

HOWTO zur MySQL-Administration

(C) 2006-2011 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>

OSTC Open Source Training and Consulting GmbH, <http://www.ostc.de>

\$Id: mysql-admin-HOWTO.txt,v 1.128 2011-10-21 14:26:14 tsbirn Exp \$

Dieses Dokument beschreibt den MySQL-Einsatz auf Administrationsseite.

HINWEIS: MySQL-Spezialitäten sind durch "MY!" oder "MY!N.M" gekennzeichnet (falls sie erst ab MySQL Version N.M (nicht mehr) verfügbar sind).

HINWEIS: Der Begriff "Datenbank" wird häufig "schwammig" verwendet. MySQL ist ein "Datenbank-Managementsystem" (DBMS), das die Verwaltung vieler "Datenbanken" gleichzeitig erlaubt. Jede Datenbank besteht aus Tabellen und weiteren Datenbank-Objekten, die zur Abbildung eines konkreten Sachverhalts verwendet werden. Statt der (langen) Begriffe "Datenbank-Managementsystem" oder (abgekürzt) "Datenbank-System" wird gerne der kurze Begriff "Datenbank" verwendet. Ein besserer Begriff für eine eigentliche "Datenbank" ist daher "Schema", so besteht keine Verwechslungsgefahr, was gemeint ist. Das DBMS Oracle z.B. vermeidet den Begriff "Datenbank" und verwendet "Schema". Für "Datenbank" wird in diesem Skript "DB" oder <Db> als Abkürzung verwendet.

INHALTSVERZEICHNIS

- 1) Das Datenbankmanagementsystem MySQL
 - 1a) Grundlegendes
 - 1b) Allgemeine Informationen
 - 1c) Eigenschaften/Einschränkungen der MySQL-Versionen
- 2) Installation der MySQL-Software
 - 2a) Allgemein
 - 2b) MySQL-Datenbankserver installieren (SuSE)
- 3) Serveradministration
 - 3a) Server starten und stoppen
 - 3b) Konfiguration: Variablen und Schalter
 - 3c) Konfigurations- und Datenbankdateien
 - 3d) Interne Verwaltungs-Datenbanken
- 4) mysqladmin-Befehle
- 5) Benutzer und Zugriffsrechte (Privileges)
- 6) Datenbank-Engines (Storage Engines/Backends)
- 7) Datenbanken erstellen und verwalten
- 8) Tabellen erstellen und verwalten
- 9) Tabellen prüfen und warten
- 10) Datenimport und -export
- 11) Datensicherung und -wiederherstellung
- 12) Überwachung und Protokolldateien
- 13) Replikation von Datenbanken (Master-Slave, Master-Master)
- 14) Gesicherte und verschlüsselte Verbindungen
- 15) Troubleshooting
- 16) Query Optimizer
- 17) Tabellen-, Index- und Query-Cache
- 18) MySQL-Proxy
- 19) Partitionierung

1) Das Datenbankmanagementsystem MySQL

1a) Grundlegendes

GROSS/Kleinschreibung wird (fast) nicht berücksichtigt:

-> mysql-HOWTO.txt -> 2) Syntax, Formatierung und Kommentar

UNIX-Zugriffsrechte und -Besitzverhältnisse sind wichtig:

* Meist Benutzer "mysql" + Gruppe "mysql"

Unterschiedliche Programme:

```
* Client:      /usr/bin/mysql
* Server:      /usr/sbin/mysqld
* Start-Skript: /etc/init.d/mysql bzw. rcmysql # start/stop/status/restart
* Aktivieren:  chkconfig -a mysql # add (SuSE)
               inserv mysql # insert
```

Optionen zu MySQL-Kommandos können auf der Kommandozeile und in Konfigurationsdateien angegeben werden. Auf der Kommandozeile ist der Bindestrich "-" als Bestandteil üblich, in den Konfigurationsdateien der Unterstrich "_". D.h. es gibt eine Kommandozeilen-Option

```
--log-bin=XXX
```

und eine (identische) Konfigurations-Option für die Konfigurations-Dateien

```
log_bin = XXX
```

Man darf aber die Unterstriche und Bindestriche gegeneinander austauschen (und sogar in einer Option "mischen")!

Optionen im richtigen Abschnitt der Konfigurationsdatei (z.B. "my.cnf") setzen (Programmname):

Abschnitt	Bedeutung
[server]	Alle Server allgemein
[client]	Alle Clients allgemein
[mysqld]	Server "mysqld" speziell
[mysql]	Client "mysql" speziell
[mysqladmin]	Client "mysqladmin" speziell
...	usw.

Zum Ein-/Ausschalten einer "Booleschen Option" OPTION gibt es mehrere Möglichkeiten:

Schreibweise	Bed.
--OPTION	An
--OPTION=1	An
--enable-OPTION	An
--OPTION=0	Aus
--disable-OPTION	Aus
--skip-OPTION	Aus

Viele Optionen sind auch im Rahmen einer bestehenden Datenbank-Verbindung änderbar:

--read-buffer-size=32M	Option
read_buffer_size = 32M	Konfigurationsdatei-Variable
SET @@read_buffer_size = 32M;	MySQL-Anweisung

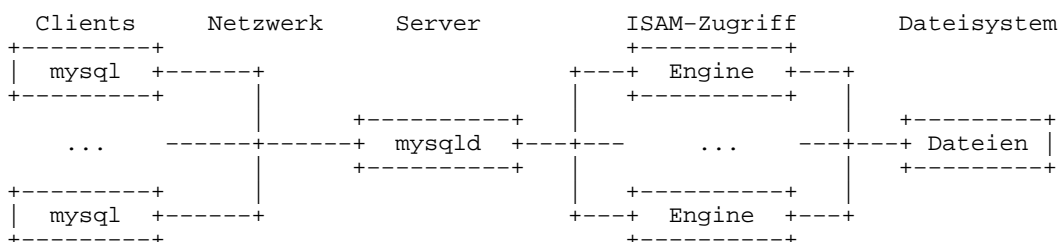
1b) Allgemeine Informationen

Links:

* http://www.mysql.de	# MySQL-Informationen (deu)
* http://www.mysql.com	# MySQL-Informationen (eng)
* http://www.mysql.org	# MySQL-Informationen (eng)
* http://dev.mysql.com	# MySQL-Referenz-Manual (eng)
* http://shop.mysql.com	# MySQL-Produkte, Support, Training (eng)
* http://www.apachefriends.org	# XAMPP (LAMP, WAMP)
* http://www.phpmyadmin.net	# phpMyAdmin
* http://www.sleepycat.com	# BDB-Hersteller (inzw. Oracle!)
* http://www.innodb.com	# InnoDB-Hersteller (inzw. Oracle!)
* http://www.infinidb.org	# InfiniDB (Calpont, http://www.calpont.com)

Architektur:

- * N Clients + 1 Server (auf anderem oder gleichen Rechner)
 - + Client: mysql (Kommandoschnittstelle, einer von vielen)
 - + Server: mysqld
- * 3 Server-Schichten
 - + Oben: Benutzerschnittstelle (SQL, DB-Management)
 - + Mitte: Storage Engine
 - + Unten: Dateisystem



Zusammenhang Rechner, DB-Server, Datenbank, Tabelle, Index, Spalte, Zeile:

R Rechner haben # (meist einer)

```

S MySQL-Server pro Rechner mit      # mysqld (meist einer)
D Datenbanken pro Server mit        # /var/lib/mysql/<Db>
T Tabellen pro Datenbank mit        # /var/lib/mysql/<Db>/<Tbl>.*
  C Spalten pro Tabelle und
  Z Datensätze/Zeilen pro Tabelle
  I Indices pro Tabelle
P Prozeduren pro Datenbank          # In Verwaltungstabelle
F Funktionen pro Datenbank          # In Verwaltungstabelle
G Trigger pro Datenbank             # In Verwaltungstabelle
V Views pro Datenbank               # In Verwaltungstabelle
E Events pro Datenbank              # In Verwaltungstabelle

```

Storage Engines (pro Tabelle festlegbar, Std: MyISAM):
-> mysql-admin-HOWTO.txt -> 6) Datenbank-Engines (Storage Engines)

Schnittstellen von MySQL:

- * Client Libraries (Programmier-APIs)
 - + PHP
 - + Perl-DBI
 - + ODBC (Windows)
 - + JDBC (Java)
 - + C/C++
 - + Ruby
 - + Python
- * Kommandozeilentools (Verwaltungswerkzeuge + Dienstprogramme)
- * Grafische Oberflächen (GUI)

Einsatzgebiete:

- * Web-Umfeld (Performanz, Verfügbarkeit, keine Transaktionen notwendig)
- * Dynamische Webseiten (in Kombination mit PHP)
- * Webserver auf LAMP-Basis (Linux, Apache, MySQL, PHP/Perl, Wikipedia)

1c) Eigenschaften/Einschränkungen der MySQL-Versionen

Vorteile von MySQL:

- * Einfache Installation/Administration
- * Schnell (durch Verzicht auf Funktionalitäten)
- * Stabil
- * Einfache Nutzung
- * Viele Schnittstellen
- * Portabel
- * Zusammen mit PHP ausgeliefert (gut integriert)
- * Unterschiedliche Engines pro Tabelle/DB möglich
- * Kosten

Nachteile von MySQL:

- * ANSI SQL-Standard nicht vollständig erfüllt
- * Referenzielle Integrität nicht sichergestellt (außer bei InnoDB)
- * ACID-Prinzip nicht durchgehalten ("Atomizität" statt Transaktionen)
(Atomic, Consistent, Isolated, Durable)
- * "Spiel/Web-Datenbank" - auf Lesen optimiert
- * Zusammen mit PHP ausgeliefert (gut integriert)
- * Performanceprobleme bei Join mehrerer Tabellen)

Defizite von MySQL (laut MySQL-Mitgründer Michael "Monty" Widenius):

- * Siehe: <http://www.scribd.com/doc/2575733/The-future-of-MySQL-The-Project>
- * Schlechte Skalierung ab 8 CPUs/Cores
- * Nicht optimale RAM-Nutzung
- * Keine externe Benutzerauthentifizierung (etwa via LDAP)
- * Ineffizienter interner SQL-Parser
- * Mangelnde Community-Beteiligung
- * Unklare Release-Politik

Versionen:

- * Enterprise Edition Kostenpflichtig
- * Community Edition Alle stabilen Defaultfunktionen (GPL)

Distribution:

- * Binär (pro OS optimiert: Server, Client, Dienstprogramme, Engines, Lizenz)
 - + Zuschneiden auf HW-Architektur + Distribution notwendig
- * Quellcode (GPL)
 - + C/C++-Compiler + Bibliotheken notwendig

Duale Lizenzierung:

- * Nur EINE Lizenz pro Maschine (auch wenn Multiprozessor!) notwendig!
 - + Keine Einschränkung bzgl. Anzahl Prozessoren und Anzahl Clients
- * Kein Kauf einer kommerziellen Lizenz nötig bei:
 - + Normalem internen Gebrauch
 - + Webserver unter Unix betreiben (auch wenn dieser kommerziell ist)
 - + Internet Service Provider (hosten Kunden-Webserver)

- + Auf MySQL beruhende Anwendung unterliegt auch der GPL
- + Client-Code in kommerzielles Programm integrieren
- * Kommerzielle Lizenz nötig:
 - + MySQL als Embedded Server benutzen
 - + Erweiterungen des MySQL-Servers verwenden
 - + Kommerz. Anwendung, die NUR mit MySQL funktioniert und es mit ausliefert
 - + Distribution von MySQL, die nicht Quelltext zur Verfügung stellt
- * Non-GPL (kommerzielle) Preise/Lizenzen:

Versionsbezeichnungen:

- * Development (Entwickler)
- * Alpha/Beta/Gamma-Test
- * Release Candidate (RC)
- * Generally Available (GA) = Production Ready

ANSI-SQL-Standardkompatibilität:

- * Option --ansi (nur ANSI-SQL akzeptieren)
 - || verkettet Zeichenketten statt OR
 - FUNC...(Leerzeichen vor "(" erlaubt (FUNCNAME dann reserviertes Wort)
 - "..." zur Quotierung von Namen die Schlüsselworte sind (statt `...`)
 - REAL = FLOAT statt DOUBLE
 - Std.-Isolationslevel ist SERIALIZABLE
- * Analog --sql-mode=REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, \
 - IGNORE_SPACE, SERIALIZE, ONLY_FULL_GROUP_BY
- * Code in /*!...*/ wird von MySQL ausgeführt, nicht von anderen Datenbanken
- * Code in /*!32302...*/ wird von MySQL-Version 3.23.02 oder neuer ausgeführt
- * Kommentar "--" MUSS Leerzeichen danach haben
- * Nur `...` als String-Quotierung erlaubt, "..." quotiert SQL-Schlüsselworte (ansonsten quotiert `...` MySQL-Schlüsselworte)

Versionsübersicht:

Eigenschaft	Ver	Bemerkung
Transakt. eingeschr.	3.X	3.20/21/22/23
SSL-Verschlüsselung	4.0	Client-Server Kommunikation
InnoDB-Engine	4.0	Standardmäßig vorhanden
Replikation	4.0	
Fulltext search	4.0	
Unions	4.0	
Transaktionen voll	4.0	Nur InnoDB und BDB
RAW-Partitionen	4.0	Nur InnoDB
Referenz. Integrität	4.0	Nur InnoDB
Subqueries	4.1	
Unicode Support	4.1	Verschiedene Zeichensatz-Collations
Prepared Statements	4.1	
Geometric Datatype	4.1	GIS=Geometric Information System (nur MyISAM)
R-tree Index	4.1	GIS=Geometric Information System (nur MyISAM)
Views	4.1	Readonly = derived table
Clustering	4.1	NDB Engine = Network Database
Konstante TRUE/FALSE	4.1	
Stored Procedures	5.0	
Functions (UDF)	5.0	User Defined Functions (extern in C program.)
Views	5.0	Updateable
Cursors	5.0	Read only
Trigger	5.0	Rudimentär
Precision Math	5.0	Kompakte DECIMAL, keine DOUBLE-Rechnung mehr
ARCHIVE Engine	5.0	
FEDERATED Engine	5.0	
BIT Datatype	5.0	
Information Schema	5.0	Metainfo in virtuellen Tabellen
XA Transaktionen	5.0	?
Trigger	5.1	Voll
Foreign keys	5.1	Ab 3.23 für InnoDB
Full outer join	5.1	Nein!
Constraints	5.1	
Event Scheduler	5.1	
XPath-Support	5.1	XML (2 neue Funk: ExtractValue, UpdateXML)
Information Schema	5.1	Stark erweitert
Partitionierung	5.1	SUN: Tabellen/Indices - RANGE/HASH/KEY
Row-based Replikation	5.1	Alt: SQL-Kommando ausgeführt, Neu: Satzbasiert
Column Level Constr.	5.1	
Online Backup	5.1	Ohne Performanz-Einbußen
Engine Plugin-API	5.1	Engines nachträglich hinzufügen/entfernen
Logs in DB ablegen	5.1	Für General+Slow Query Log (nicht Binary Log)
mysqlnd-Bibliothek	5.1	Für PHP (ersetzt libmysql)
Upgrade Program	5.1	"mysql_upgrade"
MySQL Workbench	5.1	Visueller Entwurf von DB-Schemata

Okt 22, 11 3:00

mysql-admin-HOWTO.txt

Page 5/37

MySQL Migration Toolkit		OK: Tabellen+Inhalt; NEIN: Prozed., Trigger
MySQL Enterprise Monitor		MySQL-Serverstatus verfolgen, Optimierung
Google Patches	5.4	Skalierbarkeit (InnoDB-Backend beschleunigt)
Subquery/Join	5.4	Beschleunigung durch Optimizer
Stored Procedures	5.4	SIGNAL/RESIGNAL, SQL-Fehlerbehandlung
Information Schema	5.4	PARAMETERS, ROUTINES
Prepared Statements	5.4	OUT Parameter Support
Semisynchronous Repl.	5.5	Master block. Tr. bis mind. 1 Slave bestätigt
SIGNAL/RESIGNAL	5.5	Fehler in Stored Proz/Funkt/Handler melden
Partitionierung	5.5	Zerleg. anhand meh. Spalten/String/Datum/Zeit
	5.5	(RANGE COLUMNS, LIST COLUMNS)
XML Funktionalität	5.5	Erweitert (LOAD XML)
InnoDB-Plugin	5.5	Standard-Engine (fest eingebaut)
performance_schema	5.5	Interne Tabellen zur Performanz
Authentifizierung	5.5	Alternative PAM/LDAP/..., PROXY USER
Foreign keys	6.0	Alle Engines einheitlich
Optimizer	6.0	Subqueries und Joins beschleunigen
FALCON Engine	6.0	Transaktionsfähig
Maria Engine	6.0	Transaktionsfähig
Information Schema	6.0	Erweitert (Parameters, Routines)
Online Backup+Restore	6.0	
Unicode Support	6.0	Erweitert (utf16, utf32, utf8)
LOCK TABLES Syntax	6.0	Erweitert (IN SHARE/EXCLUSIVE MODE)

2) Installation der MySQL-Software

2a) Allgemein

Binär-TAR-Archiv Linux:

```
groupadd mysql
useradd -g mysql -d /usr/local/mysql mysql
passwd mysql
mkdir /usr/local/mysql
tar xzvf TAR-ARCHIV -C /usr/local # -> mysql-SERVER-VERS-pc-linux-i686
ln -s ... mysql
mkdir /usr/local/mysql
scripts/mysql_install_db # -> System-DB "mysql" erzeugen
chown -R root /usr/local/mysql
chown -R mysql /usr/local/mysql/data
chgrp -R mysql /usr/local/mysql
chmod -R
PATH=$PATH:/usr/local/mysql/bin
```

Binär-RPM-Paket Linux:

```
rpm -i MySQL-server-VERS.i386.rpm MySQL-client-VERS.i386.rpm
Verzeichnisse
/usr/sbin          mysqld
/etc/init.d        mysql-Dienst-Skript
/usr/bin           mysql_install_db, mysql, mysqladmin, mysqlshow, mysqldump,
/usr/local/bin    ...
/var/lib/mysql    Datenbanken (mysql, test, ...)
```

Quellcode-TAR-Archiv Linux:

```
mysql-VERS.tar.gz
MySQL-VERS.src.rpm
groupadd mysql
useradd -g mysql -d /usr/local/mysql mysql
passwd mysql
mkdir /usr/local/mysql
mkdir /usr/local/archiv
tar -xzvf TAR-ARCHIV -C /usr/local/archiv/mysql-VERS
ln -s mysql-VERS mysql
cd /usr/local/mysql
./configure --prefix=/usr/local/mysql --with-mysql-user=mysql \
--with-charset=german1

./configure --help
make
make install
scripts/mysql_install_db # -> System-DB "mysql" erzeugen
chown -R root /usr/local/mysql
chown -R mysql /usr/local/mysql/data
chgrp -R mysql /usr/local/mysql
PATH=$PATH:/usr/local/mysql/bin
cp support-files/my-medium.cnf /etc/mysql/my.cnf
```

```
-> libexec/mysql
   /usr/local/mysql/var/DATABASE
   /bin/mysql
```

Quellcode-RPM-Paket Linux

```
rpm --rebuild /PFAD/ZU/MySQL-VERS.src.rpm:
-> /usr/src/packages/RPMS/i386/*.rpm
rpm -i ...
```

Windows-Verzeichnisse:

```
C:/mysql/bin
   /data
   /share
   /docs
```

2b) MySQL-Datenbankserver installieren (SuSE)

0. Benutzer "mysql" und Gruppe "mysql" (oder "daemon") anlegen, falls noch nicht vorhanden.

1. "mysql" in YaST2-Softwareinstallation suchen und alle Pakete auswählen die das Wort "mysql" enthalten (mindestens "mysql", "mysql-client" und "phpmyadmin").

2. Damit der MySQL-Datenbankserver beim Booten automatisch gestartet wird:

Ab SuSE 8.0: Im YaST2 => System => Runlevel-Editor => (*) Expertenmodus den Eintrag "mysql" in den Runleveln 2, 3 und 5 aktivieren UND sofort starten (ALT-K = "Aktivieren"). Alternativ:

```
insserv mysql           # ins=insert, serv=server oder
chkconfig -a mysql     # chk=check, -a=add
```

Bis SuSE 7.3: In "/etc/rc.config" Variable START_MYSQL auf "yes" setzen:

```
START_MYSQL="yes"
```

3. Den MySQL-Datenbankserver per Hand starten (oder neu booten), wenn er sofort benutzt werden soll (rc=run control):

```
/etc/init.d/mysql start      # Starten
/etc/init.d/mysql stop      # Anhalten
/etc/init.d/mysql restart   # Anhalten + Starten
/etc/init.d/mysql status    # Läuft er oder steht er?
```

Unter SuSE-Linux gibt es folgende Abkürzungen:

```
rcmysql start            # Starten
rcmysql stop            # Anhalten
rcmysql restart         # Anhalten + Starten
rcmysql status          # Läuft er oder steht er?
```

4. Für den MySQL-Datenbankserver wird ein eigener LINUX-Benutzer "mysql" angelegt. Seine Dateien und Prozesse sind diesem Benutzer und der Gruppe "mysql/daemon" zugeordnet.

3) Serveradministration

3a) Server starten und stoppen

MySQL-Server NICHT mit root-Rechten starten!

Linux:

```
mysql           # Client-Programm
mysqld          # Server-Dämon
mysqld_safe     # Automat. Neustart, Protokollierung
/etc/init.d/mysql # Dienst-Skript (init, start, stop, status, restart)
rcmysqld       # SuSE Dienst-Skript-Link (init)
```

In Boot-Skript einbinden:

```
cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql
ln -s ../mysql /etc/init.d/rc3.d/S99mysql
ln -s ../mysql /etc/init.d/rc3.d/K01mysql
ln -s ../mysql /etc/init.d/rc5.d/S99mysql
ln -s ../mysql /etc/init.d/rc5.d/K01mysql
```

Windows:

- * Konsole
 - mysql -nt.exe OPTIONEN
- * Dienst
 - net start mysql
 - net stop mysql
- * Desktop-Verknüpfung
- * Windowsspezifische Optionen
 - console # Ausgaben auf Konsole
 - standalone # Als Einzelanwendung starten
 - install # Dienst einrichten
 - remove # Dienst entfernen
 - use-symbolic-links # Symbolische Links erlaubt

Mehrere MySQL-Server gleichzeitig starten:

- * Grund
 - + Verschiedene MySQL-Server-Versionen gleichzeitig notwendig (Test)
 - + Strenge Trennung von Datenbanken/Tabellen
 - + Namenskollisionen von Datenbanken/Tabellen verschiedener Anwendungen
- * Unterschiedliche Konfigurationen
 - netstat -an | grep -i mysql
 - netstat -an | grep 3306
 - ps aux | grep -i mysql
- * Port + Socket zuweisen
 - port=3307 --socket=/tmp/mysql.sock
- * MÜSSEN unterschiedliche Datenverz. verwenden!
 - datadir=PATH

MySQL-Server herunterfahren:

```
mysqladmin -uroot -p shutdown
```

Kennwort für "root" festlegen und "anonymen Benutzer" entfernen:

```
mysql -uroot
USE mysql;
SELECT user, host, password FROM user WHERE user = "";
DELETE FROM user WHERE user = "";
SET PASSWORD FOR root@<Host> = PASSWORD("geheim");
FLUSH PRIVILEGES;
SELECT user, host, password FROM user WHERE user = "";
mysqladmin -uroot password geheim
```

3b) Konfiguration: Variablen und Schalter

Variablen anzeigen (einige):

```
SHOW VARIABLES;
basedir # MySQL-Verz.
datadir # Datenbank-Verz.
have_innodb # InnoDB-Tabellentypen möglich
max_connections # Anzahl gleichzeitig max. möglicher Verbindungen
port # Port
```

Konfiguration:

- * Generelle Arten (aufsteigender Vorrang!)
 - C) Umgebungsvariablen
 - B) Konfigurations-Dateien (mehrere, siehe unten)
 - A) Kommandozeile
- * Optionen: Kurzform -p und Langform --password
 - + In Konfig.dateien immer Langform ohne "--" verwenden
- * Auf Kmdo.zeile Server-Variablen setzen: --VAR=WERT
- * Konfigurations-Dateien
 - + Format: Abschnitte (Gruppen) mit Einstellungen (allgemein und speziell)
 - [server] # Alle Server allgemein
 - [client] # Alle Clients allgemein
 - [mysqld] # Server "mysqld" speziell
 - [mysql] # Client "mysql" speziell
 - [mysqladmin] # Client "mysqladmin" speziell
 - + Linux (aufsteigender Vorrang!)
 - B1) Global (alle Server) /etc/mysql/my.cnf /usr/etc/my.cnf
 - B2) Serverspezifisch /var/lib/mysql/my.cnf
 - B3) Zusätzliche Konfig.datei --defaults-extra-file=FILE
 - B4) Benutzerspezifisch ~/.my.cnf
 - + Windows (aufsteigender Vorrang!)
 - B1) Windows-Systemverzeichnis C:\Windows\system32\my.cnf
 - B2) Windows-Wurzelverzeichnis C:\my.cnf
 - B3) MySQL-Installationsverzeichnis INSTALLDIR\my.ini
 - B4) Zusätzliche Konfig.datei --defaults-extra-file=FILE
- * Abschalten der Konfigurations-Dateien
 - defaults-file=FILE

```
--no-defaults
* Beispiel für eine Konfigurations-Datei:
(weitere Beispiele in Verz. "support-files" (Linux))
```

```
[client]
password = geheim
port      = 3306
socket    = /var/lib/mysql/mysql.sock

[mysqld]
port      = 3306
socket    = /var/lib/mysql/mysql.sock
set-variable = max_allowed_packet=1M
skip-locking
```

Einige wichtige Optionen des Servers "mysqld":

Option	Bedeutung
--help/--?	Hilfe anzeigen
--ansi	ANSI-SQL erzwingen
--datadir=PATH	Verz. für Datenbank-Dateien
--log=FILE/-l FILE	Logdatei
--port=PORT/-P PORT	Portnummer (Std: 3306)
--safe-mode	Einige Optimierungen überspringen (zu Testzwecken)
--safe-show-database	Nur Datenbanken mit Zugriffsberechtigung anzeigen
--skip-show-database	Keine Anzeige vorhandener Datenbanken
--socket=FILE	Socket-Datei
--user=USER/-u USER	Benutzername für Prozess
--version/-V	Versionsnummer

3c) Konfigurations- und Datenbankdateien

Für MySQL relevante Konfigurationsdateien (die Ziffer in Klammern gibt die Reihenfolge an, in der sie beim Aufruf von "mysql" gelesen werden):

Datei	Bedeutung + Reihenfolge
UMGEBUNGSVARIABLEN	0) Werden immer überschrieben
/etc/my.cnf	1) Zentrale Konfiguration (für alle Datenbanken)
/var/lib/mysql/my.cnf	2) Zentrale Konfiguration (pro Datenbank)
~/my.cnf	3) Lokale Konfiguration (pro Benutzer)
KOMMANDOZEILEN-PARAM	4) Überschreiben (0) bis (3)
~/mysql_history	Bisher eingegebene MySQL-Befehle (pro Benutzer)
~/mysqlgui/*	Diverse Dateien für GUI-Konfiguration
~/mysqlgui/administrator/*	Konfig. "mysql-navigator"-Oberfläche
~/mysqlgui/query-browser/*	Konfig. "mysql-query-browser"-Oberfläche

Ablageort der internen Datenbank "mysql" ("privilege/private DB") und der Benutzerdatenbanken bei "MyISAM" (pro Datenbank ein Verz., pro Tabelle in einer Datenbank im Verz. zu dieser Datenbank mehrere Dateien angelegt):

Datei	Inhalt
/var/lib/mysql/mysql/*	Interne Datenbank "mysql"
/var/lib/mysql/<Db>/*	Benutzerdatenbank <Db>
/var/lib/mysql/<Db>/db.opt	Einstellungen für Datenbank <Db>
/var/lib/mysql/<Db>/<Tbl>.frm	Schema von <Tbl> (rekonstruierbar)
/var/lib/mysql/<Db>/<Tbl>.MYD	MyISAM-Daten von <Tbl> (NICHT rekonstr.)
/var/lib/mysql/<Db>/<Tbl>.MYI	MyISAM-Indices von <Tbl> (rekonstr.bar)
/var/lib/mysql/<Db>/<Tbl>.TRG	Trigger von <Tbl> (rekonstruierbar)
/var/lib/mysql/<Db>/<Tbl>.CSV	CSV-Daten von <Tbl> (im CSV-Format)
/var/lib/mysql/<Db>/<Tbl>.CSM	CSV-Metadaten von <Tbl> (im CSV-Format)
/var/lib/mysql/<Db>/<Tbl>.ISD	ISAM-Daten (veraltet)
/var/lib/mysql/<Db>/<Tbl>.ISM	ISAM-Metadaten (veraltet)
/var/lib/mysql/<Db>/<Tbl>.ibd	InnoDB-Daten + Indices (File pro Tab.)
/var/lib/mysql/ibdata1,...	InnoDB-Tablespace (alle Tab. in 1 File)
/var/lib/mysql/ib_logfile1	InnoDB-Transaktionslog (alle Tab. 1 F)
/var/lib/mysql/<Host>.err	Logging-Info. für Rechner <Host>

Ablageort der Programme und sonstiger Dateien (Linux):

Verzeichnis	Inhalt
-------------	--------

/etc/*	MySQL-Konfigurationsdateien
/usr/local/mysql/...	MySQL-Dateien bei Quellcode-Installation
/var/lib/mysql/mysql.sock	Socket für Verbindung zur Datenbank
/var/run/mysqld/mysqld.sock	(analog)
/usr/bin/*	MySQL-Client-Programme (Administration)
/usr/sbin/*	MySQL-Server-Programme
/usr/sbin/mysqld	MySQL-Datenbank-Server-Dämon
/usr/share/mysql/*	Konfigurationsdateien, Skripte, Sprachen
/usr/include/mysql/*	C-Include-Dateien (für C-Programmierung)
/usr/lib/mysql/*	Programm-Bibliotheken
/usr/share/sql-bench/*	Benchmarks (run-all-tests)

Dokumentation:

```
/usr/share/doc/packages/mysql/*
/usr/share/doc/packages/mysql-navigator/*
```

3d) Interne Verwaltungs-Datenbanken

Der MySQL-Datenbankserver besitzt zwei interne Datenbanken namens "mysql" (privilege/private DB) und "information_schema" zur internen Verwaltung mit folgenden internen Tabellen (MY!).

Interne Datenbank "mysql" (wird manipuliert und enthält echte Verwaltungsdaten):

Table	Bedeutung	Right	Dump
columns_priv	Spalten-spezifische Zugriffsrechte	J	J
db	Datenbank-spezifische Zugriffsrechte	J	J
event	Events-Verwaltung		J
func	Prozedur/Funktions-Liste		J
general_log	Allgemeine Logs		
help_...	Tabellen mit Hilfetexten		
host	Host+Datenbank-spezifische Zugriffsrechte	J	J
ndb_binlog_index	NDB-Verwaltung		
plugin	Plugin-Verwaltung		
proc	Prozedur/Funktions-Definitionen		J
procs_priv	Prozedur/Funktions-spezifische Zugriffsrechte	J	J
servers	Server-Verwaltung	?	J
slow_log	Langsame Logs		
tables_priv	Tabellen-spezifische Zugriffsrechte	J	J
time_...	Zeitzone-Tabellen		
user	Benutzer (Host, Passwort, Basis-Zugriffsrechte)	J	J

HINWEIS: In Spalte "Right" mit "J" markierte Tabellen enthalten Zugriffsrechte (GRANTS). In Spalte "Dump" mit "J" markierte Tabellen sollten im Rahmen einer Sicherung der Datenbank gesichert und beim Wiederherstellen der Datenbank zurückgespielt werden. Die restlichen Tabellen besser durch eine Neuinstallation des DB-Systems wiederherstellen.

Beispiele:

```
USE mysql;           # Zu Verwaltungsdatenbank wechseln
SHOW TABLES;       # Alle Verwaltungstabellen anzeigen
SHOW COLUMNS FROM columns_priv; # Verwaltungstabellen-Inhalt anzeigen
EXPLAIN db;          # Verwaltungstabellen-Inhalt anzeigen
EXPLAIN views;       # Verwaltungstabellen-Inhalt anzeigen
```

Interne Datenbank "information_schema" (Views auf "mysql", nicht änderbar):

Table	Bedeutung
CHARACTER_SETS	Zeichensätze
COLLATIONS	Zeichen-Abbildungen für Sortierung
COLLATION_CHARACTER_SET_APPLICABILITY	Erlaubte Kombinationen
COLUMNS	Spalten-Definitionen
COLUMN_PRIVILEGES	Spalten-Zugriffsrechte
ENGINES	Storage Engines
EVENTS	Ereignisse
FILES	
GLOBAL_STATUS	Server-Status
GLOBAL_VARIABLES	Server-Variablen
KEY_COLUMN_USAGE	Info für "Referenzielle Integrität"
PARTITIONS	Tabellen-Partitionen
PLUGINS	Plugin-Module
PROCESSLIST	Prozessliste
PROFILING	

REFERENTIAL_CONSTRAINTS		
ROUTINES		Prozedur/Funktions-Definitionen
SCHEMATA		Datenbanken
SCHEMA_PRIVILEGES		Datenbank-Zugriffsrechte
SESSION_STATUS		Sitzungs-Status
SESSION_VARIABLES		Sitzungs-Variablen
STATISTICS		Statistik-Informationen
TABLES		Tabellen-Definitionen
TABLE_CONSTRAINTS		Tabellen-Spalten-Beschränkungen
TABLE_PRIVILEGES		Tabellen-Zugriffsrechte
TRIGGERS		Trigger-Definitionen
USER_PRIVILEGES		Benutzer-Zugriffsrechte
VIEWS		View-Definitionen

HINWEIS: Diese interne Datenbank entspricht dem Standard, den auch andere Datenbanken kennen. GROSS/Kleinschreibung der Tabellennamen ist nicht relevant.

Beispiele:

```
SHOW TABLES FROM information_schema;  # Alle Verwaltungstabellen anzeigen
USE information_schema;                 # Zu Verwaltungsdatenbank wechseln
SHOW tables;                           # Alle Verwaltungstabellen anzeigen
SHOW COLUMNS FROM tables;             # Verwaltungstabellen-Inhalt anzeigen
EXPLAIN <Tbl>;                          # Tabellendefinition anzeigen
```

4) mysqladmin-Befehle

"mysqladmin" ist eine kommandozeilen-orientierte Administrations-Schnittstelle von MySQL, mit der die wichtigsten Verwaltungsaufgaben erledigt werden können. Alle seine Funktionen können auch mit "mysql" + SQL-Befehlen durchgeführt werden, "mysqladmin" bietet aber oft einen schnelleren Weg dafür an:

```
mysqladmin [OPTIONS] COMMAND [CMDOPTS] ...
```

Befehl	Bedeutung	SQL-Anweisung
create <Db>	Datenbank anlegen	CREATE DATABASE <Db>
drop <Db>	Datenbank löschen	DROP DATABASE <Db>
password <Pass>	Setzt Benutzer-Passwort auf <Pass>	SET PASSWORD=<Pass>
shutdown	Führt "mysqld" herunter	%
ping	Prüft ob "mysqld" läuft	%
status	Zeigt abgekürzten Server-Status	%
extended-status	Zeigt alle Server-Konfig.-Var.	SHOW STATUS
variables	Listet alle verfügbaren Var.	SHOW VARIABLES
version	Gibt MySQL-Version aus	%
processlist	Zeigt alle aktiven MySQL-Threads	SHOW PROCESSLIST
kill <Pid>[,...]	Beendet MySQL-Thread <Pid>,...	KILL <Pid>, ...
debug	Debug-Info in Fehler-Log schreiben	%
flush-hosts	Flusht alle gecachten Hosts	FLUSH HOSTS
flush-logs	Flusht alle Logs	FLUSH LOGS
flush-status	Flusht alle Status-Variablen	FLUSH STATUS
flush-tables	Flusht alle Tabellen	FLUSH TABLE[S]
flush-threads	Flusht den Thread-Cache	%
flush-privileges	Lädt alle GRANT-Tabellen neu	FLUSH PRIVILEGES
reload	Lädt alle GRANT-Tabellen neu	FLUSH PRIVILEGES
refresh	Flusht Tab., schliesst+öffnet Logs	%
start-slave	Startet Replikation auf Slave-Serv.	%
stop-slave	Stoppt Replikation auf Slave-Server	%

5) Benutzer und Zugriffsrechte (Privileges)

Zweck:

- * Authentifizieren von Benutzern
 - + Abhängig vom Rechner/IP-Adresse von dem/der aus Benutzer sich verbindet
- * Zuweisen von Zugriffs-Rechten (Privileges) an sie
- * Zuweisen von Administrations-Rechten (Privileges) an sie
- * Standardbenutzer
 - + "Anonymer Benutzer" (leerer Name+Passwort, nur Zugriffsrecht USAGE)
 - + "Administrator" (leeres Passwort!, alle Zugriffsrechte)
- * Rollen (Gruppen) = Zugriffsrechte für Menge von Benutzern NICHT mögl. (MY!)

HINWEIS: Die Benutzer von MySQL haben mit den Betriebssystem-Benutzern nichts

zu tun (auch wenn der Standard-Administrator "root" heißt).

Eigenschaften:

- * Alle Passworte sind verschlüsselt abgelegt!
- * Es wird unterschieden, WOHER ein Benutzer sich verbindet (Host)
- * Standardmäßig kann ein Benutzer nur selbst erstellte Datenbanken <Db> und Objekte <DbObj> bearbeiten (er ist der "Besitzer" dieser Objekte)
- * GROSS/Kleinschreibung der Benutzernamen ist relevant!
- * Passwort wird automatisch verschlüsselt!
- * Kombination Benutzer + Host (d.h. Quelle der Verbindung spielt eine Rolle)
- * In Ebenen gegliedert: Datenbank, Tabelle, Spalte, Routine
- * Additiv (OR), d.h. gegebenes Recht auf höherer Ebene nicht durch Fehlen des Rechts auf niedriger Ebene wegnehmbar
- * Zugriffsrechte mit REVOKE entziehen
 - + Transitiv (d.h. an andere Benutzer weitergegebene Rechte auch entzogen)
 - + Recht mehrfach an einen Benutzer weitergeben
 - > Alle müssen es ihm entziehen, damit er es nicht mehr hat
- * Auch Rechte für noch nicht existierende Objekte sind anlegbar
- * Regeln zu Datenbank/Tabelle werden beim Löschen dieser NICHT entfernt
- * Regeln zu Routine werden beim Löschen dieser entfernt
- * Anonymer Benutzer = leerer Name ("@"localhost")

HINWEIS: GRANT erstellt automatisch einen NEUEN Benutzer, wenn der Username neu ist (z.B. bei Schreibfehler!; nur sofern NO_AUTO_CREATE_USER Modus inaktiv)

Beschränkungen:

- * Ablehnen der Verbindung eines vorhandenen Benutzers ist NICHT möglich (Mindestrecht USAGE)
- * Anlegen/Löschen von Tabellen in einer Datenbank -> Datenbank selbst auch
- * Es gibt keine Benutzergruppen ("Rollen")!
 - + Aber Rechte sind von einem Benutzer auf andere kopierbar
- * Nach Anlegen neuer Benutzer, Ändern von Rechten, Löschen von Benutzern MUSS Berechtigungstabelle neu eingelesen werden (FLUSH PRIVILEGES).
- * Neuer MySQL-Release -> evtl. Änderungen an Rechte-Tabellen -> Update nötig
- * REVOKE löscht Benutzer nicht -> DROP/DELETE notwendig
- * REVOKE von Rechten muss exakt einem vorhergehenden GRANT entsprechen
- * Einschalten der Rechte für EINEN Benutzer aktiviert Prüfung für ALLE (d.h. Server wird etwas langsamer)
- * Einschalten der Begrenzungen für Anfragen, Updates, Verbindungen für EINEN Benutzer aktiviert Prüfung für ALLE (d.h. Server wird etwas langsamer)
- * Rechte-Änderungen während Verbindung wirkt nicht sofort (FLUSH PRIVILEGES)

Neuen Benutzer erzeugen und/oder ihm Rechte ("Privilege Types") zuweisen:

```
GRANT <Privilege> [( <Col>, ...)]           # Rechte + best. Spalten
  [, <Privilege> [( <Col>, ...)]...]       # Weitere Rechte + Spalten
ON [TABLE | PROCEDURE | FUNCTION] <DbObj> # Datenbank-Objekt
TO "<User>"@"<Host>"                       # Benutzer (Wer und woher?)
  [IDENTIFIED BY [PASSWORD] "<Pass>"]     # Passwort
  [, TO ... [IDENTIFIED BY ...]...]       # Weitere Benutzer+Passwort
  [REQUIRE <SSLOpt> [AND <SSLOpt> ...]] # Verschlüsselung
  [WITH [GRANT OPTION]                   # Weitergabe von Rechten
    [MAX_QUERIES_PER_HOUR <Zahl>]       # Beschränkung
    [MAX_UPDATES_PER_HOUR <Zahl>]       # Beschränkung
    [MAX_CONNECTIONS_PER_HOUR <Zahl>]   # Beschränkung
    [MAX_USER_CONNECTIONS <Zahl>]]      # Beschränkung
FLUSH PRIVILEGES;                          # Nicht vergessen!
```

ACHTUNG: Vorsicht mit "GRANT OPTION", zwei Benutzer mit "GRANT OPTION" können sich gegenseitig ihre Rechte weitergeben und sie so kombinieren!

TIPP: Ergänzen von Rechten ohne bereits vorhandene Rechte zu beeinflussen:
GRANT USAGE ON ... TO ... WITH ...;

HINWEIS: Nach erfolgreichem GRANT/REVOKE erscheint die Meldung "Query OK, 0 rows affected", nach schiefgegangenen GRANT/REVOKE erscheint eine Meldung "ERROR ...". Bitte nicht davon verwirren lassen.

Alle Rechte eines Benutzers anzeigen (ACHTUNG: Schreibweise "tom"@"localhost" notwendig, nicht die Hochkommas ganz aussenrum setzen):

```
SHOW GRANTS;                               # Eigene/Alle
SHOW GRANTS FOR tom;                       # Für Benutzer "tom"
SHOW GRANTS FOR @"localhost";             # Für anonymen Benutzer
SHOW GRANTS FOR "tom"@"localhost";        # Für Benutzer "tom" von Host "localhost"
```

Benutzer OHNE Zugriffsrechte anlegen (ab MY!5.0, nur USAGE-Recht, dann GRANT...):

```
CREATE USER <User>;                          # Leeres Passwort
CREATE USER <User> IDENTIFIED BY "geheim";   # Klartext-Passwort
CREATE USER <User> IDENTIFIED BY PASSWORD "46...F"; # Verschlüsselte Passw.
```

Benutzerdaten (Name, Passwort, allgemein) ändern:

```

RENAME USER <Name> TO <NewName>, ...;           # Umbenennen
SET PASSWORD [FOR <User>] = PASSWORD("geheim"); # Klartext-Passwort
SET PASSWORD = PASSWORD("geheim");             # Aktueller Benutzer
SET PASSWORD [FOR <User>] = "46...F"          # Verschlüsseltes Passw.
GRANT USAGE ON ... TO ... IDENTIFIED BY "geheim"; # Verschlüsseltes Passw.
UPDATE user SET <Col> = <Wert> WHERE <Bedingung>; # Allgemein (Vorsicht!)
FLUSH PRIVILEGES;                             # Nicht vergessen!

```

Benutzer Rechte wegnehmen (nur wenn vorher per GRANT exakt so gegeben):

```

REVOKE <Privilege> [( <Col>, ...)]              # Rechte + best. Spalten
  ON [TABLE | PROCEDURE | FUNCTION] <DbObj>    # Datenbank-Objekt
  FROM "<User>"@"<Host>", ...                  # Wer und woher?
FLUSH PRIVILEGES;                             # Nicht vergessen!

```

ALLE Rechte eines Benutzers wegnehmen (hat danach nur noch USAGE-Recht):

```

REVOKE ALL PRIVILEGES, GRANT OPTION ON *.* FROM <User>, ...;

```

Benutzer löschen (mit allen seinen Rechten; von ihm erzeugte Datenbankobjekte bleiben erhalten; aktuell mit ihm bestehende Sitzungen werden nicht beendet):

```

DROP USER <User>;                             # Inkl. Rechte (bis MY!5.0 davor REVOKE nötig!)
DELETE FROM user WHERE user = "<User>"        # Bestimmten Benutzer löschen
  AND host = "<Host>";
DROP USER ""@"localhost";                     # Anonymen Benutzer löschen!
DELETE FROM user WHERE user = "";             # Anonymen Benutzer löschen!

DELETE FROM user WHERE password = "";         # Benutzer mit leerem Passwort löschen!
DELETE FROM db WHERE host = "%";             # Anmeldung von überall her löschen

```

Beispiel: Benutzer "user" auf Rechner "localhost" das Einfügen + Lesen + Ändern auf allen Tabellen der Datenbank "first" erlauben (Löschen von Daten ist ihm dann verboten - REPLACE geht somit auch nicht!):

```

GRANT INSERT, SELECT, UPDATE
  ON first.*
  TO "tom"@"localhost";                       # Von Rechner "localhost" aus

GRANT INSERT, SELECT, UPDATE, DELETE
  ON first.*
  TO "tom"%%;                                 # Nun auch REPLACE erlaubt
  # Egal woher (auch nur "tom" möglich)

GRANT INSERT(id), SELECT(id, name)
  ON first.pers
  TO "tom"@"localhost";                       # Nur bestimmte Spalten einer
  # Tabelle erlaubt

```

Beispiel: UPDATE-Recht von Benutzer auf Datenbank "first" wegnehmen:

```

REVOKE UPDATE
  ON first.*
  FROM "tom"@"localhost";

```

Anfangs sind 2 Benutzer eingerichtet (der Benutzer "root" der MySQL-Datenbank hat nichts mit dem Benutzer "root" des Linux-Systems zu tun!):

```

* Benutzer "root":      Uneingeschränkt, leeres Passwort! -> Passwort setzen!
* Anonymer Benutzer "": Nur Verbindung, leeres Passwort! -> User löschen!

```

Als allererstes MUSS das LEERE root-Passwort geändert werden, sonst kann JEDER die Datenbank administrieren (u=user, h=host, beim 2. Mal ist schon das Passwort "geheim" notwendig!):

```

mysqladmin -uroot password "geheim"          # Rechner egal
mysqladmin -uroot -hcharlton.site -pgeheim password "geheim" # Rechner ...

```

Ebenso unbedingt den "anonymen Benutzer" löschen:

```

DELETE FROM user WHERE user = "";            # Anonymen Benutzer löschen!

```

Danach muss beim Aufruf von "mysql" und "mysqladmin" immer die Option "-p" gesetzt werden, die eine Passwortabfrage durchführt (u=user, p=password):

```

mysqladmin -uroot -p...

```

ACHTUNG: Passworte NIE auf der Kommandozeile eintippen (insbesondere das des Benutzers "root"). Grund: Die für jeden einsehbare Prozesstabelle enthält die vollständige Kommandozeile, das Passwort ist dann also "verbrannt".

ACHTUNG: Die Kommando-Historie wird in "~/.mysql_history" gespeichert. Falls

diese Datei lesbar ist, können evtl. nach PASSWORD verwendete Klartextpassworte kompromittiert sein.

Auf lokalem Rechner den Benutzer "tom" (TO "tom"@"localhost") mit Passwort "geheim" (IDENTIFIED BY "geheim") einrichten und ihm ALLE Rechte (ALL) auf ALLEN Datenbanken und Tabellen (*.*) erlauben. Weiterhin darf er seine eigenen Rechte auch an andere Benutzer weitergeben (WITH GRANT OPTION) (ACHTUNG: Maximalbeispiel, User "tom"@"localhost" darf anschließend überall alles!):

```
GRANT ALL PRIVILEGES          # Was (welche Rechte)
ON *.*                        # Welche Datenbank/Tabelle
TO "tom"@"localhost"         # Wem (User + Host)
IDENTIFIED BY "geheim"       # Passwort (optional, Hochkommas!)
WITH GRANT OPTION;           # Rechteweitergabe (optional)
```

Es können auch Einträge mit den Rechnernamen "localhost.invalid", "localhost" und "127.0.0.1" nötig sein, da solche Namen bei Verbindungsaufnahmen mit PHP, Perl oder auf OpenBSD-Systemen verwendet werden (Rechte für ALLE Rechner setzen nicht mit "tom%", sondern mit "tom")!

Anschließend kann man sich folgendermaßen als dieser Benutzer anmelden (-h=host, -u=user, -p=password):

```
mysql -hlocalhost -utom -pgeheim # Variante A (Passwort mitgegeben)
mysql -hlocalhost -utom -p        # Variante B (Passwort abgefragt)
password: geheim
```

Über folgenden Alias "mq" bzw. "mqe" (execute) kann z.B. auch ohne Passwortabfrage eine Verbindung zur Datenbank "prod" aufgenommen bzw. sofort ein Kommando an sie abgesetzt werden:

```
alias mq="mysql -hlocalhost -utom -pgeheim -Dprod" # bzw.
alias mqe="mysql -hlocalhost -utom -pgeheim -Dprod -e" # e=execute
```

Aufrufbeispiel:

```
mq          # MySQL-Prompt erscheint
mqe "SHOW DATABASES" # Ergebnis wird ausgegeben
```

Die Rechte ("Y"=erhalten, "N"=nicht erhalten) stehen in Tabelle "user" und einigen anderen ("db", "host", "table_priv", "columns_priv", "procs_priv").

Auflisten aller prinzipiell möglichen Rechte <Privilege> mit:

```
SHOW PRIVILEGES;
```

liefert folgende Tabelle (Spalte "Bezug" siehe unten):

Recht/Privilege	Bezug	Bedeutung
ALL [PRIVILEGES]	ALLE	Alle Rechte außer GRANT OPTION
USAGE	S	Keine Rechte (nur Anmelden/Verbindung möglich)
SHOW DATABASES	S	Alle Datenbanken auflisten (SHOW DATABASES)
CREATE USER	S	User erz./löschen/umbenennen + Rechte widerrufen
GRANT OPTION	ALLE	Eigene Rechte anderen User weitergeben/wegnehmen
FILE	S	Export/Import in/aus Datei auf dem Server
PROCESS	S	Prozess-Threads einsehen (ausgeführter SQL-Code)
RELOAD	S	Interne Tabellen/Logs/Rechte auffrischen (FLUSH)
SHUTDOWN	S	Datenbank-Server herunterfahren (mysqladmin)
SUPER	S	Verwalten (KILL, SET GLOBAL, CHANGE MASTER TO, PURGE BINARY LOGS)
BACKUP	S	
RESTORE	S	
PROXY	S	
SELECT	TC	Daten aus Tabellen lesen (für REPLACE nötig!)
INSERT	TC	Daten in Tabellen einfügen (für REPLACE nötig!)
UPDATE	TC	Daten in Tabellen ändern
DELETE	T	Daten aus Tabellen löschen (für REPLACE nötig!)
ALTER	T	Tabellenstruktur verändern
CREATE	DT	Datenbanken/Tabellen anlegen (Views nicht!)
DROP	DT	Datenbanken/Tabellen/Views löschen
INDEX	T	Indices anlegen/löschen
REFERENCES	DT	Referenz auf Tab. besitzen (nicht implementiert)
CREATE TABLESPACE	S	
CREATE TEMPORARY TABLES	D	Temporäre Tabellen anlegen ("stirbt" am Sitzungsende)

Okt 22, 11 3:00

mysql-admin-HOWTO.txt

Page 14/37

LOCK TABLES	D	Tabellen sperren (SELECT-Recht notwendig!)
CREATE VIEW	T V	View anlegen
SHOW VIEW	T V	View ansehen (SHOW CREATE VIEW)
TRIGGER	T G	Trigger erstellen/löschen (ab 5.1.6)
EVENT	D E	Events benutzen (ab 5.1.6)
CREATE ROUTINE	R	Anlegen von Prozeduren/Funktionen
ALTER ROUTINE	R	Ändern/Löschen von Prozeduren/Funktionen
EXECUTE	R	Ausführen von Prozeduren/Funktionen
REPLICATION CLIENT	S	Statusdaten von Master/Slave-Servern auslesen
REPLICATION SLAVE	S	Binärlog-Events von Master-Server lesen

Rechtebezug: "S" = Server (global)
 "D" = Datenbank
 "T" = Tabelle
 "C" = Spalte (column)
 ("I" = Index)
 "G" = Trigger
 "E" = Event
 "R" = Routine
 "V" = View

Schreibweise für Datenbank-Objekte <DbObj> in GRANT/REVOKE-Anweisung:

Objekt	Bedeutung
.	Alle Datenbanken + ihre Tabellen (global)
*	Alle Tabellen der mit "USE" gewählten Datenbank
<Db>.*	Alle Tabellen der Datenbank <Db>
<Db>.<Tbl>	Tabelle <Tbl> der Datenbank <Db>
<Tbl>	Tabelle <Tbl> der mit "USE" gewählten Datenbank
<Db>.<Proc>	Routine <Proc> der Datenbank <Db>
<Proc>	Routine <Proc> der mit "USE" gewählten Datenbank

Regelt Zugriff auf: Server, Datenbanken, Tabellen, Spalten, Routinen
 (stehen in der Systemdatenbank "mysql"). Typen sind:

- * Global: Server + Datenbanken
- * Datenbank: Datenbanken + Tabellen
- * Tabellen: Tabelle + Spalten
- * Spalten: Spalten
- * Routinen: Datenbanken + Routinen

Schreibweise für Benutzer in GRANT/REVOKE-Anweisung (Gänsefüßchen bei Verwendung der SQL-Wildcards "%" und "_" notwendig, legt fest, WOHER sich er sich anmelden darf). In Spalte "OK" steht "ja", wenn die entsprechende Form sicher ist, die anderen Schreibweisen sind unsicher.

Benutzer	OK	Bedeutung
"<User>"@"<Host>"	ja	Benutzer + Host-Name (kein Wildcard notwendig)
"<User>"@"<IP>"	ja	Benutzer + Host-IP (kein Wildcard notwendig)
"<User>"@"%.ostc.de"	ja	User von Host aus Domain "ostc.de"
"<User>"@"192.168.1.%"	ja	User von Host aus IP-Bereich "192.168.1.*"
%@"<Host>"		Beliebiger User von Rechner mit Host-Name
%@"<IP>"		Beliebiger User von Rechner mit Host-IP
"<User>"@%		User von bel. Host (identisch mit <User>)
<User>		User von bel. Host
localhost	ja	Nur vom Server-Rechner (UNIX Domain-Socket)
127.0.0.1	ja	Nur vom Server-Rechner (TCP-Socket)
% oder <Leer>		Beliebiger User von beliebigem Host

Beispiele für eine sinnvolle Kombination von Benutzer-Zugriffsrechten für die Benutzer "leser", "erfasser", "kontrolleur", "entwickler", "backup", "admin"

```
GRANT SELECT                                # Nur lesender Zugriff
ON first.*                                  # auf bestimmte DB
TO "leser"@"localhost"                     IDENTIFIED BY "geheim",
"erfasser"@"localhost"                     IDENTIFIED BY "geheim",
"kontrolleur"@"localhost"                  IDENTIFIED BY "geheim",
"entwickler"@"localhost"                   IDENTIFIED BY "geheim",
"backup"@"localhost"                       IDENTIFIED BY "geheim";

GRANT UPDATE, INSERT, EXECUTE,              # + Daten ändern
CREATE TEMPORARY TABLES                   # (weder Löschen noch REPLACE)
ON first.*                                  # auf bestimmter DB
TO "erfasser"@"localhost"                  IDENTIFIED BY "geheim",
"kontrolleur"@"localhost"                  IDENTIFIED BY "geheim",
"entwickler"@"localhost"                    IDENTIFIED BY "geheim";
```

```

GRANT DELETE                                # + Daten löschen (und REPLACE)
ON first.*                                  # auf bestimmter DB
TO "kontrolleur"@"localhost" IDENTIFIED BY "geheim",
   "entwickler"@"localhost" IDENTIFIED BY "geheim";

GRANT LOCK TABLES                          # + Tabellen sperren
ON first.*                                  # in bestimmter DB
TO "entwickler"@"localhost" IDENTIFIED BY "geheim",
   "backup"@"localhost" IDENTIFIED BY "geheim";

GRANT ALTER, CREATE, DROP, INDEX,           # + Tabellen + Indices verwalten
REFERENCES,                                 # + Foreign Keys verwalten
CREATE VIEW, SHOW VIEW,                    # + Views verwalten
TRIGGER,                                    # + Trigger verwalten
CREATE ROUTINE, ALTER ROUTINE               # + Stored Procedures verwalten
ON first.*                                  # in bestimmter DB
TO "entwickler"@"localhost" IDENTIFIED BY "geheim";

GRANT ALL PRIVILEGES                         # Alle Rechte (AUSSER Rechtevergabe)
ON *.*                                      # auf allen Datenbanken
TO "admin"@"localhost" IDENTIFIED BY "geheim";

```

Das letzte Statement umfasst folgende zusätzlichen Rechte für den "admin":

```

FILE                                         # + Dateien laden + speichern
SHOW DATABASES,                             # + Datenbank auflisten
CREATE USER, GRANT OPTION,                   # + Benutzer + Rechte verwalten
PROCESS, RELOAD, SHUTDOWN, SUPER            # + Datenbankserver verwalten

```

ACHTUNG: Bei der Anmeldung eines Users <User> von(!) einem Rechner <Host> wird in der obiger Reihenfolge vom exaktesten zum allgemeinsten Muster die aktuelle Anmeldung <User> + Anmelderechner <Host> mit den in der Datenbank eingetragenen verglichen (Sortierreihenfolge: <Host> zuerst, dann <User>, Zeichenketten ohne Platzhalter "_" und "%" VOR solchen mit, leere Zeichenkette bzw. "%" an letzter Stelle, IP-Adresse nach Host-Namen). Der 1. Treffer bestimmt die gesamten Rechte der Anmeldung, es findet KEINE Zusammenfassung der Rechte ALLER passenden Muster statt.

ACHTUNG:

- * Bei falscher Namensauflösung (gemäß "/etc/hosts" oder DNS) kann es Anmeldeprobleme geben (z.B. "localhost.site" statt "localhost" als erster Name zu 127.0.0.1 eingetragen).
- * -h "localhost" verbindet sich per UNIX-Socket (nur bei lokaler Anmeldung)
 - + -h <Host> weggelassen verhält sich analog
 - + --protocol TCP erzwingt Verbindung über TCP/IP-Socket
- * 127.0.0.1 verbindet sich per TCP/IP-Socket (nur bei lokaler Anmeldung)

Verschlüsselungsoptionen REQUIRE <SSLOpt>:

Option	Bedeutung
NONE	Unverschlüss. Verbindung erlaubt (Std), verschl. auch
SSL	Nur SSL-verschlüsselte Verbindungen erlaubt
X509	Gültiges Zertifikat notwendig, Issuer + Subject egal
CIPHER "<Spec>"	Verschlüsselung und Schlüssel müssen best. Bed. genügen (z.B. CIPHER "EDH-RSA-DES-CBC3-SHA")
ISSUER "<Name>"	Zertifikat muss von CA <Name> ausgestellt sein
SUBJECT "<Name>"	Zertifikat muss Subject <Name> enthalten

Optionen zu "GRANT" (Std: 0 = Kein Limit):

Option	Bedeutung
GRANT OPTION	Eigene Rechte an andere weitergebbar
MAX_QUERIES_PER_HOUR <Zahl>	Max. Anz. SQL-Anweisungen pro Stunde
MAX_CONNECTIONS_PER_HOUR <Zahl>	Max. Anz. Verbindungen pro Stunde
MAX_UPDATES_PER_HOUR <Zahl>	Max. Anz. Updates pro Stunde
MAX_USER_CONNECTIONS <Zahl>	Max. Anz. gleichzeitiger Verbindungen

Reihenfolge in der MySQL-Rechtetabellen überprüft werden, ob ein Benutzer für eine bestimmte Query Zugriffsrechte hat (absteigender Vorrang):

- 1) Benutzer-spezifische Rechte
- 2) Datenbank-spezifische Rechte
- 3) Tabellen-spezifische Rechte
- 4) Spalten-spezifische Rechte
- 5) Routinen-spezifische Rechte

SQL-Anweisung MySQL-Tabelle

```

Query + <User> ----> | host +-----+ true
      <Db>          +-----+
                        | false
                        v
+ <User>          +-----+ true
                  | db +-----+
                  +-----+
                  | false
                  v
                  +-----+ true
                  | user +-----+
                  +-----+
                  | false
                  v
+ <Tbl>          +-----+ true
                  | tables_priv +-----+
                  +-----+
                  | false
                  v
+ <Col>          +-----+ true
                  | columns_priv +-----+
                  +-----+
                  | false
                  v
+ <Proc>         +-----+ true
                  | procs_priv +-----+
                  +-----+
                  | false
                  v
Query abgelehnt

```

Direkter Zugriff auf Benutzer- und Zugriffsrechte-Verwaltungsdaten
(besser nicht, Passwort mit Funktion PASSWORD() verschlüsseln!):

```

USE mysql;
SHOW TABLES;      # -> columns_priv, db, func, host, tables_priv, user
DESCRIBE user;
SELECT * FROM user;
SELECT user, password, host FROM user;
INSERT INTO user
  SET host="sony", user="tom", password=PASSWORD("geheim");
INSERT INTO user (host, user, password)
  VALUES ("sony", "tom", PASSWORD("geheim"));

```

Datenbankanzeige des MySQL-Servers steuern:

Option	Bedeutung
--safe-show-database	Datenbanken nur mit Zugriffsberechtigung anzeigen
--skip-show-database	Keine Anzeige der vorhandenen Datenbanken

6) Datenbank-Engines (Storage Engines/Backends)

Eine Besonderheit von MySQL ist die Möglichkeit, PRO TABELLE die Art der physikalischen Speicherung (Engine) zu wählen. Damit lässt sich die Datenbank genau auf die Anforderungen zuschneiden. Standard ist MyISAM (früher ISAM). Hier eine Liste der Engines mit ihren wichtigsten Eigenschaften (Trans = Transaktionsfähig):

Engine	Trans	Beschreibung
MyISAM	--	Std, hohe Performanz, binär portabel, Volltext-Index
InnoDB	ja	Zeilenbasierte Sperrung, Foreign Keys -> ORACLE! autom. Fehlerkorrektur, konsistentes Lesen ohne Sperren
BerkeleyDB (BDB)	ja	Sleepycat, Primärschlüssel notwendig, keine AUTO_INC, seitenbasierte Sperrung (nur bis MySQL 5.0) -> ORACLE!
MEMORY	--	Hash-basiert, RAM, temp. Tabellen, Datenverlust (HEAP)
MERGE	--	Komb. ident. riesige MyISAM-Tabellen (MRG_MyISAM, UNION)
ARCHIVE	--	Große Menge, kein Index, komprimiert, selten verwendet
CSV	--	Comma Separated Values-Format, kein Index
NDBCLUSTER	--	Verteilt auf Rechner-Cluster, HA, High Performance, RAM
FEDERATED	--	Remote-Datenbank (Umlenkung, ab MySQL 5.0.3)
BLACKHOLE	--	Gespeicherte Daten verschwinden (/dev/null)
EXAMPLE	--	Programmier-Beispiel (speichert nichts, Schnittstellen)

ISAM	--	Nicht mehr verwendet (veraltet!)
FALCON	ja	MVCC (Multi Version Concurrency Control, ab MySQL 6.0)
Maria	ja	MyISAM-Erweiterung, Crash Save (ab MySQL 6.0)
InnoDB	ja	Colbased, MVCC, TA, keine Indices, Datamining, Bus.Int.
solidDB	ja	Zeilenbasierte Sperrung, MVCC (IBM)
NitroEDB	??	
PBXT	??	(PrimeBase Storage)

Grobe Unterschiede der Datenbank-Engines:

Eigenschaft	Engine
Transaktionen	InnoDB, BDB, InnoDB
Fulltext-Index (FULLTEXT INDEX)	MyISAM (nur CHAR, VARCHAR, TEXT)
Spatial-Index (SPATIAL INDEX)	MyISAM, InnoDB, NDB, BDB, ARCHIVE MyISAM (Indices)
Fremdschlüssel/Referenzielle Integrität	InnoDB
Raw-Partitionen verwendbar	InnoDB
ALTER TABLE nur teilweise möglich	InnoDB, BDB
Format betriebssystem-abhängig	ISAM
Max. Größe <= 4 GByte	ISAM

Feine Aufschlüsselung der Engine-Eigenschaften:

Eigenschaft	MyISAM	InnoDB	BDB	NDB	Archive	Memory
Max. Datenbankgröße	256TB	64TB	??	384EB	--	RAM
Betriebssystemunabh. (portabel)	Ja	??	--	??	??	--
AUTO_INCREMENT-Spalten	Ja	Ja	??	Ja	--	Ja
ALTER TABLE beliebig möglich	Ja	--	--	Ja	--	Ja
Replikationsfähig	Ja	Ja	--	Ja	Ja	Ja
Backup/point-in-time recovery	Ja	Ja	--	Ja	Ja	Ja
Raw-Partitionen statt Files	--	Ja	--	--	--	--
Referenzielle Integrität	--	Ja	--	--	--	--
Transaktionen	--	Ja	Ja	Ja	--	--
Locking-Einheit	Table	Satz	Page	Satz	Satz	Table
MVCC (Snapshot lesen)	--	Ja	--	--	Ja	--
B-tree Index	Ja	Ja	--	Ja	--	Ja
Hash Index	--	--	Ja	Ja	--	Ja
Volltext Index	Ja	--	--	--	--	--
Clustered Index	--	Ja	--	--	--	--
Statistikupdates	Ja	Ja	Ja	Ja	Ja	Ja
Geografische Datentypen	Ja	Ja	??	Ja	Ja	--
Geografischer Index (R-Tree)	Ja	--	??	--	--	--
Datencache (Tabellendaten)	--	Ja	Ja	Ja	--	--
Indexcache	Ja	Ja	Ja	Ja	--	--
Querycache (Abfragen)	Ja	Ja	Ja	Ja	Ja	Ja
Komprimierte Daten	Ja	--	--	--	Ja	--
Verschlüsselte Daten	Ja	Ja	Ja	Ja	Ja	Ja
Clusterdatenbankfähig	--	--	--	Ja	--	--

Befehle zum Abfragen/Einstellen der Engine:

```
SHOW [STORAGE] ENGINES;           # Unterstützte Engines auflisten
SET storage_engine = <Engine>;    # Default-Engine setzen
CREATE TABLE <Tbl> (...) ENGINE = <Engine>; # Engine einer Tabelle setzen
CREATE TABLE <Tbl> (...) TYPE = <Engine>; # Analog (veraltet!)
SHOW ENGINE <Engine> STATUS;      # Nur InnoDB, NDB, NDBCLUSTER
SHOW ENGINE <Engine> LOCKS;       # Nur BDB
SHOW MUTEX STATUS;                # Nur InnoDB
ALTER TABLE <Tbl> ENGINE = <Engine>; # Engine einer Tabelle ändern
ALTER TABLE <Tbl> TYPE = <Engine>; # Analog (veraltet!)
                                   # (automatisch konvertiert!)
```

MERGE-Tabelle - Eigenschaften:

- * Zusammenfassung mehrere Tabellen
- * Müssen absolut identisch sein (Reihenfolge, Spalten, Typen, Längen)
- * Z.B. für Log-Dateien (ändern sich selten, wachsen stetig)
- * Riesige Tabelle aus Teiltabellen aufbauen
- * Größenbeschränkung des OS umgehen
- * Höhere Geschwindigkeit (Daten auf mehrere Platten verteilen)
- * Effizienterer Zugriff (direkt auf relevante Teiltabelle zugreifen)

- * Tabelle identifizieren (verschiedene Namen für gleiche Tabelle)
- * EINE Tabelle umbenennen
- * Reparatur schneller

CSV Engine (Comma Separated Values, ab MY!5.1):

- * Eigenschaften:
 - + CSV-Tabellen unterstützen keine Indizierung!
 - + Partitionierung von CSV-Tabellen ist nicht mehr möglich (war mal).
 - + Keine NULL-Spalten mehr erstellbar (aber noch nutzbar falls existent)
 - + Auf Dateisystemebene füllbar (vorher FLUSH TABLE <Tbl>;)
 - + In "mysqld-max" enthalten (nicht in "mysqld")
- * Format
 - + Titelzeile mit Spaltennamen NICHT erlaubt
 - + 1. Zeile NICHT ignorierbar
 - + Trennzeichen NICHT wählbar? (z.B. TAB-getrennte Daten wenn "," in Daten)
 - + Felder müssen gequotet sein "..." (dürfen dafür "," enthalten)
 - + Spaltentrenner MUSS "," sein.
 - + Zeilenende MUSS "\n" (UNIX) sein, nicht "\r\n" (WINDOWS)!
- * Wie CSV-Engine in "mysqld" integrieren (fehlt)? - Plugin?
- + Mit "configure --with-csv-storage" den Quellcode übersetzen!
- * Ablageort der CSV-Dateien einer Tabelle <Tbl>:
 - <Tbl>.frm # Definition
 - <Tbl>.CSV # Daten
 - <Tbl>.CSM # Metadaten (Tabellenzustand, Anzahl Zeilen)
- * Beispiel


```
CREATE TABLE test (
  i INT NOT NULL,
  c CHAR(10) NOT NULL
) ENGINE = CSV;
INSERT INTO test VALUES
(1, "Satz 1"),
(2, "Satz 2");
```

Erzeugt folgende Datei "test.CSV"

```
"1","Satz 1" <NL>
"2","Satz 2" <NL>
```
- * Was passiert wenn Format kaputt?
 - + Die DB kann nichts damit anfangen!
 - + Nachricht über zerstörte Tabelle beim Zugriff
- * Prüfen und reparieren (gültige Zeilen bis 1. kaputte übernehmen)


```
CHECK TABLE test;
REPAIR TABLE test;
```

Prüft:

 - + Richtige Feldseparatoren ","
 - + Korrekte Anzahl Felder pro Zeile
 - + Korrekte Quotes "..." um Felder
 - + Metafile vorhanden und passend zu Daten
 - + 1. ungültige Zeile führt zu Fehlermeldung (mit Zeilennummer)
 - + Reparatur erfolgt bis 1. ungültige Zeile (Rest weggeworfen)

7) Datenbanken erstellen und verwalten

Standard-Datenbanken nach Installation:

Datenbank	Bedeutung
mysql	Interne Verwaltung (nicht ändern!)
information_schema	Informationsschema (Verwaltungsdaten, nicht ändern!)
test	Evtl. zu Testzwecken (leer)
tmp	Evtl. für temp. Daten (und zum Login benötigt!)

Standort:

```
/var/lib/mysql/mysql/*      Interne Datenbank "mysql"
/var/lib/mysql/<Db>/*        Benutzerdatenbank <Db>
/var/lib/mysql/<Db>/<Tbl>.frm  Format von <Tbl> (rekonstruierbar)
/var/lib/mysql/<Db>/<Tbl>.MYD  Daten von <Tbl> (NICHT rek.)
/var/lib/mysql/<Db>/<Tbl>.MYI  Indices von <Tbl> (rekonstruierbar)
/var/lib/mysql/<Host>.err     Logging-Informationen für <Host>
```

Datenbank-Namen:

- * GROSS/Kleinschreibung zählt bei Linux (bei Windows nicht!)
- * Max. 64 Zeichen
- * Alle Zeichen AUSSER "/" und "." erlaubt ("..." bzw. '...' außenrum!)

Neue Datenbank erstellen:

```
mysqladmin -uroot -pgeheim create <Db>
CREATE DATABASE <Db>;
CREATE SCHEMA <Db>;
CREATE DATABASE IF NOT EXISTS <Db>;
CREATE SCHEMA IF NOT EXISTS <Db>;
```

Datenbanken anzeigen:

```
SHOW DATABASES;
SHOW SCHEMAS;
```

Datenbank vollständig kopieren (Struktur + Daten, ohne Benutzer+Rechte):

```
mysqldump <DbOld> > dump.sql # Evtl. --no-create-info
mysql -e "CREATE DATABASE <DbNew>" # Neue Datenbank anlegen oder
mysql -e "CREATE SCHEMA <DbNew>" # Neue Datenbank anlegen oder
mysql <DbNew> < dump.sql # Evtl. --no-data
```

Datenbank löschen:

```
mysqladmin -uroot -pgeheim drop <Db> # Mit Rückfrage!
DROP DATABASE <Db>;
DROP SCHEMA <Db>;
DROP DATABASE IF EXISTS <Db>;
DROP SCHEMA IF EXISTS <Db>;
```

8) Tabellen erstellen und verwalten

Tabelle erstellen (permanente oder temporäre):

```
CREATE [TEMPORARY] TABLE <Tbl> (<Col> <Type> [<ColOpt>], ...)
    [<TblOpt>] [<Select>];
CREATE [TEMPORARY] TABLE IF NOT EXISTS <Tbl> (<Col> <Type> [<ColOpt>], ...)
    [<TblOpt>] [<Select>];
```

```
<TblOpt> = AUTO_INCREMENT = <N>
          AVG_ROW_LENGTH = <N>
          [DEFAULT] CHARACTER SET <CharSet>
          CHECKSUM = 0/1
          COLLATE <Collate>
          COMMENT "<String>"
          CONNECTION "<String>"
          DATA DIRECTORY "<Path>"
          DELAY_KEY_WRITE = 0/1
          ENGINE = <Engine>
          INDEX DIRECTORY = "<Path>"
          INSERT METHOD = {NO | FIRST | LAST}
          KEY_BLOCK_SIZE = <N>
          MAX_ROWS = <N>
          MIN_ROWS = <N>
          PACK_KEYS = 0/1/DEFAULT
          PASSWORD = "<String>"
          ROW_FORMAT = {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}
          UNION = <Tbl>
          <Partition>
```

```
<ColOpt> = [NOT] NULL
          COMMENT "<String>"
          DEFAULT <Value>
          AUTO_INCREMENT
          UNIQUE [KEY]
          [PRIMARY] KEY
          COLUMN FORRMAT [DEFAULT | FIXED | DYNAMIC]
          STORAGE [DISK | MEMORY]
          <References>
```

HINWEIS zu temporären Tabellen:

- * Pro Sitzung vorhanden (d.h. gleichnamige kollidieren nicht miteinander)
- * Automatisch gelöscht bei Sitzungsende (Trennung Client vom Server)
- * Überdecken evtl. vorhandene gleichnamige echte Tabellen!
- * Liegen im Verz. "/tmp" (bzw. def. "--tmpdir=PATH;..." oder "-t=PATH;...")
- * Nur Engine "MEMORY", "MyISAM", "MERGE" oder "InnoDB" möglich
- * Von SHOW TABLE nicht angezeigt

Tabellentyp:

- * SHOW VARIABLES LIKE "have_%"; # Unterstützte Tabellentypen anzeigen
- * Standard: "MyISAM" (Weitere: InnoDB, BDB, MERGE, HEAP, ISAM)
 - /var/lib/mysql/<Db>/<Tbl>.frm Format von <Tbl> (rekonstruierbar)
 - /var/lib/mysql/<Db>/<Tbl>.MYD Daten der Tabelle <Tbl> (NICHT rek.)
 - /var/lib/mysql/<Db>/<Tbl>.MYI Indices von <Tbl> (rekonstruierbar)
- * SQL: ... ENGINE = <Tbltype> ... # TYPE veraltet!
- * Option: default-table-type=<Tbltype>
- * ALTER TABLE <Tbl> ENGINE = "<Tbltype>" # TYPE veraltet!

Tabellen-Namen:

- * GROSS/Kleinschreibung zählt bei Linux (bei Windows nicht)
- * Max. 64 Zeichen
- * Alle Zeichen AUSSER "/" und "." erlaubt ("..." bzw. '...' außenrum!)

Spaltendefinition umfasst:

- * Spaltenname
- * Datentyp
- * Länge
- * Attribute (zum Datentyp)
- * [NOT] NULL
- * DEFAULT ...
- * AUTO_INCREMENT (nur ein Feld, Index notwendig)
- * Primärschlüssel, Index, Unique, Volltext

Datentypen von MySQL:

-> mysql-HOWTO.txt -> 4) MySQL-Datentypen

Primärschlüssel (letzten Endes durch einen Index realisiert!):

- * NUR EINER erlaubt
- * Kein NULL-Wert erlaubt
- PRIMARY KEY (...)

Tabellen auflisten:

```
USE <Db>;
SHOW TABLES;
SHOW TABLES FROM <Db>;
SHOW TABLES FROM <Db> LIKE "<Muster>";
SHOW TABLE STATUS;
SHOW TABLE STATUS FROM <Db>;
SHOW TABLE STATUS FROM <Db> LIKE "<Muster>";
```

Tabellenstruktur/definition anzeigen:

```
USE <Db>;
SHOW COLUMNS FROM <Tbl>;
DESCRIBE <Tbl>;
DESCRIBE <Tbl> <Col>;          # ... LIKE "%_priv"
EXPLAIN <Tbl>;
SHOW CREATE TABLE <Tbl>;
```

Tabelle ändern:

- * Möglichkeiten
 - + Tabellenengine ändern
 - + Tabellen + Spalten umbenennen
 - + Spalten hinzufügen + löschen (inkl. Position!)
 - + Spaltendefinition (Typ) ändern
 - + Schlüssel + Indices hinzufügen + löschen (Tuning)
- * Grundsyntax (ADD, CHANGE/MODIFY, DROP)


```
ALTER TABLE <Tbl> <Änderung>, ...;
```
- * Beispiele


```
ALTER TABLE <Name> RENAME [TO] <NewName>;
ALTER TABLE <Tbl> ENGINE = "<Tbltype>";
ALTER TABLE <Tbl> TYPE = "<Tbltype>"; # veraltet!
```
- * Großes Beispiel (<Definition> analog CREATE TABLE)


```
ALTER TABLE <Tbl>
  ADD <Col> <Definition> [FIRST | AFTER <Col>],
  ADD INDEX <Idx> (<Col>, ...),
  ADD PRIMARY KEY (<Col>),
  CHANGE <Col> <Definition>,
  DROP <Col>,
  DROP INDEX <Idx>,
  DROP PRIMARY KEY;
```

Tabelleninhalt anzeigen (DISTINCT vermeidet Ausgabe doppelter Datensätze!)

```
USE <Db>;
SELECT [DISTINCT] <Auswahl> FROM <Tbl> [<Parameter>];
```

- * Beispiele (1. Beispiel vermeiden -> belastet MySQL-Server stark!)


```
SELECT * FROM user;
SELECT user, host, password FROM user;
SELECT DISTINCT user FROM user;
```
- * Optionale Parameter (in dieser Reihenfolge!)


```
WHERE      # Auswahl-Einschränkung (Vergl, log Op, Fkt, Klammer, BETWEEN, LIKE)
GROUP BY   # Gleichartige Datensätze zusammenfassen (SUM, COUNT, MIN, MAX, AVG)
HAVING     # Weitere Einschränkung gruppierter Ergebnisse (analog WHERE)
ORDER BY   # Reihenfolge der ausgewählten Datensätze festlegen (ASC, DESC)
LIMIT      # Position und Anzahl zurückgelieferter Datensätze begrenzen
```

Tabelleninhalt löschen (Simulation von DROP + CREATE!):

```
USE <Db>;
TRUNCATE TABLE <Tbl>;
```

- * Schneller als DELETE FROM ... (da Tab. gelöscht + neu angelegt)
- * In einer Transaktion NICHT möglich

Tabelle entfernen (inklusive Inhalt!):

```
DROP TABLE <Tbl>;
```

Tabelle kopieren:

```
CREATE TABLE IF NOT EXISTS kopie LIKE pers;           # Ohne Daten, mit Indices (MY!5.0)
CREATE TABLE kopie SELECT * FROM pers;              # Mit Daten, ohne Indices
CREATE TABLE kopie SELECT * FROM pers WHERE 1=2;    # Ohne Daten, ohne Indices (Trick!);
INSERT INTO copy SELECT * FROM pers;                # Daten kopieren
```

9) Tabellen prüfen und warten

Regelmäßige Prüfung von Tabellen ist wichtig:

```
* Nur für "MyISAM" und "InnoDB" möglich
  USE <Db>;
  CHECK TABLE <Tbl> <Optionen>;                    # Eine
  CHECK TABLE <Tbl>, ... <Optionen>, ...;          # Mehrere
* Optionen:
  QUICK      # Oberflächliche Prüfung (Verknüpfungen nicht)
  FAST       # Nur nicht ordnungsgemäß geschlossene Tabellen prüfen
  CHANGED    # Nur seit letzter Prüfung geänderte Tabellen prüfen + FAST
  MEDIUM    # Alle Datensätze prüfen, Index nur per Prüfsumme (Std)
  EXTENDED  # Alle Datensätze prüfen, Index/Schlüssel komplett
* Fehlerhafte Tabellen anschließend "gesperrt", vor Verwendung zu reparieren
```

MyISAM-Tabellen prüfen + Info sammeln + optimieren + reparieren:

```
* Vorher Tabelle sichern!
* Empfehlung: Server runterfahren!
  myisamchk [<Optionen>] /var/lib/mysql/<Db>/<Tbl>.MYI # nicht .MYD!
  myisamchk [<Optionen>] /var/lib/mysql/*/*MYI        # Alle!
* Optionen
  -c/--check           # Standardprüfung (Std)
  -C/--check-only-changed # Analog "CHANGED"
  -e/--extended-check  # Analog "EXTENDED"
  -F/--fast            # Analog "FAST"
  -m/--medium-check    # Analog "MEDIUM"
  -U/--update-state    # Prüfungs-Datum + Ergebnis in Tab. speichern
  -T/--read-only       # Tabelle nicht kennzeichnen
  -s/--silent          # Nur Ergebnis ausgeben
```

MyISAM und BDB-Tabellen prüfen:

```
* Datenbankname angebar, schneller, bei laufendem Server möglich
  mysqlcheck [<Optionen>] <Db> [<Tbl>...]
* Optionen identisch zu "myisamchk" bis auf:
  -T/--read-only      # Nicht verwendbar
  -U/--update-state   # Nicht verwendbar
  -A/--all-databases  # ALLE Datenbanken prüfen (Std)
  -B/--databases ...  # Auswahl von Datenbanken prüfen
  -q/--quick          # Analog "QUICK"
  --tables ...        # Auswahl von Tabellen prüfen
```

Tabelle reparieren:

```
* Hinweise
+ Nach Prüfung mit Ergebnis "fehlerhaft" notwendig!
+ Nicht reparierbare Tabellen müssen aus Backup rekonstruiert werden!
+ Vorher Tabelle sichern!
+ Server runterfahren!
+ Es muss genügend Platz auf der Festplatte vorhanden sein!
* Möglichkeiten
A) myisamchk [<Optionen>] /var/lib/mysql/<Db>/<Tbl>.MYI # nicht .MYD!
  + Optionen (zuerst -r, dann -o probieren)
  -r/--recover           # Alles außer nichteindeutige Schlüssel (Std)
  -e/--extended-check    # Jede Zeile wiederherstellen
  -o/--safe-recover      # -r aber Index neu + langsamer + weniger Platz
  -q/--quick             # Nur Indices neu aufbauen
  -s/--silent            # Nur Ergebnis ausgeben
  -v/--verbose           # Viele Meldungen ausgeben
B) mysqlcheck [<Optionen>] <Db> [<Tbl>...]
  + Optionen
  --auto-repair          # Alles außer nichteindeutige Schlüssel (Std)
  -e/--extended-check    # Analog "--extended-check" von "myisamchk"
  -q/--quick             # Analog "--quick" von "myisamchk"
  -r/--repair            # Analog "--recover" von "myisamchk"
C) USE <Db>;
  REPAIR TABLE <Tbl> [QUICK|EXTENDED];           # Analog "myisamchk -r"
```

Tabelle optimieren:

```
* Durch Löschen/Ändern kommt es mit der Zeit zu "Fragmentierung"
+ Kleine freie Stücke Plattenplatz, die nicht mehr verwendbar sind
+ Zugriff verlangsamt, Dateigröße entspricht nicht mehr abgelegten Daten
* Optimierung = Tabellen-Neuaufbau inklusive aller Indices (Defragmentierung)
+ Gelöschte Zeilen entfernen
+ Aufgeteilte Zeilen zusammenfügen
```

```
+ Indices sortieren
+ Statistiken für MySQL-Optimierer erneuern
+ Tabelle ist in dieser Zeit "gesperrt"
* Möglichkeiten (nur "MyISAM" und "BDB")
A) myisamchk --quick --check-only-changed --sort-index --analyze \
  /var/lib/mysql/<Db>/<Tbl>.MYI # nicht .MYD!
B) mysqlcheck --optimize <Db> [<Tbl>...]
C) USE <Db>;
   OPTIMIZE TABLE <Tbl>, ...; # MyISAM, BDB, InnoDB
```

Tabellen komprimieren (nur "MyISAM"):

```
* Falls Tabelle nur gelesen wird!
* Platzbedarf reduziert
* Zugriffsgeschwindigkeit gesteigert
  myisampack [<Optionen>] /var/lib/mysql/<Db>/<Tbl>.MYI # nicht .MYD!
* Optionen
  -b/--backup # Erst Sicherung <Tbl>.old erstellen!
  -f/--force # Erzwingen
  -j/--join <Newname> # Zu großer neuer Tabelle zusammenfassen
  -t/--test # Nur simulieren, nicht durchführen
  -s/--silent # Nur Ergebnis ausgeben
  -v/--verbose # Viele Meldungen ausgeben
  -w/--wait # Warten bis Tabelle nicht mehr benutzt
* Danach MUSS der Index neu erzeugt werden!
  myisamchk -rq /var/lib/mysql/<Db>/<Tbl>.MYI # nicht .MYD!
* Danach sind die Dateien automatisch schreibgeschützt
* Soll wieder INSERT, UPDATE, DELETE möglich sein, die Tabellen entpacken
  myisamchk --unpack /var/lib/mysql/<Db>/<Tbl>.MYI # nicht .MYD!
```

Tabelle für MySQL-Optimierung analysieren (Statistik über Schlüsselverteilung)

```
A) mysqlcheck -a ...
B) myisamchk -a ...
C) USE <Db>;
   ANALYZE TABLE <Tbl>, ...;
```

10) Datenimport und -export

Einzelne Datensätze einfügen (mit MySQL-Erweiterung auch mehrere möglich!):

```
USE <Db>;

INSERT INTO <Tbl> (<Col>, ...)
  VALUES (<Wert>, ...);
INSERT INTO <Tbl> (<Col>, ...)
  VALUES (<Wert>, ...),
  (...), ...; # NUR MySQL!
INSERT INTO <Tbl> (<Col>, ...)
  SELECT <Col>, ... FROM <Tbl>;
* Alle Werte außer numerische Werte MÜSSEN in Hochkommas gesetzt sein!
```

Viele Datensätze einfügen:

```
A) Datei mit entsprechend vielen SQL-INSERT-Anweisungen + Umlenkung "<"
  mysql -uroot -p < insert-statements.sql

B) Delimited ASCII-Datei mit SQL-Anweisung "LOAD DATA..."
+ Schneller als INSERT-Anweisungen
+ Wert NULL in Datenfile -> NULL in Datenbankspalte (Codierung durch "\N")
+ Pro Datensatz eine Zeile (Zeilenvorschub NL, CR+NL)
+ Trennung der Werte durch ",", " oder ";"
+ Texte in "... " oder '...' einschließen
+ Reihenfolge der Werte = Reihenfolge der Spalten
+ CSV-Datei (Comma Separated Values)
  USE <Db>;
  LOAD DATA INFILE "<Filepath>" INTO TABLE <Tbl>;
  LOAD DATA [LOCAL] INFILE "<Filepath>"
    [REPLACE | IGNORE] # Wie doppelte Schlüssel beh.
    INTO TABLE <Tbl> [<LoadOpt>];
+ Daten auf dem Server oder auf dem Client (Option "LOCAL")
+ Doppelte Schlüssel
  - Standard: Import abbrechen
  - IGNORE: Neue Zeile ignoriert (erste zählt!)
  - REPLACE: Neue Zeile ersetzt vorhandene (letzte zählt!)
+ <LoadOpt> (in dieser Reihenfolge anzugeben!, "FIELDS" nur 1x!)
  (einzelne oder alle können auch fehlen, ebenso die Spalten)
  - FIELDS TERMINATED BY "<Trennzeichen>" # Std: "\t" = Tabulator
  ... [OPTIONALLY] ENCLOSED BY "<Zeichen>" # Std: "" = keines
  ... ESCAPED BY "<Zeichen>" # Std: "\\\" = \
  LINES STARTING BY "<String>" # Std: "" = leer
  TERMINATED BY "<String>" # Std: "\n" = \n
  IGNORE <Nnn> LINES # Erste NNN Zeilen ign.
  (<Col>, ...) # Best. Sp.+Reihenf. laden (sonst alle)
```

```

SET <Col> = <Expr>, ... # Best. Sp. fix füllen
+ Bei Windows angeben: LINES TERMINATED BY "\r\n"
+ ACHTUNG: Ohne LOCAL werden die Daten auf dem Server gesucht unter:
./FILE -> /var/lib/mysql/FILE
FILE -> /var/lib/mysql/<Db>/FILE
/PFAD/ZU/FILE -> Absoluter Pfad funktioniert auch nicht

```

C) Delimited ASCII-Datei mit Kommando "mysqlimport"

```

mysqlimport <Optionen> <Db> <DatenFile>...;
+ Dateiname ohne Extension MUSS Tabellenname entsprechen!
+ Ohne -i/-r wird bei doppeltem Schlüssel abgebrochen!
+ Optionen
-d/--delete # Tabelle erst leeren
-c=<Col>/--columns=<Col> # Auf bestimmte Felder beschränken
--fields-terminated-by=<Z>
--fields-enclosed-by=<Z>
--fields-escaped-by=<Z>
--lines-terminated-by=<Z>
-L/--local # Datei steht auf Client (Std: Server)
-i/--ignore # Ident. Zeile ignoriert (erste zählt!)
-r/--replace # Ident. Zeile ersetzt vorhandene (letzte zählt!)

```

11) Datensicherung und -wiederherstellung

Datensicherung ist notwendig:

- * Große Anzahl von Datensätzen
- * Großer finanzieller/zeitlicher Aufwand zur Wiederherstellung
- * Datenstände bestimmter Zeitpunkte wiederherstellbar (z.B. nach Bedienungs- oder Programmierfehler)
- * 2 Möglichkeiten
 - + Online im laufenden Betrieb (inkrementell)
 - + Offline bei heruntergefahrenem Server (vollständig)
 - Herunterfahren des Servers oft nicht möglich (7x24h-Betrieb!)
 - Schreiboperationen während Sicherung verhindern
- * Fehlen von Fremdschlüsseln bei best. Engines (z.B. MyISAM) erlaubt tabellenweises Sichern + Wiedereinspielen (unabhängig voneinander)
- * Tabellenstruktur ("*.frm") + Berechtigungen (DB "mysql") mitsichern!
- * Indices ("*.MYI") müssen nicht gesichert werden -> wiederherstellbar
- * Transaktionslog auch permanent sicherbar -> wieder "abspulen"

MySQL-Datensicherung:

- * Einzelne Tabellen oder ganze Datenbanken
- * Tabellen gegen Schreiben sperren während Sicherung
- * Interne Caches leeren vor Sicherung (FLUSH ...)
- * Nach Datensicherung wieder entsperren!

Tabellen gegen Zugriffe durch andere sperren (alle benötigten gleichzeitig!):

- * Wartet, bis alle anderen Zugriffe auf die Tabellen beendet sind
- LOCK TABLES <Tbl> <Type>, ...;
- * <Type>
 - READ # Schreibschutz (Lesen für alle erlaubt)
 - READ LOCAL # INSERT zugelassen, solange kein Konflikt
 - WRITE # Lese- und Schreibschutz für alle außer dem Betreiber
 - LOW_PRIORITY WRITE # Nur wenn kein READ-Lock vorhanden
 - IN SHARE MODE #
 - IN SHARE MODE NOWAIT #
 - IN EXCLUSIVE MODE #
 - IN EXCLUSIVE MODE NOWAIT #
- * WRITE hat höhere Priorität als READ

Alle von einem Benutzer gesperrten Tabellen wieder entsperren:
(oder erneutes "LOCK TABLES" bzw. Verbindungs-Ende)

```
UNLOCK TABLES;
```

Interne Tabellen-Caches leeren:

- * Schließt alle bzw. angegebene Tabellen und schreibt ihre Daten auf Datei
- FLUSH TABLE;
- FLUSH TABLES;
- FLUSH TABLE <Tbl>, ...;
- FLUSH TABLES <Tbl>, ...;

Datensicherung:

- A) Datenbank-Dateien auf Betriebssystemebene kopieren
 - + Unbedingt gegen Schreiben sperren und Caches leeren
 - + Datenbank + Tabellendefinition kopieren
 - + Berechtigungen sichern (Datenbank "mysql")
- B) BACKUP TABLE (nur "MyISAM")
 - + Kopiert Definitions- (*.frm) und Datendatei (*.MYD) in Zielverz.
 - + Nur auf Server möglich

```

    USE <Dbtable>;
    BACKUP TABLE <Tbl>, ... TO "<Dirpath>";
C) SELECT INTO OUTFILE
+ Tabellenstruktur geht verloren (nur Datensätze werden gesichert)
+ Als delimited ASCII-Datei
+ Nur auf Server möglich
+ <Col> = "*" um alle Daten zu sichern
    USE <Db>;
    SELECT <Col> INTO OUTFILE "<Filepath>" [<SaveOpt>] \
        FROM <Tbl> [<Parameter>];
+ <SaveOpt> (in dieser Reihenfolge anzugeben!, "FIELDS" nur 1x!)
(einzelne oder alle können auch fehlen, ebenso die Spalten)
- FIELDS TERMINATED BY "<Trennzeichen>" # Std: "\t" = Tabulator
... [OPTIONALLY] ENCLOSED BY "<Zeichen>" # Std: "" = keines
... ESCAPED BY "<Zeichen>" # Std: "\\" = \
LINES STARTING BY "<String>" # Std: "" = leer
TERMINATED BY "<String>" # Std: "\n" = \n
D) Dienstprogramm "mysqldump"
+ Liefert komplette Definition einer Datenbank oder Tabelle als SQL-Code
mysqldump <Optionen> <Db> [<Tbl> ...]
mysqldump <Optionen> --databases [<Db> ...]
mysqldump <Optionen> --all-databases
+ Optionen
--add-drop-table # DROP TABLE IF EXISTS vor jedem CREATE TABLE
--add-locks # LOCK + UNLOCK TABLES außenrum
-A/--all-databases # ALLE Datenbanken ausgeben
-B/--databases ... # Auswahl von Datenbanken ausgeben
-c/--complete-insert # Komplette INSERT-Anweisung (inkl. Spaltennamen)
-C/--compress # Daten bei Remoteübertragung komprimieren
-d/--no-data # Nur Tabellendefinition, keine Daten
-e/--extended-insert # Erweiterte INSERT-Form (schneller, kompakter)
-f/--force # SQL-Fehler ignorieren
-l/--lock-tables # Alle Tab. mit READ LOCAL sperren (kein Insert)
--single-transaction # Schreibop. während Dump ignorieren (nur InnoDB)
--opt # Einige Beschleunigungsoptionen:
# --add-drop-table --add-locks --create-options
# --disable-keys --extended-insert --lock-tables
# --quick --set-charset
--skip-opt # Std-Optionen abschalten
--compact # Einige Optionen zur Platzreduktion:
# --skip-add-drop-table --skip-add-locks
# --skip-comments --skip-disable-keys
# --skip-set-charset
-q/--quick # Ausgabe nicht puffern
-r/--result-file <File> # Dateiname angeben (Windows: \r\n)
-t/--no-create-info # Nur Daten, keine Tabellendefinition
--fields-terminated-by=<Z>
--fields-enclosed-by=<Z>
--fields-escaped-by=<Z>
--lines-terminated-by=<Z>
+ Komprimiert sichern
mysqldump ... | gzip > SICHERUNG.mysql.gz
+ Komprimierte Sicherung zückspielen
gunzip -c SICHERUNG.mysql.gz | mysql ...
+ Daten zwischen Datenbanken kopieren (Trick!, -c = compress)
mysqldump ... | mysql ... -h 10.1.5.123 <Db>"
mysqldump ... | ssh -e "mysql ... <Db>" root@10.1.5.123
E) DESCRIBE ... und SELECT ... mit Umlenkung auf Datei
F) Eigenes Backup-Programm in C, PHP, Perl, ... schreiben

```

Daten wiederherstellen:

```

* Tabellen sollten leer oder nicht vorhanden sein
* Danach die importierten Daten überprüfen (überschreiben, doppelt)
+ Zumindest Anzahl der Datensätze überprüfen
+ Einige Datensätze manuell ansehen
A) Datenbank-Dateien auf Betriebssystemebene kopieren
+ Zurückkopieren + Besitzer + Gruppe + Zugriffsrechte korrekt setzen
(mkdir, chown -R mysql.mysql, chmod -R o-rwx)
B) BACKUP TABLE (nur "MyISAM")
+ RESTORE TABLE <Tbl>, ... FROM "<Backupverz>";
C) SELECT INTO OUTFILE
+ LOAD DATA INFILE
+ mysqlimport
D) Dienstprogramm "mysqldump"
+ mysql ... <Db> <<Dumpfile>
E) CREATE ... und INSERT ... mit Umlenkung von Datei
F) Eigenes Restore-Programm in C, PHP, Perl, ... schreiben

```

Gutes Backup-Verfahren:

```

* FLUSH TABLES
* FLUSH TABLES WITH READ LOCK

```

- Schließt alle offenen Tabellen und sperrt sie mit einem READ-Lock
- * Backup der DB-Dateien mit LVM-Snapshot oder anderen Methode
- * UNLOCK TABLES

12) Überwachung und Protokolldateien

Typische Störungen eines MySQL-Servers sind:

- * Überlastung (viele/große Abfragen ohne Index)
- * Defekte Datenbank/Tabelle
- * Berechtigungssystem "überlistet"
- * Absturz

Überwachungsmöglichkeiten:

- * Statusinformationen
- * Prozessliste (laufende Threads)
- * Protokolldateien

Statusinformationen:

- A) SHOW STATUS;
 SHOW STATUS LIKE "CONNECTIONS";
 SHOW STATUS LIKE "ABORTED%";
- B) mysqladmin status # Das Wichtigste
 mysqladmin extended-status # Entspricht SHOW STATUS
- + Beispiele
- ABORTED_CLIENTS
 - ABORTED_CONNECTS
 - CONNECTIONS
 - FLUSH_COMMANDS
 - MAX_USED_CONNECTIONS
 - OPEN_TABLES
 - OPEN_FILES
 - QUESTIONS
 - THREADS_CONNECTED
 - UPTIME

Prozessliste (laufende Threads):

- * PROCESS-Berechtigung notwendig (sonst nur eigene Prozesse angezeigt)!
- A) SHOW PROCESSLIST; # Nur die ersten 100 Zeichen jeder Anfrage
 SHOW FULL PROCESSLIST; # Komplette Anfrage (beliebiger Länge)
- B) mysqladmin processlist

Laufende Prozesse/Threads beenden:

- * PROCESS-Berechtigung notwendig (sonst nur eigene Prozesse/Threads beendbar)!
- * Es wird nur ein Kill-Flag gesetzt, das periodisch überprüft wird (d.h. der Prozess/Threads wird nicht unbedingt sofort beendet!)
- A) KILL <Pid>, ...;
- B) mysqladmin kill <Pid>, ...

Protokolldateien (Meldungen bzw. Datenbank-Änderungen):

- * Stehen in "/var/log/mysql" oder "/var/lib/mysql"
 - * Größe von Zeit zu Zeit überprüfen (und evtl. komprimieren + wegsichern) oder mit "logrotate" verwalten
 - * Protokolldateien schließen und erneut öffnen (Cache leeren)
- FLUSH LOGS
 mysqladmin flush-logs
- A) Fehler-Log # Fehlerprotokoll # <Host>.err
 + Start, Kritische Fehler, fehlerhafte Tabellen, Herunterfahren
- B) Anfrage-Log # Allgemeine Anfragen # <Host>.log
 + Schalter -l/--log=<Logfile>
 + Variable log=<Logfile>
 + Verbindungen, Anfragen (Zuordnung über ID-Nummern)
 + Größe wächst sehr schnell (-> nur bei Bedarf aktivieren, komprimieren)
- C) Slow-Log # Langsame Anfragen # <Host>.slow.log
 + Benötigen mehr als die in Variable "long_query_time" definierte Zeit (Hinweise auf zu optimierende Anfragen/Tabellen -> Indices)
 + Schalter --log-slow-queries=<Logfile>
 --log-slow-admin-statements=<Logfile>
 --log-queries-not-using-indices=<Logfile>
 + Variable log-slow-queries=<Logfile>
 + mysqladmin extended-status | grep -i "slow queries"
- D) Update-Log # Änderungs- # <Host>.NR
 Transaktions-Log # protokoll # <Host>-bin.NR + <Host>-bin.index
 + Zeichnet alle INSERT, UPDATE, ALTER und DELETE-Anweisungen auf
 + ASCII- oder Binärform möglich
 - Binär schneller + kompakter
 - ASCII evtl. nicht mehr unterstützt
 - Binär-Log notwendig für Replikation
 + Bei bestimmten Aktionen mit neuer Nummer erstellt
 - Neustart MySQL-Server
 - mysqladmin refresh

```

- mysqladmin flush-logs
- FLUSH LOGS
+ Schalter --log-update=<Logfile>
             --log-bin=<Logfile>
             --log-bin-index=<Logfile>
+ Variable log-update=<Logfile>
             log-bin=<Logfile>
             log-bin-index=<Logfile>
+ Auslesen: mysqlbinlog <Optionen> <Logfile>
             SHOW BINLOG EVENTS [FROM <M>] [LIMIT <N>];

* Thread-Implementierung
+ User Threads (ein Prozess) # FreeBSD, SCO Unix
+ User Threads (viele Prozesse) # Linux
+ Kernel Thread (ein Prozess) # Solaris, HP-UX, AIX, OSF/1, Windows
* SHOW OPEN TABLES [FROM <Db>] [LIKE "Muster"];
* SHOW [FULL] PROCESSLIST;
* Statement-Profiling (Nutzung von Ressourcen durch Statements)
+ Pro Sitzung aufgezeichnet
+ Geht mit Sitzungsende verloren
+ SET profiling = 1
+ SET profiling_history_size = 50 (Std: 15, Max: 100, Aus: 0)
+ SHOW PROFILES
+ SHOW PROFILE [<Type>, ...] [FOR QUERY <n>]; # Letztes oder Statement n
+ Standard: Status + Duration
+ <Type> = ALL # Alle Spalten
             BLOCK IO # Datenblöcke Lese/Schreiboperationen
             CONTEXT SWITCHES # Kontextwechsel
             CPU # CPU-Nutzungsdauer User + System
             IPC # Erhaltenen/Empfangene Nachrichten
             MEMORY # Speicherverbrauch (nicht implementiert)
             PAGE FAULTS # Seitenfehler (Swap, Auslagerungsdatei)
             SOURCE # MySQL-Quellcode: Funktionsname, Dateiname, Zeilnummer
             SWAPS # Anzahl Swaps

* Prozessinformation
  Id # Verbindungs-ID des Clients
  Host # Client
  User # Client
  db # Default-Datenbank (NULL wenn keine ausgewählt)
  Command # Ausgeführtes Kommando (sehr kurz!)
  State # Thread-Zustand (sehr kurz!)
  Time # Dauer des aktuellen Zustands
  Info # Ausgeführte Anweisung

* KILL [CONNECTION|QUERY] <Pid>;
mysqladmin -utom -pcat -H127.0.0.1 ping
mysqladmin -utom -pcat -H127.0.0.1 status
mysqladmin -utom -pcat -H127.0.0.1 processlist
mysqladmin -utom -pcat -H127.0.0.1 kill 1234

* Query Caches leeren - Optionen
RESET <Opt>, ...;
  MASTER
  QUERY CACHE
  SLAVE

* Diverse Caches leeren und Tabellen schließen (evtl. mit Lock) - Optionen
FLUSH [LOCAL | NO_WRITE_TO_BINLOG] <Opt>, ...;
  DES_KEY_FILE
  HOSTS
  LOGS
  MASTER # -> RESET MASTER !!!
  PRIVILEGES
  QUERY CACHE
  SLAVE
  STATUS
  TABLE(S) <Tbl>, ...
  TABLES WITH READ LOCK
  USER_RESOURCES

```

13) Replikation von Datenbanken (Master-Slave, Master-Master)

Synchronisation von Datenbeständen zwischen verschiedenen MySQL-Servern:

- * Ein MySQL-Master-Server kann auf viele MySQL-Slave-Server replizieren
- * Der Ausfall eines Slave-Server hat keinen Einfluss auf den Master
- * Einweg-Replikation (Slaves sind read-only!)

Zweck:

- * Ausfallsicherheit erhöhen (NUR eine Stelle ist schreibbar!)

- ```

4a. Master eine eindeutige Server-ID geben (verschieden von Slaves!)
 server-id = 111
4b. Auf Master die Binär-Logdatei aktivieren
 log-bin = /var/log/mysql/mysql-bin.log
4c. Auf Master Fehlermeldungen auf Datei schreiben (Fehlersuche erleichtern)
 log_error = /var/log/mysql/mysql.err
6a. Allen Slaves eine eindeutige Server-ID geben (nicht slave_id = ... ;-))
 server-id = 113 # != MasterNr + alle Slaves verschieden!!!
6b. Auf allen Slaves folgende Konfigurations-Einträge machen
 read_only = 1 # Nicht mehr schreibbar
 master-host = 10.1.5.111 # (oder Host-Name)
 master-user = replication
 master-password = geheim
 master-port = 3306 # Auf Standard lassen
 replicate-do-db = kopie # Welche DB kopieren (auch mehrfach!)
 skip-slave-start # Slave manuell per "SLAVE START" starten
6c. Auf Slaves Fehlermeldungen auf Datei schreiben (Fehlersuche erleichtern)
 log_error = /var/log/mysql/mysql.err
6d. Auf Slaves allen Benutzern (außer "replication") die Rechte UPDATE,
 DELETE, INSERT, DROP für die Datenbank "kopie" entziehen
 REVOKE INSERT, DELETE, UPDATE, DROP
 ON kopie.*
 FROM "root"@"%
 ... FROM "kurs"@"%
 ... FROM debian-sys-maint@"localhost"
7. Master-Server starten und Slave-Server neu starten (Reihenfolge egal!)
 + Datei "/var/lib/mysql/master.info" wird auf jedem Slave erzeugt
 (enthält Info, welchen Teil der Binär-Logdatei schon abgearbeitet)
8. Änderungen werden nun automatisch vom Master zu den Slaves repliziert
9. Replikation stoppen und wieder aufnehmen (alle Änderungen werden nachgezogen)
 SLAVE STOP;
 SLAVE START;

```

## Konfigurations-Parameter

- ```

* Master
  log-bin-index=<Filepath> # Indexdatei für binären Dump
  binlog-do-db=<Db> # Zu replizierende Datenbank (Nx)
  binlog-ignore-db=<Db> # NICHT zu replizierende Datenbank (Nx)
* Slave
  log-slave-updates # Slave-Binlog für weitere Subslaves
  master-info-file=<Filepath> # Std: master.info
  replicate-do-db=<Db> # Zu replizierende Datenbank (Nx)
  replicate-ignore-db=<Db> # NICHT zu replizierende Datenbank (Nx)
  replicate-do-table=<Tbl> # Zu replizierende Tabelle (Nx)
  replicate-ignore-table=<Tbl> # NICHT zu replizierende Tabelle (Nx)
  skip-slave-start # Slave manuell per "SLAVE START" starten

```

SQL-Anweisungen

- ```

* Master
 PURGE MASTER/BINARY LOGS; #
 RESET MASTER; # Alle Binär-Logs + Index löschen + leeres neu erzeugen
 SET SQL_LOG_BIN = 0; # Binär-Log-Aufzeichnung stoppen (PROCESS-Recht!)
 SET SQL_LOG_BIN = 1; # Binär-Log-Aufzeichnung fortfahren (PROCESS-Recht!)
 SHOW MASTER/BINARY LOGS; # Bin-Log-Dateien des Masters auflisten
 SHOW BINLOG EVENTS; #
 SHOW MASTER LOGS; #
 SHOW MASTER STATUS; # Master-Status anzeigen
 SHOW SLAVE HOSTS; # Alle bekannten Slaves anzeigen
* Slave
 RESET SLAVE; # Replikationsposition im Master-Binär-Log vergessen +
 # Relay-Log löschen und neues leeres beginnen
 LOAD DATA FROM MASTER; # Initialisierung (nur bei MyISAM!)
 LOAD TABLE <Tbl> FROM MASTER; # Initialisierung (nur bei MyISAM!)
 SHOW SLAVE HOSTS; # Slave-Rechner auflisten
 SHOW SLAVE STATUS; # Slave-Status anzeigen
 SLAVE START; # Replikation fortsetzen
 START SLAVE; # Analog
 START SLAVE IO_THREAD; # Repl. Master-Bin-Log -> Slave-Relay starten
 START SLAVE SQL_THREAD; # Repl. Slave-Relay -> Slave-DB starten
 SLAVE STOP; # Repl. anhalten (später nachholbar)
 STOP SLAVE; # Analog
 STOP SLAVE IO_THREAD; # Repl. Master-Bin-Log -> Slave-Relay starten
 STOP SLAVE SQL_THREAD; # Repl. Slave-Relay -> Slave-DB stoppen

```

Slave-Einstellungen ändern (und Konfigurationsdatei "master.info" sowie "relay-log.info" updaten):

```

CHANGE MASTER TO
 MASTER_HOST = "<Host>"
 MASTER_USER = "<User>"
 MASTER_PASSWORD = "<Password>"
 MASTER_PORT = "<Port>"
 MASTER_LOG_FILE = <Filename>

```

```

MASTER_LOG_POS = <Position>
MASTER_CONNECT_RETRY = <Versuche>
RELAY_LOG_FILE = <Filename>
RELAY_LOG_POS = <Position>
MASTER_SSL = {0|1}
MASTER_SSL_CA = "<Filename>"
MASTER_SSL_CAPATH = "<Path>"
MASTER_SSL_CERT = "<Filename>"
MASTER_SSL_KEY = "<Filename>"
MASTER_SSL_CIPHER = "<Algorithm>"
MASTER_SSL_VERIFY_SERVER_CERT = {0|1}

```

Damit Master-Master-Replikation möglich ist (2 gegenseitige Master oder ein Ring aus mehreren Mastern), müssen die AUTO\_INCREMENT-Spalten auf jedem Master garantiert unterschiedliche Werte erzeugen. Dazu gibt es pro Datenbank-Server folgende Optionen:

```

auto_increment_offset = NNN
auto_increment_increment = NNN

```

Ist die Anzahl der sich gegenseitig replizierenden Master z.B. 5, so sollte jeder der Master einen der Offsets 1, 2, 3, 4, 5 erhalten und bei jedem Master der Increment auf (mindestens) 5 gesetzt werden.

ACHTUNG: Sollte an einem Master eine Tabellenzeile geändert werden, und sie wird vor ihrer Replikation dorthin auf einem anderen Master ebenfalls geändert, kann die Replikation auf den verschiedenen Mastern Abweichungen bezüglich dieser Tabellenzeile ergeben.

```

SELECT MASTER_POS_WAIT("master_log_file", master_log_pos);
 Stellt sicher, dass Slave Aktionen bis zu einer bestimmten Position im
 Masterlogfile gelesen und ausgeführt hat.
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = N;
 Überspringt die nächsten N Aktionen des Masters
 (z.B. weil ein Statement die Replikation abgebrochen hat).
 Nur gültig, wenn Slave-Thread nicht läuft.

```

Multi-Master -> Slave-Replikation wird nicht unterstützt.

Fehlermöglichkeiten:

- \* master\_id = server\_id
- \* Auf Slave slave\_id = ... statt master\_id = ... verwendet
- \* Grants nicht ausreichend
- \* Gemeinsamer Startpunkt verpasst
- \* rm /var/lib/mysql/master.info vergessen
- \* Unbedingt beide Server vor dem Kopieren der Ausgangsdatenbank runterfahren
- \* Unbedingt "log\_..." einschalten, damit man was sieht

#### 14) Gesicherte und verschlüsselte Verbindungen

-----

Eine Netzwerk-Verbindung kann abgehört und analysiert werden (Internet):

- \* Benutzername + Passwort verschlüsselt übertragen
- \* SQL-Anweisungen und Ergebnisdaten unverschlüsselt übertragen (ethereal)

Sicherheits-Maßnahmen:

- \* Kennwort für "root" sofort nach Installation vergeben
- \* Leere User (anonyme) sofort nach Installation entfernen
- \* User nur von expliziten Rechnern aus Verbindung aufnehmen lassen
  - + KEIN Hostname "%"
- \* MySQL-Server nicht als "root", sondern als Benutzer "mysql" starten
- \* Nur Benutzer "mysql" hat Lese- und Schreibzugriffe auf DB-Dateien
- \* Recht "PROCESS" und "FILE" nicht vergeben
  - + PROCESS -> Kennwortabfragen einsehen
  - + FILE -> Systemfiles mit "mysql"-Rechten in Tab. laden ("/etc/passwd")
- \* Anzahl Verbindungen auf sinnvollen Wert beschränken ("max\_connections")
- \* Einige MySQL-Optionen setzen (in "/etc/mysql/my.cnf")
  - skip-show-database # Keine Anzeige der vorhandenen Datenbanken
  - safe-show-database # Nur Datenbanken mit Zugriffsberechtigung anzeigen
  - safe-user-create # Benutzer per GRANT anlegen nur mit INSERT-Recht
    - # auf "user" möglich
  - skip-name-resolve # HOST-Namensauflösung aus (nur IPs verwendet)
  - skip-symlink # Keine symbolischen Links auf Tabellen erlaubt
- \* Dem MySQL-Server keine Netzwerk-Verbindung erlauben
  - skip-networking
- + Dann sind nur lokale Verbindungen per Socket erlaubt
- \* Verbindung zwischen MySQL-Server und Client über einen "SSH-Tunnel"
  - ssh -l <User> <Host>
  - ssh <User>@<Host>
  - /home/<User>/ssh/known\_hosts # beim ersten Mal

```
* Verschlüsselung des Protokolls zwischen MySQL-Server und Client per SSL
--with-openssl #
--with-vio #
Variable:
SHOW VARIABLES LIKE "have_openssl";
```

## 15) Troubleshooting

-----

MySQL ist eine sehr gut getestete Software, Abstürze sind selten, aber möglich:

- \* Falsche Zugriffsrechte oder Besitzverhältnisse
- \* Fehlende Verz. oder Dateien
- \* Hardware-Ausfall (Festplatten!, RAID)
- \* Angriffe
- \* Beschädigte Daten- oder Indexdateien (Reparieren)
- \* Fehlerhafte SQL-Befehle
- \* Fehlerhaft programmierte Client-Anwendungen
- \* Netzwerk-Störung/Unterbrechung
- \* Stromausfall

Ursachenforschung nach Absturz eines MySQL-Servers:

- \* Hardware überprüfen
- \* Fehler-Logdatei überprüfen
- \* Anfrage-Logdatei überprüfen
- \* Laufende Prozesse ausgeben
- \* Betriebssystem-Logdateien überprüfen
- \* Debug-Version des MySQL-Servers starten
  - Muss dazu mit "./configure" vorbereitet und übersetzt werden mit:
    - with-debug
    - with-debug=full

Performance-Optimierung:

- \* Loggen von langsamen Abfragen aktivieren
- \* Langsame/Blockierende SQL-Abfragen mit EXPLAIN ... überprüfen
- \* Indices löschen, anlegen, ändern
- \* SQL-Anweisungen ändern

Passworte zurücksetzen (z.B. falls vergessen):

- \* Benutzerkennwort (MySQL root-Passwort muss bekannt sein)
 

```
mysql -uroot -p
USE mysql;
UPDATE user SET password = PASSWORD("geheim")
WHERE user = "<User>" AND host = "<Hostame>";
FLUSH PRIVILEGES;
quit
```
- \* Root-Passwort (UNIX root-Passwort muss bekannt sein)
 

```
/etc/init.d/mysql stop # oder rcmysql stop
/usr/bin/mysqld_safe --skip-grant-tables &
mysql
USE mysql;
UPDATE user SET password = PASSWORD("geheim")
WHERE user = "root" AND host = "localhost";
FLUSH PRIVILEGES;
quit
killall mysqld_safe
/etc/init.d/mysql start # oder rcmysql start
```
- + Alternativ:
  - Datenbank-Dateien kopieren (bis auf "mysql!")
  - Datenbank neu aufsetzen
  - Datenbank-Dateien zurückkopieren (Zugriffsrechte + Besitzer setzen!)

Zugriffsverweigerung (ACCESS DENIED FOR USER: <User@Host> TO <Db>...):

- \* Benutzername nicht vorhanden
- \* Hostrechner nicht erlaubt
  - + Hostname nicht auflösbar
- \* Passwort vergessen, aber benötigt
- \* Passwort falsch
- \* Berechtigung zum Verbinden nicht vorhanden
- \* Datenbank-Zugriff nicht erlaubt
- \* UNIX\_Rechte falsch/fehlend

Häufige Fehlermeldungen:

```
CAN'T CONNECT TO MySQL SERVER
DATABASE EXISTS
FOUND WRONG PASSWORD FOR USER: ...
FOUND OLD STLYE PASSWORD FOR USER: ...
HOST <Host> IS BLOCKED BECAUSE OF TOO MANY CONNECTION ERRORS
PACKET TOO LARGE
TABLE <Tbl> DOESN'T EXIST
TOO MANY CONNECTIONS
```

UNKNOWN MySQL SERVER HOST <Host>

## 16) Query Optimizer

-----

### Eigenschaften:

- \* Laufzeit des "Query Optimizer" hängt von Anzahl verknüpfter Tabellen ab
  - + Steigt EXPONENTIELL mit der Anzahl Tabellen
  - + Für 7-10 Tabellen kein Problem, kann ab 30 Tabellen extrem lange dauern (sogar länger als die eigentliche Abfrage mit Full Table Scan)
- \* Optimizer steuern (lange Laufzeit <-> suboptimaler Plan)
  - + optimizer\_prune\_level = 1 # 1=Zu komplexe Pläne ignorieren (schneller)
  - + optimizer\_search\_depth = 0 # 0=Automatisch (je kleiner, desto schneller)
- \* Zugriffsplan des Optimizers zu <Statement> anzeigen
  - EXPLAIN <Statement>;
  - EXPLAIN EXTENDED <Statement>;

Analyse von Abfragen (was macht MySQL bei Ausführung in welcher Reihenfolge)

(Query Execution Plan = Ablaufplan des Query Optimizers):

- ```
EXPLAIN <Query>;
EXPLAIN EXTENDED <Query>;          # weitere Meldungen in Warnings
SHOW WARNINGS;                      # nach EXTENDED
```
- * Information über:
 - + Welche Indices werden benutzt
 - + Welche Join Reihenfolge
 - * Beeinflussen der Ablaufplans:


```
SELECT ... STRAIGHT_JOIN ...
USE INDEX (<Liste>)          # Nur diese Ind. nutzen (z.B. PRIMARY, auch leer!)
FORCE INDEX (<Liste>)       # Nur diese Ind. nutzen (Table Scan vermeiden!)
IGNORE INDEX (<Liste>)      # Indices nicht nutzen
max_join_size
```
 - * Evtl. Tabellenstatistik neu aufbauen sinnvoll, wenn Index nicht genutzt


```
ANALYZE TABLE <Tbl>;
myisamchk --analyze <Tbl>;
```
 - * Reihenfolge der Zeilen = Reihenfolge der Tabellenabfragen
 - + Für JEDE Datensatzkombination (Kreuzprodukt!) der vorherigen Tabellen werden Datensätze aus der nachfolgenden Tabelle gemäß "type" gelesen
 - * Ergebnis: Eine Zeile pro im SELECT verwendeter Tabelle sortiert nach der Zugriffsreihenfolge beim Bearbeiten der Query ("single-sweep-multi-join")

id	ID der Abfrage
select_type	Art der Abfrage
SIMPLE	Einfach (kein UNION oder Subquery)
PRIMARY	Äußerstes SELECT
UNION	Zweites oder späteres SELECT in UNION
DEPENDENT UNION	Zweites oder späteres SELECT in UNION (abh. von äußerer Query)
UNION RESULT	Ergebnis einer UNION
SUBQUERY	Erstes Select einer Subquery
DEPENDENT SUBQUERY	Zweites oder späteres SELECT in SUBQUERY
DERIVED	Abgeleitetes SELECT (in FROM)
UNCACHABLE SUBQUERY	Für jede Zeile einer äußeren Query neu zu generieren
UNCACHABLE UNION	Für jede Zeile einer äußeren Query neu zu generieren
table	Name der gelesenen Tabelle
type	Verknüpfung der Tabelle (Join)
system	Systemtabelle mit nur 1 Datensatz (Spezialfall von const)
const	Tabelle mit max.1 passenden Datensatz (z.B. PRIMARY KEY)
eq_ref	1 Datensatz pro Komb. vorheriger Tab. (PRIMARY KEY, UNIQUE)
ref	Alle Datensätze mit passenden Index pro ... (nicht eindeutig) (= <=>)
fulltext	FULLTEXT Index
ref_or_null	Analog "ref" + zusätzliche Suche nach NULL (IS NULL)
index_merge	Index Merge Optimierung benutzt
unique_subquery	IN Subquery
index_subquery	IN Subquery
range	Alle Datensätze im Bereich mit passendem Index pro ... (nicht eindeutig) (= <> > >= < <= IS NULL <=> BETWEEN IN)
index	Analog "all", aber nur Indexdaten lesen (schneller)
all	Full Table Scan (komplette Tabelle gelesen, schlecht!)
possible_keys	Prinzipiell benutzbare Indices (NULL = keine!)
key	Verwendeter Index (NULL = keiner!)
key_len	Breite des verwendeten Index (welche Teile verwendet)
ref	Im verwendeten Index benutzte Spalten
rows	Anzahl vermutlich zu lesender Zeilen (Produkt aller "rows" = Anzahl zu bearbeitender Zeilen!)
filtered	Prozentsatz vermutlich gefilterter Zeilen (EXPLAIN EXTENDED)
Extra	Zusatzinfo
distinct	1. passende Zeile genügt, restl. ignoriert (DISTINCT)
...	...
Using filesort	2 Durchläufe notwendig (schlecht!)
Using temporary	Temp. Tab. muss erstellt werden (GROUP BY + ORDER BY)

17) Tabellen-, Index- und Query-Cache

Indexstatistik:

- * Basiert auf "Valuegroups" = Anz. Datenzeilen mit gleichem Index
- * Durchschnittliche Größe der Valuegroups
- * Je kleiner desto besser
- * Tabellenkardinalität = Anzahl Valuegroups

N/S (N=Anzahl Datenzeilen, S=Durchschnittliche Valuegroup-Größe)

```
SHOW INDEX FROM <Tbl>;
SHOW KEYS FROM <Tbl>;
SHOW INDEX FROM <Tbl> FROM <Db>;
SHOW INDEX FROM <Db>.<Tbl>;
mysqlshow --keys <Db> <Tbl>
Table          Tabellenname
Non_unique     0=eindeutig, 1=Duplikate möglich
Key_name       Indexname
Seq_in_index   Position der Spalte im (zusammengesetzten) Index
Column_name    Spaltenname
Collation      Sortierreihenfolge (A=ascending, NULL=nein)
Cardinality    Schätzung der Anzahl eindeutiger Werte im Index
                (je höher desto besser für JOIN geeignet, ANALYZE TABLE)
Sub_part       Länge des indizierten Präfixes (NULL=vollständig)
Packed         Key gepackt (NULL=Nein)
Null           Spalte kann NULL-Werte enthalten: YES/NO
Index_type     Indexmethode: BTREE, FULLTEXT, HASH, RTREE
Comment        Kommentar
```

* MyISAM-Indexstatistik

```
mysam_stats_method = "nulls_equal"   (alle NULL-Werte gleich)      -> Valuegroup-Größe überschätzt
mysam_stats_method = "nulls_unequal" (alle NULL-Werte verschieden) -> Valuegroup-Größe unterschätzt
mysamchk --stats_method=... --analyze
ANALYZE TABLE <Tbl>, ...;
CHECK TABLE <Tbl>, ...;
OPTIMIZE TABLE <Tbl>, ...;
```

Keycache (MyISAM):

- * Systemvariable "key_buffer_size" (0=kein) -> Default Keycache (nicht löschar)
- * Mehrere möglich, Zuordnung von Indices zu best. Keycache möglich (bei jedem Serverneustart verloren)


```
CACHE INDEX <Tbl1>, ... IN <Cache1>;
SET GLOBAL <Cache1>.key_buffer_size = 128 * 1024; # Anlegen
SET GLOBAL <Cache1>.key_buffer_size = 0;         # Löschen
SET GLOBAL key_buffer_size = 0;                 # Default Cache Löschen geht nicht!
SHOW VARIABLES LIKE "key_buffer_size";          # Anzeigen
```
- * Die häufigsten Indexblöcke werden gecacht (+ OS)
 - + Nonleaf Nodes + Leaf Nodes
- * Datenblöcke werden nicht gecacht (-> OS)
- * Vorschlag: 3 Keycache für belasteten Server (blockieren sich nicht gegenseitig!)
 - + "Hot": 20% - Häufig durchsuchte Tabellen ohne Updates
 - + "Cold": 20% - Mittelgroße häufig modifizierte Tabellen (z.B. temporäre)
 - + "Warm": 60% - Default für alle anderen Tabellen
- * Initialisierung der 3 Keycache gleich beim Serverneustart:


```
key_buffer_size      = 4G
hot.key_buffer_size  = 2G
cold.key_buffer_size = 2G
init_file=/path/to/data-directory/mysqld_init.sql
CACHE INDEX db1.t1, db1.t2, db2.t3 IN hot;
CACHE INDEX db1.t4, db2.t5, db2.t6 IN cold;
```
- * Std: LRU-Strategie (Last Recently Used)
- * Midpoint Insertion Strategy


```
key_cache_division_limit = 100 # Ausgeschaltet (nur LRU)
key_cache_division_limit < 100 # Eingeschaltet
```

 - + Warm Subchain + Hot Subchain
- * Index Preloading (manuell in Key-Buffer laden)
 - + In sequentieller Reihenfolge vorausladen, wenn genug Platz


```
LOAD INDEX INTO CACHE
{<Tbl1> [INDEX (<Idx>, ...)], ...}
[IGNORE LEAVES]
[IN <Cache>]; #
```
- * Keycache Blockgröße


```
key_cache_block_size (automatisch gesetzt)
```
- * Keycache restrukturieren (durch Zuweisen einer neuen Größe)


```
SET GLOBAL cold_cache.key_buffer_size = 4 * 1024 * 1024;
```
- * Key-Hit-Ration ermitteln (möglichst nahe bei 100%):


```
SHOW GLOBAL STATUS LIKE "key_read%";
100 * (key_reads * 100 / key_read_requests)
```

Maximale Anzahl an offenen Dateien (OS-abhängige Obergrenze pro Prozesse):
 table_open_cache = Anz. Verbindungen x Max. Anzahl Tab. pro Join (Std: 64)
 max_connections
 max_tmp_tables
 --open-files-limit
 FLUSH TABLES; -> Offen Tabellen werden geschlossen
 open_tables
 opened_tables
 flush_commands

Query-Cache (Abfrage-Speicher):

- * SELECT-Anweisung + Ergebnis wird gespeichert
- * Falls vollständig identische Anweisung (GROSS/kleinschreibung, Leerzeichen) erneut auftritt, sofort Ergebnis zurückgegeben (Parsen vermieden!)
- * Sinnvoll bei Tabellen mit wenig Änderungen und vielen gleichen Abfragen (Tabellen-Änderungen -> abhängige Query-Cache Einträge gelöscht)
- * Etwa 10% Overhead, falls nie gleiche Anfrage kommt
- * Bis zu 250% schneller, falls gleiche Anfragen sehr oft kommen
- * Wird von allen Sitzungen geteilt (einer pro Server)
- * Sinnvoll: 32 - 128 MByte
- * Systemvariable:
 - maximum_query_cache_size = 0 # Std: Max. einstellbare Größe
 - query_cache_size = 0 # Std: Overhead vollständig weg
 - query_cache_type = 0/ON/DEMAND # 0/OFF = Aus
 - # 1/ON = Alle außer SQL_NO_CACHE
 - # 2/DEMAND = Nur die mit SQL_CACHE
 - query_cache_limit = 1MB # Max. Größe eines Cache-Eintrags
 - query_cache_min_res_unit = 4KB # Min. Größe eines Cache-Eintrags
 - have_query_cache = NO/YES
 - SET GLOBAL query_cache_size = 32000000 # Byte
- * SELECT-Ergebnis im Query-Cache aufheben/nicht speichern (falls möglich):
 SELECT SQL_CACHE * FROM <TABLE> ...;
 SELECT SQL_NO_CACHE * FROM <TABLE> ...;
- * Query-Cache defragmentieren (nicht leeren):
 FLUSH QUERY CACHE;
- * Query-Cache leeren:
 RESET QUERY CACHE;
 FLUSH TABLES;
- * Query-Cache Status + Auslastung beobachten:
 SHOW STATUS LIKE "qcache_%"; # bzw. "qc%"
- * Query-Einstellungen ansehen:
 SHOW VARIABLES LIKE "query%";

18) MySQL-Proxy

Zweck:

- * Profiling
- * Monitoring Execution Time/Progress
- * Connection Pooling (Round-Robin)
- * Load Balancing
- * Failover
- * Query Analysis
- * Query Filtering (Authentifizierung/Zugriffsschutz)
- * Query Manipulation/Modification/Transformation
- * Umfrage: <http://dev.mysql.com/tech-resources/quickpolls/mysql-proxy.html>

Eigenschaften:

- * Klassische "Middleware"-Komponente
- * Kann MySQL-Netzwerkprotokoll (Client und Server merken nichts vom Proxy)
- * Ermöglicht Kommunikation zwischen mehreren Clients und mehreren Server
- * Eingebaute Scriptsprache "Lua" (8-())

19) Partitionierung

Teilt Tabellen + Indices PHYSIKALISCH gemäß definierbarer Regel in Stücke auf

- * Horizontale Zerlegung der Daten (Datenzeilen)
- + Vertikale Zerlegung (Datenspalten) NICHT möglich (manuell selber machen)
- + Anderer Name: Tablespace (InnoDB)
- + Für Anwender transparent, d.h. merkt nichts davon
- + Insbesondere DATE/TIME/DATETIME-Spalten bieten sich zur Partitionierung an (bzw. Funktionen davon: TO_DAYS, YEAR, WEEKDAY, DAYOFYEAR, MONTH)
- * Arbeitsweise
 - + Partitionen IMMER von 0..N durchnummeriert
 - + Pro Datenzeile Nummer ermittelt und für Speicherung in Partition benutzt
 - + Partitionierungs-Funktion REGEL(BASIS) = MUSS Wert aus 0..N ergeben
 - Basis: Eine Integer-Spalte (auch DATE/TIME/DATETIME!)
 - Funktion auf Spalte(n) mit Integer-Ergebnis (NULL oder positiv!)
 - Regel: Wertebereiche (RANGE)

- Wertelisten (LIST)
- Hash-Funktion (HASH, KEY)
- Kombiniert (COMPOSITE = erst RANGE/LIST, dann HASH/KEY)
- * Für viele Storage Engines erlaubt (alle Partitionen einer Tabelle dieselbe)
 - + Nicht möglich: MERGE, CSV, FEDERATED
 - + Teilweise: NDBCLUSTER # Nur KEY, LINEAR KEY
- * Pro Partition Daten + Indices auf versch. Verz. verteilbar (nicht InnoDB)


```
CREATE TABLE ...
PARTITION BY ... DATA DIRECTORY "<Path>"
INDEX DIRECTORY "<Path>"
```
- * Partition-Pruning: WHERE-Bedingung schließt Partitionen sofort von Abfrage aus und minimiert somit zu verarbeitende Daten
 - + WHERE <PartCol> = <Konstante> # auch < <= > >= <>
 - + WHERE <PartCol> IN (<Konstante>, ...)
 - + WHERE <PartCol> BETWEEN <Left> AND <Right>
- * Sonstiges
 - + Bei Partitionsnamen wird GROSS/kleinschreibung ignoriert
 - + Partitionen sind reorganisierbar (Datensätze umverteilen)
 - + Partitionierungs-Operationen setzen WRITE LOCK auf Tabelle
 - + Evtl. Konfigurationsvariable "open_files_limit" hochsetzen

Syntax:

```
CREATE TABLE ... [<Part>];

<Part> = PARTITION BY {
  [LINEAR] HASH (<Expr>) |
  [LINEAR] KEY (<ColList>) |
  RANGE {(<Expr> | COLUMNS (<ColList>)} |
  LIST {(<Expr> | COLUMNS (<ColList>)}
}
[PARTITIONS <N>]
[SUBPARTITION BY {
  [LINEAR] HASH (<Expr>) |
  [LINEAR] KEY (<ColList>)
}
[SUBPARTITIONS <N>]
]
[(<PartDef> [, <PartDef>] ...)]

<PartDef> = PARTITION <PartName>
[VALUES {
  LESS THAN {(<Expr> | <ValList>) | MAXVALUE } |
  IN (<ValList> | <ValList>)
}]
{<PartOpt>, ...}
[(<SubPartDef> [, <SubPartDef>] ...)]

<SubPartDef> = SUBPARTITION <SubPartName>
{<PartOpt>, ...}
```

Zweck:

- * Plattenplatz/Dateisystem-Beschränkungen umgehen
- * Abfragen stark beschleunigen (WHERE-Bedingung schließt Partitionen aus)
- * (Un)Interessant gewordene Daten leicht entfernbar/hinzufügbar (Datum!)
- * Abfragen über mehrere Threads verteilbar (kann MySQL nicht!)
- * Abfragen mit Aggregatfunktionen parallelisierbar: SUM(), COUNT(), ...
- * Datenverteilung auf mehrere Platten kann Zugriff beschleunigen

Einschränkungen:

- * Max. 1024 Partitionen (inkl. Subpartitionen) pro Tabelle
- * Erlauben keine "Foreign Keys" (weder bei sich noch in anderen Tabellen)
- * Erlauben keinen Fulltext Index
- * Erlauben keine Spatial Datentypen (POINT, GEOMETRY, ...)
- * Temporäre Tabelle nicht partitionierbar
- * Logtabelle nicht partitionierbar
- * ALTER TABLE ... ORDER BY sortiert Sätze nur pro Partition
- * Partitioning Key: Alle Spalten im Partitioning-Ausdruck müssen Teil JEDES UNIQUE-Index + Primary-Key sein. Jeder UNIQUE-Index + Primary-Key müssen alle Spalten des Partitionierungs-Ausdrucks enthalten, AUSSER eine Tabelle enthält keine UNIQUE-Keys, dann ist jede Spalte erlaubt.

Zerlegungsregeln:

Regel	Bedeutung
RANGE	Basis: Integer-Wertebereich (z.B. 1980-1989, 1990-1999, ...) (ohne Lücken, nicht überlappend, aufsteigend)
LIST	ELSE-Zweig: PARTITION <Part> VALUES LESS THAN MAXVALUE Basis: Integer-Werteliste (z.B. 2006, 2007, 2008, 2009, ...) (ohne Lücken, nicht überlappend, aufsteigend unnötig)

HASH	ELSE-Zweig nicht möglich, alle Werte abdecken nötig Basis: Benutzerdef. Spalten-Ausdruck -> Integer + Anz. Part. Verwendet Divisionsrest MODULO Anz. Partitionen, Berechnung kostet Zeit
KEY	Ausdruck sollte "linear" sein (nichtl. verteilt ungleich) Basis: Primary Key oder UNIQUE-Index + MD5() (Gleichverteilung durch DB, auch CHAR statt INT erlaubt)
COMPOSITE	Unterzerlegung (Subpartitioning) in weitere Partitionen RANGE/LIST weiter zerlegt durch HASH/KEY Jede Partition muss gleiche Anzahl Unterpartitionen haben Max. ist eine zweistufige Zerlegung möglich
LINEAR HASH	Power-of-2 Algorithmus statt Modulo -> schneller
LINEAR KEY	Power-of-2 Algorithmus statt Modulo -> schneller

Wann welche Partitionierung?

- * RANGE: Alte Daten sollen ab und zu gelöscht werden
Abfrage basieren häufig auf Partitionierungs-Spalte
- * LIST: Analog RANGE
- * HASH: Gleichverteilung der Daten auf Partitionen
- * KEY: Analag HASH
- * COMPOSITE: Sehr große Datenmengen

Wie wird NULL behandelt?

- * RANGE: In erster Partition (kleiner als alle Integer, analog ORDER BY)
- * LIST: "NULL" muss in einer Werteliste enthalten sein (sonst verboten)
- * HASH: Analog Wert "0" behandelt
- * KEY: Analog Wert "0" behandelt

Wird Partitionierung vom MySQL-Server unterstützt?

```
SHOW VARIABLES LIKE "%partition%" # -> have_partitioning YES
SHOW PLUGINS; # -> partition ACTIVE ...
```

Beispiele:

```
CREATE TABLE members (
  vorname VARCHAR(25) NOT NULL,
  nachname VARCHAR(25) NOT NULL,
  account VARCHAR(16) NOT NULL,
  email VARCHAR(35) ,
  seit DATE NOT NULL
)
PARTITION BY RANGE(YEAR(seit)) (
  PARTITION p0 VALUES LESS THAN (1960),
  PARTITION p1 VALUES LESS THAN (1970),
  PARTITION p2 VALUES LESS THAN (1980),
  PARTITION p3 VALUES LESS THAN (1990),
  PARTITION p4 VALUES LESS THAN MAXVALUE # Rest (ELSE-Zweig)
);

CREATE TABLE t2 (
  val INT
)
PARTITION BY LIST(val) ( # ALLE möglichen Werte auflisten!
  PARTITION p0 VALUES IN (1,3,5),
  PARTITION p1 VALUES IN (2,4,6),
  PARTITION p2 VALUES IN (NULL), # Notwendig (oder NOT NULL bei val)
);

CREATE TABLE employees (
  id INT NOT NULL,
  fname VARCHAR(30),
  lname VARCHAR(30),
  hired DATE NOT NULL DEFAULT "1970-01-01",
  separated DATE NOT NULL DEFAULT "9999-12-31",
  job_code INT,
  store_id INT
)
PARTITION BY LIST(store_id) (
  PARTITION pNord VALUES IN (3,5,6,9,17),
  PARTITION pWest VALUES IN (1,2,10,11,19,20),
  PARTITION pOst VALUES IN (4,12,13,14,18),
  PARTITION pSued VALUES IN (7,8,15,16)
);

CREATE TABLE ti (
  id INT,
  amount DECIMAL(7,2),
  tr_date DATE
) ENGINE = INNODB
PARTITION BY HASH (MONTH(tr_date))
PARTITIONS 6; # Anz. Partitionen (Default: 1)
```

```
CREATE TABLE members (
  firstname VARCHAR(25) NOT NULL,
  lastname  VARCHAR(25) NOT NULL,
  username  VARCHAR(16) NOT NULL,
  email     VARCHAR(35) ,
  joined    DATE        NOT NULL
)
PARTITION BY KEY(joined)
PARTITIONS 10;                                # Anz. Partitionen (Default: 1)
```

```
CREATE TABLE ts (
  id INT, purchased DATE
)
PARTITION BY RANGE (YEAR(purchased))
SUBPARTITION BY HASH (TO_DAYS(purchased))
SUBPARTITIONS 2
(
  PARTITION p0 VALUES LESS THAN (1990),
  PARTITION p1 VALUES LESS THAN (2000),
  PARTITION p2 VALUES LESS THAN MAXVALUE
);
```

```
CREATE TABLE ts (
  id INT, purchased DATE
)
PARTITION BY RANGE (YEAR(purchased))
SUBPARTITION BY HASH (TO_DAYS(purchased))
(
  PARTITION p0 VALUES LESS THAN (1990) (
    SUBPARTITION s0,
    SUBPARTITION s1
  ),
  PARTITION p1 VALUES LESS THAN (2000) (
    SUBPARTITION s2,
    SUBPARTITION s3
  ),
  PARTITION p2 VALUES LESS THAN MAXVALUE (
    SUBPARTITION s4,
    SUBPARTITION s5
  )
);
```

Daten in Partitionen löschen:

```
TRUNCATE TABLE t;                            # Struktur+Partitionen bleiben, Daten löschen
ALTER TABLE t DROP PARTITION p0;            # Part. löschen (inkl. Daten, effizient)
```

Partitionen auflösen/hinzufügen (LIST + RANGE, Daten bleiben erhalten!):

```
ALTER TABLE t REMOVE PARTITIONING;          # Partit. entf., Daten beibehalten
ALTER TABLE t ADD PARTITION (PARTITION p3 VALUES LESS THAN (2000), ...);
                                                # RANGE: Nur hinten anhängen
ALTER TABLE t ADD PARTITION (PARTITION p3 VALUES IN (7, 14, 21), ...);
                                                # LIST: Nur NEUE Werte erlaubt
```

Partitionstyp ändern (geht nicht mit REORGANIZE):

```
ALTER TABLE t PARTITION BY KEY (id) PARTITIONS 2;
```

LIST + RANGE Partitionen mergen/splitten:

```
# Zerlegen (split), Daten beibehalten
ALTER TABLE t REORGANIZE PARTITION p0 INTO (
  PARTITION s0 VALUES LESS THAN (1960),
  PARTITION s1 VALUES LESS THAN (1970)
);
# Zusammenfügen (merge), Daten beibehalten
ALTER TABLE t REORGANIZE PARTITION s0,s1 INTO (
  PARTITION p0 VALUES LESS THAN (1970)
);
# Zusammenfügen (merge) und zerlegen (split) gleichzeitig, Daten beibehalten
ALTER TABLE members REORGANIZE PARTITION p0,p1,p2,p3 INTO (
  PARTITION m0 VALUES LESS THAN (1980),
  PARTITION m1 VALUES LESS THAN (2000)
);
```

HASH + KEY Partitionen mergen/splitten:

```
ALTER TABLE t COALESCE PARTITION 4;         # 4=Anz. zu entfernender Part.
ALTER TABLE t ADD PARTITION PARTITIONS 6;  # 6=Anz. hinzuzufügender Part.
```

Partitionen optimieren und pflegen:

```
ALTER TABLE t REBUILD PARTITION p0;      # Defragmentierung
ALTER TABLE t OPTIMIZE PARTITION p1;     # Defragmentierung + Leerraum freigeb.
                                           # CHECK + ANALYZE + REPAIR
ALTER TABLE t CHECK PARTITION p3;       # Fehler erkennen
ALTER TABLE t ANALYZE PARTITION p2;     # Key-Verteilung ermitteln
ALTER TABLE t REPAIR PARTITION p3;      # Fehler beheben
ALTER TABLE t TRUNCATE PARTITION p2;    # Partition löschen (MY!5.5)
ALTER TABLE t TRUNCATE PARTITION ALL;   # Alle Partitionen löschen (MY!5.5)
```

Partitionen anzeigen oder Benutzung erklären lassen:

```
SHOW CREATE TABLE t1;
SHOW TABLE STATUS t1;
SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
  FROM information_schema.PARTITIONS
  WHERE TABLE_SCHEMA = "p" AND TABLE_NAME = "th";
EXPLAIN PARTITIONS
  SELECT COUNT(*) FROM employees
  WHERE separated BETWEEN "2000-01-01" AND "2000-12-31"
  GROUP BY store_id;
```

Partitions-Optionen <PartOpt> (das "=" ist immer optional):

```
[STORAGE] ENGINE [=] <Engine>           # Part. einer Tab. müssen gleiche E. haben!
COMMENT [=] "<Text>"                     # Kommentar zur Partition
DATA DIRECTORY [=] "<Path>"              # Abs. Pfad für Datenablage der Partition
INDEX DIRECTORY [=] "<Path>"            # Abs. Pfad für Indexablage der Partition
MAX_ROWS [=] <N>                          # Max. Anzahl Datenzeilen (nur Hinweis!)
MIN_ROWS [=] <N>                          # Min. Anzahl Datenzeilen (nur Hinweis!)
TABLESPACE [=] (<TblSp>)                  #
NODEGROUP [=] <N>                         #
```
