren Standardmäßig ist das folgendes Verzeichnis: **/sbin/** oder **/usr/sbin/**

Wenn eines von diesen Verzeichnissen nicht vorhanden sein sollte, dann lesen Sie bitte in dem Handbuch Ihrer Distribution nach. Der Kopiervorgang sieht dann so aus:

```
user@linux ~/ # cp mysql.server /sbin/mysqld (mit "Enter" bestätigen)
```

Um damit Arbeiten zu können müssen Sie die Datei mit den entsprechenden Zugriffsrechten versehen. Normalerweise sollte dies so sein, daß der Besitzer der Datei alle Rechte hat, und der Rest nur Lese- und Ausführrechte hat.

Geben Sie nun folgenden Befehl ein:

```
user@linux ~/ # chmod 755 /sbin/mysqld
```

Ihr Dämon ist nun fertig zum starten:

```
user@linux ~/ # mysqld start
```

Damit Sie mit *MySQL* arbeiten können, benötigen Sie noch die Standard-Datenbanken (was genau das ist, erkläre ich Ihnen später) Zuerst wechseln Sie in das Verzeichnis: **/usr/src/mysql-4.0.0/scripts/** und geben anschließend diesen Befehl ein:

```
user@linux /usr/src/mysql-4.0.0/scripts/ # ./mysql_install_db
```

Nun werden die Datenbanken "mysql" und "test" angelegt Ihr System ist nun bereit konfiguriert und administriert zu werden.

7 Der erste Kontakt

Um die zu erfahren ob der Server aktiv ist, geben sie bitte folgendes Kommando in die Konsole ein:

```
user@linux ~/ # mysqladmin -u root -h localhost ping
```

Nun sollten Sie die Meldung "mysqld is alive" erhalten. Ist dem so, dann läuft Ihr Server. Andernfalls erhalten Sie eine Hinweismeldung, daß die Verbindung nicht hergestellt werden konnte und wie Sie eventuell Fehler beseitigen können. Wenn Sie das ausprobieren möchten, kann müssen Sie den Datenbankserver herunterfahren:

```
user@linux ~/ # mysqladmin -u root -h localhost shutdown
```

Anschließend noch einmal den "ping"-Befehl eingeben und Sie müssten die entsprechende Fehlermeldung erhalten. Anschließend den Server bitte wieder mit dem Befehl `mysqld start` starten, damit wir damit arbeiten können.

Abschließend zu den Test's geben Sie noch folgenden Befehl ein:

```
user@linux ~/ # mysqladmin -u root -h localhost status
```

Der Befehl liefert bei laufendem Server noch einige Informationen zum System. Sie erkennen auf einfache Art, wie lange der Server schon aktiv ist, wie viele Threads er gerade bearbeitet und all die anderen Dinge.

Soweit der erste Kontakt mit dem MySQL-Datenbankserver. Nun starten wir gemeinsam den Client mysql. Dazu geben Sie am Prompt den Befehl

```
user@linux ~/ # mysql -u root -h localhost
```

ein und bestätigen mit "Enter". Eine Bildschirmmeldung begrüßt Sie und dort, wo Ihr Eingabecursor blinkt, steht nun mysql>

```
user@linux ~/ # mysql -u root -h localhost

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 4.0.0

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> _
```

Durch die Begrüßung wurde Ihnen mitgeteilt, daß alle Befehle mit einem Semikolon abgeschlossen werden müssen. Wenn Sie sich einmal vertippen sollten, machen Sie sich keinen Kopf, *MySQL* schmeißt lediglich eine Fehlermeldung aus, sonst passiert aber nichts weiter. Gehen Sie nun folgende Befehle durch, und verlassen den Clienten mit dem Befehl `quit` wieder.

```
user@linux ~/ # mysql -u root -h localhost

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 4.0.0

Type 'help;' or '\h' for help. Type '\c' to clear the buffer
```

```
mysql>; USE mysql;
Database changed
mysql>; SHOW TABLES;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| host            |
| tables_priv     |
| user            |
+-----------------+
5 rows in set  <0.00 sec>

mysql>; quit;
Bye

user@linux ~/ #
```

Nun haben Sie bereits den Datenbankserver und den Client verwendet. Zum Server ist grundsätzlich nicht mehr viel zusagen, als daß er gestartet werden muß. Dies geschieht automatisch oder manuell, wie sie das schon ausprobiert haben. Der Client hingegen wird Sie nun noch eine Weile beschäftigen.

8 Administration

In diesem Kapitel erfahren Sie die wichtigsten Aufgaben eines Administrators. Ich zeige Ihnen die wichtigsten Werkzeuge und erläutere die ersten Schritte bei der Erstellung einer kleinen Datenbank. Sie erfahren weiterhin die nötigen Grundlagen und die Reihenfolge der notwendigen Arbeitsschritte. Sie können sich mit dem System erst einmal vertraut machen, obwohl Sie den wichtigsten Schritt vornehmen: Die Absicherung des Datenbank-Management-System. Dieser Abschnitt ist wichtig, damit ihr Datenbanksystem vor unberechtigten Zugriff geschützt ist. Gerade bei Verwendung im Netzwerk (oder im WWW) ist wichtig, daß Sie Ihr System absichern.

8.1 Aufgaben eines Administrators

Der Begriff Administrator hört sich immens wichtig an, aber anders als bei vielen anderen Begriffen, trifft das in diesem speziellen Fall sogar zu. Sie sind wie bei Linux selbst der Allmächtige! Diese folgenden Aufgaben kommen dabei auf Sie zu:

* Installation und Updates
* Systemkonfiguration
* Verwaltung von Benutzer- und Zugriffsrechten
* Pflege und Wartung des Systems
* Fehlerlokalisierung und Fehlerbehebung
* Durchführen von Datensicherungen

Für diese Aufgaben ist alleine der Administrator zuständig und nur er alleine hat den vollständigen Zugriff auf das System. Das heißt, daß Sie mit dem Privileg an keinerlei Beschränkungen gebunden sind und eine große Verantwortung haben. Aus diesem Grunde sollten Sie mit den administrativen Rechten nur arbeiten, wenn es wirklich nicht mehr anders geht.

Unter Linux/Unix ist der Benutzer "root" ein alter Bekannter. root ist der Superuser, also gewissermaßen der Obermotz des gesamten Systems und genau einen solchen finden Sie auch bei *MySQL*. Dieser Kerl existiert bereits, nachdem sie *MySQL* auf die Platte gezwängt haben, und Mutter Datenbank fackelt da nicht lange mit der Aufgabenverteilung. Somit kann er kurz nach seiner Geburt schon die folgenden Arbeiten verrichten:

* Erstellen und Verwalten von Benutzerkonten sowie Passwortvergabe
* Anlage von neuen Datenbanken
* Löschen von vorhandenen Datenbanken
* Allgemeine Kontrollfunktionen

Zu diesem Zweck stellt Ihnen *MySQL* ein Tool Namens "mysqladmin" zur Verfügung. Die meisten der vorher aufgeführten Aufgaben verrichten Sie mit diesem Programm. Der Start Ihrer Karriere als MySQL-Administrator beginnt mit der Absicherung Ihres Systems.

8.2 Absichern des Systems

Nach der Neuinstallation ist Ihr MySQL-System offen wie das Brandenburger-Tor. Wenn Sie nun mit einem Netzwerk verbunden sind, und auf Ihr Netzwerk auch von außerhalb zugegriffen werden kann (Remote-Technik), dann haben Sie einen kritischen Punkt. Theoretisch könnte Ihnen ein Fremder alle administrative Rechte nehmen und Ihr System lahm legen. Sie haben vorher schon einen Blick in die vorhandenen Tabellen geworfen, diese Sie mit dem Benutzerkonto "root" OHNE Passwort aufrufen konnten. Sämtliche Zugriffsrechte und Benutzerkonten werden in *MySQL* in Form einer dafür angelegten Datenbank verwaltet. In diesen Tabellen stehen die Benutzernamen, die Passwörter und die zugeteilten Rechte an den vorhandenen Datenbanken und Tabellen. Schreib- und Leserechte werden mit so genannten "Flags" erteilt.

MySQL verwaltet die Benutzerkonten sowie die Rechtevergabe in einer Datenbank mit dem sinnigen Namen *mysql*. In

dieser Datenbank finden Sie die folgenden Tabellen:

columns_priv	In und mit der Tabelle vergeben Sie Zugriffsbeschränkungen auf einzelne Tabellenspalten und die darauf anwendbaren Befehle
db	In dieser Tabelle werdenZugriffsrechte für die gesamte Datenbank gespeichert. Dort wird gespeichert, welcher Benutzer auf welche Datenbank Zugriff hat und welche weiteren Rechte er darauf hat.
host	Hier besteht die Möglichkeit einzelnen Rechnern, also den Hosts, den Zugriff auf MySQL zu beschränken/verbieten.
user	Hier werden die Benutzer mitsamt ihren Passwörtern eingetragen

Mit dem Befehl mysqlshow bekommen Sie eine Liste aller verfügbaren Datenbanken:

```
user@linux ~/ # mysqlshow -u root -h localhost

+-----------+
| Databases |
+-----------+
| mysql     |
| test      |
+-----------+

user@linux ~/ #
```

Wenn Sie nun die Tabelle user aus der Datenbank mysql auf dem Bildschirm ausgeben lassen würden, dann hätten Sie folgende Einträge:

```
+-----------+--------+-------------------+
| Host      | User   | Password          |
+-----------+--------+-------------------+
| localhost | root   |                   |
| %         |        |                   |
| localhost |        |                   |
| %         | root   |                   |
| localhost | tbanus | 68d4f47c49a579c9 |
+-----------+--------+-------------------+
```

Kurz zur Erklärung: Mit Localhost ist der Rechner gemeint, auf dem auch der MySQL-Server installiert wurde. Wahrscheinlich der, vor dem Sie sitzen. Das %-Zeichen ist ein Platzhalter für beliebige andere Rechner

In der Spalte User finden Sie zwei Benutzernamen den root und eventuell finden Sie wo der Benutzer tbanus steht Ihren Vornamen. Dies Bedeutet, daß es auf dem System einen Benutzer mit dem Namen tbanus gibt. Da der Benutzer nicht mit einem Prozentzeichen aufgelistet wird, kann dieser *MySQL* nicht von einem anderen Rechner aus bedienen.

Die vier Zeilen, die den Administrator root betreffen, ergeben alle Kombinationen, die notwendig sind, damit Sie sich als root immer auch ohne Passwort von jedem System aus zugreifen können. Das ist äußerst brisant und wird nun schnellstens abgestellt.

Als ersten Schritt vergeben Sie dem root ein Passwort. Danach löschen Sie alle Datensätze, in denen Prozentzeichen oder leere Datenfelder vorkommen. Erst dann ist Ihr System abgesichert.

Sie müssen sich nun ein sicheres Kennwort überlegen. Verwenden Sie dabei bitte nicht Ihr Kennwort von *root* bei Linux oder eines der anderen Benutzerkennwörter. Die folgende Befehlszeile geben Sie in Ihrer Console ein, wobei Sie das Wort "rootpasswort" durch Ihr Passwort ersetzen

```
user@linux ~/ # mysqladmin -u root -h localhost password "rootpasswort"
```

Wenn nach der Bestätigung des Befehls nix passiert, dann hat es geklappt. Wenn Sie nun als root auf eine Funktion zugreifen wollen und nicht Ihr Passwort mit angeben `mysql -u root` geschieht folgendes:

```
ERROR 1045: Access denied for user: ...
```

Verwenden Sie stattdessen `mysql -u root -p`, werden Sie nach Ihrem Passwort gefragt. Nachdem Sie es eingegeben haben stehen Sie wieder im Clienten.

Genau dort müssen Sie nun für folgende Schritte der Systemabsicherung auch hin. Nun befolgen Sie bitte die folgenden Anweisungen:

Um die Datenbank MYSQL zu aktivieren, geben Sie den SQL-Befehl `USE mysql;` ein. Anschließend verwenden Sie den Befehl `SELECT Host, User, Password FROM user;` und erhalten eine Liste von Datensätzen, die ähnlich der vorherigen Tabelle aussehen sollte, nur das bei dem Benutzer root nun auch eine Passwort-Kodierung aufgeführt ist.

Mit dem folgenden Befehl löschen Sie nun alle Datensätze, die zu einem Teil leer sind oder ein Prozentzeichen besitzen

```
DELETE FROM user WHERE Host="%";
DELETE FROM user WHERE User="";
```

Das Ergebnis können Sie sich mit einem erneuten Aufruf von dem `SELECT` Befehl anschauen. Zu guter Letzt verlassen Sie den Client und starten den MySQL-Server neu, damit die Änderungen übernommen werden.

```
user@linux ~/ # mysqladmin -u root -h localhost -p reload
```

Dadurch wird das System neu initialisiert und ist ab sofort vor anonymen Zugriff geschützt. Es können sich nun nur noch die Benutzer root und in meinem Beispiel der Benutzer tbanus einloggen und damit arbeiten, weil diese alleine Eingetragen sind.

Das folgende Listing zeigt Ihnen noch einmal alle Schritte der Reihe nach, so daß Sie das Beispiel komplett nachvollziehen können:

```
user@linux ~/ # mysqladmin -u root -h localhost password "rootpasswort"
```

```
user@linux ~/ # mysql -u root -h localhost -p
Enter password: *******
```

```
Welcome to the MySQL monitor. Commands end with ; or \g
Your MySQL connection id is 12 to server version 4.0.0
Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> USE mysql;
Database changed
mysql> SELECT Host, User, Password FROM user;
+-----------+--------+------------------+
| Host      | User   | Password         |
+-----------+--------+------------------+
| localhost | root   | 111e97222dab1dd7 |
```

```
| %         |        |                   |
| localhost |        |                   |
| %         | root   |                   |
| localhost | tbanus | 68d4f47c49a579c9 |
+-----------+--------+-------------------+
5 rows in set (0.03 sec)

mysql> DELETE FROM user WHERE Host="%";
Query OK, 2 rows affected (0,57 sec)

mysql> DELETE FROM user WHERE User="";
Query OK, 1 row affected (0,36 sec)

mysql> SELECT Host, User, Password FROM user;
+----------+---------+-------------------+
|Host      | User    | Password          |
+----------+---------+-------------------+
|localhost | root    | 111e97222dab1dd7 |
|localhost | michael | 68d4f47c49a579c9 |
+----------+---------+-------------------+
2 rows in set (0.01 sec)

mysql> quit
Bye

user@linux ~/ # mysqladmin -u root -h localhost -p reload
Enter password: *******
user@linux ~/ #
```

9 Die Datenbank-Planung

Auf den Punkt Datenbank-Planung gehe ich in diesem Dokument nicht näher ein, aber es wird in den späteren Releasen von Selflinux nachgeholt; Versprochen ;) Damit dennoch Ihre Datenbanken planen können, können Sie der einfachheitshalber auf folgende Tools zurückgreifen: Die letzten 2 Einträge sind Mysql-Frontends (Grafische Oberflächen zur Verwaltung und Konfiguration von MySQL)

Programm	System	Version	Preis	Bezugsquelle
ERwin	Win/Linux	Demoversion	unbekannt	http://www.cai.com
DeZign	Windows	Demoversion	130 US$	http://www.datanamic.com/dezign
MySQLFront	Windows	Freeware	kostenlos	http://www.anse.de
KMySql	Linux	Freeware	kostenlos	bei den Distributionen dabei.

10 Die Datenbank

Gut nun haben Sie MySQL erfolgreich installiert und konfiguriert. Prima!!! Aber was bringt Ihnen das beste und sicherste Datenbank-System, wenn Sie nix damit anfangen können? In diesem Kapitel werde ich Ihnen Zeigen, wie Sie eine Datenbank anlegen, verwalten, zuweisen und löschen können.

10.1 Eine Datenbank erzeugen

```
mysql> CREATE DATABASE [IF NOT EXISTS] db_name;
```

Mit diesem Kommando erstellen Sie eine neue Datenbank. Falls diese schon vorhanden sein sollte, erhalten Sie eine entsprechende Fehlermeldung. Wenn Sie keine Fehlermeldung erhalten wollen, dann sollten Sie die Option in den Eckigen Klammern verwenden. Sie können auch alternativ mittels mysqladmin unter der Eingabeaufforderung eine neue Datenbank anlegen.

10.2 Datenbanken löschen

```
mysql> DROP DATABASE [IF EXISTS] db_name;
```

Auch hier können Sie mittels der Option `IF EXISTS` die Fehlermeldung einer nicht vorhandenen Datenbank verbergen lassen.

Vorsicht:
Mit DROP löschen Sie alle Tabellen einer Datenbank! Es gibt keine Möglichkeit die Daten wieder zurückzuholen!

10.3 Datenbankwechsel

```
mysql> USE db_name
```

Wenn Sie mit einer Datenbank arbeiten wollen, müssen Sie diese zunächst aktivieren. Wenn sie während Ihrer Arbeit die Datenbanken wechseln wollen, können sie mit dem Befehl `USE` genau das erreichen.

11 Das Backup

Die wichtigste Aufgabe eines Administrators ist die Sicherung der Datenbanken. Der Befehl mysqldump erzeugt eine Textdatei, in der unter anderem typische MySQL-Kommandos gespeichert werden, die eine Wiederherstellung der Datenbanken ermöglichen.

11.1 Sichern der Daten

```
shell> mysqldump [OPTIONS] database [tables]
oder   mysqldump [OPTIONS] --database [OPTIONS] DB1 [DB2 DB3...]
oder   mysqldump [OPTIONS] --all-databases [OPTIONS]
```

Die vielen Optionen die mysqldump Ihnen anbietet erhalten Sie wenn Sie den Befehl mit dem Parameter --help aufrufen. Für den Anfang reicht es, wenn Sie den Parameter --opt mit anhängen. Diese Option enthält alle nötigen Einstellungen, die bei einer Sicherung berücksichtigt werden müssen. Diesen Parameter sollten Sie immer setzen wenn Sie eine vollständige Sicherung durchführen wollen.

Beispiel-Befehl:

```
user@linux ~/ # mysqldump -u root -p --opt --all-databases > backup.sql
```

In diesem Fall bewirkt die Option `--opt` einen read lock für alleTabellen! Das heisst, daß sie für eine kurzen Augenblick für alle Zugriffe gesperrt werden. Die Backup-Datei wird dabei möglichst klein gehalten und es werden Kommandos eingefügt, die bei einer Wiederherstellung eventuell vorhanden Tabellen zuerst löschen, um sie dann neu erzeugen zu können. Während dieser Operation ist die komplette Datenbank auch blockiert.

11.2 Wiederherstellen der Daten

```
user@linux ~/ # mysql -u root -p < backup.sql
```

Mit diesem Befehl wird ein Restore (Wiederherstellung) durchgeführt

```
user@linux ~/ # mysqldump -u root --password=testpassword --opt gelddatenbank
< save_gdb.sql
```

Damit können Sie einen bestimmten Datensatz wiederherstellen.

12 Tabellen (Relationen)

Tabellen speichern Informationen. Wie Sie aber die Tabellen erstellen, löschen und verändern, zeige ich Ihnen in den folgenden Kapiteln

12.1 Tabellen erzeugen

Es kommt gleich eine mordsmäßige Syntax-Definition. Bitte nicht erschrecken! Der CREATE-Befehl ist mit Abstand der aufwändigste Befehl, den *MySQL* zu bieten hat. Das ganze könnte nun ein wenig"trocken" werden.

12.1.1 Syntax-Definition

```
mysql> CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
       [(erstellungs-definition,...)] [tabellen_optionen]
       [select_statement]
```

Erstellungs-Definition:

```
spaltenname datentyp [NOT NULL | NULL] [DEFAULT default_value]
[AUTO_INCREMENT] [PRIMARY KEY] [reference_definition]
  oder    PRIMARY KEY (index_col_name,...)
  oder    KEY [index_name] (index_col_name,...)
  oder    INDEX [index_name] (index_col_name,...)
  oder    UNIQUE [INDEX] [index_name] (index_col_name,...)
  oder    FULLTEXT [INDEX] [index_name] (index_col_name,...)
  oder    [CONSTRAINT symbol] FOREIGN KEY index_name
          (index_col_name,...)[reference_definition]
  oder    CHECK (expr)
```

index_col_name:

```
  col_name [(length)]
```

reference_definition:

```
      REFERENCES tbl_name [(index_col_name,...)]
                 [MATCH FULL | MATCH PARTIAL]
                 [ON DELETE reverence_option]
                 [ON UPDATE reverence_option]
```

reference_option:

```

```

```
       RESTRICT | CASCADE | SET NULL | NO AKTION | SET DEFAULT
```

tabellen_optionen:

```
TYPE = {BDB | HEAP | ISAM | InnDB | MERGE | MRG_MYISAM | MYSISAM }
  oder  AUTO_INCREMENT = #
  oder  AVG_ROW_LENGTH = #
  oder  CHECKSUM = {0 | 1}
  oder  COMMENT = "string"
  oder  MAX_ROWS = #
  oder  MIN_ROWS = #
  oder  PACK_KEYS = {0 | 1 | DEFAULT}
  oder  PASSWORD = "string"
  oder  DELAY_KEY_WRITE = {0 | 1}
  oder  ROW_FORMAT = { default | dynamic | fixed | compressed }
  oder  RAID_TYPE = {1 | STRIPED | RAIDO} RAID_CHUNKS=#
        RAID_CHUNKSIZE=#
  oder  UNION = {table_name.[table_name...])
  oder  INSERT_METHOD = { NO | FIRST | LAST }
  oder  DATA DIRECTORY="directory"
  oder  INDEX DIRECTORY="directory"

select_statement:
  [INGORE | REPLACE] SELECT ...  (Irgendein korrektes SELECT-Kommando
  ist zulässig)
```

12.1.2 Anwendungsbeispiel

Als kleines Beispiel für ich Ihnen vor, wie Sie eine Adressentabelle erstellen können. Folgende Anforderungen soll diese mindestens erfüllen:

* Es gibt eine Kundennummer die als Primärschlüssel definiert wird. Die Datenbank soll maximal 100 Kunden aufnehmen und automatisch mittels des Primärschlüssels hochgezählt werden. Es gibt keine negativen Schlüssel.
* Es dürfen keine Datensätze ohne Name, Vorname, PLZ und Ort angelegt werden.
* Die Postleitzahl ist ein Zahlenwert und darf nicht negativ sein
* Es soll nach dem Namen und Ort gesucht werden. Deswegen sollen diese mit einem Index verknüpft werden

```
mysql> CREATE TABLE kunden(
    -> kunden_nr TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    -> name CHAR(30) NOT NULL,
    -> vorname CHAR (30) NOT NULL,
    -> strasse CHAR (30),
    -> plz INT(5) UNSIGNED NOT NULL,
    -> ort CHAR (30) NOT NULL,
    -> PRIMARY KEY(kunden_nr),
    -> INDEX(name, ort);

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW TABLES;
+-------------------------+
| Tables_in_testdatenbank |
+-------------------------+
```

```
| kunden                   |
+--------------------------+
1 row in set (0.00 sec)

mysql>
```

Mit dem Befehl `SHOW TABLES` können Sie nachsehen, ob die Tabelle angelegt wurde.

Ein weiterer sehr interessanter Befehl ist `EXPLAIN`. Mit diesem Kommando können sie die Tabellendefinition überprüfen.

12.2 Tabellen (Relationen) löschen

```
mysql> DROP TABLE [IF EXISTS] tbl_name [, tbl_name, ...]
```

Alles was kommt muß auch mal wieder gehen. Manche netten Nachbarn nehmen den Zeitpunkt zwar nicht immer so genau, aber bei einer Datenbank bestimmen Sie, wann es Zeit ist, zu verschwinden.

Bei *MySQL* ist es nicht wie bei anderen Programmen, bei denen Sie 75 Mal gefragt werden, ob Sie sich auch wirklich sicher sind! *MySQL* löscht nach bestätigen der Eingabetaste gnadenlos.

Aus diesem Grunde ist es ratsam, bevor Sie mit der Arbeit an einer Datenbank anfangen eine Sicherheitskopie anzulegen. (Wie, wird in den vorangegangenen Kapiteln beschrieben)

12.3 Indexierung

```
mysql> CREATE [UNIQUE|FULLTEXT] INDEX index_name ON tbl_name
       (spalten_name{(länge)]....];
```

Sie legen normalerweise die Indizes bereist bei Tabellenerstellung fest. Hin und Wieder kommt es aber vor, daß Sie einen weiteren Index brauchen.
Löschen können Sie einen Index mittels des Befehls `DROP INDEX`.

12.4 Tabellen verändern

```
mysql> ALTER TABLE tbl_name tabellen_optionen;
```

Wenn Sie bei einer bestehenden Tabelle eine Spalte hinzufügen wollen, brauchen Sie nicht eine neue Tabelle erzeugen und diese verknüpfen. Sie können mittels des Befehls `ALTER` Veränderungen an bestehenden Tabellen vornehmen.

Folgen zeige ich Ihnen einige Syntax, an denen Sie sehen können, was `ALTER` alles kann.

```
mysql> ALTER TABLE kunden ADD email CHAR(30);
```

Damit legen erzeugen Sie eine neue Spalte namens Email.

```
mysql> ALTER TABLE kunden DROP email;
```

Keine gute Idee? Dann löschen Sie doch einfach die Spalte Email wieder...

```
mysql> ALTER TABLE kunden DROP INDEX name;
```

Damit löschen Sie den Index der bei dem Nachnamen angelegt wurde.

```
mysql> ALTER TABLE kunden DROP PRIMERY KEY;
```

Somit wird der Primärschlüssel der auf die Kundennummer gelegt war, gelöscht.

```
mysql> ALTER TABLE kunden ADD PRIMERY KEY (kunden_nr);
```

War ein Fehler? Dann erstellen Sie ihn einfach wieder.

```
mysql> ALTER TABLE kunden MODIFY ort CHAR(50);
```

Mit der MODIFY-Option ändern Sie den Datentyp. In dem Beispiel wir die Zeichenlänge bei der Spalte ort von 30 auf 50 angehoben.

```
mysql> ALTER TABLE kunden CHANGE ort wohnort CHAR(50);
```

Hiermit können Sie nicht nur den Datentyp sondern auch gleich die Bezeichnung ändern

```
mysql> ALTER TABLE kunden RENAME kunden;
```

Sie können sogar ganze Tabellen umbenennen. Aber achten Sie bitte auf die von Ihnen gesetzten Beziehungen! Diese ändert *MySQL* leider nicht automatisch.

```
mysql> ALTER TABLE kunden ALTER plz SET DEFAULT 85253;
```

Mit dieser Option können Sie der Postleitzahl einen Vorgabewert von 85253 zuweisen.

13 Das Arbeiten mit dem Datensätzen

Nun haben Sie gelernt wie sie Datenbanken und Tabellen erstellen, löschen und verändern können. Da dies aber nicht der einzige Sinn und Zweck von *MySQL* ist, sehen Sie jetzt wie sie diese Datensätze auch noch füllen, löschen und ändern.

13.1 Das Füllen

```
mysql> INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] tbl_name
       [(spaltenname]...)] VALUES (datum1,...),(...).....;
```

oder:

```
mysql> INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] tbl_name
       [(spaltenname]...)] SELECT ...;
```

oder:

```
mysql> INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] tbl_name
       SET spaltenname1=datum1, spaltenname2=datum2, ....;
```

Das INSERT-Kommando dient dazu, (wie sie bereits ahnen) neue Daten in die Tabelle aufzunehmen. Sie können zwischen den drei Varianten wählen. Aber der wohl am meisten verwendete Befehl wird der erste sein. Mit diesem Befehl können Sie die Daten einfügen die Sie gerne möchten. An dem folgenden Beispiel möchte ich Ihnen gerne zeigen, wie das in etwa aussieht: (Ich beziehe mich hierbei wieder auf die kunden-beispiels-Datenbank)

```
mysql> INSERT INTO kunden (name, vorname, strasse, plz, ort) VALUES
    -> ('Sommerfeld', 'Frank', 'Waldweg 12', 85221, 'Dachau');
```

Die Kundennummer wird (da ja mit der Option auto_increment belegt) automatisch von 1 weg hochgezählt.

Möchten Sie z.B. mehrere Datensätze auf einmal eingeben, dann verwenden Sie einfach mehrere Klammerpaare die durch ein Komma getrennt sind:

```
mysql> INSERT INTO kunden (name, vorname, strasse, plz, ort) VALUES
    -> ('Brunninger', 'Jasmin', 'Giechstrasse 1', 81249, 'Muenchen'),
    -> ('Stadler', 'Matthias', 'Glonntalstrasse 15', 85253, 'Erdweg'),
    -> ('Sistig', 'Marianne', 'Linztal 1', 82149, 'Olching');
```

Die zweite Methode ist die Option SELECT. Meist werden SELECT-Befehle mittels eines MySQL-Frontends ausgeführt. Diese können in einer Internetseite eingebunden sein. Dabei wird ein Script die Tipparbeit für Sie

abnehmen. Aber um Ihnen zu zeigen, wie Sie auch "händisch" damit arbeiten können, zeige ich Ihnen 2 Beispiele wie Sie das bewerkstelligen:

```
mysql> SELECT * FROM kunden;
```

damit erhalten Sie (das ganze ist nun sehr unübersichtlich) alle Spalten und Datensätze der Tabelle kunden aufgelistet.

```
mysql> SELECT kunden_nr, vorname, name, ort FROM kunden;
```

Sie können auch einzelne Spalten auswählen!

13.2 Die Select-Anweisung

Sie erhalten hier wiederum den Syntax von dem Befehl SELECT. Ich werde Ihnen das Kommando anschließend anhand von Beispielen erklären und zeigen was es für Möglichkeiten gibt.

```
mysql> SELECT [STRAIGHT_JOIN] [SQL_SMALL_RESULTS] [SQL_BIG_RESULTS]
       [SQL_BUFFER_RESULTS]
       [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]

    select_ausdruck....
    [INTO {OUTFILE | DUMPFILE} 'datei_name' export_optionen]
    [FROM tbl_name
        [WHERE where_definitionen]
        [GROUP BY {unsigned_integer | spalten_name | formula} [ASC | DESC],
 ...]
        [HAVING where_definition]
        [ORDER BY {unsigned_integer | spalten_name | formula} [ASC | DESC],
 ...]
        [LIMIT [offset,] zeilen]
        [PROCEDURE prozedur_name]
        [FOR UPDATE | LOCK IN SHARE MODE]]
```

Die Ausgabe von SELECT erzeugt immer eine Tabelle. Der wohl simpelste Befehl lautet SELECT 1;.

Ich führe Ihnen nun einige Beispiele auf, die Ihnen die Möglichkeiten von SELECT zeigen sollen.

```
mysql> SELECT kunden_nr AS Kundennummer, vorname AS Vorname, name AS
       Name, -> ort AS Ort FROM kunden;
```

Damit vergeben Sie für die vorhanden Spaltenbezeichnungen, schönere und einprägsamere Namen.

```
mysql> SELECT COUNT(kunden_nr) FROM kunden;
```

Mit COUNT können Sie die Anzahl der Datensätze anzeigen lassen.

```
mysql> SELECT * FROM kunden LIMIT 0, 5;
```

Ein tolles feature ist LIMIT. Sie können so die Anzahl der zu anzeigenden, Datensätze auf eine bestimmte Anzahl (hier 5) beschränken lassen. In diesem Beispiel heisst es: Zeige mir maximal 5 Treffer ab den ersten Datensatz. Folgend wäre es dann: LIMIT 5,5 LIMIT 10,5 usw. somit erhalten Sie immer die nächsten 5 Sätze auf Ihrem Bildschirm. Verwendet wird diese Anweisung meist bei Suchmaschinen.

```
mysql> SELECT auftrags_nr, wert AS Netto, wert*1.16 AS Brutto FROM
auftraege;
```

Sie können auch mittels SELECT Berechnungen ausgeben lassen.

```
mysql> SELECT vorname, name, strasse, plz, ort FROM kunden WHERE
    -> ort = 'Dachau';
```

Wer wohnt eigentlich alles von den Kunden in Dachau?

```
mysql> SELECT vorname, name, strasse, plz, ort FROM kunden WHERE
    -> name LIKE 'Sch%';
```

Hier werden alle Nachnahmen mit Sch..... aufgelistet.

```
mysql> SELECT SUM(wert) FROM auftraege WHERE
    -> kunden_nr = 45350
```

Lassen Sie MySQL einfach mal für Sie rechnen

13.3 Das löschen von Datensätzen

Es soll auch mal vorkommen, daß Sie Datensätze löschen müssen. Dies ist eigentlich weiter nicht tragisch. Sie erhalten vom mir wieder einen Syntax und ein Beispiel dazu.

```
mysql> DELETE [LOW_PRIORITY | QUICK] FROM tbl_name
           [WHERE where_definition]
           [ORDER BY ...]
           [LIMIT rows]
```

Beispiel:

```
mysql> DELETE FROM auftraege WHERE kunden_nr = 1000
```

Damit löschen Sie den angegebenen Datensatz... Viel Erfolg ;)

14 Die Zusammenfassung der administrativen Befehlsketten

Passwortvergabe für den Benutzer root:

```
user@linux ~/ # mysqladmin -u root -h localhost password rootpasswort
```

MySQL Absichern:

```
user@linux ~/ # mysql -u root -p
Enter password: *******

        Welcome to the MySQL monitor, ...

        mysql> USE mysql;
        Database changed

        mysql> DELETE FROM user WHERE users='root' AND host='%';
        Query OK, 1 rows affected (0,00 sec)

        mysql> FLUSH PRIVILEGES;
        Query OK, 0 rows affected (0,00 sec)

        mysql> quit
```

ANMERKUNG: Mit FLUSH PRIVILEGES machen Sie die Änderungen wirksam. Sie könnten genauso den Befehl mysqladmin mit dem Parameter -reload benutzen.

Für den Standard-Benutzer die uneingeschränkten Rechte entziehen

```
user@linux ~/ # mysql -u root -p
Enter password: *******

        Welcome to the MySQL monitor, ...

        mysql> USE mysql;
        Database changed

        mysql> REVOKE ALL ON *.* FROM ''@localhost;
        Query OK, 1 rows affected (0,00 sec)

        mysql> REVOKE GRANT OPTION ON *.* FROM ''@localhost;

        mysql> FLUSH PRIVILEGES;
        Query OK, 0 rows affected (0,00 sec)

        mysql> quit
```

ANMERKUNG: Wenn Sie die Absicherung bereits durchgeführt haben, dann wird Ihnen MySQL mitteilen, daß es keinen User gibt, dem Sie die Rechte entziehen können.

Aktionen ohne Kennwort verbieten:

```
user@linux ~/ # mysql -u root -p
```

```
Enter password: *******

        Welcome to the MySQL monitor, ...

        mysql> USE mysql;
        Database changed

        mysql> DELETE FROM user WHERE user='' AND host='localhost';
        Query OK, 1 rows affected (0,00 sec)

        mysql> DELETE FROM user WHERE user='' AND host='%';
        Query OK, 1 rows affected (0,00 sec)

        mysql> FLUSH PRIVILEGES;
        Ouery OK, 0 rows affected (0,00 sec)

        mysql> quit
```

15 MySQL Sprachreferenz

15.1 Datentypen

Bezeichner	Erklärung
TINYINT[(x)]	1 Byte
SMALLINT[(x)]	2 Byte
MEDIUMINT[(x)]	3 Byte
INT[(x)]	4 Byte
BIGINT[(x)]	8 Byte
FLOAT[(x[, y])]	4 Byte
DOUBLE[(x[, y])]	8 Byte
DECIMAL(x, y)	x Byte bei MySQL<3.23,x+2 Byte bei >=3.23
DATE	Datums-Format JJJJ-MM-TT
TIME	Zeit-Format hh:mm:ss
DATETIME	Kombination JJJJ-MM-TT hh:mm:ss
TIMESTAMP	Zeitstempel JJJMMTThhmmss
YEAR	Vierstellige Jahresangabe JJJJ
CHAR(x)	Zeichenkette mit fester Länge x
VARCHAR(x)	Zeichenkette mit variabler Länge x (x+1 Byte)
TINYTEXT	Zeichenkette mit variabler Länge 1 (1+1 Byte)
TINYBLOB	Binärdaten mit variabler Länge 1 (1+1 Byte)
TEXT	Zeichenkette mit variabler Länge 1 (1+2 Byte)
BLOB	Binärdaten mit variabler Länge 1 (1+2 Byte)
MEDIUMTEXT	Zeichenkette mit variabler Länge 1 (1+3 Byte)
MEDIUMBLOB	Binärdaten mit variabler Länge 1 (1+3 Byte)
LONGTEXT	Zeichenkette mit variabler Länge 1 (1+4 Byte)
LONGBLOB	Binärdaten mit variabler Länge 1 (1+4 Byte)
ENUM("wert1","wert2",.)	Aufzählung von maximal 65.535 Zeichenketten
SET("wert1","wert2",)	Aufzählung von maximal 255 Zeichenketten

15.2 Administration

SQL-Kommando	Erklärung
CREATE DATABASE	Erzeugt eine neue Datenbank
DROP DATABASE	Löscht eine vorhandene Datenbank incl. aller Tabellen
FLUSH	Leert und aktualisiert interne Caches
GRANT	Vergabe von Zugriffsrechten
KILL	Beendet einen angegebenen Prozess
REVOKE	Schränkt vergebene Zugriffsrechte wieder ein
USE	Wechselt zu einer anderen Datenbank

15.3 Definition

SQL-Kommando	Erklärung
ALTER TABLE	Ändert oder fügt neue Spalten, Indices etc. hinzu
CREATE FUNCTION	Gibt eine benutzerdefinierte Funktion (UDF) an, die geladen werden soll
CREATE INDEX	Fügt einer Tabelle einen neuen Index hinzu
CREATE TABLE	Erzeugt eine neue Tabelle
DESCRIBLE	Liefert Informationen über die Tabellenstruktur

DROP INDEX	Löscht einen Tabellenindex
DROP TABLE	Löscht eine Tabelle inkl. aller Datensätze
RENAME TABLE	Benennt eine existierende Tabelle um
SHOW	Zeigt Informationen über Datenbanken,Tabellen, Zugriffsrechten etc. an
EXPLAIN	Wie DESCRIBLE, kann aber auch Informationen über das Ergebnis einer SELECT-Anweisung liefern

15.4 Manipulation

SQL-Kommando	Erklärung
DELETE	Löscht angegebene Datensätze aus einer Tabelle
INSERT	Neuen Datensatz in eine Tabelle einfügen
LOAD DATA	Nimmt neue Datensätze in eine Tabelle auf, die in einer ASCII-Datei definiert wurden
REPLACE	Ersetzt einen bereits vorhandenen Datensatz durch einen anderen
SELECT	Fragt vorhandene Datensätze in Tabellen ab
TRUNCATE TABLE	Löscht alle Datensätze einer Tabelle
UPDATE	Verändert den Inhalt eines vorhandenen Datensatzes

15.5 Wartung

SQL-Kommando	Erklärung
BACKUP TABLE	Erstellt ein Backup einer Tabelle
LOCK TABLE	Blockiert eine Tabelle für Schreib- und Lesezugriffe
RESTORE TABLE	Stellt eine mit BACKUP gesicherte Tabelle wieder her
UNLOCK TABLE	Gibt eine mit LOCK blockierte Tabelle wieder frei
ANALYZE TABLE	Liefert Informationen über die Aktualität der Schlüssel# einer Tabelle. Funktioniert nur mit MyISAM und BDB-Tabellen
CHECK TABLE	Überprüft eine MyISAM-Tabelle auf Fehler
OPTIMIZE TABLE	Optimiert die Speicherzuordnung einer MyISAM- und BDB-Tabelle
REPAIR TABLE	In absoluten Notfällen hilft vielleicht dieses Kommando, denn es versucht, beschädigte MyISAM-Tabellen zu reparieren

15.6 Funktionen

SQL-Kommando	Erklärung
CEILING(x)	Liefert nächsthöhere, aufgerundete ganze Zahl x
FLOOR(x)	Liefert nächstkleinere, abgerundete Zahl x
MOD(x, y)	Liefert Modulo von x und y
PI( )	Liefert die Konstante PI
POW(x, y)	Liefert das Ergebnis von x{hoch y}
RAND()	Liefert eine Zufallszahl zwischen 0.0 und 0.1
ROUND(x)	Rundet zur nächstliegenden Ganzzahl
ROUND(x, y)	Rundet auf y Nachkommastellen
SIGN(x)	Erwirkt einen Vorzeichenwechsel
TRUNCATE(x)	Liefert den ganzzahligen Teil einer Kommazahl
TRUNCATE(X, y)	Liefert eine Kommazahl mit y Nachkommastellen
CREATEST(x, y, ...)	Liefert das Maximum von x,y,...
IFNULL(adr1, adr2)	Liefert adr2, wenn adr1 NULL ist, ansonsten adr1
ISNULL(x)	Liefert 1, wenn x NULL ist, ansonsten 0

`STRCMP(str1, str2)`	Liefert 1, wenn str1 gleich str2 ist
`CURDATE()`	Liefert das aktuelle Datum als Zeichenkette
`CURTIME()`	Liefert die aktuelle Zeit als Zeichenkette
`DATE_ADD(dat, x)`	Liefert die Addition von dat + x als Datum
`DATE_FORMAT(format)`	Formatiert einen Datum- oder Zeitwert
`HOUR(x)`	Liefert die Stunde einer Zeitangabe x
`MINUTE(x)`	Liefert die Minute einer Zeitangabe x
`MONTH(x)`	Liefert den Monat des Datums x
`NOW()`	Liefert das aktuelle Datum und die aktuelle Uhrzeit
`SECOND(x)`	Liefert die Sekunden einer Zeitangabe x
`TIME_FORMAT(formate)`	Formatiert eine Zeitangabe
`YEAR(x)`	Liefert das Jahr des Datums x
`AVG(x)`	Ermittelt den arithmetischen Mittelwert einer Spalte x
`COUNT(x)`	Ermittelt die Anzahl von Datensätzen in der Spalte x
`MAX(x)`	Ermittelt den Maximumwert einer Spalte x
`MIN(x)`	Ermittelt den Minimumwert einer Spalte x
`SUM(x)`	Liefert die Summa aller Werte in Spalte x
`DATABASE()`	Liefert den Namen der aktuellen Datenbank als Zeichenkette
`LAST_INSERT()`	Liefert den zuletzt verwendeten AUTO-INCREMENT zurück
`PASSWORT(x)`	Verschlüsselt den Wert x
`USER()`	Liefert den aktuellen Benutzernamen des Clients
`VERSION()`	Liefert die aktuelle Version des Datenbank-Clients
`CHAR_LENGTH(x)`	Liefert die Länge der Zeichenkette x
`CONCAT(s1, s2, ...)`	Verbindet die einzelnen Zeichenketten s1,s2 zu s
`INSERT(s1, p, 0, s2)`	Fügt die Zeichenkette s2 an der Position p in der Zeichenkette s1 ein
`LENGTH(x)`	Liefert die Anzahl der Zeichen in x
`LOWER(x)`	Wandelt alle Großbuchstaben in Kleinbuchstaben um
`LTRIM(x)`	Entfernt vorangestellte Leerzeichen von x
`RTRIM(x)`	Entfernt nachgestellte Leerzeichen von x
`TRIM(x)`	Entfernt vorangestellte und nachfolgende Leerzeichen von x
`UCASE(x)`	Wandelt alle Zeichen in x in Großbuchstaben um
`BIN(x)`	Liefert den Dualcode der Dezimalzahl x
`HEX(x)`	Liefert den Hexadezimalcode der Dezimalzahl x
`OCT(x)`	Liefert den Oktalcode der Dezimalzahl x