

Everybody's Database: MySQL

KAPITEL

7

Die zweifellos im Internet am weitesten verbreitete Open-Source-Datenbank ist MySQL. Sie basiert auf einer Entwicklung von Michael »Monty« Widenius. Die schwedische Firma TcX, bei der er arbeitet, ist heute offizieller Vertreiber und Supporter des Datenbank-Systems. Es wird geschätzt, daß es von MySQL über eine halbe Million Server-Installationen gibt.

Manchmal hört man, daß es sich bei MySQL um eine Weiterentwicklung des mSQL-Systems handle. Das stimmt nur insofern, als Widenius bei der Entwicklung von den mSQL-Entwurfszielen ausging und die APIs entsprechend gestaltete. Es handelt sich jedoch beim Datenbankkern um eine völlig eigenständige Entwicklung, die einige Schwächen von mSQL umgeht.

Die Ähnlichkeit der APIs ist eine Erleichterung für mSQL-Entwickler, die auf die »große Schwester« umsteigen möchten.

Äußerliche Ähnlichkeiten im Handling der beiden Datenbanken haben sogar dazu geführt, daß es eine gemeinsame Veröffentlichung über beide Produkte gibt ([36]).

Neueste Informationen und Tips zu MySQL gibt es unter

<http://www.mysql.de/>



7.1 MySQL-Features im Überblick

Eines der wichtigsten Ziele beim Entwurf von MySQL war die Performance: der Datenbank-Kernel sollte mindestens genauso schnell sein wie sein Vorbild mSQL, obwohl er ihm gegenüber durch eine große Anzahl von Features erweitert werden sollte.

Weitere Design-Ziele waren einfaches Handling und Robustheit.

Die Schnelligkeit wurde vor allem durch das Konzept des Multithreading erreicht, das MySQL von mSQL unterscheidet. Clients müssen nicht auf die Beendigung anderer Abfragen warten; eine große Anzahl von Abfragen kann parallel bearbeitet werden. Dabei nutzt MySQL auch die Fähigkeiten von Multi-processor-Maschinen optimal aus.

Bezüglich der Abfragesprache ist MySQL ebenfalls reicher ausgestattet als mSQL. Es verfügt über mehr als zehn Datentypen und unterstützt SQL-Funktionen. Über ANSI-SQL hinausgehend sind Funktionen wie *ENCRYPT*, *IF* und *WEEK-DAY* implementiert. Sehr nützlich ist die Fähigkeit, mit automatisch aufsteigend nummerierten Feldern zu arbeiten (*AUTO_INCREMENT*, *LAST_INSERT_ID*).



MySQL unterscheidet zwischen Groß- und Kleinschreibung. So sind *feld1*, *Feld1* und *FELD1* drei unterschiedliche Spaltennamen. Dies ist vom Programmierer und Nutzer zu beachten.

Bewußt verzichtet wurde auf performancemindernde Features wie

- referentielle Integrität,
- Transaktionsverarbeitung,
- Speicherung von Viewdefinitionen in der Datenbank,
- Stored Procedures.

In neueren Versionen werden diese Merkmale nach und nach eingearbeitet, jedoch – wie die Entwickler versprechen – so, daß man sie auch abschalten kann, damit nur Installationen, die sie wirklich benötigen, dann auch die damit verbundenen Performance-Nachteile in Kauf nehmen müssen.

7.1.1 Ein Wort zur MySQL-Lizenz

Im Herbst 2000 ging ein Raunen durch die Web-Gemeinde: »MySQL ist jetzt Open Source ...« Schön und gut. Aber Open Source ist nicht gleich GPL (die General Public License von GNU). Vielmehr steht MySQL unter einer eigenen, etwas komplizierteren Lizenz, der MySQL Free Public License. Diese Lizenz sollte sich jeder zu Gemüte führen, der beabsichtigt, Lösungen auf Basis von MySQL kommerziell anzubieten oder zu supporten.

Sie dürfen auf alle Fälle

- den Server installieren und jederzeit kostenlos nutzen,
- eigene Anwendungen für MySQL schreiben und kostenlos selbst nutzen,
- selbstgeschriebene Anwendungen auf der Basis von MySQL kostenlos weitergeben und veröffentlichen.



Doch bereits bei der kostenlosen Weitergabe von MySQL-basierten, eigenen Anwendungen gibt es einige Lizenzbedingungen zu beachten, die in der Datei *PUBLIC* (auf der Buch-CD im Verzeichnis *doc/mysql*) nachzulesen sind. Vollends kompliziert wird es, wenn jemand mit den eigenen Schöpfungen auch noch Geld verdienen möchte. Dann ist ein gründliches Studium der Lizenzdatei unumgänglich.

7.2 Installation und Inbetriebnahme

Die Installation der Server-Software gestaltet sich meistens unkompliziert. In vielen Linux-Distributionen sind entsprechende Binärpakete schon enthalten und können mit dem vorhandenen Paketmanager leicht eingebunden werden.

Günstig ist es, gleich daran zu denken, daß wahrscheinlich auch eine Skriptsprache gebraucht wird, um MySQL im Web zu nutzen. Entsprechende PHP- oder Perl-Pakete sind dann gleich mit zu installieren. Ein funktionierender Webserver versteht sich von selbst.

Im Falle der SuSE-Distribution handelt es sich um die Pakete in Tabelle 7.1.



Paketname	Serie	Inhalt
apache	n	Apache-Webserver
mod_php4	n	PHP4-Modul für Apache
mod_php4-core	n	PHP4-Grundausrüstung (serverunabhängig)
phpdoc	doc	Dokumentation für PHP4
phpMyAdmin	n	PHP-Administrationsskripte für MySQL
mysql	ap	MySQL-Datenbankserver
mysql-client	ap	Einfache Client-Anwendungen für MySQL
mysql-devel	ap	Include-Dateien und Bibliotheken zur Anwendungsentwicklung
mysql-navigator	ap	Ein einfacher Datenbank-Browser für MySQL
mysql-shared	ap	Laufzeitbibliotheken für MySQL-Anwendungen

Tabelle 7.1: Installationspakete für eine MySQL-Webdatenbank (SuSE)

Wer lieber Perl für die Web-Anbindung verwendet, installiert statt der PHP-Module die Pakete *mod_perl* (Serie *n*) und *perlref* (Serie *doc*).

Es gibt ein Paket *mysql-bench*, das einige Benchmark-Tests für die Datenbank enthält. Zur Nutzung einiger neuerer Features wie Transaktionsverarbeitung steht zusätzlich das Paket *mysql-Max* zur Verfügung. Diese werden bei Bedarf ebenfalls installiert.

Die Paketnamen weichen von Distribution zu Distribution etwas ab; generell gibt es immer Pakete mit den angegebenen Inhalten, deren exakte Namen mit rpm-Kommandos zu ermitteln sind (insbesondere *rpm -qa | grep mysql*).

Während der Installation wird ein neuer Linux-Benutzeraccount *mysql* angelegt, unter dessen Berechtigung später der Datenbankserver läuft.

Nachdem die MySQL-Pakete installiert sind, kann der Datenbankserver sofort gestartet werden. Die meisten Linux-Distributoren liefern dazu entsprechende Startup-Skripte mit.

Abbildung 7.1:
Erster Start der
MySQL-Datenbank

```

mystix:~ # /etc/rc.d/mysql start
Creating MySQL privilege database and starting MySQL.
Creating db table
Creating host table
Creating user table
Creating func table
Creating tables_priv table
Creating columns_priv table

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USERS !
This is done with:
/usr/bin/mysqladmin -u root -h localhost password 'new-password' -p
AND
/usr/bin/mysqladmin -u root -h mystix.paradix.own password 'new-password' -p
See the manual for more instructions.

Please report any problems with the /usr/bin/mysqlbug script!

The latest information about MySQL is available on the web at http://www.mysql.c
om
Support MySQL by buying support/licenses at http://www.tcx.se/license.htm.

Starting mysqld daemon with databases from /var/mysql
done
mystix:~ #

```

Der Start des Servers erfordert einige Kommandozeilen-Parameter, die das jeweilige Skript aus der Konfigurationsdatei */etc/my.cnf* entnimmt.

Beim ersten Start sind eine Reihe von Tabellen anzulegen und zu initialisieren. Auch dies übernimmt das Startup-Skript in den meisten Distributionen automatisch. Bild 7.1 zeigt den Ablauf beim Start des MySQL-Dämons aus einer SuSE-8.0-Distribution heraus.

Startup-Skripte liegen je nach Distribution im Verzeichnis */etc/rc.d*, */etc/init.d* oder */etc/rc.d/init.d*. Die Namen können geringfügig abweichen. So heißt das MySQL-Skript bei Red Hat *mysqld*, bei SuSE *mysql*. Ein Blick in den Verzeichnisbaum gibt den nötigen Aufschluß.

Ein erster Test des Datenbankservers erfolgt mit dem Client-Programm *mysql*, das jeder Linux-Benutzer ausführen kann. Es verfügt über eine eigene, kleine Kommandosprache und einen interaktiven Eingabeprompt (Bild 7.2).

Antwortet der Server wider Erwarten nicht, ist mittels der Prozeßstatusanzeige zu kontrollieren, ob er ordnungsgemäß gestartet wurde:

```

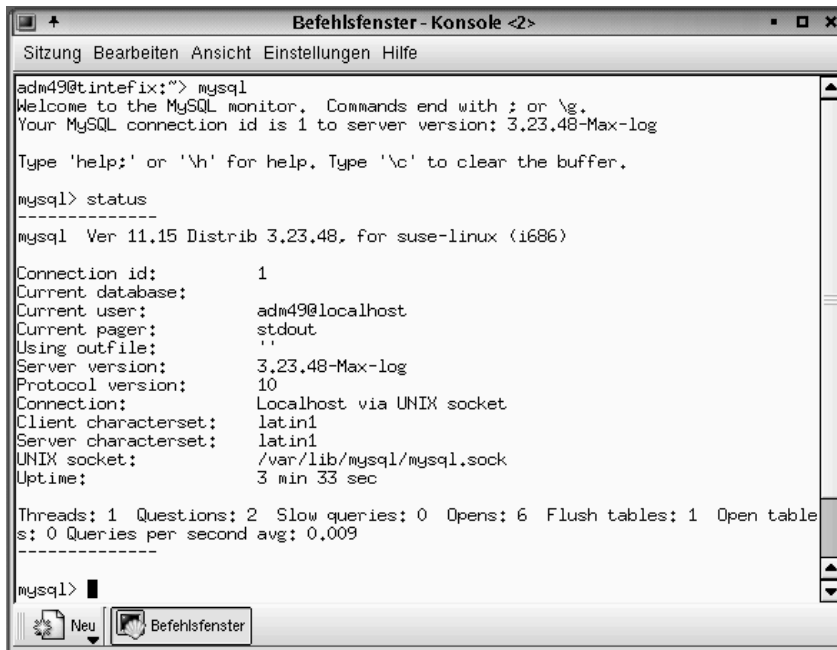
mystix > ps ax | grep mysqld
1837 pts/1    S      0:00 sh /usr/bin/safe_mysqld ...
1849 pts/1    SN    0:00 /usr/sbin/mysqld ...

```

```

1851 pts/1    SN      0:00 /usr/sbin/mysqld ...
1852 pts/1    SN      0:00 /usr/sbin/mysqld ...
mystix >

```



```

Befehlsfenster - Konsole <2>
Sitzung Bearbeiten Ansicht Einstellungen Hilfe
adm49@tintefix:~$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.48-Max-log

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

mysql> status
-----
mysql Ver 11.15 Distrib 3.23.48, for suse-linux (i686)

Connection id:          1
Current database:
Current user:           adm49@localhost
Current pager:          stdout
Using outfile:          ''
Server version:         3.23.48-Max-log
Protocol version:       10
Connection:             Localhost via UNIX socket
Client characterset:    latin1
Server characterset:    latin1
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 3 min 33 sec

Threads: 1  Questions: 2  Slow queries: 0  Opens: 6  Flush tables: 1  Open table
s: 0  Queries per second avg: 0.009
-----

mysql>

```

Abbildung 7.2:
Ein erster Zugriff auf
den MySQL-Server

7.3 Benutzerverwaltung und Sicherheit

Anfangs ist der MySQL-Server absolut ungeschützt. Jeder Linux-Benutzer kann sich ohne Paßwort als Datenbank-Administrator *root* einloggen und den Server verwalten.

Auf diesen Umstand wird bei der Inbetriebnahme hingewiesen (Bild 7.1).

»Nur mal so zum Testen« kann dieser Zustand durchaus nützlich sein, indem er den Zugriff erleichtert. Für ernsthafte Anwendungen ist es jedoch erforderlich, Benutzeraccounts für den Datenbankserver anzulegen und mit Paßwörtern zu schützen.

MySQL bietet ein fein differenziertes System, um Zugriffsrechte sowohl auf die Datenbank allgemein als auch auf einzelne Objekte festzulegen und zu verwalten. Das MySQL Reference Manual nennt diesen Teil des DBMS »Access Privilege System«. Dabei wird mit ACLs (Access Control Lists) gearbeitet, wie von anderer Server-Software her bekannt. Intern werden die ACLs, wie bei SQL-Systemen allgemein üblich, als Tabellen der Datenbank verwaltet.

*Access Privilege
System*

7.3.1 Ein erster Riegel

Um potentiellen Angriffen auf die Datenbank gleich einen Riegel vorzuschieben, ist die erste Aktion des Datenbank-Verwalters die Einrichtung eines Paßworts für *root*.



Für eine Datenbank sollten nie dieselben Paßwörter verwendet werden wie für den Zugang zum Linux-System allgemein, also die Benutzeraccounts des Betriebssystems; selbst dann oder gerade dann nicht, wenn mit gleichen Benutzernamen gearbeitet wird. Wir werden uns folglich für den Datenbank-*root* ein anderes Kennwort ausdenken als das des Linux-Superusers.

Dieses Paßwort ist über das MySQL-Verwaltungsprogramm *mysqladmin* zweimal einzugeben, einmal für lokale Verbindungen auf der Server-Maschine über einen Unix-Socket und zum anderen für die TCP/IP-Verbindungen von anderen Arbeitsstationen aus.

```
# mysqladmin -u root -h localhost password GehHe*mText
# mysqladmin -u root -h <Hostname> password GehHe*mText
#
```

Statt *<Hostname>* wird der DNS-Name des Server-Hosts, statt »GehHe*mText« das neue MySQL-Root-Paßwort eingesetzt.

Ab sofort ist vom Root-Account des Linux-Systems aus kein Zugriff ohne Paßwort auf die Datenbank möglich. Das ist leicht zu prüfen, indem in einer Root-Shell versucht wird, den MySQL-Client *mysql* zu starten:

```
# mysql
ERROR 1054: Access denied for user: 'root@localhost' (Using password: NO)
#
```

Ganz anders, wenn das Programm mit dem Schalter *-p* gestartet und das festgelegte Paßwort verwendet wird:

```
# mysql -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.48
```

Type 'help' for help

```
mysql> quit
Bye
#
```

Statt des *mysql*-internen Kommandos *quit* kann auch die Tastenkombination [Strg]+[D] verwendet werden.

7.3.2 Alle Löcher zustopfen

Probiert man jedoch einen Zugang als anderer Linux-Benutzer, stellt man fest, daß scheinbar (fast) alles beim alten ist:

```
bernhard > mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.22.32

Type 'help' for help

mysql>
```

Die Datenbank antwortet sogar auf eine Abfrage:

```
mysql> select user();
+-----+
| user()          |
+-----+
| bernhard@localhost |
+-----+
1 row in set (0.00 sec)

mysql>
```

Das sieht schlimmer aus als es ist. Zwar haben nach wie vor alle lokalen Linux-Benutzer außer *root* Zugang zum Datenbankserver, und das ohne Paßwort, aber keinerlei Rechte auf Tabellen oder andere Datenobjekte. Das läßt sich einfach ausprobieren, indem eine Verbindung zur Datenbank *mysql* versucht wird, in der interne Informationen des Servers abgelegt sind:

```
mysql> use mysql
ERROR 1044: Access denied for user: '@localhost' to database 'mysql'
mysql>
```

Scheinbar alles in Ordnung. Oder doch nicht? Probieren wir es mit der Datenbank *test*:

```
mysql> use test
Database changed
mysql>
```

Aha. Ein schwacher Trost, daß diese Datenbank nichts zu enthalten scheint:

```
mysql> show tables;
Empty set (0.00 sec)
mysql>
```

In der Tat ist es jedoch so, daß jeder lokale Linux-Benutzer im Auslieferungszustand des MySQL-Servers uneingeschränkten Zugriff auf alle Datenbanken hat, die mit der Zeichenfolge »test« beginnen. Das Reinschauen wäre vielleicht nicht so schlimm; was aber, wenn jemand in einer solchen Datenbank Tabellen anlegt und nach Herzenslust mit Daten vollschreibt? Dann kann es auf der Festplatte unter Umständen eng werden.



Abhilfe schafft hier die Sperrung aller Linux-Benutzeraccounts für den Datenbankserver. Leider ist das nicht so einfach mit der Vergabe eines Benutzerpaßwortes getan wie bei *root*. Wie dem obigen Codeschnipsel (ERROR 1044) zu entnehmen ist, werden alle übrigen Accounts des Linux-Systems in MySQL auf die leere Zeichenkette gemappt. Eine Art anonymer Zugang, und dem kann man leider kein Paßwort zuweisen.

mysqladmin fällt daher als Werkzeug hier aus. Vielmehr muß direkt in die Systemtabelle *user* der Datenbank *mysql* eingegriffen werden, die die Benutzerrechte verwaltet. Die zwei Zeilen mit dem leeren Benutzernamen sind aus der Tabelle zu löschen (Bild 7.3)

Abbildung 7.3:
»Gefährliche« Benutzeraccounts aus der Datenbank löschen

```

mystix:~ # mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 3.22.32

Type 'help' for help.

mysql> use mysql
Database changed
mysql> delete from user where user='';
Query OK, 2 rows affected (0,01 sec)

mysql> flush privileges
-> ;
Query OK, 0 rows affected (0,00 sec)

mysql> quit
Bye
mystix:~ #
  
```

Das Bedien-Interface *mysql* verarbeitet beliebige SQL-Kommandos, einschließlich der MySQL-spezifischen Erweiterungen. Ein fertiges SQL-Statement wird an die Datenbank abgesetzt, wenn ein abschließendes Semikolon erkannt wurde. Im Beispiel des *flush*-Statements fehlt dieses in Bild 7.3; deshalb der Folgeprompt *->*, an dem das fehlende Zeichen einzugeben ist.



Weiterhin ist aus diesem und den vorigen Beispielen zu erkennen, daß Groß- und Kleinschreibung bei *mysql* keine Rolle spielen. Statt

```
delete from user where user=''
```

ist auch

```
DELETE FROM user WHERE user=''
```

oder eine andere Schreibweise möglich.

Die Anweisung *flush privileges* wird benötigt, da der MySQL-Server aus Gründen der Performance eine Kopie der Steuerdatenbank *mysql* im RAM hält; und in dieser Kopie kommen unsere Änderungen erst an, wenn sie durch das *flush*-Kommando aktualisiert wird.

Bei dem Befehl *use* handelt es sich nicht um eine SQL-Erweiterung, sondern um ein Kommando der Bedienoberfläche *mysql*. Daher wird diese Zeile nicht mit einem Semikolon abgeschlossen.

7.3.3 Die kontrollierte Pforte

Jetzt ist es an der Zeit, mindestens einen paßwortgeschützten Datenbank-Benutzeraccount mit definierten Zugangsberechtigungen einzurichten, der für die tägliche Arbeit mit den Daten benötigt wird. Schließlich soll nicht jeder einzelne Datenspeicherungsvorgang mit DBA-Rechten ablaufen.

Wir bleiben zu diesem Zweck in der als *root* gestarteten *mysql*-Oberfläche und legen mit

```
mysql> CREATE DATABASE PiPaPo;
```

unsere schon aus anderen Kapiteln bekannte Beispiel-Datenbank an.

Der Benutzeraccount wird durch eine *GRANT*-Anweisung angelegt:

```
mysql> GRANT ALL ON PiPaPo.* TO agentin IDENTIFIED BY 'mit_herz';
```

Damit ist ein Benutzer, der sich unter dem Namen *agentin* mit dem Paßwort *mit_herz* anmeldet, berechtigt, alle Objekte der Datenbank PiPaPo zu lesen (*SELECT*), zu verändern (*INSERT*, *UPDATE*, *DELETE*) zu erzeugen und zu zerstören (*CREATE*, *DROP*) sowie eine Reihe weiterer Verwaltungsarbeiten an der Datenbank vorzunehmen, mit Ausnahme des Anlegens neuer Benutzer und Weitergabe der eigenen Rechte (*GRANT*). Soll dies zusätzlich auch noch erlaubt werden, ist die obige Anweisung in der Form

```
mysql> GRANT ALL ON PiPaPo.* TO agentin IDENTIFIED BY 'mit_herz' WITH GRANT OPTION;
```

zu verwenden.

Sehr leicht lassen sich weitere Benutzer mit eingeschränktem Rechtesatz anlegen. Der Benutzer *hilfsagent*, erzeugt durch die Anweisung

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON PiPaPo.* TO hilfsagent IDENTIFIED BY 'Wrd!brmft';
```

darf mit allen Tabellen und Views uneingeschränkt arbeiten, ohne jedoch selbst welche anlegen oder löschen zu können, während ein Benutzer *auskunft* mit

```
mysql> GRANT SELECT ON PiPaPo.* TO auskunft IDENTIFIED BY 'P1p8P0';
```

nur Leserechte bekommt.