MySQL 5.5 & 5.6

Michael Kofler

Sicherheit ■ Werkzeuge
Datentypen ■ SQL-Eigenheiten
Backups, Logging und Replikation
InnoDB-Tablespace ■ Tuning

ebooks.kofler

MySQL 5.5 & 5.6

Autor	Michael Kofler
Korrektorat	Markus Hinterreither
ISBN	978-3-902643-08-7
Verlag	ebooks.kofler, Schönbrunngasse 54c, 8010 Graz, Austria

Die PDF-Ausgabe dieses Buchs ist hier erhältlich:
http://kofler.info/ebooks/mysql/

Inhaltsverzeichnis

Vorwort

MySQL ist das populärste Open-Source-Datenbanksystem. Unzählige Content Mangagement Systeme (CMS), Wikis, Foren und Web-Shops verwenden MySQL zur Verwaltung der Daten. Dafür gibt es mehrere Gründe:

- MySQL lässt sich einfach einrichten und warten.
- MySQL beansprucht relativ wenig Ressourcen und ist schnell.
- MySQL kooperiert ausgezeichnet mit PHP sowie mit unzähligen weiteren Programmiersprachen.
- MySQL ist für viele (aber nicht für alle!) Anwendungen kostenlos.

Dieses eBook beschreibt, wie Sie MySQL 5.5 und 5.6 optimal und sicher einrichten und effizient administrieren. Die Themenpalette reicht von Backups über Replikation bis hin zu relativ neuen MySQL-Funktionen wie der Partitionierung.

Ein eigenes Kapitel gibt einen Überblick über die wichtigsten Administrationswerkzeuge, ein weiteres beschreibt MySQL-spezifische Datentypen und andere Besonderheiten, auf die Sie beim Design von MySQL-Datenbanken achten sollten.

Auch die vielen SQL-Eigenheiten von MySQL kommen nicht zu kurz: MySQL verhält sich in manchen Details anders als etablierte Datenbanksysteme. Das ist nicht unbedingt ein Nachteil – solange man diese Besonderheiten kennt!

Das Administrationskapitel enthält eine Menge Details zu den diversen Backup-, Logging- und Replikationsverfahren, die Ihnen MySQL zur Auswahl stellt. Ebenfalls behandelt werden dort der Import und Export von Textdateien sowie das Management der InnoDB-Tablespace-Dateien. Ein kurzes Kapitel zum Thema *Tuning* rundet das eBook ab.

Konzentration auf das Wesentliche

Dieses eBook setzt den Fokus auf MySQL. Auf eine allgemeine Einführung in den Umgang mit relationalen Datenbanksystemen habe ich dabei verzichtet. Sie finden in diesem eBook also z. B. eine Zusammenfassung der MySQL-spezifischen Erweiterungen der Datenbanksprache SQL, aber keine umfassende Beschreibung aller SQL-Kommandos und der daraus resultierenden Möglichkeiten. Ebenso beschreibe ich MySQL-Besonderheiten beim Einrichten neuer Tabellen, ohne aber auf die Theorie des richtigen Datenbankdesigns einzugehen (Normalisierungsregeln etc.).

Dieses eBook kann und will das im Web verfügbare MySQL-Handbuch nicht ersetzen! Die PDF-Ausgabe des MySQL-Handbuchs für Version 5.5 umfasst bereits schwer verdauliche 4600 Seiten! Für Version 5.6 gab es im Dezember 2012 noch keine PDF-Ausgabe, aber es ist abzusehen, dass diese bei Fertigstellung 5000 Seiten überschreiten wird.

Beim Verfassen dieses eBooks habe ich mich dagegen bemüht, auf vergleichsweise wenigen Seiten einen guten Überblick über MySQL zu geben. Für den Fall, dass Sie doch einmal mehr Details benötigen, biete ich an vielen Stellen Links auf die entsprechende Webseite im MySQL-Handbuch.

Damit wünsche ich Ihnen viel Erfolg mit MySQL!

Michael Kofler im Dezember 2012
http://kofler.info

1 Installation

Unter Linux ist die Installation von MySQL normalerweise in einer Minute erledigt. Alle gängigen Linux-Distributionen stellen dazu fertige MySQL-Pakete zur Verfügung.

Auch unter OS X bereitet die Installation von MySQL wenig Probleme. Unerfreulich waren hingegen meine Erfahrungen unter Windows: Das Installationsprogramm sieht zwar hübsch aus, liefert aber je nach Windows-Konfiguration zum Abschluss der Installation einen Fehler. Wenn Sie auf ähnliche Probleme stoßen, finden Sie in diesem Kapitel eine Anleitung, wie Sie diesen Fehler beheben können.

Generell gehe ich im weiteren Verlauf dieses Buchs aber davon aus, dass Sie unter Linux arbeiten. Produktiv wird der MySQL-Server in aller Regel auf einem Linux-System laufen. Daher sollten Sie nach Möglichkeit auch während der Entwicklung unter Linux arbeiten – und sei es in einer virtuellen Maschine, wenn Sie auf Windows oder OS X pardout nicht verzichten wollen.

Welches MySQL soll es denn sein?

Ursprünglich wurde MySQL von der relativ kleinen Firma *MySQL AB* in Schweden entwickelt. 2008 wurde MySQL AB von Sun gekauft, 2010 Sun von Oracle. Diese beiden Eigentümerwechsel und die mitunter wenig Open-Source-freundliche Firmenpolitik von Oracle führten zu starken Spannungen mit der MySQL-Community und letztlich dazu, dass es mittlerweile neben dem "offiziellen" MySQL von Oracle diverse weitere MySQL-Varianten (Forks) gibt. Die drei wichtigsten sind:

- **Percona Server:** Diese MySQL-Variante wird von der Firma Percona zur Verfügung gestellt, die sich auf MySQL-Consulting spezialisiert hat. Percona Server 5.5 bietet gegenüber dem originalen MySQL 5.5 vor allem Effizienzverbesserungen im InnoDB-Tabellentreiber, der in Percona "XtraDB" heißt.
- **MariaDB:** Diese MySQL-Variante wird von einigen ehemaligen MySQL-Gründern bzw. -Entwicklern gepflegt. Die aktuelle MariaDB-Version 5.5 enthält gegenüber dem originalen MySQL 5.5 diverse Zusatzfunktionen und Effizienzverbesserungen. MariaDB enthält auch den XtraDB-Treiber von Percona.
- **Drizzle:** Drizzle ist eine speziell für den Cloud-Einsatz optimierte Version von MySQL. Viele Funktionen des originalen MySQL-Servers wurden in Drizzle explizit entfernt, um zu einem kompakteren Code zu gelangen, der besser gewartet werden kann. Aus diesem Grund ist Drizzle nur eingeschränkt kompatibel zu MySQL.

In diesem eBook gehe ich davon aus, dass Sie das originale MySQL einsetzen. Entsprechende Installationspakete können Sie als *MySQL Community Server* kostenlos von der MySQL-Website herunterladen. Wenn Sie unter Linux arbeiten, können Sie sich die Mühe einer manuellen Installation ersparen und stattdessen die MySQL-Pakete Ihrer Distribution verwenden, die auf demselben Quellcode wie die Community-Version von MySQL basieren.

MySQL ist gratis, oder?

Ganz so einfach ist es nicht! Tatsächlich untersteht MySQL *zwei* Lizenzen, der Open-Source-Lizenz GPL (GNU Public License) sowie einer kommerziellen Lizenz von Oracle. Sie dürfen MySQL nur kostenlos verwenden, solange Sie sich an die Regeln der GPL halten. Der Einsatz von MySQL ist kostenlos, wenn Sie MySQL zuhause, in Ihrer Firma oder auf Ihrer Website einsetzen. Das gilt auch für kommerzielle Websites.

Komplizierter ist es, wenn Sie ein Programm entwickeln und weiterverkaufen, das das zur Ausführung MySQL voraussetzt: Nur wenn es sich bei Ihrem Programm um ein Open-Source-Programm gemäß der GPL handelt und Sie also auch den Quellcode weitergeben (und Ihr Kunde den Quellcode ebenfalls weitergeben darf), ist die Nutzung von MySQL weiterhin frei. Wenn Sie hingegen den Quellcode nicht weitergeben, benötigt Ihr Kunde

eine kommerzielle MySQL-Lizenz von Oracle! Details zu den Lizenzbedingungen von MySQL finden Sie hier:

http://www.mysql.com/about/legal/

Oracles kommerzielles MySQL-Angebot, also die MySQL Enterprise-Edition, unterscheidet sich nicht nur hinsichtlich der Lizenz von der Community-Edition von MySQL. Die Käufer einer Enterprise-Lizenz erhalten zudem Support sowie einige Zusatzfunktionen bzw. Zusatzprogramme, die nicht kostenlos erhältlich sind. Dazu zählen Backup- und Monitoring-Werkzeuge sowie zusätzliche Authentifizierungs-Plugins.

Versionen

Dieses eBook konzentriert sich auf Community-Editionen von MySQL 5.5 und 5.6 sowie auf frei verfügbare MySQL-Administrationswerkzeuge. **MySQL 5.5** wurde Ende 2010 veröffentlicht und ist momentan am stärksten verbreitet. Alle aktuellen Linux-Distributionen enthalten Pakete für MySQL 5.5.

Die nächste MySQL-Version wird die Nummer 5.6 tragen. Es gibt bereits einen *Release Candidate*, also eine fast fertige Version, Oracle hat aber noch keinen Zeitplan angegeben, wann **MySQL 5.6** fertiggestellt werden soll (voraussichtlich Anfang bis Mitte 2013). Erfahrungsgemäß dauert es nach der Fertigstellung einer grundlegend neuen MySQL-Version aber nochmals geraume Zeit, bis diese in für den Server-Einsatz relevanten Linux-Distributionen Eingang findet (also Debian, Ubuntu LTS, Red Hat Enterprise Linux). Insofern werden Sie als Datenbankadministrator in absehbarer Zukunft also überwiegend mit MySQL 5.5 zu tun haben.

Zu guter Letzt gelten große Teile dieses eBooks auch noch für **MySQL 5.1**. Auf diese MySQL-Version werden Sie bei Enterprise-Distributionen (z. B. RHEL) sowie bei älteren Linux-Installationen stoßen. Wenn man von vielen Effizienzverbesserungen einmal absieht, sind die Unterschiede zwischen MySQL 5.1, 5.5 und 5.6 überraschend gering. Die Neuerungen in den Versionen 5.5 und 5.6 betreffen überwiegend Spezialfunktionen, die vor allem für sehr große Datenbanken im Enterprise-Einsatz relevant sind.

Die überwiegende Mehrheit aller MySQL-Installationen betrifft die Verwaltung von relativ kleinen Datenmengen, z. B. für ein CMS oder Wiki, wobei "klein" durchaus etliche GByte

sowie mehr als hundert Abfragen pro Sekunde meinen kann. Wenn Sie also nicht gerade Websites wie Yahoo!, Facebook oder Twitter betreiben bzw. ganz besondere Anforderungen haben, werden Sie von den Verbesserungen in MySQL 5.5 und 5.6 nur wenig spüren.

Version	Wesentliche Neuerungen
5.1 (Ende 2008)	Partitionen. Events. Row-based Replikation. Plugin-API.
5.5 (Ende 2010)	InnoDB wird Standard-Engine. *SIGNAL* und *RESIGNAL* für Stored Procedures. Authentifizierungs-Plugins. Semi-synchrone Replikation.
5.6 (2013)	memcached-API. Volltextsuche für InnoDB. SHA-256-Verschlüsselung für Passwörter und andere Sicherheitsverbesserungen. *TIME*-, *DATETIME*- und *TIMESTAMP*-Zeitauflösung verbessert (Mikrosekunden).

Tabelle 1.1: MySQL-Versionen

Vielleicht fragen Sie sich, was zwischen Version 5.1 und 5.5 liegt. Die knappe Antwort: Nichts! Es gab eine Weile Pläne für eine MySQL-Version 5.4, diese Version wurde aber nie fertiggestellt. Einen guten Überblick darüber, welche Zusatzfunktionen bzw. Neuerungen ab welcher MySQL-Version zur Verfügung stehen, gibt diese Seite:

http://dev.mysql.com/doc/refman/5.6/en/development-history.html

1.1 Installation unter Linux

Die MySQL-Installation ist bei den meisten Distributionen mühelos. Es müssen lediglich die richtigen MySQL-Pakete installiert werden. Allerdings bekommen Sie auf diese Weise nicht immer die aktuellste MySQL-Version. Die meisten Linux-Distributionen ändern die MySQL-Hauptversion während des mehrjährigen Wartungszeitraum nicht.

Um ein konkretes Beispiel zu nennen: Debian 6 wurde im Februar 2011 fertiggestellt und mit der schon damals nicht mehr aktuellen Version MySQL 5.1.49 ausgeliefert. Im

Rahmen der Sicherheits-Updates wurde die MySQL-Version mittlerweile (Dezember 2012) bis zur Version 5.1.63 aktualisiert. Debian 6 wird aber nie offizielle MySQL-5.5-Pakete enthalten. Wenn Debian 7 voraussichtlich Anfang 2013 frei gegeben wird, wird diese Debian-Version Pakete für MySQL 5.5 enthalten. Debian-Anwender, die MySQL 5.6 einsetzen möchten, müssen entweder auf Debian 8 warten, oder die MySQL-Pakete aus den *testing*- oder *unstable*-Paketquellen beziehen, sobald sie dort verfügbar sind.

Die folgende Tabelle fasst kurz zusammen (Stand Dezember 2012), welche MySQL-Version Sie mit welcher Distribution erhalten:

Distribution	MySQL
Debian 6	MySQL 5.1.n
Debian 7	MySQL 5.5.n
Fedora 15 bis 18	MySQL 5.5.n
openSUSE 11.2 bis 11.4	MySQL 5.1.n
openSUSE 12.1 und 12.2	MySQL 5.5.n
SUSE Enterprise 10	MySQL 5.0.n
SUSE Enterprise 11	MySQL 5.0.n
RHEL 5.n	MySQL 5.0.n
RHEL 6.n	MySQL 5.1.n
Ubuntu 10.04 LTS	MySQL 5.1.n
Ubuntu 12.04 LTS	MySQL 5.5.n

Tabelle 1.2: MySQL-Versionen in populären Linux-Distributionen

Debian und Ubuntu

Zur Installation des MySQL-Servers sowie der Client-Werkzeuge (also der Kommandos `mysql`, `mysqladmin` etc.) führen Sie das folgende Kommando aus:

```
apt-get install mysql-client mysql-server
```

Während der Installation müssen Sie das `root`-Passwort für den MySQL-Server angeben. Wählen sie ein anderes Passwort als jenes für Ihren eigenen Login bzw. für den Root-Login! Der MySQL-Server wird sofort nach der Installation gestartet und auch in Zukunft nach jedem Rechnerneustart automatisch ausgeführt.

Fedora und Red Hat Enterprise Linux (RHEL)

Zur Installation von MySQL unter Fedora oder RHEL führen Sie das nachstehende Kommando aus. (Alle Informationen zu RHEL gelten gleichermaßen auch für Oracle Linux, CentOS und Scientific Linux.)

```
yum install mysql mysql-server
```

Hinweis

RHEL 6.n stellt leider nur MySQL-Pakete für Version 5.1 zur Verfügung (Stand Dezember 2012, RHEL-Version 6.3). Natürlich können Sie von der MySQL-Website RPM-Pakete für MySQL 5.5 herunterladen und manuell installieren; das ist aber umständlich, außerdem müssen Sie sich selbst um Updates kümmern. Wesentlich einfacher ist es, die inoffizielle, aber zuletzt gut gewartete REMI-Paketquelle einzurichten und die MySQL-Pakete von dort zu beziehen. Das Kommando zur Installation lautet dann `yum install --enablerepo=remi mysql mysql-server`.

Ganz egal, ob Sie MySQL-Pakete direkt von MySQL oder aus der REMI-Paketquelle verwenden – auf jeden Fall verlieren Sie damit (für diese Pakete) die langjährige Update-Garantie durch Red Hat. Wenn Sie also an den MySQL-Server keine besonderen Anforderungen stellen und nicht auf Neuerungen von MySQL 5.5 angewiesen sind, kann es durchaus vernünftiger sein, bei MySQL 5.1 zu bleiben!

Der MySQL-Server wird standardmäßig nicht gestartet. Um den Server einmalig manuell zu starten, führen Sie das folgende Kommando aus:

```
service mysql start
```

Damit der MySQL-Server bei jedem Rechnerstart automatisch ausgeführt wird, führen Sie je nach Distribution eines der beiden folgenden Kommandos aus:

```
systemctl enable mysqld.service              (Fedora)
chkconfig --level 35 mysqld on               (RHEL 6)
```

Unterschiedliche Kommandos sind deswegen erforderlich, weil sich unter Fedora das Init-System Systemd um den Start des MySQL-Servers kümmert, unter RHEL 6 hingegen ein traditionelles Init-V-Script.

Achtung

Der MySQL-Server ist unter RHEL bzw. Fedora anfänglich nicht abgesichert. Ein *root*-Login ist ohne Passwort möglich. Sie sollten baldmöglichst ein *root*-Passwort festlegen (siehe den Abschnitt Basisabsicherung)!

openSUSE

Zur Installation führen Sie das folgende Kommando aus:

```
zypper install mysql-community-server \
  mysql-community-server-client
```

Der MySQL-Server wird nicht automatisch gestartet. Zum manuellen Start führen Sie dieses Kommando aus:

```
service mysql start
```

Für den Start ist das Init-V-Script `/etc/init.d/mysql` zuständig. (Der MySQL-Start wurde also bis einschließlich openSUSE 12.2 *nicht* auf Systemd portiert.) Damit der MySQL-Server in Zukunft bei jedem Rechnerstart automatisch ausgeführt wird, ist ein weiteres Kommando erforderlich. Damit werden Init-V-Links für die Runlevel 3 und 5 eingerichtet:

```
insserv mysql
```

Achtung

Auch unter (open)SUSE ist der MySQL-Server anfänglich nicht abgesichert. Ein *root*-Login ist ohne Passwort möglich. Sie sollten baldmöglichst ein *root*-Passwort festlegen (siehe den Abschnitt Basisabsicherung)!

MySQL 5.6 installieren

Wie bereits erwähnt, gibt es zur Zeit keine Linux-Distribution mit integrierten MySQL-5.6-Paketen. Wenn Sie MySQL 5.6 dennoch unter Linux ausprobieren möchten, ist Handarbeit erforderlich. Auf der Website http://dev.mysql.com/downloads finden Sie speziell für Debian (und Ubuntu) bzw. für Red Hat und SUSE Enterprise vorbereitete Pakete. Im Regelfall benötigen Sie drei Pakete, die den *MySQL Server*, die *Client Utilities* und die *Shared Components* enthalten. Achten Sie darauf, dass Sie sich für die richtige Architektur entscheiden (32 oder 64 Bit).

Für dieses eBook habe ich die Installation von MySQL 5.6 unter CentOS 6.3 getestet. Das größte Problem sind dabei die Paketabhängigkeiten. Das Paket `MySQL-shared` lässt sich wegen Konflikten mit dem in der Regel standardmäßig installierten Paket `mysql-libs` nicht installieren. Eine Deinstallation von `mysql-libs` scheitert aber daran, dass die darin enthaltenen Bibliotheken von diversen anderen Paketen benötigt werden. Abhilfe schafft `rpm -e --nodeps`. Allerdings ist nicht garantiert, dass alle installierten Programme mit MySQL-Abhängigkeit zu den neuen MySQL-Bibliotheken kompatibel sind. Auf einem Produktions-Server sind derartige Kommandos also absolut nicht zu empfehlen!

```
rpm -e mysql-libs
  Fehler: Fehlgeschlagende Abhängigkeiten:
  libmysqlclient.so.16()(64bit) wird benötigt von
    (installiert) redland-1.0.7-11.el6.x86_64
  ...
rpm -e --nodeps mysql-libs
rpm -i MySQL-client-n.n.rpm MySQL-server-n.n.rpm \
  MySQL-shared-n.n.rpm
```

2 Hello World!

Dieses Kapitel versucht, Ihnen eine erste Vorstellung zu vermitteln, wie verschiedene Techniken und Werkzeuge bei der Arbeit mit MySQL zusammenspielen. Aus didaktischer Sicht wäre es natürlich wünschenswert, MySQL in aufeinander aufbauenden Schritten zu beschreiben. In der Praxis funktioniert das aber nicht. Damit ich Ihnen erklären kann, wie Sie administrative Werkzeuge einsetzen, müssen Sie das MySQL-Sicherheitssystem in Grundzügen kennen. Um umgekehrt MySQL erstmals abzusichern, sollten Sie zumindest mit dem Kommando `mysql` umgehen können. Wo also anfangen?

Der Ausweg aus diesem Dilemma ist dieses Kapitel: Es fasst einige Besonderheiten von MySQL ganz kurz zusammen und stellt einige grundlegende Werkzeuge vor. Zu allen hier präsentierten Informationen folgen in den weiteren Kapiteln mehr Details. Aber um einen ersten Überblick zu geben, sollte dieses Kapitel ausreichen.

Hinweis

Der Begriff »MySQL« hat je nach Kontext verschiedene Bedeutungen. Für dieses eBook gelten die folgenden Regeln:

MySQL bezeichnet das Datenbanksystem als Ganzes. `mysqld` meint den MySQL-Server (den MySQL-Dämon). `mysql` ist ein MySQL-Kommandointerpreter, also ein administratives Werkzeug für die Kommandozeile. *mysql* kann aber auch eine besondere MySQL-interne Datenbank bezeichnen, die zur Speicherung der Zugriffsrechte aller MySQL-Benutzer verwendet wird.

2.1 Erste Experimente (Hello World!)

In diesem Abschnitt zeige ich Ihnen, wie Sie eine Testdatenbank einrichten, eine darin enthaltene Tabelle mit einigen Daten befüllen und eine erste Abfrage durchführen.

Der Einfachheit halber führen Sie alle Arbeiten als MySQL-Benutzer *root* aus. Wie Sie im Kapitel Sicherheit sehen werden, ist es normalerweise zweckmäßiger und sicherer, für jede Datenbank zumindest einen eigenen MySQL-Benutzer einzurichten, der nur auf diese eine Datenbank Zugriff hat.

Administrationswerkzeuge

Der MySQL-Server ist ein Programm ohne Benutzeroberfläche. Es läuft im Hintergrund als Dienst (Windows) bzw. "Dämon" (Linux) und ist gewissermaßen eine unsichtbare Black Box.

Um den Server zu steuern, um neue Datenbanken anzulegen, um Backups durchzuführen sowie für viele weitere Aufgaben brauchen Sie Administrationswerkzeuge. Das für MySQL-Einsteiger attraktivste Programm ist vermutlich die MySQL Workbench. Ihre grafische Benutzeroberfläche hilft bei vielen administrativen Arbeiten und beim Design neuer Datenbankschemata.

Persönlich bin ich kein großer Fan der MySQL Workbench. Ihre Bedienung ist nicht immer schlüssig und setzt auf jeden Fall grundlegende Datenbank- und MySQL-Kenntnisse voraus. Wenn Sie diese bereits haben, kommen Sie mit dem Kommando `mysql` oft wesentlich schneller zum Ziel. Dieses Kommando führen Sie unter Linux oder OS X in einem Terminalfenster aus. Unter Windows öffnen Sie das Fenster *Eingabeaufforderung* und starten dort das Programm `mysql.exe`. Eventuell müssen Sie den vollständigen Pfad des Programms angeben. Unter Windows befindet sich `mysql.exe` in der Regel im Verzeichnis `C:\Program Files\MySQL\MySQL Server 5.n\bin`).

Administrative Aufgaben setzen oft voraus, dass Sie sich beim MySQL-Server mit dem Benutzernamen *root* anmelden. Wenn Sie mit dem Kommando `mysql` arbeiten, geben Sie den gewünschten Benutzernamen mit der Option `-u` an.

```
mysql -u root
  ERROR 1045 (28000): Access denied for user 'root'@'localhost'
  (using password: YES)
```

Die obige Fehlermeldung bedeutet, dass der administrative MySQL-Benutzer durch ein Passwort abgesichert ist. Das ist gut so! Sollte mit dem obigen Kommando ein *root*-Login ohne Fehlermeldung gelingen, liegt ein schwerer Sicherheitsmangel vor, den Sie möglichst rasch beheben sollten (siehe den Abschnitt Basisabsicherung).

Damit `mysql` Ihnen die Möglichkeit gibt, dieses Passwort einzugeben, müssen Sie zusätzlich die Option `-p` angeben. In der Regel sieht der Start des Kommandos `mysql` so aus:

```
mysql -u root -p
  Enter password: ******
```

Nach dem Login erwartet `mysql` die Eingabe Ihrer SQL-Kommandos, die Sie mit einem Strichpunkt und Return abschließen müssen. Einige Spezialkommandos funktionieren auch ohne Strichpunkt, es ist aber vernünftig, wenn Sie sich die Verwendung des Strichpunkts gleich angewöhnen.

Als ersten Test, ob der MySQL-Server funktioniert, können Sie das Kommando `status` ausführen. Das Ergebnis sollte so ähnlich wie im folgenden Listing aussehen:

```
mysql -u root -p
  Enter password: ******
  Welcome to the MySQL monitor.  Commands end with ; or \g.
  Your MySQL connection id is 246487
  Server version: 5.5.24-0ubuntu0.12.04.1 (Ubuntu)
  ...
mysql> status;
  mysql  Ver 14.14 Distrib 5.5.24, for debian-linux-gnu (x86_64)
    using readline 6.2
  Connection id:          246487
  Current database:
  Current user:           root@localhost
  SSL:                    Not in use
```

```
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.5.24-0ubuntu0.12.04.1 (Ubuntu)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    utf8
Conn.  characterset:    utf8
UNIX socket:            /var/run/mysqld/mysqld.sock
Uptime:                 39 days 16 hours 6 min 15 sec
Threads: 1  Questions: 27081252  Slow queries: 0  Opens: 503
  Flush tables: 1  Open tables: 144
  Queries per second avg: 7.900
```

Um das Kommando `mysql` zu beenden, geben Sie `exit` ein oder drücken Strg+D (nur unter Linux und OS X).

Eine ausführliche Beschreibung des Kommandos `mysql` finden Sie im Kapitel Administrationswerkzeuge. Dort stelle ich Ihnen auch eine Menge weiterer Kommandos und Benutzeroberflächen vor.

"Hello World!" in der MySQL Workbench

Nach dem Start der MySQL Workbench müssen Sie zuerst eine Verbindung zum MySQL-Server herstellen. Dazu klicken Sie in der linken Spalte des Startdialogs auf OPEN CONNECTION TO START QUERYING. Beim Verbindungsaufbau geben Sie den Benutzernamen *root* und das dazugehörende Passwort an. Sobald die Verbindung zum MySQL-Server gelungen ist, zeigt die Workbench das neue Dialogblatt SQL EDITOR an. Alle weiteren Schritte spielen sich in diesem Dialogblatt ab.

Um eine neue, leere Datenbank einzurichten, klicken Sie in der Symbolleiste auf den Button CREATE A NEW SCHEMA. Anschließend geben Sie den gewünschten Namen der neuen Datenbank an (z. B. *helloworld1*) und klicken auf APPLY. Die Workbench zeigt nun

3 Sicherheit

Im Abschnitt Basisabsicherung habe ich bereits ganz kurz die Grundzüge des MySQL-Sicherheitskonzepts vorgestellt. Die Absicherung erfolgt in zwei Schritten:

- **Login:** Bevor ein Client-Programm mit dem MySQL-Server kommunizieren kann, muss es sich authentifizieren. Dabei werden drei Informationen ausgewertet: der Login-Name, das Passwort und der Hostname bzw. die IP-Adresse des Rechners, auf dem das Client-Programm läuft.
- **Privilegien:** In einer MySQL-internen Datenbank wird für jeden Benutzer eine Liste sogenannter Privilegien gespeichert. Diese Liste legt fest, was der Benutzer darf und was nicht. Damit ist es z. B. möglich, dass ein Benutzer nur die Tabellen einer bestimmten Datenbank lesen kann, während ein anderer alle Datenbanken lesen und verändern darf.

Dieses Kapitel erklärt die Grundlagen des MySQL-Sicherheitssystems. Sie lernen, wie Sie neue MySQL-Benutzer einrichten und diesen die erforderlichen Rechte (Privilegien) geben, um bestimmte Aufgaben zu erledigen. Außerdem erkläre ich Ihnen, wie diese Informationen MySQL-intern gespeichert werden.

Daneben kann der Zugriff auf den MySQL-Server durch einige weitere Verfahren beschränkt, abgesichert oder verschlüsselt werden, etwa durch Änderungen in der MySQL-Konfigurationsdatei, durch SSL-Verschlüsselung oder SSH-Tunnel, durch Authentifizierungs-Plugins etc. Einen Überblick über diese Möglichkeiten finden Sie am Ende dieses Kapitels.

3.1 Das Wichtigste zuerst

In der Praxis wird das fein abgestufte Sicherheitssystem von MySQL selten benötigt. Der Normalfall sieht vielmehr so aus, dass auf einem MySQL-Server mehrere Datenbanken eingerichtet sind, die von unterschiedlichen Programmen genutzt werden – z. B. eine Datenbank für das CMS der Website A und eine zweite für das Wiki von Website B.

Damit nun jedes Programm auf seine Datenbank zugreifen kann (und nur auf diese), gibt es für jede Datenbank einen MySQL-Benutzer mit uneingeschränkten Rechten auf diese Datenbank. Der Benutzer *root* wird nur für zentrale Administrationsarbeiten und für Backups verwendet. (Sicherheitsbewusste MySQL-Administratoren richten für Backups noch einen weiteren Benutzer ein.)

Zum Einrichten neuer Benutzer sowie zum Festlegen der Zugriffsrechte sieht MySQL eine ganze Reihe von Kommandos vor (*CREATE USER*, *DROP USER*, *RENAME USER*, *SET PASSWORD*, *GRANT* und *REVOKE* sowie *SHOW GRANTS*). Die vollständige Syntax dieser Kommandos ist ziemlich komplex und detailliert im MySQL-Benutzerhandbuch beschrieben. In der Praxis werden Sie zumeist mit *GRANT* und *DROP USER* das Auslangen finden und auch bei diesen Kommandos nur einen Bruchteil der möglichen Syntaxvarianten nutzen.

Die Syntax der Kommandos *GRANT*, *REVOKE* und *DROP USER* sieht vereinfacht dargestellt folgendermaßen aus:

```
GRANT  privilegien
ON     [database.]table  oder  database.spname
TO     user@host [IDENTIFIED BY 'password']
[WITH GRANT OPTION];

REVOKE privilegien
ON     [database.]table  oder  database.spname
FROM   user@host;

DROP USER user1@host1, user2@host2;
```

Wenn Sie die Zugriffsrechte für alle Tabellen einer Datenbank verändern möchten, lautet die Schreibweise *ON database.**. Wenn Sie globale Privilegien verändern möchten, geben Sie *ON *.** an. Es ist nicht zulässig, im Datenbanknamen Jokerzeichen zu verwenden. Als *user* können Sie `''` angeben, wenn alle Benutzer eines bestimmten Rechners gemeint sind (z. B. `''@rechnername`). Bei *host* muss dagegen die Schreibweise `'%'` verwendet werden (z. B. `benutzername@'%'`).

Im folgenden Beispiel wird der neuen Benutzer *username* eingerichtet, falls dieser noch nicht existiert. Dieser Benutzer erhält uneingeschränkte Rechte auf alle Tabellen der Datenbank *dbname*. (Damit Sie das *GRANT*-Kommando ausführen können, müssen Sie sich vorher als *root* beim MySQL-Server anmelden.)

```
GRANT ALL ON dbname.* TO username@localhost IDENTIFIED BY 'xxx';
```

Der Benutzer *username* darf nur dann eine Verbindung zum MySQL-Server herstellen, wenn die Verbindung auf demselben Rechner initiiert wird, auf dem auch der MySQL-Server läuft. Gerade bei Webanwendungen ist das häufig der Fall, wenn sowohl Apache als auch MySQL auf demselben Rechner laufen. Wenn *username* hingegen von einem anderen Rechner aus auf den MySQL-Server zugreifen soll, müssen Sie im *GRANT*-Kommando anstelle von *localhost* den Host-Namen oder die IP-Adresse angeben:

```
GRANT ALL ON dbname.* TO username@firma-abc.de IDENTIFIED BY 'xxx';
```

Sie können *GRANT ALL* auch mehrfach ausführen, um MySQL-Logins von einer Reihe verschiedener Rechner zuzulassen. Noch liberaler ist das folgende Kommando, das einen Login von jedem beliebigen Hostnamen aus erlaubt. Dabei ist entscheidend, dass das Jokerzeichen `%` für den Hostnamen in Anführungszeichen gestellt wird.

```
GRANT ALL ON dbname.* TO username@'%' IDENTIFIED BY 'xxx';
```

Hinweis

Die Kommunikation zwischen einem Benutzer (Client-Programm) und dem MySQL-Server über das Netzwerk setzt voraus, dass der Port 3306 nicht durch eine Firewall blockiert ist und dass der MySQL-Server überhaupt Netzwerkverbindungen akzeptiert. Viele MySQL-Installationen sind aber aus Sicherheitsgründen standardmäßig

so eingerichtet, dass keine Netzwerkverbindungen – oder nur solche von `localhost` – zugelassen werden. Entscheidend sind die Einträge `skip-networking` oder `bind-adress=127.0.0.1` in der MySQL-Konfigurationsdatei. Welche Funktion die beiden Schlüsselwörter haben, ist im Abschnitt Sicherheitseinstellungen in my.cnf bzw. my.ini beschrieben.

Wenn Sie einen zweiten Administrator-Account einrichten möchten, also einen MySQL-Benutzer, der uneingeschränkte Rechte auf alles hat, gehen Sie so vor. *ALL ON *.** bedeutet, dass *root2* alle Privilegien für alle Tabellen in allen MySQL-Datenbanken hat. Die zusätzliche Klausel *WITH GRANT OPTION* gibt dem Benutzer *root2* darüber hinaus die Möglichkeit, selbst neue Benutzer einzurichten, die über dieselben Privilegien wie *root2* verfügen können.

```
GRANT ALL ON *.* TO root2@localhost IDENTIFIED BY 'xxx'
  WITH GRANT OPTION;
```

Wenn Sie den Überblick darüber verloren haben, welche Privilegien ein bestimmter Benutzer hat, hilft das Kommando *SHOW GRANTS* weiter:

```
SHOW GRANTS FOR peter@localhost;
  Grants for peter@localhost:
  GRANT SELECT ON mydb.* TO 'peter'@'localhost'
```

Mit `REVOKE` können Sie zuvor vergebene Privilegien wieder entziehen. Der Benutzer bleibt aber in der MySQL-Benutzerdatenbank und kann sich weiterhin anmelden. Mit dem folgenden Kommando verliert er aber alle Privilegien, die den Zugriff oder die Bearbeitung der Tabelle *dbname* ermöglichen:

```
REVOKE ALL ON dbname.* FROM username@'%';
```

Um einen Benutzer vollständig aus der MySQL-Benutzerdatenbank zu entfernen, müssen Sie das Kommando *DROP USER* verwenden. Beachten Sie dabei, dass Sie jede Kombination aus Benutzername und Hostname exakt angeben müssen, also beispielsweise so:

```
DROP USER username@localhost, username@firma-abc.de, username@'%';
```

4 Werkzeuge

Der MySQL-Server selbst hat weder eine grafische Benutzeroberfläche noch eine text-orientierte Schnittstelle zur Bedienung. Sämtliche Administrationsarbeiten müssen daher mit Werkzeugen erfolgen, die via TCP/IP oder über eine Socket-Datei mit dem MySQL-Server kommunizieren. An derartigen Programmen herrscht kein Mangel: Neben Kommandos wie `mysql`, `mysqladmin`, `mysqldump` oder `mysqlimport` existieren auch webbasierte (phpMyAdmin, Chive) und grafische Benutzeroberflächen (MySQL Workbench). Dieses Kapitel gibt einen Überblick über die wichtigsten Programme.

4.1 mysql

Das Kommando `mysql` wird in der Regel als SQL-Kommandointerpreter verwendet. Sie können also nach einem Verbindungsaufbau mit dem MySQL-Server SQL-Kommandos ausführen und deren Ergebnisse in einem Terminalfenster ansehen. Daneben kann `mysql` aber auch dazu verwendet werden, um Datenbank-Backups wieder einzuspielen oder um Administrationsarbeiten in Scripts automatisiert auszuführen. `mysql` ist damit das unscheinbarste und zugleich das wichtigste Administrationsprogramm für MySQL.

Das erklärt auch, warum der Abschnitt zu `mysql` so ausführlich ausfällt. Daneben gibt es einen weiteren Grund: Alle Optionen von `mysql`, die den Verbindungsaufbau zum MySQL-Server betreffen, gelten gleichermaßen auch für die meisten anderen `mysqlxxx` -Kommandos, also beispielsweise für `mysqladmin` oder `mysqldump`. Insofern ist dieser Abschnitt über die eigentliche Anwendung von `mysql` hinaus wichtig.

Hinweis

Es ist Ihnen sicher schon aufgefallen – der Begriff *mysql* hat je nach Kontext unterschiedliche Bedeutungen. Ich verwende in diesem eBook verschiedene Schreibweisen für die Begriffe:

- MySQL bezeichnet das Datenbanksystem bzw. den MySQL-Server in seiner Gesamtheit.
- `mysql` meint das gleichnamige Kommando, das MySQL-intern übrigens häufig auch *MySQL Monitor* genannt wird.
- *mysql* bezieht sich auf die gleichnamige Datenbank, in der der MySQL-Server Zugriffsrechte, Privilegien und eine Menge weiterer Verwaltungsinformationen für den internen Gebrauch speichert (siehe auch das Sicherheits-Kapitel).

mysql starten

Unter Unix/Linux bzw. OS X starten Sie `mysql` einfach in einem Terminalfenster. Sollte das nicht funktionieren, vergewissern Sie sich unter Linux, dass das Paket mit `mysql` installiert ist. (Bei vielen Distributionen ist `mysql` getrennt vom MySQL-Server im Paket `MySQL-client` enthalten.) Die wahrscheinlichste Fehlerursache unter OS X besteht darin, dass `mysql` üblicherweise in das Verzeichnis `/usr/local/mysql/bin` installiert wird, dieses Verzeichnis aber nicht in der Umgebungsvariablen `PATH` enthalten ist. Abhilfe: Geben Sie beim Start von `mysql` den gesamten Pfad an, oder fügen Sie `/usr/local/mysql/bin` zu `PATH` hinzu (am besten in `.bash_profile`).

Wenn Sie unter Windows arbeiten, führen Sie `mysql` innerhalb des Eingabeaufforderungsfensters aus. Damit das funktioniert, müssen Sie entweder den vollständigen Pfad angeben (in der Regel `C:\Program Files\MySQL\MySQL Server n.n\bin`) oder diesen Pfad zur Umgebungsvariablen `Path` hinzufügen (siehe auch das Kapitel Installation).

Verbindung zum MySQL-Server herstellen

Die Nutzung von `mysql` setzt voraus, dass das Programm eine Verbindung zu einem MySQL-Server herstellen kann. Der MySQL-Server darf durchaus auch auf einem Rechner laufen. Deswegen werden beim Start von `mysql` in der Regel zwei oder drei Optionen angegeben:

- `-u name` gibt an, mit welchem Benutzernamen `mysql` sich beim MySQL-Server identifizieren soll. Fehlt diese Information, verwendet `mysql` unter Unix, Linux und OS X den aktuellen Benutzernamen, unter Windows `ODBC`.
- `-p` bewirkt, dass `mysql` Sie nach dem Passwort fragt. Alternativ können Sie das Passwort auch unmittelbar an die Option `-p` hinzufügen (ohne Leerzeichen, also z. B. `-pganzGeheim`), das ist aber unsicher und sollte vermieden werden!
- Mit `-h host` geben Sie den Hostnamen des Rechners an, auf dem der MySQL-Server läuft. Verzichten Sie auf diese Option, nimmt `mysql` an, dass Sie mit `localhost` kommunizieren möchten.
  Ein Verbindungsaufbau zu einem externen MySQL-Server ist nur dann möglich, wenn der Server so konfiguriert ist, dass er nicht-lokale Verbindungen überhaupt zulässt. (Das ist aus Sicherheitsgründen oft nicht der Fall.) Außerden muss der TCP/IP-Port 3306 verwendbar sein. Stellen Sie sicher, dass dieser Port nicht durch eine Firewall blockiert wird!

In seltenen Fällen müssen Sie zwei weitere Optionen angeben:

- `--protocol=name` gibt an, welches Kommunikationsprotokoll Sie verwenden möchten. Wenn das Programm mysql und der MySQL-Server auf unterschiedlichen Rechnern laufen, kommt als einziges Protokoll `tcp` in Frage. Wenn `mysql` hingegen auf dem selben Rechner wie der MySQL-Server läuft, existieren je nach MySQL-Server-Konfiguration die Alternativen `socket` (Unix/Linux) bzw. `pipe` oder `memory` (beide Windows). Ohne die Option `-h` verwendet `mysql` unter Windows standardmäßig das Protokoll `tcp`, unter Unix/Linux das Protokoll `socket`.
- `-P n` gibt an, welche TCP/IP-Portnummer verwendet werden soll. Diese Option ist nur erforderlich, wenn der MySQL-Server *nicht* über den Defaultport 3306 kommuniziert.

Optional können Sie beim Start von `mysql` noch einen Datenbanknamen angeben. Diese Datenbank gilt dann für alle innerhalb von `mysql` ausgeführten Kommandos als Defaultdatenbank. Sie können die Defaultdatenbank jederzeit durch *USE datenbankname* ändern.

Ein typisches Kommando zum Start von `mysql` sieht so aus:

```
mysql -u root -p mycms
  Enter password: *********
```

Damit stellt `mysql` eine Verbindung zum MySQL-Server auf dem lokalen Rechner her. Beim Verbindungsaufbau müssen Sie das Passwort des MySQL-Benutzers *root* angeben. Nach dem Verbindungsaufbau gilt *mycms* als Defaultdatenbank!

Tipp

Sollte es Probleme beim Verbindungsaufbau geben, finden Sie im Abschnitt Mögliche Ursachen für Verbindungsprobleme eine Auflistung der häufigsten Fehlerursachen sowie Tipps zu deren Behebung. Wenn Sie unter Linux explizit eine TCP/IP-Verbindung zum lokalen MySQL-Server herstellen möchten, müssen Sie entweder `--protocol=tcp` oder `-h lokaler-hostname` bzw. `-h lokale-ip-adresse` angeben. Nach dem Verbindungsaufbau können Sie mit `status` feststellen, welche Art der Verbindung vorliegt.

```
mysql -u root -p --protocol=tcp
  Enter password: *********
  Welcome to the MySQL monitor. ...
mysql> status
  ...
  Connection:  uranus via TCP/IP
  TCP port:    3306
  ...
```

5 Datenbank-Design

Keine Angst, Inhalt dieses Kapitels ist nicht die hunderdste Wiederholung der Normalisierungsregeln! Vielmehr geht es hier um MySQL-spezifische Besonderheiten, die beim Entwurf neuer Datenbanken zu beachten sind. Sie finden in diesem Kapitel Informationen zu den folgenden Themen:

- Datentypen inklusive ENUM, SET und TIMESTAMP
- Zeichensatz und Sortierordnung
- AUTO_INCREMENT-Spalten
- GIS-Daten
- Primary Keys und Indizes
- Foreign Keys
- Volltextindizes
- Views
- Partitionen
- Stored Procedures, Trigger und Events
- Die information_schema-Datenbank

Ich gehe in diesem Kapitel davon aus, dass Sie in Ihren Datenbanken InnoDB-Tabellen verwenden. Das ist zwar schon seit MySQL 5.5 die Defaulteinstellung, auf überraschend vielen MySQL-Systemen kommen aber noch immer MyISAM-Tabellen zum Einsatz. Das ist mit wenigen Vorteilen, aber vielen Nachteilen verbunden (siehe auch den Abschnitt Engines).

5.1 CREATE TABLE

Neue Tabellen richten Sie entweder mit einem grafischen Administrationswerkzeug oder direkt durch die Ausführung von *CREATE TABLE* aus. Die Syntax von *CREATE TABLE* ist ziemlich komplex, weil MySQL dabei unzählige Sonderfälle und Varianten unterstützt. Die Beschreibung im MySQL-Handbuch erstreckt sich über viele Seiten.
Anstatt hier alle Syntaxvarianten zu wiederholen, geben die folgenden Zeilen drei Beispiele für die typische Anwendung von *CREATE TABLE*:

```
CREATE TABLE tname (spalte1 INT, spalte2 VARCHAR);

CREATE TABLE uploads (
  id INT NOT NULL AUTO_INCREMENT,
  orig_name VARCHAR(100) NOT NULL,
  filename VARCHAR(100) NOT NULL,
  mime VARCHAR(100) NOT NULL,
  filedate DATETIME DEFAULT NULL,
  thumb LONGBLOB,
  ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id),
) DEFAULT CHARSET=utf8;

CREATE TABLE pdfmail (
  id INT NOT NULL AUTO_INCREMENT,
  pdf INT NOT NULL,
  email VARCHAR(255),
  downcode VARCHAR(32),
  ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id),
  KEY fk_pdf (pdf),
  CONSTRAINT fk_pdf FOREIGN KEY (`pdf`) REFERENCES `pdf` (`id`));
```

Details zu den hier genannten Schlüsselwörtern folgen im weiteren Verlauf dieses Kapitels. Wenn Tabellen- oder Spaltennamen Sonderzeichen enthalten, können Sie den

Namen in schräge Apostrophe stellen, also z. B. *CREATE TABLE ‘tabellenname mit leerzeichen‘ (...)*.

Tipp

Mit dem Kommando *SHOW CREATE TABLE tname* können Sie jederzeit ermitteln, wie das Kommando *CREATE TABLE* für eine bereits existierende Tabelle aussieht. Das Kommando *CREATE TABLE neu LIKE alt* gibt Ihnen die Möglichkeit, auf unkomplizierte Weise eine neue Tabelle einzurichten, deren Aufbau exakt jenem einer bereits existierenden Tabelle entspricht.

5.2 Datentypen

Beim Einrichten neuer Tabellen haben Sie für jede Spalte die Wahl zwischen diversen Datentypen. Dieser Abschnitt fasst die Eckdaten aller MySQL-Datentypen zusammen. Beachten Sie, dass MySQL im Gegensatz zu vielen anderen Datenbanksystemen keine Möglichkeit bietet, selbst eigene Datentypen zu definieren!

Ganze Zahlen

Bei den *INT*-Datentypen sind normalerweise sowohl positive als auch negative Zahlen erlaubt. Mit dem nachgestellten Attribut *UNSIGNED* kann der Zahlenbereich ausschließlich auf positive Zahlen eingeschränkt werden. Details zum Datentyp *SERIAL*-Spalten folgen im Abschnitt AUTO_INCREMENT-Spalten.

Optional kann bei der Definition eines Integerfelds die gewünschte Spaltenbreite in *SELECT*-Ergebnissen angegeben werden (Maximum Display Size bzw. Stellenanzahl), also z. B. *INT(4)*. Beachten Sie aber, dass dadurch weder der zulässige Zahlenbereich noch die erlaubte Stellenanzahl einschränkt wird. Trotz INT(4) können Sie also beispielsweise Zahlen größer als 9999 speichern.

Datentyp	Bedeutung
TINYINT	Integer-Zahl (1 Byte)
SMALLINT	Integer-Zahl (2 Byte)
INT, *INTEGER*	Integer-Zahl (4 Byte)
BIGINT	Integer-Zahl (8 Byte)
SERIAL	entspricht *BIGINT AUTO_INCREMENT NOT NULL PRIMARY KEY*

Tabelle 5.1: Datentypen für ganze Zahlen

Achtung

Wenn Sie beim Einfügen neuer Datensätze den zulässigen Zahlenbereich überschreiten, ersetzt MySQL Ihre Werte standardmäßig durch die größte bzw. kleinste zulässige Zahl. Dieses Verhalten weicht von anderen Datenbanksystemen ab und lässt sich durch die Einstellung der Variablen *sql_mode* verändern (siehe den Abschnitt SQL-Modus).

Fließkommazahlen

Die Datentypen *FLOAT* und *DOUBLE* entsprechen den in vielen Programmiersprachen verfügbaren IEEE-Zahlentypen für einfache und doppelte Genauigkeit.

Datentyp	Bedeutung
FLOAT	Fließkommazahl (4 Byte)
DOUBLE	Fließkommazahl (8 Byte)
REAL	entspricht *DOUBLE*

Tabelle 5.2: Datentypen für Fließkommazahlen

Optional kann die gewünschte Anzahl von Stellen *FLOAT(M, D)* bzw. *DOUBLE(M, D)* eingestellt werden. In diesem Fall gibt *M* die Anzahl der Stellen vor dem Dezimalpunkt an, *D*

6 SQL

Einführungen in die *Standard Query Language* (SQL) gibt es schon genug – da will ich in diesem eBook nicht noch einen Aufguss hinzufügen. Ich gehe davon aus, dass Sie mit den Grundkonzepten von SQL vertraut und in der Lage sind, *SELECT*-, *INSERT*- und *UPDATE*-Kommandos zu formulieren.

Stattdessen konzentriert sich dieses Kapitel auf die vielen MySQL-spezifischen Besonderheiten in der SQL-Syntax bzw. deren Interpretation. Themen dieses Kapitels sind:

- *LIMIT*
- *GROUP_BY_CONCAT*
- XML-Funktionen
- Volltextsuche
- Transaktionen
- Variablen
- Sonstige Kommandos
- MySQL-Eigenheiten (SQL-Modus)

6.1 LIMIT

Mit *LIMIT* können Sie bei *SELECT*-Abfragen die Anzahl der Ergebnisdatensätze einschränken. Das folgende Kommando liefert 10 Datensätze aus der Tabelle *tname*:

```
SELECT * FROM tname LIMIT 10;
```

In der Regel sollten Sie *LIMIT* nur in Kombination mit *ORDER BY* einsetzen. Ohne *ORDER BY* ist nämlich nicht vorhersehbar, welche 10 Datensätze Sie erhalten. Das folgende Kommando liefert die zehn Datensätze mit dem jüngsten Rechnungsdatum:

```
SELECT * FROM tname ORDER BY invoicedate DESC LIMIT 10;
```

Häufig werden Sie *LIMIT* verwenden, um auf einer Website seitenweise Ergebnisse anzuzeigen, z. B. auf der Startseite die zehn aktuellsten Blogbeiträge, mit WEITER die nächsten zehn Beiträge etc. Dazu übergeben Sie *LIMIT* zwei Parameter, also *LIMIT offset, n*. Dabei gibt *offset* an, mit welchem Datensatz begonnen werden soll. Die Zählung der Datensätze beginnt mit 0! Ein Offset von 10 bedeutet daher, dass mit dem elften Datensatz begonnen wird.

```
--- die 10 aktuellsten Blog-Beiträge
SELECT * FROM blogtable ORDER BY publdate DESC LIMIT 10;

--- die nächsten 10 Blog-Beiträge
SELECT * FROM blogtable ORDER BY publdate DESC LIMIT 10, 10;

--- Position 21 bis 30
SELECT * FROM blogtable ORDER BY publdate DESC LIMIT 20, 10;
```

SQL_CALC_FOUND_ROWS, FOUND_ROWS

Wenn Sie eine *SELECT*-Abfrage mit *LIMIT* durchführen, erhalten Sie nur eine Teilmenge des Gesamtergebnisses. Gerade zum seitenweisen Anzeigen von Suchergebnissen wäre es aber oft hilfreich zu wissen, wie viele Datensätze insgesamt zur Verfügung stehen.

Diese Information gibt Ihnen die SQL-Funktion *FOUND_ROWS*, die Sie im Anschluss an die *SELECT*-Abfrage auswerten müssen. *FOUND_ROWS* liefert allerdings nur dann ein Ergebnis, wenn Sie *SELECT* mit der zusätzlichen Option *SQL_CALC_FOUND_ROWS* ausführen:

```
SELECT SQL_CALC_FOUND_ROWS * FROM blogtable
ORDER BY publdate DESC LIMIT 10;
  ...
SELECT FOUND_ROWS();
 110
```

blogtable enthält also 110 Blog-Einträge, von denen das *SELECT*-Kommando wegen *LIMIT* aber nur die ersten zehn zurückgibt.

Die Anwendung von *CALC_FOUND_ROWS* und *FOUND_ROWS* ist insbesondere bei komplexen Abfragen sinnvoll, bei denen ein separates *SELECT COUNT* zur Ermittlung der Datensatzanzahl zu zeitaufwändig wäre. Beachten Sie aber, dass die Option *CALC_FOUND_ROWS* gewisse Optimierungen verhindert, die MySQL bei *LIMIT*-Abfragen durchführt. Verwenden Sie *CALC_FOUND_ROWS* also nur, wenn Sie anschließend *FOUND_ROWS()* tatsächlich auswerten!

Datensätze ändern (UPDATE, DELETE)

So wie Sie mit *SELECT* die ersten/letzten *n* Datensätze durch *ORDER BY* und *LIMIT* auswählen können, so können Sie diese Datensätze auch mit *UPDATE* bzw. mit *DELETE* ändern.

Um Platz zu sparen, ersetzt *UPDATE* im folgenden Beispiel bei den ältesten 100 Nachrichten in der Tabelle *messages* den Nachrichtentext durch *message no longer available*. Wichtig ist dabei die Anweisung *ts=ts* für die *TIMESTAMP*-Spalte, die sicherstellt, dass *ts* unverändert bleibt. (Andernfalls würden aus den ältesten 100 Nachrichten scheinbar die 100 neuesten!)

```
UPDATE messages
SET msgText="message no longer available", ts=ts
ORDER BY ts LIMIT 100;
```

Noch einfacher ist es, die letzten 100 Nachrichten einfach zu löschen:

```
DELETE messages ORDER BY ts LIMIT 100;
```

6.2 GROUP_CONCAT-Aggregatsfunktion

Wie alle anderen Datenbanksysteme kann auch MySQL *SELECT*-Ergebnisse mit *GROUP BY* zusammenfassen und auf jede Gruppe von Datensätzen Aggregatsfunktionen wie *SUM* oder *COUNT* anwenden. Die folgende Abfrage ermittelt z. B., wie viele Nachrichten es von jedem Nachrichtentyp (Spalte *mtype* in der Spalte *messages*) gibt:

```
SELECT mtype, COUNT(*) FROM messages
GROUP BY mtype;
```

Eine ausgesprochen praktische Besonderheit von MySQL ist die Aggregatsfunktion *GROUP_CONCAT*: Sie ermöglicht es, die Zeichenketten einer Gruppe zusammenzufassen. An die Funktion übergeben Sie im einfachsten Fall nur den Spaltennamen. MySQL fügt die Zeichenketten einer Gruppe dann in willkürlicher Reihenfolge aneinander und trennt sie durch ein Komma. Durch *ORDER BY* können Sie die Zeichenketten sortieren, *SEPARATOR* gibt die gewünschten Trennzeichen an.

Für *GROUP_CONCAT* gibt es vielfältige Anwendungsmöglichkeiten, wie das folgende Beispiel beweist. Es liefert eine Tabelle mit allen Buchtiteln, die von mehr als einem Autor geschrieben wurden, samt einer Zeichenkette mit den alphabetisch geordneten Autoren. Ausgangspunkt für das Beispiel sind die Tabellen *titles* mit Buchtiteln, *authors* mit Autorennamen sowie *rel_title_author* zur n:m-Verknüpfung zwischen Titeln und Autoren (also mit der Information, welche Autoren an welchen Büchern mitgearbeitet haben).

```
SELECT title,
       GROUP_CONCAT(authname ORDER BY authname SEPARATOR ', ')
         AS authors,
       COUNT(authors.authID) AS cnt
FROM authors, titles, rel_title_author
WHERE authors.authID = rel_title_author.authID
AND   titles.titleID = rel_title_author.titleID
GROUP BY titles.titleID
HAVING cnt>1
ORDER BY title;
```

7 Administration

Zu den wichtigsten Administrationsthemen zählen das Erstellen von Backups, der Umgang mit Logging-Dateien, das Einrichten von Replikationssystemen, der Import und Export von Textdateien sowie die Verwaltung der MySQL- bzw. InnoDB-Tabellendateien.

7.1 Backup-Verfahren und -Werkzeuge

Grundsätzlich gibt es zwei Methoden, um ein Backup einer Datenbank zu erstellen: Sie können eine Sicherungskopie der Datenbankdateien vornehmen oder über den Server die eigentlichen Daten auslesen und so speichern, dass später ein einfaches Wiedereinspielen möglich ist (zumeist in Form von *INSERT*-Kommandos).

Bei beiden Varianten gibt es ein Problem: Die Dateien bzw. die Tabellen dürfen sich während des Backups nicht ändern! Andernfalls besteht die Gefahr, dass das Backup nicht konsistent ist (also beispielsweise eine Tabelle auf nicht mehr existente Datensätze einer anderen Tabelle verweist) bzw. die gesicherten Dateien gar korrupt sind.

Je nach Verfahren gibt es eine einfache Lösung: Sie fahren die Datenbank herunter, bevor Sie die Dateien kopieren, oder Sie stellen durch *LOCK TABLE* (MyISAM) bzw. durch eine Transaktion (InnoDB) sicher, dass sich die Daten während des Auslesens nicht ändern. Der Nachteil: Während des Backups ist die Datenbank mehr oder weniger stark beeinträchtigt. Da größere Backups unter Umständen stundenlang dauern, ist das selten eine akzeptable Lösung.

Nachdem nun klar ist, warum ein Datenbank-Backup nicht so einfach ist, wie es auf den ersten Blick aussieht, lohnt ein Blick auf die folgende Tabelle. Sie gibt einen Überblick über die wichtigsten Backup-Werkzeuge für MySQL.

Werkzeug	Beschreibung, Vor- und Nachteile
`mysqldump`	Dieses mit MySQL mitgelieferte Kommando ist der Klassiker unter den Backup-Werkzeugen. Sein größter Vorteil: Die resultierenden SQL-Dateien sind zwischen unterschiedlichen MySQL-Versionen weitgehend portabel. Leider ist das Kommando für große Datenbanken ungeeignet: Die Backup-Dateien sind riesig, sowohl das Backup als auch das Wiedereinspielen der Daten dauert schier endlos.
`mylvmbackup`	Dieses Open-Source-Script kann nur unter Linux eingesetzt werden, und auch nur dann, wenn sich das MySQL-Datenverzeichnis (zumeist `/var/lib/mysql`) in einem Logical Volume befindet. Sind diese Voraussetzungen erfüllt, ist `mylvmbackup` aber eine extrem effiziente Backup-Lösung, die gleichermaßen für InnoDB- und MyISAM-Tabellen geeignet ist.
`mysqlhotcopy`	Dieses früher sehr populäre Perl-Script wird mit MySQL mitgeliefert. Es führt *LOCK TABLE* aus und kopiert dann die Datenbankdateien. Es ist nur für MyISAM-Tabellen geeignet, läuft nur unter Unix/Linux und gilt als veraltet. Nicht empfehlenswert!
InnoDB Hot Backup	Dieses Backup-Werkzeug von Oracle erlaubt es, ein Backup von InnoDB-Tabellen im laufenden Betrieb durchzuführen. InnoDB Hot Backup bietet damit eine ähnliche Funktionalität wie `mylvmbackup`, ist aber nicht auf LVM oder ein anderes, snapshot-fähiges Dateisystem angewiesen. Sein größter Nachteil besteht darin, dass es nur Kunden der kommerziellen MySQL-Enterprise-Version zur Verfügung steht. Die Funktionsweise ist im InnoDB Hot Backup Manual beschrieben.

Tabelle 7.1: MySQL-Backup-Werkzeuge und -Verfahren

Werkzeug	Beschreibung, Vor- und Nachteile
Percona XtraBackup	Hierbei handelt es sich um eine Open-Source-Alternative zu InnoDB Hot Backup. Das Programm kann nicht nur für den Percona Server, sondern auch für gewöhnliche MySQL-Installationen verwendet werden. Eine gute Vergleichstabelle zwischen XtraBackup und InnoDB Hot Backup sowie Download-Links finden Sie auf der Percona-Website. Beachten Sie, dass XtraBackup als einziges Betriebssystem Linux unterstützt.
GUIs	Nahezu jedes MySQL-Administrations-Werkzeug mit grafischer Benutzeroberfläche bietet die Möglichkeit, Backups durchzuführen (MySQL Workbench, phpMyAdmin etc.). Für MySQL-Einsteiger ist das zumeist die einfachste Form, ein Backup zu erstellen. Die Backup-Methode ist aber selten im Hinblick auf maximale Effizienz optimiert und lässt sich zumeist nicht automatisieren (z. B. ein Backup jeden Sonntag nachts). Außerdem können die Backups vielfach nur mit dem jeweiligen Administrations-Werkzeug wieder eingelesen werden, was ein gravierender Nachteil ist.

Tabelle 7.1: MySQL-Backup-Werkzeuge und -Verfahren (Forts.)

Dieses Kapitel konzentriert sich im Folgenden auf zwei Kommandos: `mysqldump`, weil es der De-facto-Standard ist und jeder MySQL-Administrator damit umgehen können muss, und `mylvmbackup`, weil es die zur Zeit ideale Backup-Lösung für Linux-Anwender ist.

Backups versus Logging versus Replikation

Die Begriffe Backups, Logging und Replikation stiften bisweilen Verwirrung, weil sie alle in der einen oder anderen Form mit der Rekonstruktion von Daten zu tun haben. Am einfachsten ist sicherlich der Begriff Backup zu verstehen: Er bezeichnet eine zu einem bestimmten Zeitpunkt durchgeführte Sicherung aller Daten. Ein Backup ist bei großen Datenbanken ein aufwändiger Prozess, der zumeist nur einmal täglich oder gar nur einmal wöchentlich ausgeführt wird.

Was aber ist mit allen Änderungen, die zwischen dem letzten Backup und dem Ausfall eines Datenbank-Servers durchgeführt wurden? Bei MySQL können Sie alle Datenänderungen im sogenannten *Update Log* protokollieren. Die resultierenden Logging-Dateien in einem kompakten, binären Format ermöglichen es, alle Änderungen seit dem letzten Backup Schritt für Schritt nachzuvollziehen. Es ist zweckmäßig, die Logging-Dateien regelmäßig auf einen anderen Rechner zu kopieren, damit bei einem Totalausfall des Servers nicht auch die Logging-Dateien zerstört werden.

Von Replikation spricht man, wenn zwei Datenbanken auf unterschiedlichen Rechnern synchron gehalten werden. Jede Änderung am Master wird nahezu sofort auch am Replikations-Slave ausgeführt. Entgegen landläufiger Meinung ersetzt Replikation kein Backup: Wenn Sie am Master irrtümlich *DROP TABLE tname* ausführen, wird die Tabelle Sekunden später auch am Slave gelöscht! Replikation kann aber eine ideale Ergänzung zu regelmäßigen Backups sein, wenn jederzeit ein Ersatz-Server einsatzfähig sein soll: Fällt der Haupt-Server aus, kann der bisherige Replikations-Slave diese Rolle praktisch sofort übernehmen. Als Administrator müssen Sie nur ein paar Konfigurationszeilen verändern. Dagegen dauert das Einspielen eines großen Backups auf einem anderen Rechner unter Umständen Stunden oder Tage.

8 Tuning

In diesem abschließenden Kapitel geht es darum, MySQL möglichst effizient zu betreiben und vielleicht noch ein bisschen mehr Performance herauszukitzeln. Unter anderem behandelt dieses Kapitel die folgenden Themen:

- Server-Parameter (Grundeinstellungen, Tuning)
- Query Cache
- performance-Datenbank

Tipp

Dieses Kapitel gibt nur eine erste Einführung in das komplexe Materie des MySQL-Tunings! Das ultimative Referenzwerk zu diesem Thema ist das bereits in der dritten Auflage im O'Reilly Verlag erschienene 800-seitige Buch *High Performance MySQL* (amazon-Link).

8.1 Server-Parameter

my.cnf bzw. my.ini

Unzählige Parameter des MySQL-Servers werden beim Start des Programms aus der Konfigurationsdatei my.cnf (Unix/Linux) bzw. my.ini (Windows) gelesen. Der genaue Speicherort dieser Datei variiert. Üblicherweise befindet sich diese Datei an den folgenden Orten:

```
/etc/mysql/my.cnf oder /etc/my.cnf              (Linux)
/etc/my.cnf                                     (OS X)
C:\ProgramData\MySQL\MySQL Server 5.6           (Windows)
```

Unter Unix/Linux stellen Sie mit dem folgenden Kommando fest, welche Konfigurationsdatei(en) der MySQL-Server berücksichtigt:

```
mysqld --help --verbose | grep -A 1 'Default options'
  Default options are read from the following files in the given order:
  /etc/my.cnf /etc/mysql/my.cnf /usr/local/mysql/etc/my.cnf ~/.my.cnf
```

Achten Sie darauf, dass `my.cnf` nur von `root` veränderbar ist! Konfigurationsdateien, die für jederman schreibbar sind, werden vom MySQL-Server aus Sicherheitsgründen ignoriert.

Einstellungen in der zuletzt gelesenen Konfigurationsdatei haben Vorrang! Die serverspezifischen Optionen in der Konfigurationsdatei beginnen mit der Zeile `[mysqld]`. Die Einstellung `datadir = ...` gibt beispielsweise an, wo der MySQL-Server die Datenbankdateien speichert. Diese und eine Menge weiterer Server-Variablen werden im weiteren Verlauf dieses Kapitels vorgestellt.

```
# Beispiel für den serverspezifischen Teil von
# my.cnf (Linux) bzw. my.ini (Windows)
[mysqld]
...
datadir          = /var/lib/mysql
```

Änderungen an dieser Datei werden erst durch einen Neustart des MySQL-Servers wirksam! Unter Linux führen Sie dazu `service mysql restart` aus. Bei einigen Server-Variablen ist aber auch eine Veränderung im laufenden Betrieb durch *SET varname* möglich:

```
SET GLOBAL varname = ...   -- gilt, bis der MySQL-Server beendet wird
SET varname = ...          -- gilt nur für die aktuelle Verbindung
```

Den aktuellen Zustand der Server-Variablen stellen Sie mit *SHOW VARIABLES* fest. Die lange Ergebnisliste können Sie durch *LIKE '%name%'* einschränken.

Achtung

Wenn Sie in einer Konfigurationsdatei eine Option angeben, die `mysqld` nicht kennt (beispielsweise wegen eines banalen Tippfehlers), kann der Server nicht gestartet werden. Achten Sie also auf die korrekte Schreibweise! Fehlermeldungen beim Start werden in Error Log protokolliert. Die Logging-Datei finden Sie unter Linux zumeist in `/var/lib/mysql/hostname.err` oder in `/var/log/mysql/error.log`.

Grundeinstellungen

Die folgenden Tabellen fassen die wichtigsten Parameter zur Grundeinstellung des MySQL-Servers zusammen. Die Tabellen enthalten jeweils ein Beispiel für eine unter Linux übliche Einstellung bzw. für die Defaulteinstellung für MySQL 5.6.

Beachten Sie, dass alle hier aufgezählten Parameter auch in Form von Optionen beim Start des `mysqld`-Prozesses angegeben werden können (also z. B. `--tmpdir=xxx`). Eine Referenz aller Konfigurationsparameter bzw. `mysqld`-Optionen (es sind Hunderte!) inklusive deren Defaultwerte finden Sie wie üblich im MySQL-Handbuch.

Hinweis

Innerhalb von `my.cnf` können mehrteilige MySQL-Parameter wahlweise in der Form `bind_address` oder `bind-address` angegeben werden, also mit Unterstrich oder mit Bindestrich. Beide Schreibweisen sind gleichwertig!

Parameter	Bedeutung
`basedir = /usr`	gibt das Basisverzeichnis der MySQL-Installation (Binärdateien) an.
`datadir = /var/lib/mysql`	gibt das Verzeichnis an, in dem die Datenbankdateien gespeichert werden.
`tmpdir = /tmp`	gibt den Ort für temporäre Dateien an.

Tabelle 8.1: Grundeinstellungen

Die Einstellung des Parameters `basedir` in `my.cnf` wird in MySQL 5.6 nicht mehr empfohlen. In vielen Fällen ist die Angabe dieses Parameters gar nicht erforderlich; falls doch, ist es sicherer, das Basisverzeichnis beim Start von `mysqld` mit der Option `--basedir=...` einzustellen.

Kommunikation und Sicherheit

Parameter	Bedeutung
`socket = /var/run/mysqld/mysqld.sock`	gibt den Ort der Socketdatei an.
`port = 3306`	gibt an, über welchen IP-Port der MySQL-Server kommuniziert.
`skip-networking`	deaktiviert Netzwerkverbindungen. Die Kommunikation muss über die Socket-Datei erfolgen.
`bind-address = 127.0.0.1`	akzeptiert Netzwerkverbindungen ausschließlich von localhost.
`user=mysql`	gibt den Unix/Linux-Account an, unter dem der `mysqld`-Prozess ausgeführt werden soll.
`skip-grant-tables`	ignoriert die eingestellten MySQL-Zugriffsrechte. Jeder kann sich ohne Passwort beim MySQL-Server anmelden.

Tabelle 8.2: Kommunikation und Sicherheit

Wenn auf einem Web-Server sowohl Apache als auch MySQL ausgeführt werden, ist in der Regel die Einstellung `bind-address = 127.0.0.1` empfehlenswert. Sie verhindert, dass Netzwerkverbindungen von außen hergestellt werden können.

Die Option `skip-grant-tables` ist nur für Reparaturmaßnahmen gedacht (z. B., wenn Sie das root-Passwort vergessen haben). Sie sollte mit `skip-networking` kombiniert werden.

Über ebooks.kofler

ebooks.kofler ist entstanden, um kleinere Textmengen in Form von eBooks zu vermarkten. ebooks.kofler bietet qualitativ hochwertige, kompakte, preisgünstige, lesefreundliche Texte, die von Anfang an als eBooks konzipiert sind.

Die Vorteile gegenüber herkömmlichen Informationsmedien sind einerseits die rasche Verfügbarkeit (kein umständlicher Bestell- und Versandprozess wie bei Büchern), andererseits der hohe Qualitätsanspruch (keine veralteten Informationen auf unübersichtlichen, mit Werbung überladenen Web-Seiten).

eBooks für Programmierer, Administratoren und IT-Profis

eBooks von ebooks.kofler richten sich an Computer-Anwender, Programmierer oder Administratoren. Sie liefern konkrete Lösungsanleitungen und helfen dabei, anstehende Aufgaben rasch und effizient zu lösen.

Themenfokussiert

Nichts ist mühsamer, als lange Texte am Computer zu lesen. Deswegen sind die eBooks von ebooks.kofler auf ein Thema fokussiert. Die eBooks beginnen nicht bei Adam und Eva, sondern da, wo es interessant wird. Und es besteht keine Notwendigkeit, den Text auf eine für den Buchhandel taugliche Länge »aufzublasen«, wie es in der Praxis leider oft passiert. (Vor dem Buchregal gilt vielfach die Devise: Dicker ist besser.) Ihr Vorteil: einerseits ein günstiger Preis, andererseits Textmengen, die gut am Bildschirm bewältigt werden können.

Bildschirm-optimiertes Layout

Nahezu alle momentan erhältlichen IT-eBooks sind »Recycling-Produkte«. Das Rezept: Man nehme ein EDV-Fachbuch, verpacke es in eine PDF-Datei und verkaufe es dann als eBook. Das Layout solcher eBooks ist jedoch für den Druck optimiert.

Doch nicht alles, was in gedruckter Form gut aussieht, ist für die Lektüre am Bildschirm geeignet. eBooks von ebooks.kofler sind von Anfang an für das Lesen am Bildschirm konzipiert:

- ausreichende Schriftgröße
- serifenlose Schriften
- Farben
- anklickbare Web-Links und Querverweise im Buch

Das Layout von ebooks.kofler wurde dahingehend entwickelt, dass eBooks in einer Doppelseiten-Ansicht auf einem (Notebook)-Monitor mit 1280*800 Pixeln gut lesbar sind.

Hoher Qualitätsanspruch

eBooks von ebooks.kofler erfüllen hohe Qualitätsansprüche: inhaltlich, sprachlich und ästhetisch! Alle Texte werden professionell korrekturgelesen.

PDF ohne DRM

eBooks von ebooks.kofler werden als PDF-Dateien frei von DRM-Schutz ausgeliefert. Damit können Sie Ihr eBook dort lesen, wo Sie es brauchen - auf Ihrem Computer. In optimaler Darstellungsqualität - auch bei Tabellen, Listings, Abbildungen etc.

Sie dürfen die PDF-Datei auf beliebig viele Computer/Geräte kopieren, ausdrucken, Textpassagen (Listings) per Cut&Paste kopieren etc. Es gibt nur eine Einschränkung: Sie dürfen das eBook nicht an andere Personen weitergeben, weder in elektronischer Form noch als Ausdruck.

Autor werden

Wollen Sie selbst eBooks für ebooks.kofler schreiben? Schreiben Sie eine E-Mail an kontakt@kofler.info!