HOWTO zur MySQL-Verwaltung (DB-Administrator)

(C) 2006-2018 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 1.248 2020/02/24 06:36:18 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-Datenbank-Server installieren (SuSE)
3) Serveradministration
3a) Server starten und stoppen
3b) Konfiguration: Variablen und Optionen
3c) Konfigurations- und Datenbankdateien
3d) Interne Verwaltungsdatenbanken
3e) SQL-Modus (sql_mode)
4) MySQL Kommandozeilen-Programme
4a) mysqladmin
4b) mysqlshow
5) Benutzer und Berechtigungen (Privileges)
6) Datenbank-Engines (Storage Engines/Backends)
6a) InnoDB Engine
6b) MyISAM Engine
6c) MEMORY Engine (HEAP)
6d) CSV Engine (Comma Separated Values, ab MY!5.1)
6e) ARCHIVE Engine
6f) BLACKHOLE Engine
6g) MERGE Engine
6h) FEDERATED Engine
6i) EXAMPLE Engine
6j) Weitere Engines
7) Datenbanken erstellen und verwalten
8) Tabellen erstellen und verwalten
8a) Temporäre Tabellen
9) Tabellen und Indices 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 (EXPLAIN)
17) Tabellen-, Index- und Query-Cache
18) Partitionierung
19) MySQL Hilfsprogramme
19a) MySQL Utilities
19b) Percona Toolkit
20) MySQL Cluster

1) Das Datenbankmanagementsystem MySQL

1a) Grundlegendes

GROSS/Kleinschreibung wird in MySQL (fast) nicht berücksichtigt:
  --> mysql-user-HOWTO.txt --> 2g) GROSS/Kleinschreibung

UNIX-Zugriffsrechte und -Besitzverhältnisse sind wichtig:
  * Meist gehört alles Benutzer "mysql" + Gruppe "mysql"

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

Optionen zu MySQL-Kommandos können auf der Kommandozeile und in
Konfigurations-Dateien angegeben werden. Auf der Kommandozeile ist der
Bindestrich "-" als Bestandteil üblich, in Konfigurations-Dateien 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")!

ACHTUNG: Optionen im richtigen ABSCHNITT (Gruppe) der Konfigurations-Datei
(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
[mysqldump]	Client "mysqldump" 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:

Schreibweise	Bedeutung
--read-buffer-size=32M	Option
read_buffer_size = 32M	Konfigurations-Datei

```
  | SET @@read_buffer_size = 32M; | MySQL-Anweisung      |
  +-------------------------------+----------------------+
```

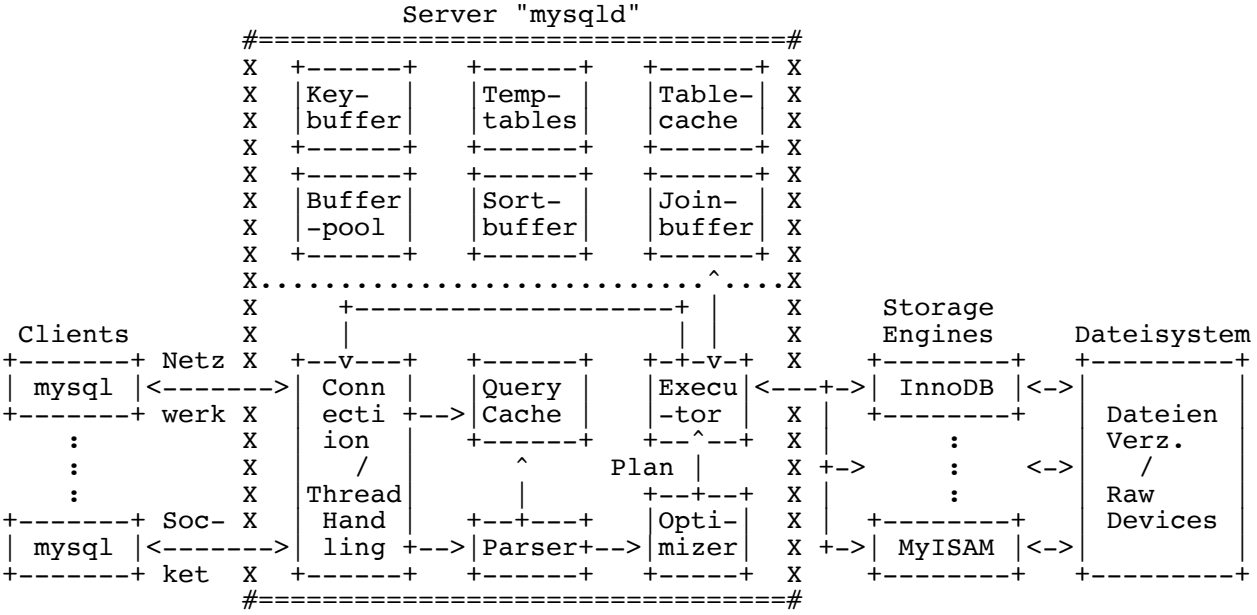
1b) Allgemeine Informationen

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)
```

Architektur:

```
  * N MySQL-Clients + 1 MySQL-Server (auf anderem oder gleichen Rechner)
    + Client: mysql (Kommandoschnittstelle, eine von vielen)
    + Server: mysqld (Daemon, Instanz)
  * MySQL-Server-Schichten
    + Oben:  Benutzer-Schnittstelle (API, SQL, DB-Management)
    + Mitte: Storage Engine
    + Unten: Dateisystem
```



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

```
  n Hosts (Rechner) haben                    # (meist nur einer)
    s MySQL-Server (Instanzen) mit           # mysqld (nur einer auf Port 3306)
      d Datenbanken pro MySQL-Server mit     # /var/lib/mysql/<Db>
        t Tabellen pro Datenbank mit         # /var/lib/mysql/<Db>/<Tbl>.*
          c Spalten pro Tabelle und          #
          z Datensätze/Zeilen/Rows pro Tab. und
          i Indices pro Tabelle              #
        p Prozeduren pro Datenbank           # In Verwaltungstabelle
        f Funktionen pro Datenbank           # In Verwaltungstabelle
        g Trigger pro Datenbank-Tabelle      # In Verwaltungstabelle
        v Views pro Datenbank                # In Verwaltungstabelle
        e Events pro Datenbank               # In Verwaltungstabelle
```

Storage Engines (pro Tabelle, STD: InnoDB seit MY!5.5.1, vorher MyISAM):

```
  --> 6) Datenbank-Engines (Storage Engines/Backends)
```

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)

* Bekannte Nutzer (inzw. teilweise Wechsel zu MariaDB):
  + Amazon
  + Facebook
  + Google
  + Pinterest
  + Twitter
  + Wikipedia

1c) Eigenschaften/Einschränkungen der MySQL-Versionen

Vorteile von MySQL:
  * Einfache Installation/Administration
  * Schnell (durch Verzicht auf Funktionalitäten)
  * Stabil
  * Portabel
  * Einfache Nutzung
  * Viele Schnittstellen
  * Gute Integration mit PHP
  * Unterschiedliche Engines pro Tabelle/DB möglich
  * Keine/geringe Kosten

Nachteile von MySQL:
  * ANSI SQL-Standard nicht vollständig erfüllt
  * Referenzielle Integrität nicht sichergestellt (außer bei InnoDB)
  * ACID-Prinzip nicht durchgehalten ("Atomizität" statt Transaktionen)
    (ACID = Atomic, Consistent, Isolated, Durable)
  * Nur für "Spiel/Web-Datenbank" --> auf Lesen optimiert
  * Performanceprobleme bei Join mehrerer Tabellen
  * Optimizer vergleichsweise einfach
  + Standardmäßig keine GUIs (reines DBMS mit Kommandozeilen-Tools)

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 Benutzer-Authentifizierung (etwa via LDAP)
  * Ineffizienter interner SQL-Parser
  * Mangelnde Community-Beteiligung
  * Unklare Release-Politik (Oracle)

Versionen:
  * Enterprise Edition   # Kostenpflichtig
  * Community Edition   # Alle stabilen Defaultfunktionen (GPL)

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

Duale Lizensierung:
  * Nur EINE Lizenz pro Maschine (auch wenn Multiprozessor!) notwendig!
    + Keine Einschränkung bzgl. Anzahl Prozessoren und Clients
  * Kein Kauf einer kommerziellen Lizenz nötig bei:
    + Internem 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:
  * DMR = Development Milestone Release (Entwickler)
  *

```
  * RC  = Release Candidate
  * GA  = Generally Available (Production Ready)

ANSI-SQL-Standard nur eingeschränkt erfüllt (keine 100% Kompatibilität):
  * Option --ansi (nur ANSI-SQL akzeptieren)
      ||            # Verkettet Zeichenketten statt OR
      FUNCNAME (    # Leerzeichen vor "(" OK (FUNCNAME dann reserviertes Wort)
      "..."         # Quotierung von Namen die Schlüsselworte sind (statt `...`)
      REAL          # Entspricht FLOAT statt DOUBLE
      SERIALIZABLE  # STD.-Isolationslevel statt REPEATABLE READ
      GROUP BY      # Alle selektierten Spalten müssen in GROUP BY vorkommen
  * Analog --sql_mode=REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, \
                      IGNORE_SPACE, ONLY_FULL_GROUP_BY
           --transaction-isolation=SERIALIZABLE
  * Analog als SQL-Kommando:
      SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
      SET GLOBAL sql_mode = "ANSI";
  * 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
    (sonst quotiert `...` SQL-Schlüsselworte)

Versionsübersicht:

  +----------------------------+-------------------------------------------------+
  | Eigenschaft                | Bemerkung                                       |
  +----------------------------+-------------------------------------------------+
  | Version 3.23 (2001, erste brauchbare)                                        |
  +----------------------------+-------------------------------------------------+
  | MyISAM-Engine              | Statt ISAM-Engine                               |
  | InnoDB-Engine              | Verfügbar (aber nicht im Standard-Paket)        |
  | Fulltext Indexing          | Mit MyISAM                                      |
  | Statement based Replication| Killer feature (1 Thread auf Replika)!          |
  | Transaktionen              | Eingeschränkt (nur für InnoDB und BDB)          |
  +----------------------------+-------------------------------------------------+
  | Version 4.0 (2003)                                                           |
  +----------------------------+-------------------------------------------------+
  | SSL-Verschlüsselung        | Client-Server Kommunikation                     |
  | InnoDB-Engine              | Standardmäßig vorhanden                         |
  | Statement based Replication| Neu geschrieben mit 2 Threads auf Replika       |
  | Fulltext search            | Mit MyISAM                                      |
  | Unions                     |                                                 |
  | Multi-Table DELETE         |                                                 |
  | Transaktionen              | Voll (nur für InnoDB und BDB)                   |
  | RAW-Partitionen            | Nur für InnoDB                                  |
  | Referenzielle Integrität   | Nur für InnoDB (Foreign Keys)                   |
  | ROW-level Locking          | Nur für InnoDB                                  |
  | Query-Cache                |                                                 |
  +----------------------------+-------------------------------------------------+
  | Version 4.1 (2005)                                                           |
  +----------------------------+-------------------------------------------------+
  | Unicode Support            | Verschiedene Zeichensatz-Collations             |
  | Character Collation        | Zeichensatz + Sortierregeln pro Zeichenwert     |
  | Prepared Statements        |                                                 |
  | Geometric Datatype         | GIS=Geometric Information System (nur MyISAM)   |
  | RTREE Index                | GIS=Geometric Information System (nur MyISAM)   |
  | Views                      | Read-only (Derived Table)                       |
  | Clustering                 | NDB-Engine (Network DataBase)                   |
  | Konstante TRUE/FALSE       |                                                 |
  | Binary Protocol            |                                                 |
  | INSERT INTO ... ON DUPLICATE KEY UPDATE | Verbessertes REPLACE               |
  +----------------------------+-------------------------------------------------+
  | Version 5.0 (2006)                                                           |
  +----------------------------+-------------------------------------------------+
  | Stored Procedures          |                                                 |
  | Functions (UDF)            | User Defined Functions (extern in C program.)   |
  | Views                      | Updateable                                      |
  | Cursors                    | Read only                                       |
  | Trigger                    | Rudimentär                                      |
  | Precision Math             | Kompakte DECIMAL, keine DOUBLE-Rechnung mehr    |
  | ISAM-Engine                | Entfernt (MyISAM ist vollständiger Ersatz)      |
  | ARCHIVE-Engine             |                                                 |
  | FEDERATED-Engine           | Fernzugriff auf Tabellen auf externem Server    |
  | BIT Datentyp               |                                                 |
  | INFORMATION_SCHEMA         | Metadaten in virt. Tab. (Inhalt von "mysql")    |
```

XA Transaktionen	
Version 5.1 (2008, erste unter SUN Microsystems)	
Trigger	Voll
Foreign keys	Ab 3.23 für InnoDB
Full outer Join	Nein!
Constraints	
Event Scheduler	Periodisch automatisch Tätigkeit durchführen
XPath-Support	XML (2 neue Funk.: ExtractValue, UpdateXML)
INFORMATION_SCHEMA	Stark erweitert
Partitionierung	Tabellen/Indices per RANGE/HASH/KEY
Row-based Replikation	Alt: SQL-Kommando ausgeführt, Neu: Satzbasiert
Column Level Constraint	
Online Backup	Ohne Performanz-Einbußen
Engine Plugin-API	Engines nachträglich hinzufügen/entfernen
BDB-Engine entfernt	
Logs in DB ablegen	Für General+Slow Query Log (nicht Binary Log)
mysqlnd-Bibliothek	Für PHP (ersetzt libmysql)
Upgrade Tool	"mysql_upgrade"
MySQL Workbench	Visueller Entwurf von DB-Schemata
MySQL Migration Toolkit	OK: Tabellen+Inhalt; NEIN: Prozed., Trigger
MySQL Enterprise Monitor	MySQL-Serverstatus verfolgen, Optimierung
Version 5.4 (2009)	
Google Patches	Skalierbarkeit (InnoDB-Backend beschleunigt)
Subquery/Join	Beschleunigung durch Optimizer
Stored Procedures	SIGNAL/RESIGNAL, SQL-Fehlerbehandlung
INFORMATION_SCHEMA	PARAMETERS, ROUTINES
Prepared Statements	OUT Parameter Support
Version 5.5 (2010, erste unter Oracle)	
Semisynchronous Replikation	Master block. Tr. bis mind. 1 Slave bestätigt
SIGNAL/RESIGNAL	Fehler in Stored Proz/Funkt/Handler melden
Partitionierung	Zerleg. anhand meh. Spalten/String/Datum/Zeit (RANGE COLUMNS, LIST COLUMNS)
XML Funktionalität	Erweitert (LOAD XML)
InnoDB-Plugin	Standard-Engine (ab MY!5.5.1, fest eingebaut)
PERFORMANCE_SCHEMA	17 interne Tab. zur Performanz-Messung
Authentifizierung	Alternative PAM/LDAP/..., PROXY USER
Version 5.6 (2013) (fehlenden Funktionen in InnoDB nachrüsten, Serverleistung verbessern)	
InnoDB	Volltextsuche, portable Table Spaces, DDL-Anweisungen online (ohne Tabellenkopie), Multi-Range-Read+Batched Key Access DISCARD/IMPORT TABLESPACE
Query Optimizer	Subqueries verbessert (nach Jahren das 1.Mal); Anfragen mit wenig sort. Ergebniszeilen bei LIMIT, IN(...) beschleunigt; Statistiken genauer und stabil (dauerhaft); ICP (Index Condition Pushdown)
PERFORMANCE_SCHEMA	17 --> 49 interne Tab. zur Performanz-Messung
Views	TEMPTABLE-View WHERE-optimiert + Indiziert
JSON Format Output	Ergebnisausgabe in JavaScript Object Notation
GTID	Global Transaction Identifier (Replikation)
EXPLAIN	Für INSERT, UPDATE, DELETE; Ausgabe verbessert
TIMESTAMP, TIME, DATETIME	Sekundenbruchteile (.NNNNNN)
Replikation	Semi-synchron, verzögert, Row Image Control
Sonstiges	mysql_config_editor, ~/.mylogin.cnf Password-Validierungs-Plugin Neue GIS-Tests (ST_Crosses, ST_Intersects)
Version 5.7 (Okt 2015)	
Trigger	Mehrere pro Event + Reihenf. (FOLLOWS <Trig>)
Generated Columns	Spalte per Ausdruck berech. (virtuell/stored)
JSON-Datentyp	Inkl. Funktionen
InnoDB	Optimierungen
Performance-Schema	Erweitert und optimiert
SYS-Schema	Vereinfacht Zugriff auf Performance-Schema
Security	Erweiterungen
Replikation	Verbesserungen

Version 6.0 (abgebrochen, Teile davon in 5.4/5.5/5.6 integriert)	
Foreign Keys	Alle Engines einheitlich (sonst nur InnoDB)
Query Optimizer	Subqueries und Joins beschleunigen
FALCON-Engine	Transaktionsfähig
Maria-Engine	Transaktionsfähig
INFORMATION_SCHEMA	Erweitert (Parameters, Routines)
Online Backup+Restore	
Unicode Support	Erweitert (utf16, uf32, utf8)
LOCK TABLES Syntax	Erweitert (IN SHARE/EXCLUSIVE MODE)
Thread Pooling	
Version 8.0 (	

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/mysqld
   /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:\Program Files\MySQL\MySQL Server 5.6\bin      # MySQL-Programme
                                      ...\lib      # DLL-Dateien
                                      ...\share    # Fehler- + Hilfetexte
                                      ...\docs     # Dokumentation
                                      ...\my.ini   # Server-Konfig.datei
                                      ...\data     # Datenbanken + Tabellen


2b) MySQL-Datenbank-Server 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-Datenbank-Server 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-Datenbank-Server 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-Datenbank-Server 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
      mysqld-nt.exe OPTIONEN
  * Dienst (Stdname: "MySQL")
      "C:\Programme\MySQL\MySQL Server 5.6\bin\mysqladmin" -u root shutdown
      "C:\Programme\MySQL\MySQL Server 5.6\bin\mysqld" --install MySQL56
      "C:\Programme\MySQL\MySQL Server 5.6\bin\mysqld" --install MySQL56 --defaults-
file="C:\ProgramData\MySQL\MySQL Server 5.6\my.ini"
      "C:\Programme\MySQL\MySQL Server 5.6\bin\mysqld" --install-manual MySQL56
      "C:\Programme\MySQL\MySQL Server 5.6\bin\mysqld" --remove MySQL56
      net start MySQL56
      net stop MySQL56
  * 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 in verschiedenen 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 Benutzer "root" + Anmeldung vom lokalen Rechner aus festlegen:

  mysqladmin -uroot password geheim

Unsichere Anmeldungen in MySQL entfernen ("anonymer Benutzer" (leerer Name),
Benutzer mit Anmeldung von belieb. Host "%" und Benutzer mit leerem Passwort):

  mysql -uroot "
    USE mysql;
    SELECT user, host, password FROM user
      WHERE user = "" OR host = "%" OR password = "";
    SET PASSWORD FOR "root"@"localhost" = PASSWORD("geheim");
    SET PASSWORD FOR "root"@"127.0.0.1" = PASSWORD("geheim");
    SET PASSWORD FOR "root"@"::1"       = PASSWORD("geheim");
    DELETE FROM user
      WHERE user = "" OR password = "" OR host = "%";
    FLUSH PRIVILEGES;
    SELECT user, host, password FROM user;
  "


3b) Konfiguration: Variablen und Optionen
-----------------------------------------

Einige Variablen/Optionen sind nicht dynamisch zur Laufzeit des Servers
änderbar (z.B. "basedir", "datadir", ...), zu ihrer Änderung bedarf es eines
Neustarts des Servers.

Einige Variablen haben sowohl einen globalen Server-Wert als auch einen
sitzungsbezogenen Session-Wert. Beim Start einer Sitzung wird der globale
Server-Wert in den sitzungsbezogenen Wert kopiert. Spätere Änderungen am
sitzungsbezogenen Wert haben nur Auswirkung auf diese Sitzung, nicht aber auf
den globalen Server-Wert oder die Werte anderer Sitzungen. Spätere Änderungen
am globalen Server-Wert haben nur Auswirkungen auf DANACH neu gestartete
Sitzungen, bereits laufende Sitzungen behalten ihren sitzungsbezogenen Wert.
```

Variablen anzeigen (einige):

```
  SHOW VARIABLES;           # Globale oder sessionbezogenen Server-Variablen
  SHOW GLOBAL VARIABLES;    # Globale Server-Variablen
  SHOW SESSION VARIABLES;   # Sessionbezogenen Server-Variablen
  SHOW LOCAL VARIABLES;     # Sessionbezogenen Server-Variablen
```

Beispiele:

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

Konfiguration:

```
  * Generelle Arten (aufsteigender Vorrang!)
      A) Eingebaut in Server/Client
      B) Umgebungsvariablen
      C) Konfigurations-Dateien (mehrere, siehe unten)
      D) 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!)
        C1) Global (alle Server)             # /etc/mysql/my.cnf  /usr/etc/my.cnf
        C2) Serverspezifisch                 # /var/lib/mysql/my.cnf
        C3) Zusätzliche Konfig.datei         # --defaults-extra-file=FILE
        C4) Benutzerspezifisch               # ~/.my.cnf
    + Windows (aufsteigender Vorrang!)
        C1) Windows-Systemverzeichnis        # C:\Windows\system32\my.cnf
        C2) Windows-Wurzelverzeichnis        # C:\my.cnf
        C3) MySQL-Installationsverzeichnis   # INSTALLDIR\my.ini
        C4) Zusätzliche Konfig.datei         # --defaults-extra-file=FILE
  * Abschalten der Konfigurations-Dateien
      --defaults-file=FILE
      --no-default
  * Beispiel für eine Konfigurations-Datei:
    (weitere Beispiele in Verz. "support-files" (Linux))
```

```
    +-------------------------------------------------+
    | [client]                                        | Optionen für
    | password      = geheim                          | alle Clients
    | port          = 3306                            |
    | socket        = /var/lib/mysql/mysql.sock       |
    | sigint-ignore = 1                               |
    |                                                 |
    | [mysqld]                                        | Optionen für
    | bind-addr          = 127.0.0.1                  | mysqld-Server
    | port               = 3306                       |
    | socket             = /var/lib/mysql/mysql.sock  |
    | basedir            = /usr                       |
    | datadir            = /var/lib/mysql             |
    | max_allowed_packet = 1M                         |
    | max_connectios     = 150                        |
    | skip-locking       = 1                          |
    +-------------------------------------------------+
```

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 Zugriffs-Berechtigung anzeigen
--skip-show-database	Keine Anzeige vorhandener Datenbanken
--skip-grant-tables	Rechteverwaltung ignorieren
--socket=FILE	Socket-Datei
--user=USER/-u USER	Benutzername für Prozess
--version/-V	Versionsnummer

3c) Konfigurations- und Datenbankdateien

Für MySQL relevante Konfigurations-Dateien (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	1a) Zentrale Konfiguration (für alle Datenbanken)
/etc/mysql/my.cnf	1b) Zentrale Konfiguration (für alle Datenbanken)
/etc/mysql/conf.d/*.cnf	1c) Per "!includedir /etc/mysql/conf.d" (alle DB)
/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 Benutzer-Datenbanken bei "MyISAM" (pro Datenbank ein Verz., pro Tabelle in einer Datenbank im Verz. zu dieser Datenbank mehrere Dateien angelegt). Den Startpunkt <DataDir> legt die Option "--datadir" fest:

Linux: "/var/lib/mysql/data/"
Windows: "C:\ProgramData\MySQL\MySQL Server 5.6\data\"

Verz.pfad/Datei.Extension	Inhalt
<DataDir>/mysql/	Interne Verwaltungsdatenbank "mysql"
<DataDir>/performance_schema/	Interne DB zur Performance Messung
<DataDir>/test/	Testdatenbank "test" (leer)
<DataDir>/sakila/	Bsp.datenbank "sakila" (Flugbuchungen)
<DataDir>/world/	Bsp.datenbank "world" (Länder, Städte)
<DataDir>/menagerie/	Bsp.datenbank "menagerie" (Zoo)
<DataDir>/<Db>/	Benutzer-Datenbank <Db>
<DataDir>/<Db>/db.opt	Einstellungen für Datenbank <Db>
<DataDir>/<Db>/<Tbl>.frm	MyISAM Tabellen-Definition
.MYD	MyISAM Daten
.MYI	MyISAM Indices
<DataDir>/<Db>/<Tbl>.frm	CSV Tabellen-Definition
.CSV	CSV Daten (im CSV-Format)
.CSM	CSV Metadaten (im CSV-Format)
<DataDir>/<Db>/<Tbl>.frm	ISAM Tabellen-Definition
.ISD	ISAM Daten (veraltet)
.ISM	ISAM Metadaten (veraltet)
<DataDir>/<Db>/<Tbl>.frm	InnoDB Tabellen-Definition
.ibd	InnoDB Daten + Indices (EINE Tab+Ind)
<DataDir>/<Db>/<Tbl>.frm	Archive Tabellen-Definition
.ARZ	Archive Daten
.ARM	Archive Metadaten
<DataDir>/<Db>/<Tbl>.frm	Merge Tabellen-Definition
.MRG	Merge-Index-Dateien (Liste)

<DataDir>/<Db>/<Tbl>.TRG .TRN	Trigger Definition zu Tabelle Triggername
<DataDir>/ibdata1 ibdata2 ibdataN	InnoDB 1. Tablespace (VIELE Tab+Ind) InnoDB 2. Tablespace (VIELE Tab+Ind) InnoDB N. Tablespace (VIELE Tab+Ind)
<DataDir>/ib_logfile0 ib_logfile1	InnoDB Transaktionslog (alternierend) (alternierend)
<DataDir>/<Host>-bin.000001 <Host>-bin.000002 <Host>-bin.NNNNNN	Binärer Transaktionslog 1. Datei 2. Datei NNNNNN. Datei
<DataDir>/<Host>.pid .err .log -slow.log	Prozessnummer des MySQL-Servers Logging Fehlermeldungen Logging allgemeine Meldungen Logging langsamer SQL-Anweisungen

Linux --- Ablageort der Programme und sonstiger Dateien:
(datadir = "/var/lib/mysql/data/")

Verzeichnis	Inhalt
/etc/*	MySQL-Konfigurations-Dateien (Variante A)
/etc/mysql/*	MySQL-Konfigurations-Dateien (Variante B)
/usr/local/mysql/...	MySQL-Dateien bei Quellcode-Installation
/var/lib/mysql/mysql.sock	Socket für Datenbank-Verbindung (Var A)
/var/run/mysqld/mysqld.sock	Socket für Datenbank-Verbindung (Var B)
/usr/bin/*	MySQL-Client-Programme (Administration)
/usr/bin/mysql	MySQL-Client "mysql"
/usr/sbin/*	MySQL-Server-Programme
/usr/sbin/mysqld	MySQL-Datenbank-Server-Dämon "mysqld"
/usr/lib/mysql/*	Programm-Bibliotheken
/usr/lib/mysql/plugin/*	Engine-Plugins
/usr/share/sql-bench/*	Benchmarks (run-all-tests)
/usr/share/mysql/*	Konfigurations-Dateien, Skripte, Sprachen
/usr/include/mysql/*	C-Include-Dateien (für C-Programmierung)
/usr/share/doc/packages/mysql/	Dokumentation

Windows --- Ablageort der Programme und sonstiger Dateien:
(datadir = "C:\\ProgramData\\MySQL\\MySQL Server 5.6\\data")

Verzeichnis (Datei)	Inhalt
C:\Program Files\MySQL\MySQL Server 5.6\bin\...	MySQL-Programme
...\mysqld	MySQL-Server
...\mysql	MySQL-Client
...\lib	DLL-Dateien
...\share	Fehler- + Hilfetexte
...\my-default.ini	Default-Konfig.-Datei
C:\Program Files\MySQL\Connector C++ 1.1.3	C++ Connector
\Connector ODBC 5.3.4	ODBC Connector
\MySQL Connector C 6.1.5	C Connector
C:\Program Files (x86)\MySQL\Connector J 5.1.31	Java Connector
...\Connector NET 6.8.3	.NET Connector
...\MySQL Documentation 5.6.20	MySQL Doku
...\MySQL For Excel 1.2.1	Excel Connector
...\MySQL Installer	MySQL Installer
...\MySQL Notifier 1.1.5	MySQL Notifier
...\MySQL Utilities	MySQL Utilities
...\MySQL Workbench CE 6.1.7	MySQL Workbench
...\Samples and Examples 5.6.20	MySQL Beispiele
C:\ProgramData\MySQL\MySQL Server 5.6\my.ini	Server-Konfig.datei
...\data	Datenbanken + Tabellen

C:\Users\Administrator\Documents\dumps	Workbench Export Verz.

3d) Interne Verwaltungsdatenbanken

Der MySQL-Datenbank-Server besitzt 3 interne Verwaltungsdatenbanken:

Datenbank	Bedeutung
mysql	Interne Verwaltungsdaten ("Dictionary", immer MyISAM)
INFORMATION_SCHEMA	Verwaltungsdaten (generiert aus "mysql", read-only!)
PERFORMANCE_SCHEMA	Performancezähler (Messdaten, read-only)

Die interne Datenbank "mysql" enthält die Verwaltungsdaten bzw. das "Dictionary" in Form von Verzeichnissen (Datenbanknamen) und Dateien (Tabellennamen + Tabellenstruktur in "*.frm"-Dateien)

Tabelle	Bedeutung	Right	Dump
/<Db>	Datenbanken (Verz.name unter "--datadir")		D
<Tab>.frm	Tabellen einer Datenbank unter <Db>-Verz.		D
user	Benutzer (Passwort, Basis-Rechte) (H U)	R	D
db	Datenbank-spez. Berechtigungen (H D U)	R	D
tables_priv	Tabellen-spez. Berechtigungen (H D U T)	R	D
columns_priv	Spalten-spez. Berechtigungen (H D U T C)	R	D
procs_priv	Proz./Funk.-spez. Berechtigungen (H D U R)	R	D
proxies_priv	Proxy-spez. Berechtigungen (H U)	R	D
event	Event-Verwaltung (D SM)		D
func	Prozedur/Funktionen-Liste (D SM)		D
proc	Prozedur/Funktionen-Definitionen (D SM)		D
general_log	Allgemeine Logs (als CSV-Datei abgelegt)		
slow_log	Langsame Logs (als CSV-Datei abgelegt)		
plugin	Plugin-Verwaltung		
servers	Server-Verwaltung (S H D U P)	?	D
help_category	Hilfetext zu ...		
help_keyword	Hilfetext zu ...		
help_relation	Hilfetext zu ...		
help_topic	Hilfetext zu ...		
time_zone	Zeitzone		
time_zone_leap_second	Zeitzone		
time_zone_name	Zeitzone		
time_zone_transition	Zeitzone		
time_zone_transition_type	Zeitzone		
innodb_index_stats	InnoDB		
innodb_table_stats	InnoDB		
ndb_binlog_index	NDB-Verwaltung		
slave_master_info	Replikation		
slave_relay_log_info	Replikation		
slave_worker_info	Replikation		

HINWEIS: In Spalte "Right" mit "R" markierte Tabellen enthalten Berechtigungen (GRANTs). In Spalte "Dump" mit "D" 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. Die restlichen Abkürzungen in obiger Tabelle: "H" = Host, "D" = Database, "U" = User, "T" = Table, "C" = Column, "R" = Stored Routine, "S" = Server, "P" = Plugin)

Beispiele:

```
  USE mysql;                          # In 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) dient zur einfacheren Abfrage von Informationen über die MySQL-Datenbank per SELECT-Statements mit allen SQL-Möglichkeiten (allgemeiner als die SHOW-Befehle auf Verwaltungsdatenbank "mysql"):

Tabelle	Bedeutung
CHARACTER_SETS	Zeichensätze
COLLATIONS	Zeichen-Abbildungen für Sortierung
COLLATION_CHARACTER_SET_APPLICABILITY	Erlaubte Kombinationen
COLUMNS	Spalten-Definitionen
COLUMN_PRIVILEGES	Spalten-Berechtigungen
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-Berechtigungen
SESSION_STATUS	Sitzungs-Status
SESSION_VARIABLES	Sitzungs-Variablen
STATISTICS	Statistik-Informationen
TABLES	Tabellen-Definitionen
TABLE_CONSTRAINTS	Tabellen-Spalten-Beschränkungen
TABLE_PRIVILEGES	Tabellen-Berechtigungen
TRIGGERS	Trigger-Definitionen
USER_PRIVILEGES	Benutzer-Berechtigungen
VIEWS	View-Definitionen

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

Zunächst 28 Tabellen in MY!5.0, ab MY!5.1 49 Tabellen, usw.

Beispiele:

```
  SHOW TABLES FROM INFORMATION_SCHEMA;   # Alle Verwaltungstabellen anzeigen
  USE INFORMATION_SCHEMA;                # In Verwaltungsdatenbank wechseln
  SHOW TABLES;                           # Alle Verwaltungstabellen anzeigen
  SHOW COLUMNS FROM TABLES;              # Verwaltungstabellen-Inhalt anzeigen
```

Interne Datenbank "PERFORMANCE_SCHEMA" dient zur Messung von Performance-Daten, für sie existiert eine gleichnamige spezielle Engine "PERFORMANCE_SCHEMA".

    TODO

3e) SQL-Modus (sql_mode)

Ursprünglich war die MySQL-Syntax nicht ANSI-kompatibel und die Reaktion von MySQL beim Einfügen fehlerhafter Daten großzügig (d.h. fehlertolerant). Andere Datenbanken verhalten sich hier mehr ANSI-konform und deutlich restriktiver. Durch den "SQL-Modus" kann das Verhalten von MySQL an andere Datenbanken angeglichen werden, falls dies notwendig sein sollte.

Der SQL-Modus legt also fest, welche SQL-Syntax im Detail akzeptiert wird und welche Datenüberprüfungen bei ungültigen/fehlenden Daten stattfinden. Damit kann ein MySQL-Server einfacher in unterschiedlichen Umgebungen und zusammen mit anderen SQL-Datenbanken eingesetzt werden. Definiert wird der SQL-Modus durch eine Optionen-Liste getrennt durch "," (STD: leer = keine Option =

MySQL-Verhalten).

HINWEIS: Option "innodb_strict_modes" aktiviert weitere Prüfungen beim Einsatz von InnoDB-Tabellen.

Der MySQL-Server kann global oder Client-abhängig in verschiedenen SQL-Modi operieren. Der SQL-Modus wird folgendermaßen festgelegt:

```
--sql_mode="..."                # Beim Aufruf von "mysqld (Server-global)
sql_mode = "..."                # In Konfig.-Datei "my.cnf" (Server-global)
SET GLOBAL sql_mode = "...";    # Server-global (SUPER-Recht nötig!)
SET SESSION sql_mode = "...";   # Sitzungsbezogen (Client-lokal)
```

Den aktuellen globalen oder sitzungsbezogenen Wert erhält man mit:

```
SELECT @@GLOBAL.sql_mode;     # Globaler Wert
SELECT @@SESSION.sql_mode;    # Sitzungsbezogener Wert
```

Wird "GLOBAL sql_mode" verändert, dann gilt die neue Einstellung für alle MySQL-Clients, die sich danach NEU mit dem MySQL-Server verbinden.

Der SQL-Modus ist aus folgenden Einstellungen kombinierbar. Ab MY!5.6 sollen diese Werte nicht mehr einzeln verwendet werden, sondern über die vordefinierten Zusammenfassungen (siehe weiter unten):

Option	Bedeutung
ALLOW_INVALID_DATES	Alle Komb. von Mon=1..12 + Tag=1..31 erlaubt
ANSI_QUOTES	"..." + `...` zur Schlüsselwort-Quotierung
ERROR_FOR_DIVISION_BY_ZERO	Fehler statt Warnung bei Division durch 0
HIGH_NOT_PRECENDENCE	"NOT" hat gleich hohen Vorrang wie "!"
IGNORE_SPACE	Leerz. in "FUNCTION (" OK (dann res. Wort)
NO_AUTO_CREATE_USER	GRANT legt nicht autom. neuen Benutzer an
NO_AUTO_VALUE_ON_ZERO	NULL erz. neuen AUTO_INCREMENT-Wert, 0 nicht
NO_BACKSLASH_ESCAPES	"\" ist normales Zeichen in String '...'
NO_DIR_IN_CREATE	Tab. erstellen ignoriert INDEX/DATA DIRECTORY
NO_ENGINE_SUBSTITUTION	Tab.-Engine bei CREATE nicht vorh. --> Fehler
NO_FIELD_OPTIONS	MySQL-Spalten-Opt. bei SHOW CREATE TABLE ign.
NO_KEY_OPTIONS	MySQL-Index-Opt. bei SHOW CREATE TABLE ign.
NO_TABLE_OPTIONS	MySQL-Tabellen-Opt. bei SHOW CREATE TABLE ig.
NO_UNSIGNED_SUBTRACTION	Subtraktion von UNSIGNED-Wert ergibt SIGNED
NO_ZERO_DATE	00-00-00 verboten (YYYY-01-00/YYYY-00-01 OK)
NO_ZERO_IN_DATE	Monat=00 und Tag=00 nicht erlaubt
ONLY_FULL_GROUP_BY	Alle selekt. Sp. müssen in GROUP BY vorkommen
PAD_CHAR_TO_FULL_LENGTH	CHAR-Spalten mit Leerzeichen auffüllen
PIPES_AS_CONCAT	Operation "||" verkettet Strings (sonst OR)
REAL_AS_FLOAT	REAL entspricht FLOAT (statt DOUBLE)
STRICT_ALL_TABLES	Ungültige/fehlende Werte bei allen Tab. zur.w.
STRICT_TRANS_TABLE	Ungültige/fehlende Werte bei Transakt. zurückw

Folgende Zusammenfassungen von SQL-Modi sind vordefiniert, die den MySQL-Server auf das spezielle SQL-Verhalten anderer Datenbanken umschalten:

Kombination	Liste der Einzeloptionen
ANSI	REAL_AS_FLOAT, PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE
DB2	PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS
MAXDB	PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER
MSSQL	PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER
MYSQL323	NO_FIELD_OPTIONS, HIGH_NOT_PRECEDENCE
MYSQL40	NO_FIELD_OPTIONS, HIGH_NOT_PRECEDENCE
ORACLE	PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER

POSTGRESQL	PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS
TRADITIONAL	STRICT_TRANS_TABLE, STRICT_ALL_TABLES, NO_ZERO_IN_DATE, NO_ZERO_DATE, ERROR_FOR_DIVISION_BY_ZERO, NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION

* TRADITIONAL = Fehler statt Warnung ausgeben beim Einfügen ungültiger Werte
  (DATE, DATETIME, ENUM, SET)

* Strict Modus = Mind. STRICT_TRANS_TABLES oder STRICT_ALL_TABLES aktiviert
  + Ungültiges Datum nicht akzeptiert
  + Datum mit Tag/Monat "0" nicht akzeptiert,
  + ENUM mit ungültigem Wert nicht akzeptiert (bzw. NULL)
  + Zu langer String nicht akzeptiert

4) MySQL Kommandozeilen-Programme

Es gibt eine Reihe von mitgelieferten Verwaltungs- und Dienstprogrammen:
  * Zur Datenbank-Erstellung und -Bearbeitung
  * Zur Administration + Überwachung
  * Zur Datenbank- bzw. Tabellen-Reparatur

Damit ist Fernzugriff auf MySQL betriebssystem-unabhängig möglich:
  * Netzwerkverbindung muss prinzipiell vorhanden sein
  * Firewall muss Verbindung von außen auf Port 3306 erlauben
  * Dienstprogramm muss auf lokalem Rechner installiert sein
  * Anmeldedaten müssen bekannt sein: -h<Host> -u<User> -p<Pass> -P<Port>

Liste der mitgelieferten Dienstprogramme (meist MySQL-Clients die eine Verbindung zum MySQL-Server aufbauen, manche arbeiten auch direkt auf den Datenbankdateien):

  S = Serverprogramm
  CS = Clientprogramm, das auf Server zugreift
  C = Clientprogramm unabhängig vom Server (Dateizugriff)
  I = Installationsprogramm
  H = Hilfsprogramm
  T = Testprogramm
  P = Programmierung

	Kommando	Bedeutung
CS	mysql	Interaktiver Befehlszeilen-Client (auch Batch, alle Kmdos)
CS	mysqlaccess	Berechtigungen für USER+HOST+DB-Kombinationen überprüfen
CS	mysqladmin	Datenbank-Administration (nur Admin-Befehle)
C	mysqldumpslow	Logdatei mit "langsamen SQL-Anweisungen" zus.fass.+anzeig.
CS	mysqlreport	Bericht über MySQL Status erstellen (SHOW STATUS)
CS	mysqlshow	Struktur von DB-Objekten ansehen (DB, Tabelle, Spalte)
CS	mysqlcheck	Tabellen analysieren/prüfen/optimieren/reparieren (online)
CS	mysqlanalyze	Symb. Link auf "mysqlcheck" (-a = analysieren)
CS	mysqloptimize	Symb. Link auf "mysqlcheck" (-r = optimieren)
CS	mysqlrepair	Symb. Link auf "mysqlcheck" (-o = reparieren)
C	innochecksum	InnoDB-Tabellen prüfen (offline)
C	myisamchk	MyISAM-Tabellen prüfen/optimieren/reparieren (offline)
C	myisampack	MyISAM-Tabellen komprimieren für Read-Only-Zugr. (offline)
C	myisam_ftdump	Info über MyISAM Fulltext-Indices ausgeben
C	myisamlog	MyISAM-Logdateien verarbeiten
C	isamlog	MyISAM-Logdateien verarbeiten
C	mysqlbinlog	Anweisungen aus Binärlog lesen (Rebuild nach Absturz)
C	mysql_explain_log	SQL-Querylog analysieren (per EXPLAIN)
C	mysql_find_rows	SQL-Anw. aus Dateien extrahieren (Updatelog, RegEx)
CS	mysqldump	Backup von Datenbanken/Tabellen in SQL-Befehlsform
CS	mysqlimport	Textdatei in div. Formaten importieren (Name = <Tab>name)
CS	mysqlhotcopy	Backup MyISAM/ARCHIVE-Tab. binär (onl., Filezgr. Serverlok.)

IS	mysql_install_db	Datenverz. + System-Tab. (GRANT) des Servers erstellen
S	mysqlmanager	MySQL Instance Manager ("mysqld_multi"-Ersatz, -MY!5.5)
S	mysqld	Server (zum Start den Wrapper "mysqld_safe" verwenden)
S	mysqld_safe	Server-Startskript (früher "safe_mysqld")
S	mysqld.server	Server-Startskript (SysV-Systeme)
S	mysqld_multi	Server-Startskript (Manager für mehrere Server)
IS	comp_err	MySQL Fehlernachrichten-Datei übersetzen (Installation)

C	perror	Bedeutung von Fehlercodes ausgeben ("print error")
H	replace	Textersetzung in Dateien durchführen
H	resolveip	Abbildung Hostname <-> IP-Adresse auflösen
I	mysqlbug	Interaktiv Fehlerbeschreib. erstellen + an MySQL mailen
I	msql2mysql	mSQL-C-Funktionsaufrufe --> mySQL-C-Funkt. konvertieren
	my_print_defaults	Konfig.opt. einer Konf.Gruppe ([client], ...) ausgeben

S	mysql_waitpid	Server-Prozess beenden (und darauf warten)
S	mysql_zap	Server-Prozesse beenden (per RegEx-Muster)

CS	mysql_convert_table_format	Tabellen in andere Storage Engine konvertieren
CS	mysql_fix_extensions	MyISAM-Dateiext. konvert. frm/MYI/MYD/ISD/ISM
CS	mysql_secure_installation	Sicherheit einer Installation verbessern
CS	mysql_setpermission	Berechtigungen in GRANT-Tab. interaktiv setzen
CS	mysql_tableinfo	Aus Medadat. erz. (MY!5.0 INFORMATION_SCHEMA)
CS	mysql_tzinfo_to_sql	Zeitzonen-Tabelle in Server laden
CS	mysql_upgrade	(Rechte-)Systemtab. upgraden (neue Version)
CS	mysql_upgrade_shell	" " " " " "
CS	fix_priv_tables	" " " " (generiert)

T	mysqlslap	Server-Lasttest per Simulation vieler Clients
T	mysql-test-run.pl	Server-Test per "mysqltest" steuern (Skript)
T	mysqltest	Server-Test und Vergleich mit anderen Ergeb.
T	mysqltest_embedded	Server-Test und Vergleich mit anderen Ergeb.
T	mysql_client_test	Test der MySQL-Client-API (Skript)
T	mysql_client_test_embedded	Test der MySQL-Client-API (Skript)

P	mysql_config	Compiler-Optionen zum Übersetzen extrahieren
P	make_binary_distribution	Binärpaket aus MySQL-Install. erzeugen (Unix)
P	make_win_bin_dist	Binärpaket aus MySQL-Install. erzeugen (Windows)
P	resolve_stack_dump	Numerischen Stack Trace Dump in Symbole auflösen

mysql-binlog-dump	?
mysql-myisam-dump	?

Zur Überwachung der MySQL-Prozesse und -Engines sind folgende Kommandozeilentools verfügbar:

Tool	Bedeutung
innotop	MySQL- und InnoDB Transaktions/Threadliste (analog "top")
mtop	MySQL-Threadliste (analog "top")
mytop	MySQL-Threadliste (analog "top")

4a) mysqladmin

Das Client-Programm "mysqladmin" ist eine kommandozeilen-orientierte Administrations-Schnittstelle von MySQL, mit der die wichtigsten Verwaltungsaufgaben erledigt werden können. Alle Funktionen können auch mit "mysql" + SQL-Befehlen durchgeführt werden, "mysqladmin" bietet aber oft einen einfacheren + 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>
status	Zeigt abgekürzten Server-Status	STATUS (Client)

extended-status	Zeigt alle Server-Konfig.-Var.	SHOW STATUS
variables	Listet alle verfügbaren Var.	SHOW VARIABLES
version	Gibt MySQL-Version aus	STATUS (Client)
processlist	Zeigt alle aktiven MySQL-Threads	SHOW PROCESSLIST
kill <Pid>[,...]	Beendet MySQL-Thread <Pid>,...	KILL <Pid>, ...
ping	Prüft ob "mysqld" läuft	%
shutdown	Fährt "mysqld" herunter	%
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	%

4b) mysqlshow

Das Client-Programm "mysqlshow" erlaubt die Anzeige von Datenbank-Elementen von der Kommandozeile aus. Alle Funktionen können auch mit "mysql" + SQL-Befehlen durchgeführt werden, "mysqlshow" bietet aber oft einen einfacheren + schnelleren Weg dafür an.

```
mysqlshow [OPTS] [<Db> [<Tbl> [<Col>]]]
```

Im jeweils letzten Argument sind die Platzhalter * ? % _ erlaubt (quotieren um sie vor der Shell zu schützen):

* Ohne Datenbank <Db> werden alle Datenbanken angezeigt
* Ohne Tabelle <Tbl> werden alle Tabellen angezeigt
* Ohne Spalte <Col> werden alle Spalten + ihr Datentyp angezeig

Einige Optionen sind möglich:

Option	Bedeutung
-i/--status	Zusatzinfo zu Tabellen anzeigen
-k/--keys	Indices zu Tabellen anzeigen
-t/--show-table-type	Spalte mit Tabellentyp anzeigen
--count	Anzahl Zeilen pro Tabelle anzeigen (langsam!)

5) Benutzer und Berechtigungen (Privileges)

Zweck:

* Authentifizieren von Benutzern
  + Abhängig von Rechner/IP-Addresse von dem/der aus Benutzer sich verbindet
* Zuweisen von Zugriffs-Rechten
* Zuweisen von Administrations-Rechten
* Standardbenutzer
  + Anonymer Benutzer: Name+Passwort leer, nur Zugriffsrecht USAGE
  + Administrator: Passwort leer, alle Berechtigungen
* Rollen (Gruppen) = Berechtigungen für Menge von Benutzern NICHT mögl. (MY!)

HINWEIS: MySQL-Benutzer haben NICHTS mit Betriebssystem-Benutzern zu tun (auch wenn der Administrator von Linux und MySQL identisch "root" heißt). D.h. die Benutzer- und Berechtigungs-Verwaltung von MySQL ist völlig unabhängig von der des Betriebssystems.

Ein MySQL-Server läuft in der Regel unter Benutzer "mysql" (und Gruppe "mysql") alle MySQL-Verz. + Dateien werden diesem Benutzer zugeordnet.

Der Benutzer "mysql" kann auf dem Linux-System nicht administrativ tätig werden meist kann er sich sogar nicht darauf anmelden (kein Passwort vergeben oder Login-Shell deaktiviert). Er hat somit keinen Zugriff auf die Daten anderer

Benutzer und andere Benutzer (außer "root") haben keine Zugriffsrechte auf die dem MySQL-Server zugordneten Dateien (Datensicherheit!).

Linux	MySQL	Bemerkung
root	--	Linux-Administrator
--	root	MySQL-Administrator
mysql	--	Linux-Besitzer von mysqld-Prozess/Daten
tsbirn	tsbirn	Benutzer+Passwort verschieden
...	...	

Eigenschaften:
* GROSS/kleinschreibung der Benutzernamen relevant!
* Passworte werden verschlüsselt abgelegt!
* Kombination Benutzer + Host (d.h. Quelle der Verbindung spielt eine Rolle)
  --> Es wird unterschieden, WOHER ein Benutzer sich verbindet (Host)
* Standardmäßig kann ein Benutzer nur SELBST ERSTELLTE Datenbanken <Db> und Datenbank-Objekte <Obj> bearbeiten (er ist "Besitzer" dieser Objekte)
* 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
* Berechtigungen mit REVOKE entziehen
  + Transitiv (d.h. an andere Benutzer weitergegebene Rechte auch entzogen)
  + Recht mehrfach an einen Benutzer weitergegeben
    --> Alle müssen es ihm entziehen, damit er es nicht mehr hat
* Auch Rechte für NICHT existierende Objekte sind anlegbar
* Rechte zu Datenbank/Tabelle werden beim Löschen dieser NICHT entfernt
* Regeln zu Routine werden beim Löschen dieser entfernt
* Anonymer Benutzer = leerer Name (""@"localhost")
  + Jeder Benutzer, der MySQL nicht bekannt ist, wird als dieser angemeldet
  + Hat nur Recht "USAGE" (Anmeldung)
  + Anmeldename USER() dann ungleich angemeldeter User CURRENT_USER()
* Berechtigungen ignorieren (nur beim Server-Start):
  --skip-grant-tables

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 USER/DELETE notwendig
* REVOKE von Rechten muss nicht vorhergehendem GRANT entsprechen
* Einschalten der Rechte für EINEN Benutzer aktiviert Rechteprüfung für ALLE (d.h. Server wird langsamer)
* Einschalten der Begrenzungen MAX_... für Anfragen, Updates, Verbindungen für EINEN Benutzer aktiviert Prüfung für ALLE (d.h. Server wird langsamer)
* Rechte-Änderungen während Verbindung wirkt nicht sofort (FLUSH PRIVILEGES)

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

```
GRANT <Priv> [(<Col>, ...)]                          # Rechte + best. Spalten
      [, <Priv> [(<Col>, ...)]...]                   # Weitere Rechte + Spalten
  ON [TABLE | PROCEDURE | FUNCTION] <Obj>            # Datenbank-Objekt
    TO <User>@<Host>                                 # Benutzer (Wer und woher?)
      [IDENTIFIED BY [PASSWORD] <Pass>]              # Passwort
    [, TO ... [IDENTIFIED BY ...]...]                # Weitere Benutzer+Passwort
        [REQUIRE <SSLOpt> [AND <SSLOpt> ...]]        # Verschüsselung
        [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
```

ACHTUNG: Zwei Benutzer mit dem Recht "GRANT OPTION" können sich gegenseitig ihre Rechte weitergeben und sie so kombinieren!

Minimale Anweisungen um einem Benutzer alle Rechte an allen Tabellen zu geben (alle 5 Einträge sind notwendig!):

```
  GRANT ALL PRIVILEGES ON *.* TO tom@"localhost";  # Lokal per Socket
  GRANT ALL PRIVILEGES ON *.* TO tom@"127.0.0.1";  # Lokal per IPv4
  GRANT ALL PRIVILEGES ON *.* TO tom@"::1";        # Lokal per IPv6
  GRANT ALL PRIVILEGES ON *.* TO tom@"HOSTNAME";   # Echter Rechnername
  GRANT ALL PRIVILEGES ON *.* TO tom@"localhost.invalid";   # FQHN
```

HINWEIS: Nach erfolgreichem GRANT/REVOKE erscheint die Meldung "Query OK, 0 rows affected (0.00 sec)", nach schiefgegangenem GRANT/REVOKE erscheint eine Meldung "ERROR ...". Bitte nicht davon verwirren lassen!

Alle Rechte eines Benutzers anzeigen (ACHTUNG: Schreibweise "tom"@"localhost" notwendig, Gänsefüßchen ganz aussenrum sind falsch!):

```
  SHOW GRANTS;                        # Für angemeldeten Benutzer
  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 Berechtigungen 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 Passwort
```

Benutzer-Daten (Name, Passwort, allgemein) ändern:

```
  RENAME USER <Name> TO <NewName>, ...;                # Benutzer umbenennen
  SET PASSWORD [FOR <User>] = PASSWORD("geheim");      # Klartext-Passwort
  SET PASSWORD = PASSWORD("geheim");                   # Aktueller Benutzer
  SET PASSWORD [FOR <User>] = "*46...F"                # Verschlüss. Passw.
  GRANT USAGE ON ... TO ... IDENTIFIED BY "geheim";    # Klartext-Passw.
  UPDATE mysql.user SET password = "*46...F"           # Tabelle direkt
    WHERE user = "root";                               #   ändern (Vorsicht!)
  FLUSH PRIVILEGES;                                    # Nicht vergessen!
```

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

```
  REVOKE <Priv> [(<Col>, ...)]                  # Rechte + best. Spalten
    [ON [TABLE | PROCEDURE | FUNCTION] <Obj>]   # Datenbank-Objekt (ohne=alle)
      FROM <User>@<Host>, ...                   # Wer und woher?
```

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

```
  REVOKE ALL PRIVILEGES, GRANT OPTION FROM <User>@<Host>, ...;   # Ohne ON ...!
```

Benutzer löschen (nach MY!5.0 mit allen seinen Rechten; von ihm erzeugte Datenbank-Objekte 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 = "";     # Ben. mit leerem Passwort löschen!
  DELETE FROM user WHERE host = "%";        # ... mit Anmeldung von überall 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            # Nur INSERT, SELECT, UPDATE
    ON first.*
      TO "tom"@"localhost";               # Von Rechner "localhost" aus

  GRANT SELECT                            # Nur SELECT
    ON first.*
      TO "tom"@"arbeitsplatz";            # Von Rechner "arbeitsplatz" aus

  GRANT INSERT, SELECT, UPDATE, DELETE    # Nun auch REPLACE erlaubt
    ON first.*
```

```
      TO "tom"@"%";                         # Egal woher (auch nur "tom" möglich)

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

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

```
  REVOKE UPDATE
    ON first.*
      FROM "tom"@"localhost";               # FROM statt TO!
```

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

```
  * Administrator "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" (leerer Benutzername) 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.

Beispiel: Auf lokalem Rechner den Benutzer "tom" (TO "tom"@"localhost") mit Passwort "geheim" (IDENTIFIED BY "geheim") einrichten und ihm ALLE Rechte (ALL PRIVILEGES) 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 analog zum Administrator "root"):

```
  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" "127.0.0.1" (IPv4) und "::1" (IPv6) nötig sein, da solche Namen bei der Verbindungsaufnahme 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 folgenden Tabellen der Verwaltungsdatenbank "mysql" und regeln den Zugriff auf Server, Datenbanken, Tabellen, Spalten und Routinen:

Tabelle	Schlüssel	Bedeutung
host	(nicht genutzt seit MY!4.1, nicht vorhanden seit MY!5.6.7)	
user	Host User	Password, globale Rechte, SSL + max + ...
db	Host User Db	DB-spezifische Rechte
tables_priv	Host User Db Table	Table-spez. Rechte
columns_priv	Host User Db Table Column	Spalten-spez. Rechte
procs_priv	Host User Db Routine	Routinen-spez. Rechte
proxies_priv	Host User Proxy-Host Proxy-User	Mapping ext. --> int. User

Auflisten aller prinzipiell möglichen Rechte <Priv> mit:

```
SHOW PRIVILEGES;
```

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

Recht/Privilege	Bezug	Bedeutung
ALL [PRIVILEGES]	ALLE	Alle Rechte außer GRANT OPTION
USAGE	S	Anmelden/Verbindung (minimal notwendiges Recht)
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
SUPER	S	Verwalten (KILL, SET GLOBAL, CHANGE MASTER TO, PURGE BINARY LOGS)
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!)
LOCK TABLES	D	Tabellen sperren (SELECT-Recht notwendig!)
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	? (MY!5.5)
CREATE TEMPORARY TABLES	D	Temporäre Tabellen anlegen ("stirbt" am Sitzungsende)
CREATE VIEW	TV	View anlegen
SHOW VIEW	TV	View ansehen (SHOW CREATE VIEW)
TRIGGER	TG	Trigger erstellen/löschen (ab 5.1.6)
EVENT	D E	Events benutzen (ab 5.1.6)
CREATE ROUTINE	R	Prozeduren/Funktionen anlegen
ALTER ROUTINE	R	Prozeduren/Funktionen ändern/löschen
EXECUTE	R	Prozeduren/Funktionen ausführen
REPLICATION CLIENT	S	Statusdaten von Master/Slave-Servern auslesen
REPLICATION SLAVE	S	Binärlog-Events von Master-Server lesen
BACKUP	S	? (MY!5.5)
RESTORE	S	? (MY!5.5)
PROXY	S	? (MY!5.5)

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

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

Objekt	Bedeutung
.	ALLE Datenbanken + ALLE 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

Schreibweise für Benutzer in GRANT/REVOKE-Anweisung (Gänsefüßchen um <User> und <Host> bei Verwendung der SQL-Wildcards "%" und "_" notwendig). In Spalte "OK" steht "ja", wenn entsprechende Form sicher ist, "--" bezeichnet unsichere Form.

Benutzer	OK	Bedeutung	
"<User>"@"<Host>"	ja	Benutzer + Host-Name	X
"<User>"@"<IP>"	ja	Benutzer + Host-IP	X
"<User>"@"%.ostc.de"	ja	User von Host aus Domain "ostc.de"	X
"<User>"@"10.32.1.%"	ja	User von Host aus IP-Bereich "10.32.1.*"	X
"%"@"<Host>"	--	Beliebiger User von Rechner mit Host-Name	X
"%"@"<IP>"	--	Beliebiger User von Rechner mit Host-IP	X
"<User>"@"%"	--	User von beliebigem Host	X
"<User>"	--	(identisch)	X
""@<Host>	--	Anonymer Benutzer	X
@"localhost"	ja	Nur vom Server-Rechner (UNIX Domain-Socket)	X
@"127.0.0.1"	ja	Nur vom Server-Rechner (TCP-Socket, IPv4)	X
@"::1"	ja	Nur vom Server-Rechner (TCP-Socket, IPv6)	X
@"<Host>"	ja	Nur vom Server-Rechner (UNIX Domain-Socket)	X
% oder <Leer>	--	Beliebiger User von beliebigem Host	V

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.

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

```
  GRANT ALL PRIVILEGES                    # Alle Rechte (AUSSER Rechtevergabe)
    ON first.*                            # auf bestimmte DB
      TO "admin"@"localhost";

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

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

  GRANT DELETE                            # + Daten löschen (und REPLACE)
```

```
    ON first.*                          # auf bestimmter DB
      TO "kontrolleur"@"localhost",
         "entwickler"@"localhost";

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

  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";
```

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      # + Datenbank-Server verwalten
```

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" oder "::1" eingetragen).
- -h 127.0.0.1 verbindet sich per TCP/IP-Socket (nur bei lokaler Anmeldung)
- -h ::1 verbindet sich per TCP/IP-Socket (nur bei lokaler Anmeldung)
- -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

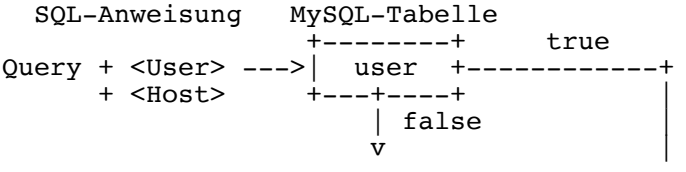
Verschlüsselungsoptionen REQUIRE <SSLOpt>:

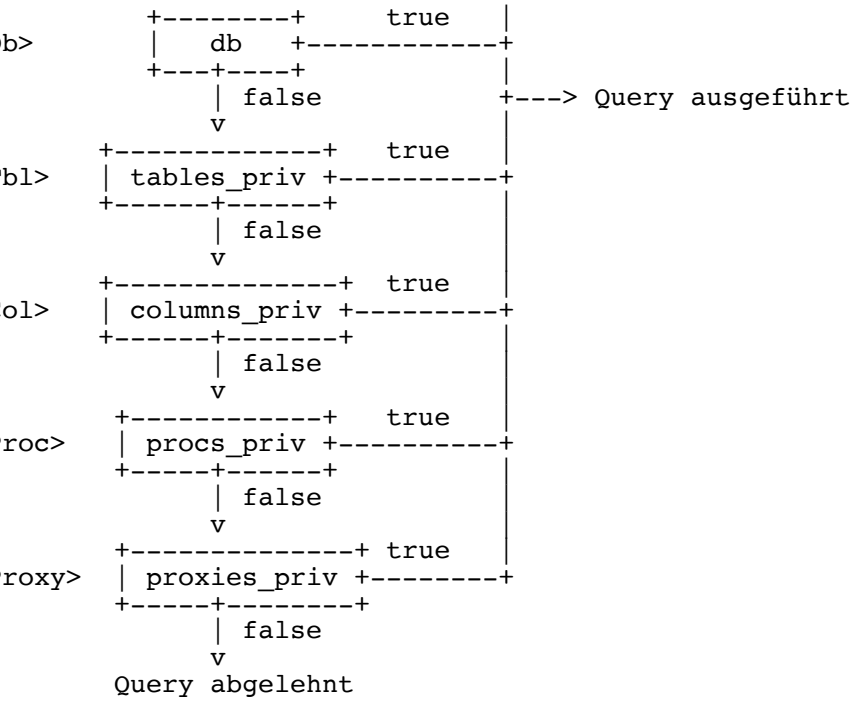
Option	Bedeutung
NONE	Unverschlüss. Verbindung erlaubt (STD), verschl. auch
SSL	Nur SSL-verschlüsselte Verbindungen erlaubt
X509	Gültiges Zertifikat notwendig, Issuer + Subject egal
CIPHER <Spec>	Verschlüsselung und Schlüssel müssen best. Bed. genügen (z.B. CIPHER "EDH-RSA-DES-CBC3-SHA")
ISSUER <String>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

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

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

Reihenfolge in der MySQL-Rechtetabellen überprüft werden, ob ein Benutzer für eine bestimmte Query Berechtigungen hat (absteigender Vorrang):
1) Benutzer-spezifische Rechte
2) Datenbank-spezifische Rechte
3) Tabellen-spezifische Rechte
4) Spalten-spezifische Rechte
5) Routinen-spezifische Rechte
6) Proxy-spezifische Rechte





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

```
USE mysql;
SHOW TABLES;       # --> user, db, tables_priv, columns_priv, procs_priv, ...
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 Zugriffs-Berechtigung 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 InnoDB (seit MY!5.5.1, vorher MyISAM, noch früher ISAM). Hier eine Liste der Engines mit ihren wichtigsten Eigenschaften (TA = Transaktionsfähig):

Engine	TA	Beschreibung
MyISAM	--	Hohe Performanz, binär portabel, Volltext-Index
InnoDB	JA	Zeilenbasierte Sperrung, Foreign Keys --> ORACLE!, MVCC, autom. Fehlerkorrektur, konsistentes Lesen ohne Sperren
MEMORY(HEAP)	--	Hash-basiert, RAM, temp. Tabellen, Datenverlust (HEAP)
MERGE	--	Komb. identischer MyISAM-Tabellen (MRG_MyISAM, UNION)
ARCHIVE	--	Große Menge, kein Index, komprimiert, Read-Only
FEDERATED	--	Remote-Datenbank (Umlenkung, ab MY!5.0.3)
CSV	--	Comma Separated Values-Format, kein Index
BLACKHOLE	--	Gespeicherte Daten verschwinden (/dev/null), für Tests
NDB(CLUSTER)	JA	Verteilt auf Rechner-Cluster, HA, High Performance, RAM

| EXAMPLE | -- | Programmier-Beispiel (speichert nichts, Schnittstellen) |

Fremdentwicklungen anderer Hersteller:

Engine	TA	Beschreibung
FALCON	JA	MVCC (Multi Version Concurrency Control, ab MY!6.0)
Aria	JA	MyISAM-Erweiterung, Crash Save (MY!6.0, früher "Maria")
BrightHouse	JA	Data Warehouse orientiert (keine Indices, Komprimierung)
InfiniDB	JA	Colbased, MVCC, TA, keine Indices, Datamining, Bus.Int.
NitroEDB	??	Very large Databases (VLDB), Echtzeit, Netzwerksicherung
PBXT	??	Foreign Keys, MVCC, Rowlocks (PrimeBase XT Storage)
solidDB	JA	Zeilenbasierte Sperrung, MVCC (IBM)
XtraDB	JA	InnoDB Fork, verbessert, "XtraBackup" für hot backup, mehr Laufzeitinfo, IO-Durchsatz bei mehr Threads besser
memcached	JA	Caching im Speicher
IBM DB2	JA	
FederatedX	JA	
FederatedODBC	JA	
Revision	JA	Automatische Versionierung (DDEngine)

Nicht mehr eingesetzte Engines:

Engine	TA	Beschreibung
ISAM	--	Vorgänger von MyISAM (veraltet!)
BerkeleyDB (BDB)	JA	Sleepycat, Primärschlüssel notwendig, keine AUTO_INC, seitenbasierte Sperrung (nur bis MY!5.0) --> ORACLE!

Grobe Unterschiede der Datenbank-Engines:

Eigenschaft	Engine
Transaktionen	InnoDB, BDB, NDB
Fulltext-Index (FULLTEXT INDEX)	MyISAM (nur CHAR, VARCHAR, TEXT) InnoDB (seit 5.6)
Spatial-Index (SPATIAL INDEX)	MyISAM, InnoDB, NDB, BDB, ARCHIVE MyISAM (Indices)
Fremdschlüssel/Referenzielle Integrität	InnoDB, BDB
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	Archiv	Memory
Max. Tabellengröße	256TB	64TB	??	384EB	--	RAM
Betriebssystem-unabh. (port.)	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 auf Basis von...	Table	Satz	Page	Satz	Satz	Table
MVCC (Snapshot lesen)	--	JA	--	--	JA	--
BTREE Index	JA	JA	--	JA	--	JA
HASH Index	--	--	JA	JA	--	JA
Volltext Index	JA	(JA)*	--	--	--	--

!!!
PITR

Clustered Index	--	JA	--	--	--	--
Statistikupdates	JA	JA	JA	JA	JA	JA
Geografische Datentypen	JA	JA	??	JA	JA	--
Geografischer Index (RTREE)	JA	--	??	--	--	--
Daten-Cache (Tabellendaten)	--	JA	JA	JA	--	--
Index-Cache	JA	JA	JA	JA	--	--
Query-Cache (Abfragen)	JA	JA	JA	JA	JA	JA
Komprimierte Daten	JA	--	--	--	JA	--
Verschlüsselte Daten	JA	JA	JA	JA	JA	JA
Clusterfähig	--	--	--	JA	--	--

*) Volltextindex bei "InnoDB" ab MY!5.6

Befehle zum Abfragen/Einstellen der Engine:

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

Die Engine einer Tabelle lässt sich jederzeit ändern, dabei wird die Tabelle umkopiert (doppelte Platzbedarf, Zeitaufwand):

```
ALTER TABLE <Tbl> ENGINE = <Engine>;             # Engine einer Tabelle ändern
ALTER TABLE <Tbl> TYPE = <Engine>;               # Analog (veraltet!)
                                                 # (automatisch konvertiert!)
```

6a) InnoDB Engine

Eigenschaften:
- Standard-Engine seit MY!5.5.1
- Immer Index-organisiert (auch wenn kein Index im Schema definiert)
  (Datensätze nach Primärschlüssel sortiert)
- Multi-Range-Read + Batched Key Access nur hier verfügbar
- Weitere Indices verweisen auf Primärschlüssel --> 2-stufiger Zugriff
- Daten können auf RAW-Partition liegen (kein Dateisystem dazwischen)
- Unterstützt
  + Referenzielle Integrität (Fremdschlüssel, Foreign Keys)
  + Row-Level Locking
  + Konsistentes Lesen ohne Sperren
  + Autom. Fehlerkorrektur
  + Spatial-Index
- KEIN Fulltext-Index (erst ab MY!5.6)
- ALTER TABLE eingeschränkt
- Max. Größe: 64 TByte

* Ablageort von "InnoDB"-Datenbanken und -Indices:
  A) Ein zentraler Tablespace im Dateisystem:
     ibdata1
     ibdata2
     ...
     <Tbl>.frm
  B) Pro Tabelle ein Tablespace im Dateisystem:
     <Tbl>.ibd
     <Tbl>.frm
  C) Rohe Partitionen:
     innodb_data_file_path=/dev/hdd1:3Gnewraw;/dev/hdd2:2Gnewraw   # Init
     innodb_data_file_path=/dev/hdd1:3Graw;/dev/hdd2:2Graw   # Betrieb

* InnoDB-Einstellungen

```
innodb_data_home_dir        = /var/lib/mysql              # Startpunkt Tablespaces
innodb_data_file_path       = <Path>:<MByte>,...,         # ibdata1, ibdata2, ...
(fixe Größe)
                               [<Path>:<MByte>:autoextend]  # Letztes ibdata bel.
groß
innodb_data_file_path       = ibdata1:10M:autoextend      # ibdata1 ab 10MB
```

```
    innodb_file_per_table       = 1                             # Daten+Indexdatei
"*.ibd" pro Tabelle
    innodb_autoextend_increment = 8                             # Vergrößerungseinheit
```

6b) MyISAM Engine

Eigenschaften:
- Standard-Engine bis MY!5.1 (seitdem InnoDB)
- Interne Verwaltungsdatenbank "mysql" nur per MyISAM-Engine möglich
- Keine Transaktionen
- Locks nur auf kompletter Tabellen-Ebene (paralleler Zugriff)
- Sehr effizient bei vorwiegenden Lesezugriffen
- Flexibelste Autoincrement-Eigenschaft aller Speicherengines
- In komprimierte read-only Tabellen konvertierbar
- Feste Zeilenlänge möglich --> schnellere Zeilensuche
- Verwendbar für MERGE-Tabellen
- Portabel da alle Datenformate maschinenunabhängig
  + ".frm"-Datei (Tabellenstruktur)
  + ".MYD"-Datei (Daten)
  + ".MYI"-Datei (Indices)
- Mächtige Volltextsuche

6c) MEMORY Engine (HEAP)

Eigenschaften:
- Tabellen-Definition liegt in ".frm"-Datei
- Tabellen-Daten liegen im RAM
- Für temporäre Tabellen
- Fixe Satzlänge
- Index ist HASH-basiert
- Replikation nicht möglich
- Nicht alle Datentypen unterstützt
  + Kann keinen BLOB- oder TEXT-Datentyp enthalten

6d) CSV Engine (Comma Separated Values, ab MY!5.1)

Eigenschaften:
- CSV-Tabellen unterstützen keine Indizierung!
- Partitionierung von CSV-Tabellen nicht mehr möglich (war mal)
- NULL-Spalten nicht mehr erstellbar (aber noch nutzbar falls existent)
  --> ERROR 1178 (42000): The storage engine for the table doesn't support nullable columns
- Auf Dateisystemebene füllbar (vorher FLUSH TABLE <Tbl>;)
- Darstellung der NULL durch "\N"
- Mit "configure --with-csv-storage" Quellcode übersetzen!

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)!

Dateinamen der CSV-Dateien einer Tabelle <Tbl>:

```
  <Tbl>.frm     # Tabellen-Definition
  <Tbl>.CSV     # Tabellen-Daten
  <Tbl>.CSM     # Tabellen-Metadaten (Tabellenzustand, Anzahl Zeilen)
```

Beispiel:

```
  CREATE TABLE csv_test (
      i INT      NOT NULL,
      c CHAR(10) NOT NULL
  ) ENGINE = CSV;

  INSERT INTO csv_test VALUES
      (1, "Satz 1"),
      (2, "Satz 2"),
```

```
      (3, "Satz 3");
```

  Erzeugt folgende Datei "csv_test.CSV":

```
    +-----------------+
    |"1","Satz 1" <NL>|
    |"2","Satz 2" <NL>|
    |"3","Satz 3" <NL>|
    +-----------------+
```

Was passiert wenn Format kaputt?
  * Die DB kann nichts mehr damit anfangen!
  * Nachricht über zerstörte Tabelle beim Zugriff

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

```
  CHECK TABLE csv_test;
  REPAIR TABLE csv_test;
```

6e) ARCHIVE Engine TODO

Für Speicherung großer Datenmengen bei geringem Platzbedarf per Kompression. Keine Indizes möglich, d.h. Zugriff per "Table Scan". Nur INSERT und SELECT werden unterstützt. Der schnelle Zugriff auf die Daten steht hier nicht im Vordergrund.

Vor dem Speichern der Daten auf dem Speichermedium werden diese zunächst in einem Kompressionspuffer gesammelt. Wenn eine Serie von Einfüge-Operationen beendet wird, wird der optimale Kompressionsalgorithmus ermittelt und die Daten werden komprimiert ausgegeben.

Falls während einer Sequenz von Einfüge-Operationen von einem anderen Benutzer eine SELECT-Anfrage kommt, wird eine vorzeitige Kompression und Ausgabe der im Kompressionspuffer gespeicherten Daten erzwungen.

6f) BLACKHOLE Engine

Entwickelt, um die Syntax von SQL-Anweisungen zu prüfen und ein Binärlog zu schreiben. Die Daten werden nicht gespeichert ("schwarzes Loch"). Dadurch ist die Syntaxprüfung von SQL-Anweisungen ohne Speicherplatz-Bedarf möglich. Die Ausgabe des Binärlogs kann über einen Parameter aktiviert und deaktiviert werden.

Ideal für:
  * Syntaxprüfung von Dump-Dateien
  * Test der Datenreplikation durch Binärlog-Vergleich auf Master- + Slave
  * Zeitmessungen Aufwand für Schreiben des Binärlogs.
  * Keine Schreibzugriffe auf Master, nur auf Slaves

6g) MERGE Engine

Eigenschaften:
  * Zusammenfassung mehrerer physikalischer Tabellen zu einer logischen Tabelle
  * Müssen absolut identisch sein (Reihenfolge, Spalten, Typen, Längen)
  * Merge-Tabellen sind eine frühe (einfache) Form von Partitionierung
    + Untertabellen sichtbar
    + Pruning manuell durchzuführen
    --> 18) Partitionierung

Zweck:
  * 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)
  * Reparatur schneller
  * Tabelle identifizieren (verschiedene Namen für gleiche Tabelle) --> View
  * EINE Tabelle umbenennen --> RENAME, View

Beispiel 1:

```
    CREATE TABLE t1 (a INT NOT NULL PRIMARY KEY) ENGINE=MyISAM;
    CREATE TABLE t2 (a INT NOT NULL PRIMARY KEY) ENGINE=MyISAM;
    INSERT INTO t1 VALUES (1), (2), (3);
    INSERT INTO t2 VALUES (4), (5), (6);

    CREATE TABLE mrg(a INT NOT NULL PRIMARY KEY)
      ENGINE=MERGE UNION=(t1, t2) INSERT_METHOD=LAST;   # Oder FIRST
```

Beispiel 1:

```
    CREATE TABLE log_01 (
        pkey      INT(11) NOT NULL AUTO_INCREMENT,
        a         INT,
        b         VARCHAR(12),
        timeEnter TIMESTAMP(14),
        PRIMARY KEY (pkey)
    ) ENGINE = MyISAM;

    CREATE TABLE log_02 (
        pkey      INT(11) NOT NULL AUTO_INCREMENT,
        a         INT,
        b         VARCHAR(12),
        timeEnter TIMESTAMP(14),
        PRIMARY KEY (pkey)
    ) ENGINE = MyISAM;

    CREATE TABLE log_sum (
        pkey      INT(11) NOT NULL AUTO_INCREMENT,
        a         INT,
        b         VARCHAR(12),
        timeEnter TIMESTAMP(14),
        PRIMARY KEY (pkey)
    ) ENGINE = MERGE UNION(log_01, log_02) INSERT_METHOD = LAST;
```

6h) FEDERATED Engine

Bietet Zugriff auf Tabellen, die auf einem anderen Server liegen (müssen dort bereits existieren). Der lokale Server verhält sich als Client, der auf den entfernten Server zugreift. Er speichert selbst nicht die Daten, sondern gewährt nur den Zugriff auf den entfernten Server, auf dem ebenfalls direkt auf die Daten zugegriffen werden kann.

ACHTUNG: Die Zugangsdaten zum entfernten Datenbank-Server stehen unverschlüsselt in der lokalen "*.frm"-Datei. Der Zugriff auf das mysql-Datenverzeichnis sollte also auf Betriebssystem-Ebene eingeschränkt werden, um das Auslesen der Zugangsdaten durch Unbefugte zu verhindern.

6i) EXAMPLE Engine

Code-Beispiel für die Entwicklung einer eigenen Speicher-Engine. Kennt Funktionen zum Erstellen einer Tabelle, die Funktionen zum Schreiben und Lesen der Datensätze sind nur angedeutet. Ein SELECT-Statement liefert immer eine leere Ergebnismenge.

6j) Weitere Engines

* BDB (Berkley Data Base): Seit MY!5.1 nicht mehr unterstützt
* NDB (Network Data Base): MySQL Cluster-Engine

7) Datenbanken erstellen und verwalten

Standard-Datenbanken nach der Installation:

Datenbank	Bedeutung
mysql	Interne Verwaltungsdaten (immer MyISAM)
test	Testdatenbank (leer)
INFORMATION_SCHEMA	Verwaltungsdaten (generiert aus "mysql", read-only!)
PERFORMANCE_SCHEMA	Performancezähler (Messdaten, read-only)

Standort:

```
/var/lib/mysql/mysql/*              Interne Datenbank "mysql"
/var/lib/mysql/<Db>/*               Datenbank <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 <Db> erstellen:

```
mysqladmin -uroot -pgeheim CREATE <Db>                  # Variante A
mysql      -uroot -pgeheim -e "CREATE DATABASE <Db>"    # Variante B
CREATE DATABASE <Db>;                                   # Im "mysql"-Client
CREATE SCHEMA   <Db>;                                   # Im "mysql"-Client
CREATE DATABASE IF NOT EXISTS <Db>;                     # Im "mysql"-Client
CREATE SCHEMA   IF NOT EXISTS <Db>;                     # Im "mysql"-Client
```

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
mysql -e "CREATE SCHEMA   <DbNew>"                      # Neue Datenbank anlegen
mysql     <DbNew> < dump.sql                            # Evtl. --no-data
```

Datenbank löschen (VORSICHT: mit allen Tabellen+Indices darin!):

```
mysqladmin -uroot -pgeheim DROP <Db>                    # Variante A (mit Rückfrage!)
mysql      -uroot -pgeheim -e "DROP DATABASE <Db>"      # Variante B
DROP DATABASE <Db>;                                     # Im "mysql"-Client
DROP SCHEMA   <Db>;                                     # Im "mysql"-Client
DROP DATABASE IF EXISTS <Db>;                           # Im "mysql"-Client
DROP SCHEMA   IF EXISTS <Db>;                           # Im "mysql"-Client
```

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>    # STD-Zeichensatz für alle Sp.
           CHECKSUM = 0/1                       #
           COLLATE <Collate>                    # STD-Sortierung für alle Sp.
           COMMENT <String>                     # Beliebiger Kommentar
           CONNECTION <String>                  #
           DATA DIRECTORY <DirPath>             # Ablageverz. Daten
           DELAY_KEY_WRITE = 0/1                #
           ENGINE = <Engine>                    # Engine
           INDEX DIRECTORY = <DirPath>          # Ablageverz. Indices
           INSERT METHOD = {NO | FIRST | LAST}  #
           KEY_BLOCK_SIZE = <N>                 #
           MAX_ROWS = <N>                       # Für Partitionierung
           MIN_ROWS = <N>                       # Für Partitionierung
```

```
            PACK_KEYS = 0/1/DEFAULT              #
            PASSWORD = <String>                  #
            ROW_FORMAT = {<RowFmt>}              # Mehrere
            UNION = <Tbl>                        #
            <Partition>                          #

<RowFmt> = DEFAULT      #
            DYNAMIC      #
            FIXED        #
            COMPRESSED   #
            REDUNDANT    #
            COMPACT      #

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

Tabellentypen/Engines:
  * 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 = <Engine> ...    # TYPE veraltet!
  * Option: default-table-type=<Engine>
  * ALTER TABLE <Tbl> ENGINE = <Engine>   # 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-user-HOWTO.txt --> 4) MySQL-Datentypen

Primärschlüssel (letzten Endes durch einen Index realisiert!):
  * NUR EINER erlaubt
  * Kein NULL-Wert erlaubt
    PRIMARY KEY (...)

Tabellen und ihren Status auflisten (Engines, Partitionen, ...):
  USE <Db>;
  SHOW TABLES;                                  # Aktuelle Datenbank
  SHOW TABLES FROM <Db>;                        # Datenbank <Db>
  SHOW TABLES FROM <Db> LIKE <Muster>;          # Passende Datenbanken
  SHOW TABLE STATUS;                            # Aktuelle Datenbank
  SHOW TABLE STATUS FROM <Db>;                  # Datenbank <Db>
  SHOW TABLE STATUS FROM <Db> LIKE <Muster>;    # Passende Datenbanken

Tabellenstruktur/definition anzeigen:
  USE <Db>;                         #
  SHOW COLUMNS FROM <Tbl>;          #
  DESCRIBE <Tbl>;                   #
  DESCRIBE <Tbl> <Col>;             # ... LIKE "%_priv"
  EXPLAIN <Tbl>;                    #
  EXPLAIN <Tbl> <Col>;              #
  SHOW CREATE TABLE <Tbl>;          # Inkl. Engine

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 = <Engine>;
      ALTER TABLE <Tbl> TYPE = <Engine>;   # 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
  * Bei Fremdschlüsselbezug von <Tbl> doch einzelsatzweise

Tabelle entfernen (inklusive Inhalt!):

  DROP TABLE <Tbl>;             # Fehler falls nicht existent
  DROP TABLE IF EXISTS <Tbl>;   # Kein Fehler falls nicht existent

Tabelle kopieren (Struktur, -=ohne Daten/Indices, +=mit Daten/Indices):

  CREATE TABLE IF NOT EXISTS copy LIKE pers;        # -Daten, +Indices (MY!5.0)
  CREATE TABLE copy SELECT * FROM pers;             # +Daten, -Indices
  CREATE TABLE copy SELECT * FROM pers WHERE 1=2;   # -Daten, -Indices (Trick!)
  INSERT INTO  copy SELECT * FROM pers;             # Daten kopieren


8a) Temporäre Tabellen
----------------------

Es gibt zwei Varianten von Temporären Tabellen:
  * IMPLIZIT von MySQL angelegte für Zwischenergebnisse von
    Joins, Sortierung, Gruppierung, DISTINCT, ...
  * EXPLIZIT angelegte im Rahmen einer Sitzung

Temporäre Tabelle EXPLIZIT anlegen + löschen:

  CREATE TEMPORARY TABLE pers (      # Temp. Tabelle (stirbt bei Sitzungsende)
    nr      INT          DEFAULT 0   # (VERDECKT evtl. gleichnamige echte Tab.!)
    vorname VARCHAR(30),             #
    name    VARCHAR(30)              #
  ) CHARACTER SET latin1;            # Standard-Zeichensatz für Textspalten
                                     #
  DROP TEMPORARY TABLE pers;         # Temporäre Tabelle löschen (nicht echte!)

Eigenschaften:
  * Kein "*.frm"-Datei dafür erzeugt (da nicht permanent)
  * Pro Sitzung vorhanden (d.h. gleichnamige in mehreren Sitzungen erlaubt)
```

```
  * Automatisch gelöscht am Sitzungsende (Trennung Client vom Server)
  * Dürfen genauso heißen wie eine echte Tabelle
    --> VERDECKEN gleichnamige echte Tabelle bis temp. Tabelle wieder gelöscht
  * Liegen im Verz. "/tmp" (bzw. def. "--tmpdir=PATH;..." oder "-t=PATH;...")
  * Nur mit Engine "MEMORY", "MyISAM", "MERGE" oder "InnoDB" möglich
  * Kann keinen BLOB- oder TEXT-Datentyp enthalten (MY?)
  * Von SHOW TABLES nicht angezeigt

* Bei statement-based Replikation nicht verwendbar:
  + Absturz der Kopie  --> Temp. Tabellen verschwinden
  + Neustart der Kopie --> Temp. Tabellen fehlen

* Ersatz "Pseudotemporäre Tabellen":
  + CONNECTION_ID() als Teil des Tabellennamens verwenden
  + Langsamer da "*.frm" auf Platte muss (evtl. --sync_frm-Option deaktivieren)
  + Percona Toolkit "pt_find" löscht autom. pseudotemp. Tabellen auf Basis von:
      --connection-id
      --server-id

* Konfigurations-Variablen:
    --tmp_table_size   # Max. für temp. Tab. zur Verfügung gestellter Speicher
    --tmpdir           # Verz. für temp. Tab. ab bestimmter Größe
    --max_tmp_tables   # Max. Anzahl an temp. Tab. gleichzeitig

* Tabelle als gleichnamige temp. Tab. in Speicher holen --- bringt das etwas?
  + Für Tests Original-Tabelle verkleinert in gleichnamige temp. Tab. ablegen
  + Im Speicher ist wenig Platz
  + Kein Zugriff per Index möglich
  + Caching normaler Tabellen holt diese auch schon in den Speicher


9) Tabellen und Indices 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 InnoDB-Tabellen prüfen:
  * Datenbankname angebbar, 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" (Zersplitterung)
    + 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 "InnoDB"!)
    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"): auch bei InnoDB
  * Falls Tabelle nur gelesen wird!
  * Platzbedarf reduziert
  * Zugriffsgeschwindigkeit gesteigert
      myisampack [<Optionen>] /var/lib/mysql/<Db>/<Tbl>.MYI   # Nicht .MYD!
  * Optionen
      +---------------------+-----------------------------------------+
      | Option              | Bedeutung                               |
      +---------------------+-----------------------------------------+
      | -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 per SQL-Anweisung einfügen (per MySQL-Erweiterung auch
mehrere, alle Werte außer numerischen MÜSSEN in Hochkommas gesetzt sein!):

  INSERT INTO <Tbl>               # Einzelsatz (alle Spalten, Werte in
```

```
    VALUES (<Wert>, ...);          #             Spaltenreihenfolge)
  INSERT INTO <Tbl> (<Col>, ...)   # Einzelsatz (ausgewählte Spalten)
    VALUES (<Wert>, ...);          #
  INSERT INTO <Tbl> (<Col>, ...)   # Mehrere Datensätze (MY!)
    VALUES (<Wert>, ...),          #   1. Satz
           (...),                  #   2. Satz
           ...;                    #   usw.
  INSERT INTO <Tbl> (<Col>, ...)   # Datensätze aus anderer Tabelle kopieren
  SELECT <Col>, ... FROM <Tbl>;

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

  B) Delimited ASCII-Datei per SQL-Anweisung "LOAD DATA..."
     + Komplement zu SELECT ... INTO OUTFILE
     + Schneller als INSERT-Anweisungen
     + Wert "\N" in Datenfile --> NULL in Datenbankspalte (Codierung)
     + 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)
         LOAD DATA [LOCAL] INFILE <FilePath>    # Quelldatei
                   [REPLACE | IGNORE]           # Wie doppelte Schlüssel beh.
                   INTO TABLE <Tbl>             # Zieltabelle
                   [<LoadOpt>...];              # Optionen
     + Daten liegen auf Client ("LOCAL") oder Server (kein "LOCAL")
     + Behandlung doppelter Schlüssel:
       - Standard: Import abbrechen
       - IGNORE:   Neue Zeile ignoriert          --> erste gilt!
       - REPLACE:  Neue Zeile ersetzt vorhandene --> letzte gilt!
     + Optionen <LoadOpt> (in dieser Reihenfolge, "FIELDS/LINES" nur 1x!)
       (einzelne oder alle können auch fehlen, ebenso die Spalten)
         FIELDS TERMINATED BY <Char>              # STD: "\t" = Tabulator
                [OPTIONALLY] ENCLOSED BY <Char>   # STD: ""   = keines
                ESCAPED BY <Char>                 # STD: "\\" = \
         LINES STARTING BY <String>               # STD: ""   = leer
               TERMINATED BY <String>             # STD: "\n" = \n
         IGNORE <Nnn> LINES                       # Erste NNN Zeilen ignorieren
         (<Col>, ...)                             # Best. Sp. in Reihenf. laden (sonst
alle)
         SET <Col> = <Expr>, ...                  # Best. Sp. fix füllen
     + Zeilenenden sind abhängig vom Betriebssystem (STD: "\n"):
       - Linux:   LINES TERMINATED BY "\n"
       - Windows: LINES TERMINATED BY "\r\n"
       - MacOS X: LINES TERMINATED BY "\r"
     + ACHTUNG: Ohne LOCAL wird die Datei auf dem Server gesucht unter:
         ./FILE        --> /var/lib/mysql/FILE
         FILE          --> /var/lib/mysql/<Db>/FILE
         /PFAD/ZU/FILE --> Absoluter Pfad funktioniert NICHT!

  C) Dienstprogramm "mysqlimport"
       mysqlimport <Optionen> <Db> <FilePath>...;
     + Dateiname ohne Extension MUSS Tabellenname entsprechen!
     + CSV-Datei (Comma Separated Values)
     + 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=<Char>   # STD: "\t" = Tabulator
         --fields-enclosed-by=<Char>     # STD: ""   = keines
         --fields-escaped-by=<Char>      # STD: "\\" = \
         --lines-terminated-by=<Char>    # STD: "\n" = \n
         -L/--local             # Datei steht auf Client (STD: Server)
         -i/--ignore            # Ident. Zeile ignoriert (erste zählt!)
         -r/--replace           # Ident. Zeile ersetzt vorhandene (letzte zählt!)

Datensätze per SQL-Anweisung ausgeben:

  A) SELECT <Col>, ... FROM <Tbl>, ...;
     + Ausgabe auf Standard-Ausgabe (Client)
     + Umlenken auf Datei

  B) SELECT INTO OUTFILE...
     + Als delimited ASCII-Datei
```

```
    + Nur auf Server möglich
    + <Col> = "*" um alle Daten zu sichern
        SELECT <Col> INTO OUTFILE <FilePath> [<SaveOpt>]
                     FROM <Tbl> [<Parameter>];
    + Optionen <SaveOpt> (in dieser Reihenfolge anzugeben!, "FIELDS" nur 1x!)
      (einzelne oder alle können auch fehlen, ebenso die Spalten):
        FIELDS TERMINATED BY <Char>              # STD: "\t" = Tabulator
               [OPTIONALLY] ENCLOSED BY <Char>   # STD: ""   = keines
               ESCAPED BY <Char>                 # STD: "\\" = \
        LINES STARTING BY <String>               # STD: ""   = leer
              TERMINATED BY <String>             # STD: "\n" = \n


11) Datensicherung und -wiederherstellung
-----------------------------------------

Datensicherung ist notwendig:
  * Große Anzahl von Datensätzen
  * Großer finanzieller/zeitlicher Aufwand zur Wiederherstellung
  * Datenstände bestimmer 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 Threads sperren
(alle benötigten Tabellen 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 Beantrager
      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...
```

D) Dienstprogramm "mysqldump"
   + Liefert komplette Definition einer Datenbank oder Tabelle als SQL-Code
     mysqldump <Optionen> <Db> [<Tbl> ...]   # Eine Datenbank
     mysqldump <Optionen> --databases [<Db> ...] # Mehrere Datenbanken
     mysqldump <Optionen> --all-databases   # Alle Datenbanken
   + Optionen

Option (*=On)		Bedeutung
-A/--all-databases		ALLE Datenbanken ausgeben
-B/--databases DB...		Auswahl von Datenbanken ausgeben
--tables TAB...		Auswahl von Tabellen angeben
-d/--no-data		Nur Tabellen-CREATE, keine Daten
-t/--no-create-info		Nur Daten, kein Tabellen-CREATE
--triggers	*	Trigger ausgeben
--skip-triggers		(abschalten)
-R/--routines		Routines (Stored Procedures+Functions) ausg.
-E/--events		Events ausgeben
-l/--lock-tables	*	Alle Tabelle einer DB gegen Schreiben sperr.
--skip-lock-tables		(abschalten)
-x/--lock-all-tables		Alle Tab. aller DB locken (global read lock) (--single-transaction + --lock-tables ausg.)
--single-transaction		Konsistenten Snapshot aller Tabellen durch dumpen in einer Transaktion (nur InnoDB) (schaltet --lock-tables ab, währenddessen ALTER/DROP/RENAME/TRUNCATE TABLE verboten
--add-drop-database		DROP DATABASE IF EXISTS am Anfang
--add-drop-table	*	DROP TABLE IF EXISTS vor CREATE TABLE
--skip-add-drop-table		(abschalten)
--add-drop-trigger		DROP TRIGGER IF EXISTS vor CREATE TRIGGER
--add-locks	*	LOCK + UNLOCK TABLE um INSERT Daten
--skip-add-locks		(abschalten)
-a, --create-options	*	All MySQL-spez. TABLE CREATE-Opt. ausg.
--skip-create-options		(abschalten)
--flush-privileges		FLUSH PRIVILEGES als Abschluss erzeugen
-K/--disable-keys	*	DISABLE KEYS vor + ENABLE KEYS nach INSERT
--skip-disable-keys		(abschalten)
-n/--no-create-db		CREATE DATABASE IF NOT EXISTS weglassen
-t/--no-create-info		CREATE TABLE IF NOT EXISTS weglassen
-c/--complete-insert		Komplette INSERT-Anw. (inkl. Spaltennamen)
-e/--extended-insert	*	Erweiterte INSERT-Form (schnell, kompakt)
--skip-extended-insert		(abschalten)
--hex-blob		Binärdaten ((VAR)BINARY, BLOB) hexadez. dump
--dump-date	*	Dumpdatum am Dateiende ausgeben
--skip-dump-date		(abschalten)
--comments	*	Pgm+Server.version + Host als Kommentar aus.
--skip-comments		(abschalten)
--order-by-primary		Tabellen-Daten nach PK sortieren (langsam)
-C/--compress		Daten bei Remoteübertragung komprimieren
-F/--flush-logs		Logdaten vor Dump rausschreiben
-f/--force		SQL-Fehler ignorieren
--max-allowed-packet=N		Max. SQL-Anweisungs-Länge
--opt	*	Optionen zur Beschleunigung an/aus:
--skip-opt		--add-drop-table --add-locks --create-options --disable-keys --extended-insert --lock-tables --quick --set-charset
--compact		Optionen zur Platzreduktion an/aus:
		--skip-add-drop-table --skip-add-locks --skip-comments --skip-disable-keys --skip-set-charset

```
      | -q/--quick               | Datensätze einzeln holen statt auf 1x       |
      | -r/--result-file=<File>  | Dateiname angeben (Windows: \r\n --> \n)    |
      +--------------------------+-----+---------------------------------------+
      | --fields-terminated-by=<Char> |                                       |
      | --fields-enclosed-by=<Char>   |                                       |
      | --fields-escaped-by=<Char>    |                                       |
      | --lines-terminated-by=<Char>  |                                       |
      +-------------------------------+---------------------------------------+
    + Komprimiert sichern
        mysqldump ... | gzip > SICHERUNG.mysql.gz
    + Komprimierte Sicherung zückspielen
        gunzip -c SICHERUNG.mysql.gz | mysql ...
        zcat 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, C#/.NET, ... 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 <DirPath>;
  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, C#/.NET, ... 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

* Sonstige Verfahren
  + mysqlhotcopy
  + mylvmbackup
  + mydumper + myloader
  + mysqldbimport + mysqldbexport
  + mysql-myisam-dump
  + mysql-binlog-dump
  + mysqlbinlog


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) STATUS;                          # Server-Status + einige Server-Infos
  B) SHOW STATUS;                     # Liste der Variablen/Optionen + Wert
     SHOW STATUS LIKE "CONNECTIONS";  # Teilmenge davon
     SHOW STATUS LIKE "ABORTED%";     # Teilmenge davon
  C) mysqladmin status                # Das Wichtigste zum Server-Status
       Uptime: 285286  Threads: 1  Questions: 138  Slow queries: 0  Opens: 103
```

```
      Flush tables: 1  Open tables: 24  Queries per second avg: 0.0
  D) 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 oder mysqld.log
    + Start, Kritische Fehler, fehlerhafte Tabellen, Herunterfahren
    + "safe_mysqld" fängt mysqld-Fehlermeldung auf stderr ab und schreibt
      sie auf Datei
  B) Anfrage-Log # Allgemeine Anfragen # <Host>.log
    + Option -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)
    + Option --log-slow-queries=<Logfile>                #
             --log-slow-admin-statements=<Logfile>       #
             --log-queries-not-using-indices=<Logfile>   #
             --long-query-time=<N>                       # STD: 10s (1,2,...)
    + 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
    + Option --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,
Zeilnnummer
               SWAPS                # Anzahl Swaps

  * Prozessinformationen:
      +---------+-----------------------------------------------+
      | Spalte  | Bedeutung                                     |
      +---------+-----------------------------------------------+
      | 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 -pgeheim -h127.0.0.1 ping
      mysqladmin -utom -pgeheim -h127.0.0.1 status
      mysqladmin -utom -pgeheim -h127.0.0.1 processlist
      mysqladmin -utom -pgeheim -h127.0.0.1 kill 1234

  * Diverse Caches leeren und Tabellen schließen (evtl. mit Lock)
    (STD: in Binlog geschrieben und damit auf Slave repliziert):

      FLUSH [NO_WRITE_TO_BINLOG | LOCAL] <Opt>, ...;

   + Signal SIGHUP (1) an MySQL-Server löst einige FLUSH-Operationen aus
   + Erzeugt automatisch COMMIT der aktuellen Transaktion
   + FLUSH TABLES (Plural) hat Synonym FLUSH TABLE (Singular)
   + NO_WRITE_TO_BINLOG oder LOCAL zeichnet Operation nicht in Binlog auf

      +------------------+-----------------------------------------+------+
      | Option <Opt>     | Beschreibung                            |Binlog|
      +------------------+-----------------------------------------+------+
      | DES_KEY_FILE     | DES-Schlüssel des Servers neu laden     |  ja  |
      | HOSTS            | Bei IP-Adress-Änderung oder Blockade    |  ja  |
      | [<LogType>] LOGS | Log-Dateien schließen + wieder öffnen:  |  --  |
      |    <LogType>     |   BINARY  = Binlog Dateien + Inkrement  |  --  |
      |                  |   ENGINE  = Alle Engine-Logs (InnoDB)   |  --  |
      |                  |   ERROR   = Errorlog Datei              |  --  |
      |                  |   GENERAL = General Querylog Datei      |  --  |
      |                  |   RELAY   = Relaylog Datei              |  --  |
      |                  |   SLOW    = Slowlog Datei               |  --  |
      | PRIVILEGES       | Berechtigungen neu einlesen (nach Änd.) |  ja  |
      | QUERY CACHE      | Querycache defragment. (nicht löschen!) |  ja  |
      | STATUS           | Session Status und Keycache-Zähler lö.  |  ja  |
      | TABLES <Tbl>, ...| Tabellendaten auf Platte schreiben +    |  --  |
      |                  | Query-Cache löschen (RESET QUERY CACHE) |  --  |
      |   FOR EXPORT     |   globaler Lock für Export (InnoDB)     |  --  |
      |   WITH READ LOCK |   globaler Lock für Backup/Snapshot     |  --  |
      | USER_RESOURCES   | Benutzer-Zähler pro Stunde zurücksetzen |  ja  |
      +------------------+-----------------------------------------+------+

  * Stärkere Version von FLUSH für einige Operationen:
   + Erzeugt automatisch COMMIT der aktuellen Transaktion
```

```
RESET <Opt>, ...;
```

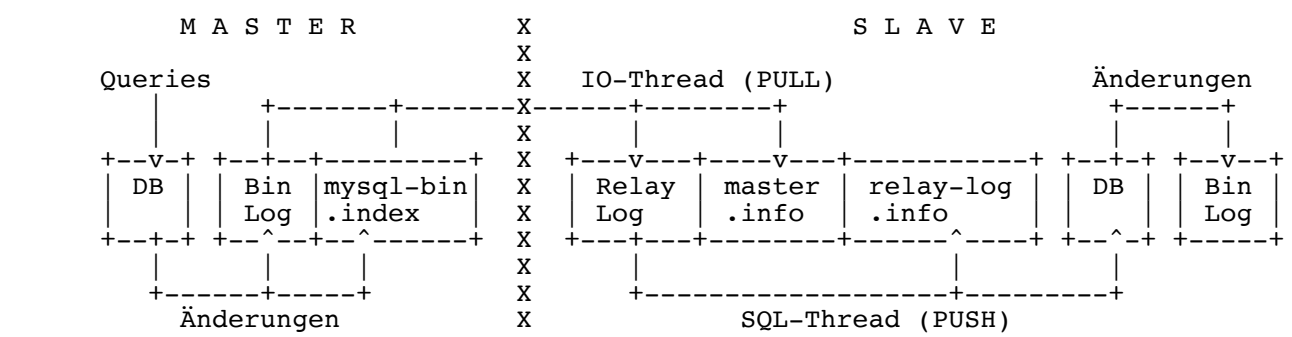
MASTER	Binlog in Index-Datei löschen + Binlog-Index-Datei leeren + neue Binlog-Datei beginnen
SLAVE	Master-Binlog-Position vergessen + Relaylog-Dateien löschen + neue Relaylog-Datei beginnen
QUERY CACHE	Query-Cache löschen

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

Synchronisation von Datenbeständen zwischen verschiedenen MySQL-Servern:
- * EIN Master kann auf VIELE Slaves replizieren
- * Abschalten/Ausfall von Slave-Server hat keinen Einfluss auf Master
- * Einweg-Replikation (Master --> Slave)

Zweck:
- * Ausfallsicherheit erhöhen (NUR eine Stelle ist schreibbar!)
- * Datenverlustrisiko verringern
- * Stabilität steigern
- * Lastverteilung bei lesenden Anfragen --> Zugriffsgeschwindigkeit erhöht
- * Master = Änderung/Abfrage, Slave = Aggregation/Analyse/Backup
- * Slave oft nur lesbar (auf "read-only" gesetzt, --read-only)

Ablauf:
- * Slave enthält eine 1:1-Kopie des Datenbestandes vom Master
  (zu einem Zeitpunkt, während dort keine Änderungen stattfinden!).
- * Master speichert Queries bzw. resultierende Änderungen in Binär-Logdatei
  --> Position des Binlog steht in Datei "mysql-bin.index"
- * Slave liest per IO-Thread die vom Master im Binär-Log gespeicherten
  Queries/Änderungen und speichert sie in Relay-Log bei sich zwischen (PULL)
  --> Position der Relay-Replikation steht in Datei "relay-log.info".
- * Slave liest per SQL-Thread den zwischengespeicherten Relay-Log
  und wendet die Änderungen auf seine Datenbank-Kopie an (PUSH).
  --> Position der DB-Replikation steht in Datei "master.info".
- * Slave erzeugt evtl. für seine DB-Änderungen eigenen Binär-Log,
  wenn er als Master für weiteren Slave fungiert ("log-slave-updates").



Voraussetzung:
- * Absolut identische Kopie der Master-Datenbank auf Slave
  zum Zeitpunkt ab dem Binär-Log vom Master zum Slave geholt wird
- * Gleiche MySQL-Version (darf bei Slave auch höher sein)
- * Gleicher Zeichensatz
- * Eindeutige "server_id" auf Master und Slave vorhanden
  (da alle Aktionen mit ID des auslösenden Servers gekennzeichnet sind)
  --> auto.conf + UUID muss ebenfalls verschieden sein
  (--server-uuid = ... bzw. --replicate-same-server-id = 1)

Hinweise:
- * Zu replizierende Datenbanken/Tabellen sind auswählbar:

```
 --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)
```

- * MyISAM und InnoDB möglich
- * FLUSH-Befehle werden nicht repliziert

```
    --> Nach Änderungen auf Master auf Slave manuell ausführen
  * LOAD DATA INFILE       --> repliziert
  * LOAD LOCAL DATA INFILE --> NICHT repliziert
  * RAND-Funktion wird nicht korrekt repliziert
    --> TIMESTAMP als Argument für RAND-Funktion verwenden
  * Format der Binärlogs (--binlog-format):
    + STATEMENT-basiert (schnell)
    + ROW-basiert       (langsam)
    + MIXED-Based Replication (MBR)
      - Normalerweise STATEMENT-basiert
      - Nur in bestimmten Ausnahmen ROW-basiert
        . RAND()
        . NOW()
        . NOW()
  * Table-Engines von Master und Slave müssen nicht identisch sein
  * Multi-Master --> Slave-Replikation wird nicht unterstützt.

Einrichten (Konfiguration in "my.cnf/ini" eintragen)
  (Bsp: Datenbank "kopie" von Rechner 10.1.5.111 --> 10.1.5.113 replizieren)
  0. Auf Slave Datenbank vollständig löschen (falls bereits vorhanden)
       rm -r /var/lib/mysql/...    # Oder alternativ umbenennen
       mv /var/lib/mysql/kopie         /var/lib/mysql/weg1
       mv /var/lib/mysql/ibdata1       /var/lib/mysql/weg2
  1. Auf Master neuen Benutzer "repl" mit folgenden Rechten einrichten
     (auf Slave nicht notwendig!)
       GRANT FILE, REPLICATION CLIENT, REPLICATION SLAVE
         ON kopie.*
           TO 'repl'@'%'
             IDENTIFIED BY "geheim";
  1b. Für LOAD DATA ... noch folgende Rechte nötig: SELECT, RELOAD, SUPER
  2a. Master eindeutige Server-ID geben (verschieden von Slave!)
       server-id = 111
  2b. Auf Master Binär-Logdatei aktivieren
       log-bin   = /var/log/mysql/mysql-bin.log
  2c. Auf Master Fehlermeldungen auf Datei schreiben (Fehlersuche erleichtern)
       log_error = /var/log/mysql/mysql.err
  2d. Master neu starten, damit Änderungen an "my.cnf" gelesen werden
  3a. Master gegen Änderungen sperren und seine Daten auf Platte schreiben
      (Slave runterfahren ist nicht notwendig und auch nicht erwünscht!)
       FLUSH TABLES WITH READ LOCK;   # Verb. A: sperrt Master gegen Schreiben
       SHOW MASTER STATUS;            # Verb. B: Binärlog-Position ermitteln
                                      # z.B. --> 'raum05pc03-bin.000001' + 290
  3b. DB-Verz. von Master rekursiv kopieren
       cd /var/lib/mysql                                  # Linux
       tar dbkopie.tgz czvf .                             # Linux
       cd "C:\ProgramData\MySQL\MySQL Server 5.6\data"    # Windows
       zip a dbkopie.zip .                                # Windows
  3c. Änderungen auf Master wieder erlauben
       UNLOCK TABLES;                 # Verb. A: erlaubt Schreiben auf Master
  4a. Datenbank-Kopie des Masters auf Slave kopieren und auspacken
       scp -rp dbkopie.tgz    10.1.5.113:                 # Linux
       ssh 10.1.5.113                                     # Linux
       cd /var/lib/mysql                                  # Linux
       tar xzvf ~/dbkopie.tgz                             # Linux
       cd "C:\ProgramData\MySQL\MySQL Server 5.6\data"    # Windows
       zip x dbkopie.zip                                  # Windows
  4b. Auf Slave die Besitzverhältnisse der kopierten Dateien auf Benutzer
      "mysql" und Gruppe "mysql" ändern und die Zugriffs-Rechte anpassen
       chown -R mysql.mysql /var/lib/mysql                          # Linux
       find /var/lib/mysql -type d -print0 | xargs -0 chmod 770     # Linux
       find /var/lib/mysql -type f -print0 | xargs -0 chmod 660     # Linux
  5a. Slave eindeutige Server-ID geben (nicht slave_id = ... ;-)
       server-id       = 113          # != MasterNr
  5b. Auf Slave folgende Konfigurations-Einträge machen
       read_only       = 1            # Nicht mehr schreibbar
       master-host     = 10.1.5.111   # (oder Host-Name)
       master-user     = repl         #
       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
  5c. Auf Slave Fehlermeldungen auf Datei schreiben (Fehlersuche erleichtern)
       log_error       = /var/log/mysql/mysql.err
  5d. Slave neu starten, damit Änderungen an "my.cnf" gelesen werden
  6a. Verbindung zum Master dem Slave mitteilen (wird gespeichert):
       SLAVE STOP;
```

```
        CHANGE MASTER TO
          MASTER_HOST     = '10.1.5.111',              # ab MY!5.6
          MASTER_USER     = 'repl',                    # ab MY!5.6
          MASTER_PASSWORD = 'geheim',                  # ab MY!5.6
          MASTER_LOG_FILE = 'raum05pc03-bin.000001',   # aus "SHOW MASTER STATUS"
          MASTER_LOG_POS  = 290;                       # aus "SHOW MASTER STATUS"
        SLAVE START;
  6b. Auf Slave allen Benutzern (außer "repl") für Datenbank "kopie" die
      Rechte UPDATE, DELETE, INSERT, DROP entziehen
        REVOKE INSERT, DELETE, UPDATE, DROP
          ON kopie.*
            FROM "root"@"%";
        ... FROM "kurs"@"%";
        ... FROM debian-sys-maint"@"localhost";
  7. Änderungen werden nun automatisch vom Master zu den Slaves repliziert
  8. Der Slave darf jederzeit angehalten werden (um z.B. einen Backup zu machen)
     nach Neustart der Replikation werden alle Änderungen seit Stop nachgezogen)
        SLAVE STOP;
        # Backup durchführen
        SLAVE START;

Konfigurations-Parameter
  * Master
      log-bin          = <File>       # Binäre Logdatei
      log-bin-index    = <File>       # 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      = 1      # Slave-Binlog für Sub-Slaves
      master-info-file       = <File> # 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       = 1      # Slave manuell per "SLAVE START" starten

SQL-Anweisungen
  * Master
      PURGE MASTER/BINARY LOGS;       #
      RESET MASTER;                   # Binär-Logs + Index löschen + leeres neu
erzeugen
      SET SQL_LOG_BIN = 0;            # Binär-Log-Aufz. stoppen (PROCESS-Recht!)
      SET SQL_LOG_BIN = 1;            # Binär-Log-Aufz. 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
      START SLAVE;                    # Replikation fortsetzen
      START SLAVE IO_THREAD;          # Repl. Master-Bin-Log --> Slave-Relay starten
      START SLAVE SQL_THREAD;         # Repl. Slave-Relay --> Slave-DB starten
      STOP SLAVE;                     # Repl. anhalten (später nachholbar)
      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 Konfigurations-Datei "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               = <FilePath>
      MASTER_LOG_POS                = <Position>
      MASTER_CONNECT_RETRY          = <N>          # Anz. Wiederholungs-Versuche
      RELAY_LOG_FILE                = <FilePath>
      RELAY_LOG_POS                 = <Position>
      MASTER_SSL                    = {0|1}
```

```
      MASTER_SSL_CA                 = <FilePath>
      MASTER_SSL_CAPATH             = <Path>
      MASTER_SSL_CERT               = <FilePath>
      MASTER_SSL_KEY                = <FilePath>
      MASTER_SSL_CIPHER             = <String>      # Algorithmus
      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    = <N>
  auto_increment_increment = <N>
```

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

```
  auto_increment_offset    = 1    # 2, 3, 4, 5
  auto_increment_increment = 5
```

Die Server erzeugen dann folgende Auto-Increment-Werte:

```
  Master 1:  1,  6, 11, 16, ...
  Master 2:  2,  7, 12, 17, ...
  Master 3:  3,  8, 13, 18, ...
  Master 4:  4,  9, 14, 19, ...
  Master 5:  5, 10, 15, 20, ...
```

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 erlaubt, wenn Slave-Thread nicht läuft.
```

Fehlermöglichkeiten:
  * master_id von Master und Slave gleich
    (oder --server-uuid von Master und Slave gleich bzw.
     --replicate-same-server-id = 0)
  * Auf Slave slave_id = ... statt master_id = ... verwendet
  * Grants für Replikations-Benutzer nicht ausreichend (REPLICATE
  * Anmeldung von Slave an Master scheitert (User+Host oder Passwort falsch)
  * Gemeinsamer Startpunkt verpasst
  * rm /var/lib/mysql/master.info vergessen
  * Unbedingt beide Server vor dem Kopieren der Ausgangsdatenbank runterfahren
    (bei Master genügt Flushen + Schreib-LOCK)
  * Unbedingt "log_error = FILE" aktivieren, damit man Fehler besser erkennt
  * GROSS/kleinschreibung der Dateinamen von Binär-Log ist relevant
  * Tippfehler Dateiname

14) Gesicherte und verschlüsselte Verbindungen

Eine Netzwerk-Verbindung kann abgehört und analysiert werden (Internet):
  * Benutzername + Passwort verschlüsselt übertragen
  * SQL-Anweisungen und Ergebnisdaten verschlüsselt übertragen (SSL)

Sicherheits-Maßnahmen:
  * Kennwort für "root" sofort nach Installation vergeben/ändern
  * Anonymer Benutzer (leerer Name) sofort nach Installation entfernen
  * Benutzer 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
  * Folgende Rechte NICHT vergeben:

Recht	Bedeutung

```
    +-------------------+-----------------------------------------------------+
    | PROCESS           | Kennwortabfragen einsehen                           |
    | FILE              | Systemfile mit "mysql"-Rechten in Tab. laden        |
    | GRANT OPTION      | Eigene Rechte an andere weitergebbar                |
    | SUPER             | Verw.op. erlaubt (KILL, SET GLOBAL)                 |
    | CREATE USER       | User erz./löschen/umbenennen + Rechte widerrufen    |
    | SHOW DATABASES    | Alle Datenbanken auflisten (SHOW DATABASES)         |
    | RELOAD            | Interne Tabellen/Logs/Rechte auffrischen (FLUSH)    |
    | SHUTDOWN          | Datenbank-Server herunterfahren (mysqladmin)        |
    | DROP              | Datenbanken/Tabellen/Views löschen                  |
    | CREATE TABLESPACE | InnoDB-Tablespace anlegen                           |
    | LOCK TABLES       | Tabellen sperren (SELECT-Recht notwendig!)          |
    | CREATE VIEW       | View anlegen                                        |
    | SHOW VIEW         | View ansehen (SHOW CREATE VIEW)                     |
    | TRIGGER           | Trigger erstellen/löschen (ab MY!5.1.6)             |
    | CREATE ROUTINE    | Anlegen von Prozeduren/Funktionen                   |
    | ALTER ROUTINE     | Ändern/Löschen von Prozeduren/Funktionen            |
    +-------------------+-----------------------------------------------------+
  * Anzahl Verbindungen auf sinnvollen Wert beschränken ("max_connections")
  * Einige MySQL-Optionen setzen (in "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
  * Keine Netzwerk-Verbindung erlauben --> Nur lokale Verb. per Socket erlaubt
      --skip-networking
  * Nur Netzwerk-Verbindungen aus bestimmten Netzwerk erlauben
      --bind-addr = IP/MASK
  * Alte Passwort-Verschlüsselung abschalten (unsicher):
      --secure_auth   = 1     --> 16 Z. nicht mehr erlaubt (ab MY!5.7 fix)
      --old_passwords = 0/1/2 --> Verhalten von PASSWORD()-Fkt. steuern
  * 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 = <Host>;
        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
  * Password vergessen, aber benötigt
  * Password falsch
  * Berechtigung zum Verbinden nicht vorhanden
  * Datenbank-Zugriff nicht erlaubt
  * UNIX_Rechte falsch/fehlend
  * Berechtigungs-Verwaltung ignorieren: --skip-grant-tables

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 (EXPLAIN)
-----------------------------

Der Query Optimizer erstellt zu jeder SQL-Anweisung auf der Basis von
statistischen Informationen über die Tabellen-Inhalte, die vorhandenen Indices,
den Verknüpfungen zw. den Tabellen im der SQL-Anweisung sowie sonstigen
Informationen einen Ablaufplan für dieses Statement (Query Execution Plan):

  * Welche Tabellen werden benutzt?
  * Welche Join Reihenfolge der Tabellen?
  * Welche Indices werden benutzt (und welche nicht)?
  * Wieviele Datensätze werden gelesen?

Zugriffsplan/Ablaufplan des Query Optimizers zu <Stmt> anzeigen
(vor MY!5.6.3 nur SELECT, ab dann auch INSERT, UPDATE, DELETE, REPLACE
und sogar PARTITION-bezogen):

  EXPLAIN <Stmt>;                 # Einfache Form
  EXPLAIN EXTENDED <Stmt>;        # Erweiterte Form, weitere Meldungen...
  SHOW WARNINGS;                  # ...in Warnings
  EXPLAIN PARTITIONS <Stmt>;      # Bei part. Tabelle inkl. Partitions-Nutzung
  EXPLAIN <Tbl>;                  # Tabellendefinition anzeigen
  EXPLAIN <Stmt> FORMAT = JSON;   # JSON-Format (inkl. EXTENDED + PARTITIONS)
                                  # (STD. FORMAT = TRADITIONAL)

Eigenschaften:
  * MySQL-Optimizer macht immer LEFT JOIN von links nach rechts
    --> Sortiert letzten Endes die Tabellen für den Zugriff!
  * Pro Tabellenzugriff wird max. ein Index benutzt
```

```
  * Zeilen-Reihenfolge im Ablaufplan = Reihenfolge der Tabellenabfragen
    + Die im SELECT verwendeten Tabellen werden aufgrund der Bewertung linear
      angeordnet und von links nach rechts paarweise verknüpft. Die entstehende
      Zwischentabelle wird mit der nächsten Tabelle verknüpft usw. bis die
      Gesamtergebnismenge vorliegt:
      - UNION    --> Zwei solche Ergebnismengen werden erstellt + addiert
      - Subquery --> Innere Ergebnismenge erstellt + mit äußerer verknüpft
    + Für JEDE Datensatzkombination (KREUZPRODUKT!) der vorherigen Tabellen
      werden Datensätze aus der nachfolgenden Tabelle gemäß "type" gelesen
  * Laufzeit des "Query Optimizer" hängt von Anzahl verknüpfter Tabellen ab
    + Steigt EXPONENTIELL mit Anzahl Tabellen
    + Bis 7-10 Tabellen kein Problem, kann ab 30 Tabellen extrem lange dauern
      (sogar länger als die eigentliche Abfrage mit Full Table Scan)

Steuern des Optimizers generell (lange Laufzeit <-> suboptimaler Plan):

  optimizer_prune_level  = 1     # 1=Zu komplexe Pläne ignorieren (schneller)
  optimizer_search_depth = 0     # 0=Automatisch (je kleiner, desto schneller)
  optimizer_switch = "..."       # Welche Operation im Ablaufplan nutzbar
  max_join_size = N              # Anz. in JOIN benutzter Datensätze begrenzen

Steuern des Optimierers pro Statement ("Hint"):

  SELECT ... STRAIGHT_JOIN ...   # Optim. ignor., in JOIN-Reihenfolge arbeiten
                                 # (linke Tab. immer vor rechter gelesen)
                                 # (zwischen Tabellen anstelle des Kommas)
  <Tbl> USE INDEX (<Liste>)      # Nur diese Ind. für <Tbl> nutzen
                                 # (z.B. PRIMARY, auch leer!)
  <Tbl> FORCE INDEX (<Liste>)    # Nur diese Ind. für <Tbl> nutzen
                                 # (Table Scan vermeiden!)
  <Tbl> IGNORE INDEX (<Liste>)   # Diese Indices NICHT für <Tbl> nutzen

Evtl. sinnvoll Tabellenstatistik neu aufzubauen, wenn Index ignoriert wird:

  ANALYZE TABLE <Tbl>;
  myisamchk --analyze <Tbl>;

Ausgabe von EXPLAIN [EXTENDED] ist eine Zeile pro in der SQL-Anweisung
verwendeter Tabelle sortiert nach der Zugriffsreihenfolge bei der
Query-Bearbeitung ("single-sweep-multi-join") mit folg. Info:

    +----------------------+-----------------------------------------------------+
    | Spalte               | Bedeutung                                           |
    +----------------------+-----------------------------------------------------+
    | id                   | ID der Abfrage (1 + #UNION + #Subqueries)           |
    +----------------------+-----------------------------------------------------+
    | select_type          | Art der Abfrage                                     |
    |   SIMPLE             | Einfach (kein UNION oder Subquery)                  |
    |   PRIMARY            | Äußerstes SELECT                                    |
    |   UNION              | Zweites/späteres SELECT in UNION                    |
    |   DEPENDENT UNION    | Analog (abhängig von äußerer Query)                 |
    |   UNION RESULT       | Ergebnis einer UNION                                |
    |   SUBQUERY           | Erstes Select einer Subquery                        |
    |   DEPENDENT SUBQUERY | Zweites/späteres SELECT in SUBQUERY                 |
    |   DERIVED            | Abgeleitetes SELECT (in FROM)                       |
    |   UNCACHABLE SUBQUERY| Für jede Zl. der äußeren Query neu zu generieren    |
    |   UNCACHABLE UNION   | Für jede Zl. der äußeren Query neu zu generieren    |
    +------------------+---+-----------------------------------------------------+
    | table            | Name der gelesenen Tabelle                              |
    +------------------+---------------------------------------------------------+
    | type             | Verknüpfung der Tabelle (Join, optimal --> schlecht)    |
    | : system (GOOD)  | Systemtabelle mit 1 Satz (Spezialfall von const)        |
    | : const          | Tabelle mit max. 1 pass. Satz (z.B. PRIMARY KEY)        |
    | : eq_ref         | 1 Satz pro Komb. vorh. Tab. (PRIMARY KEY, UNIQUE)       |
    | : ref            | Alle Sätze mit pass. Index (nicht eindeut. = <=>)       |
    | : fulltext       | FULLTEXT Index                                          |
    | : ref_or_null    | Analog "ref" + zusätzliche Suche nach NULL (IS NULL)    |
    | : index_merge    | Index Merge Optimierung benutzt (ab MY!5.0)             |
    | : unique_subquery| Bei "... IN (Subquery)" (Unique Index)                  |
    | : index_subquery | Bei "... IN (Subquery)" (Non-Unique Index)              |
    | : range          | Alle Sätze im Bereich mit pass. Index (nicht eind.)     |
    | :                | (= <=> <> > >= < <= IS NULL <=> BETWEEN IN)             |
    | : index          | Analog "all", aber nur Index lesen (etwas besser!)      |
    | v all (BAD)      | Full Table Scan (komplette Tab. 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 vermutl. gefilt. Zeilen (EXPLAIN EXTENDED)  |
    +-----------------+---------------------------------------------------------+
    | Extra           | Zusatzinfo                                              |
    |   Distinct      | 1. passende Zeile genügt, restl. ign. (DISTINCT)        |
    |   ...           | ...                                                     |
    |   Using filesort| 2 Durchläufe notwendig (schlecht!)                      |
    |   Using temporary| Temporäre Tab. notwendig (GROUP BY + ORDER BY)         |
    |   Using index   | IndexSCAN!                                              |
    +-----------------+---------------------------------------------------------+
```

Weitere Optimizer-Einstellungen (seit MySQL!5.6):

```
  +-----------------------------+-----------------------------------------+
  | Server-Konf.-Variable       | Bedeutung/Werte                         |
  +-----------------------------+-----------------------------------------+
  | optimizer_switch            | "index_merge=on,                        |
  |                             |  index_merge_union=on,                  |
  |                             |  index_merge_sort_union=on,             |
  |                             |  index_merge_intersection=on,           |
  |                             |  engine_condition_pushdown=on,          |
  |                             |  index_condition_pushdown=on,           |
  |                             |  mrr=on,                                |
  |                             |  mrr_cost_based=on,                     |
  |                             |  block_nested_loop=on,                  |
  |                             |  batched_key_access=off,                |
  |                             |  materialization=on,                    |
  |                             |  semijoin=on,                           |
  |                             |  loosescan=on,                          |
  |                             |  firstmatch=on,                         |
  |                             |  subquery_materialization_cost_based=on,|
  |                             |  use_index_extensions=on"               |
  +-----------------------------+-----------------------------------------+
  | optimizer_trace             | "enabled=off,                           |
  |                             |  one_line=off"                          |
  | optimizer_trace_features    | "greedy_search=on,                      |
  |                             |  range_optimizer=on,                    |
  |                             |  dynamic_range=on,                      |
  |                             |  repeated_subselect=on"                 |
  | optimizer_trace_limit       | 1                                       |
  | optimizer_trace_max_mem_size| 16384                                   |
  | optimizer_trace_offset      | -1                                      |
  +-----------------------------+-----------------------------------------+
```

17) Tabellen-, Index- und Query-Cache

Cache für geöffnete Dateien (OS-abhängige Obergrenze pro Prozess/Session):

```
  table_open_cache       = <N>   # Verb. Max. Anzahl Tab. pro Join (STD: 64)
  table_definition_cache = <N>   # Anz. gecachter Tabellendefinitionen
  max_connections                #
  max_tmp_tables                 #
  --open-files-limit             #
  FLUSH TABLES;                  # --> Offene Tabellen werden geschlossen
  open_tables                    #
  opened_tables                  #
  flush_commands                 #
```

Indexstatistik:

```
  * Basiert auf "Valuegroups" = Anz. Datensätze mit gleichem Index
  * Durchschnittliche Größe der Valuegroups
  * Je kleiner desto besser
  * Tabellenkardinalität = Anzahl Valuegroups
    N/S (N=Anzahl Datensätze, 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>
  * Ausgabespalten:
     +--------------+-------------------------------------------------------+
```

```
    | Spalte       | Bedeutung                                                  |
    +--------------+------------------------------------------------------------+
    | 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    | Sortierregel/(reihen)folge (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
      myisam_stats_method = "nulls_equal"   (alle NULL-Werte gleich)      -->
Valuegroup-Größe überschätzt
      myisam_stats_method = "nulls_unequal" (alle NULL-Werte verschieden) -->
Valuegroup-Größe unterschätzt
      myisamchk --stats_method=... --analyze
      ANALYZE TABLE <Tbl>;    # Key-Verteilung ermitteln (Statistik)
      CHECK TABLE <Tbl>;      # Fehler erkennen
      CHECKSUM TABLE <Tbl>;   # Prüfsumme bilden
      OPTIMIZE TABLE <Tbl>;   # Defragmentierung + Leerraum freigeben
                              # (CHECK + ANALYZE + REPAIR)
      REPAIR TABLE <Tbl>      # Fehler beheben

Keycache (MyISAM):
  * Systemvariable "key_buffer_size" (0=kein) --> Default Keycache (nicht löschbar)
  * Mehrere möglich + Zuordnung von Indices zu best. Keycache möglich
    (bei jedem Serverneustart verloren):
      CACHE INDEX <Tbl1>, ... IN <Cache>;
      SET GLOBAL <Cache>.key_buffer_size = 128 * 1024;  # Anlegen
      SET GLOBAL <Cache>.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 Keycaches für hohe Last (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 Keycaches 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.key_buffer_size = 4 * 1024 * 1024;
  * Key-Hit-Ration ermitteln (möglichst nahe bei 100%):
      SHOW GLOBAL STATUS LIKE "key_read%";
      100 * (key_read_requests - key_reads) / key_read_requests

Query-Cache (Abfrage-Speicher):
  * NUR SELECT-Anweisung + Ergebnis-Menge darin gespeichert
    + Muss deterministisch sein (NEW(), UUID(), RAND(), CONNECTION_ID() --> NEIN)
```

```
    + Falls exakt identisches SELECT (inkl. GROSS/kleinschreibung, Leerzeichen)
      erneut auftritt, sofort Ergebnis zurückgegeben (Parsen vermieden!)
    + Partitionierte Tabellen --> NICHT verwendet (MY!5.6.5)
    + Cursor                  --> NICHT verwendet
    + Prepared Statements     --> verwendet (seit MY!5.1)
  * Sinnvoll bei Tabellen mit vielen gleichen Abfragen und wenig Änderungen
    + INSERT/UPDATE/DELETE/REPLACE-Änderung auf Tabellen
      --> Davon abhängige Query-Cache Einträge sofort gelöscht
    + Bis zu 250% schneller, falls gleiche Anfrage oft kommt
    + Etwa 10% Overhead, falls nie gleiche Anfrage kommt
  * Wird von allen Sitzungen geteilt (pro Server)
    + Findet vor dem Parsen statt
    + Benutzer-Rechte werden überprüft
  * Sinnvolle Größe:
    + 32-128 MByte
    + Fragmentiert im Laufe der Zeit stark (zersplittert)
    + Lohnt sich nur, wenn sich wirklich viele Anweisungen wiederholen
  * Systemvariable:
      maximum_query_cache_size = 0    # STD: Max. einstellbare Größe
      query_cache_size = 0            # STD: Overhead vollständig weg
      query_cache_type = 0/1/2        # 0/OFF    = Aus
                                      # 1/ON     = Bei SELECT SQL_NO_CACHE... (STD)
                                      # 2/DEMAND = Bei SELECT 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 <Tbl> ...;      # In Query-Cache
      SELECT SQL_NO_CACHE * FROM <Tbl> ...;   # Nicht in Query-Cache
  * 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_cache_%";

TIP: Performance Tuning Key for MySQL (Schritt-für-Schritt-Anleitung)
     --> http://www.fromdual.com/mysql-performance-tuning-key


18) Partitionierung
-------------------

Teilt Tabellen + Indices PHYSIKALISCH gemäß definierbarer Regel in Stücke auf
  * EINE logische Tabelle besteht aus MEHREREN physikalischen Untertabellen
    + In Wirklichkeit nur "Wrapper"
    + Indices pro Subtabelle, kein Gesamtindex
  * Annahmen:
    + Partitionierung ist "billig"
    + Pruning beim Zugriff ist möglich (z.B. aufgrund WHERE-Filter)
  * Einsatzgebiete:
    + Riesige Tabellen "splitten"
    + Historische Daten "rollieren"
    + Tabelle physikalisch auf mehrere Platten verteilen
    + Backup/Restore "stückweise" ermöglichen
    + Aufwand für folgende Operationen geringer:
      - Alte Daten verwerfen
      - Index neu aufbauen
      - Zugriff auf Tabellenteile
  * Horizontale Zerlegung der Daten (Datensätze)
    + Vertikale Zerlegung (Datenspalten) NICHT möglich (manuell selber machen)
      - NoSQL-Datenbanken haben diese Struktur!
      - Pro Spalte eine physikalische Tabelle
      - Stark komprimierbar (wg. gleicher Werte)
      - Zugriff erfordert viele Joins
    + Anderer Name: Berechneter Tablespace (InnoDB)
    + Für Anwender/Anwendung transparent, d.h. merkt nichts davon
    + Insbesondere DATE/TIME/DATETIME-Spalten als Zerlegungskriterium sinnvoll
      (bzw. Funktionen davon: TO_DAYS, YEAR, WEEKDAY, DAYOFYEAR, MONTH)
  * Arbeitsweise
    + Partitionen IMMER von 0..N nummeriert
    + Pro Datensatz Nummer ermittelt und für Speicherung in Partition benutzt
```

```
    + Partitionierungs-Funktion REGEL(BASIS) MUSS ganzzahligen Wert 0..N ergeben
      - REGEL: Wertebereiche: RANGE [COLUMNS]
               Wertelisten:   LIST [COLUMNS]
               Hash-Funktion: [LINEAR] HASH/KEY
               Kombiniert:    COMPOSITE (erst RANGE/LIST, dann HASH/KEY)
      - BASIS: Eine/mehrere Integer-Spalten (auch DATE/TIME/DATETIME!)
               Funktion auf Spalte(n) mit Integer-Ergebnis (NULL oder positiv!)
  * Für viele Storage Engines erlaubt
    + Partitionen einer Tabelle haben gleiche Engine
    + Nicht möglich: MERGE, CSV, FEDERATED
    + Teilweise:     NDB(CLUSTER)               # Nur [LINEAR] KEY
  * Pro Partition Daten + Indices auf versch. Verz. verteilbar (nicht InnoDB)
      CREATE TABLE ...
        PARTITION BY ... DATA DIRECTORY <DirPath>
                         INDEX DIRECTORY <DirPath>
  * Partition-Pruning: WHERE-Bedingung schließt Partitionen sofort von Abfrage
                       aus und minimiert somit zu verarbeitende Daten bei:
      WHERE <PartCol> = <Const>                # Auch < <= > >= <>
      WHERE <PartCol> IN (<Const>, ...)
      WHERE <PartCol> BETWEEN <LeftVal> AND <RightVal>
  * Sonstiges
    + Bei Partitionsnamen wird GROSS/kleinschreibung ignoriert
    + Partitionen jederzeit reorganisierbar (Datensätze umverteilen)
    + Partitionierungs-Änderungen setzen WRITE LOCK auf gesamte Tabelle
    + Konfigurationsvariable "--open_files_limit" evtl. hochsetzen


Syntax:

  CREATE TABLE ... [<Part>];

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

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

  <SubPartDef> = SUBPARTITION <SubPart>
                 {<PartOpt>, ...}

Zweck:
  * Grobe Indizierung mit wenig Overhead und Datenzusammenfassung
    + Weglassen großer Tabellenteile
    + Zusammengehörenden Daten nahe beinander speichern
  * Plattenplatz/Dateisystem-Beschränkungen umgehen (riesige Tabellen)
  * Abfragen stark beschleunigen (WHERE-Bedingung schließt Partitionen aus)
    + Datum/Zeitreihen
    + Geografische Unterteilung
    + Standorte, Abteilungen, PLZ, Telefon-Vorwahl, ...
  * Nur Locking ausgewählter Partitionen, nicht gesamte Tabelle
    ("Partition Lock Pruning", noch nicht in MY!5.5, erst seit MY!5.6.6)
  * (Un)Interessant gewordene Daten leicht entfernbar/hinzufügbar (Datum!)
  * Abfragen über mehrere Threads verteilbar (kann MySQL nicht!)
  * Abfragen mit Aggregatfunk. COUNT(), ... parallelisierbar (kann MySQL nicht!)
  * Verteilung auf mehrere Platten möglich --> kann Zugriff beschleunigen
  * Plattenschaden betrifft nur eine Partition, Restl. Partitionen noch OK
  * Zugriff direkt auf Partitionen statt Tabelle möglich (ab MY!5.6)
```

  * Vermeiden von "dynamischem SQL" für Zerlegen von Tabellen

Einschränkungen:
  * Max. 1024 Partitionen (inkl. Subpartitionen) pro Tabelle
    (ab MY!5.6.7 max. 8192 Partitionen pro Tabelle)
  * Partitionen nicht einzeln ansprechbar (Partition-Selection)
    (ab MY!5.6 einzeln oder zusammengefasst statt Gesamttabelle ansprechbar)
  * Alle Partitionen einer Tabelle haben gleiche Engine
  * Keine "Foreign Keys" (weder bei sich noch in anderen Tabellen)
  * Kein Fulltext Index
  * 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 muss
    alle Spalten des Partitionierungs-Ausdrucks enthalten, AUSSER eine Tabelle
    enthält keine UNIQUE-Keys, dann ist jede Spalte erlaubt.
  * Partitioning-Ausdruck:
    + Muss deterministisch sein
    + Muss Ganzzahl ergeben (MY!5.1)
    + Kann auf Spalten basieren (MY!5.5)

Zerlegungsregeln:

Regel	Bedeutung
RANGE	Basis: Integer-Bereich (z.B. 1980-1989, 1990-1999, ...) (ohne Lücken, nicht überlappend, aufsteigend) ELSE-Zweig: PARTITION <Part> VALUES LESS THAN MAXVALUE
RANGE COLUMNS	Erweiterung von RANGE, mehrere Spalten erlaubt, aber keine Funktion auf den Spalten erlaubt (MY!5.5)
LIST	Basis: Integer-Liste (z.B. 2006, 2007, 2008, 2009, ...) (ohne Lücken, nicht überlappend, aufsteigend unnötig) ALLE Werte abdecken nötig, ELSE-Zweig nicht möglich
LIST COLUMNS	Erweiterung von LIST, mehrere Spalten erlaubt, aber keine Funktion auf den Spalten erlaubt (MY!5.5)
HASH	Basis: Ben.def. Spalten-Ausdruck --> Integer + Anz. Part. Verwendet Divisonsrest MODULO Anz. Partitionen, Berechnung kostet Zeit Ausdruck sollte "linear" sein (sonst ungleich verteilt)
KEY	Analog HASH, aber Primary Key/UNIQUE-Index + Hash MD5() (Gleichverteilung durch DB, auch CHAR statt INT erlaubt)
LINEAR HASH LINEAR KEY	Power-of-2 Algorithmus statt Modulo --> schneller Power-of-2 Algorithmus statt Modulo --> schneller
COMPOSITE	Unterzerlegung (Subpartitioning) in weitere Partitionen RANGE/LIST weiter zerlegt durch HASH/KEY Jede Partition muss gleiche Anzahl Unterpartitionen haben Max. ist 2-stufige Zerlegung möglich

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: (analog HASH)
  * COMPOSITE: Sehr große Datenmengen (2-fach unterteilen)

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

Wird Partitionierung vom MySQL-Server unterstützt?
  * Beim Compilieren einschalten: -DDWITH_PARTITION_STORAGE_ENGINE
  * Beim Server-Start ausschalten: --skip-partition
  * Abfragen:

```
SHOW PLUGINS;                       # --> partition ACTIVE STORAGE ENGINE NULL GPL
```

```
  SHOW VARIABLES LIKE "%partition%"   # --> have_partitioning YES (bis MY!5.6.1)
```

Beispiele:

```
  # Nachträgliche Zerlegung nach DATE-Spalte "seit" in Jahresbereiche NNNN-MMMM
  DROP TABLE IF EXISTS members;
  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    # Partitionierungskriterium
  );

  ALTER TABLE members
    PARTITION BY RANGE(YEAR(seit)) (          # 7 Partitionen:
      PARTITION p0 VALUES LESS THAN (1960),   #    0..1959
      PARTITION p1 VALUES LESS THAN (1970),   # 1960..1969
      PARTITION p2 VALUES LESS THAN (1980),   # 1970..1969
      PARTITION p3 VALUES LESS THAN (1990),   # 1980..1989
      PARTITION p4 VALUES LESS THAN (2000),   # 1990..1999
      PARTITION p5 VALUES LESS THAN (2050),   # 2001..2049
      PARTITION p6 VALUES LESS THAN MAXVALUE  # 2050..     (Rest = ELSE-Zweig)
    );

  # Zerlegung nach 3 Spalten gleichzeitig in mehrere Bereiche
  CREATE TABLE t1 (
    a INT,
    b INT,
    c CHAR(3),
    d INT
  )
  PARTITION BY RANGE COLUMNS(a,d,c) (             # 4 Partitionen:
    PARTITION p0 VALUES LESS THAN ( 5, 10, 'g'),  # 0,0,'' ..  4,  9, 'f'
    PARTITION p1 VALUES LESS THAN (10, 20, 'm'),  #        ..  9, 19, 'l'
    PARTITION p2 VALUES LESS THAN (15, 30, 's'),  #        .. 14, 29, 'r'
    PARTITION p3 VALUES LESS THAN (MAXVALUE, MAXVALUE, MAXVALUE)  # Rest = ELSE
  );

  # Zerlegung nach INT-Wert "val" in als Liste aufgezählte Bereiche (mit NULL)
  CREATE TABLE t2 (
    val INT
  )
  PARTITION BY LIST(val) (             # 3 Part., ALLE mögl. 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)
  );

  # Zerlegung nach INT-Wert "store" in als Liste aufgezählte Bereiche (ohne NULL)
  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          NOT NULL
  )
  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)
  );

  # Zerlegung nach 2 Spalten gleichzeitig in als Liste aufgezählte Bereiche (ohne NULL)
  CREATE TABLE kunden (
    first_name VARCHAR(25),
    last_name  VARCHAR(25),
    street_1   VARCHAR(30),
    street_2   VARCHAR(30),
    city       VARCHAR(15),
    renewal DATE
  )
  PARTITION BY LIST COLUMNS(first_name, last_name) (
```

```
    PARTITION pRegion_1 VALUES IN('C', 'C'),
    PARTITION pRegion_2 VALUES IN('F', 'F'),
    PARTITION pRegion_3 VALUES IN('M', 'M'),
    PARTITION pRegion_4 VALUES IN('Z', 'Z'),
  );

  # Zerlegung nach DATE-Spalte "tr_date" in 6 Bereiche per HASH auf Monat(DATE)
  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)

  # Zerlegung nach DATE-Spalte "tr_date" in 10 Bereiche per HASH auf DATE
  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)

  # Zerlegung nach YEAR-Spalte in mehrere Jahresbereiche NNNN-MMMM,
  # Unterzerlegung jedes Jahresbereichs in 2 Unterpartitionen nach Tag
  CREATE TABLE tr (
    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
    )
  );
```

Zugriff auf Partitionen ist für folgende SQL-Anweisungen möglich (ab MY!5.6):

```
    SELECT * FROM t1 PARTITION (p0);
    DELETE FROM t2 PARTITION (p1, p2, p3) WHERE name LIKE "T%";
    INSERT INTO t3 PARTITION (p2) (name, vorname, alter)
                          VALUES ("Tom", "Cat", 43);
    REPLACE INTO t4 PARTITION (p1) (nr, vorname, name)
      VALUES (1, "Thomas", "Birnthaler"),
             (2, "Hans",   "Dampf");
    UPDATE
    UPDATE user PARTITION (p3, p8) SET alter = alter+1 WHERE name LIKE "T%";
    LOAD DATA INFILE "/tmp/test.csv" REPLACE
              INTO TABLE t5 PARTITION (p8);
    LOAD XML ...
```

Daten in Partitionen löschen:

```
  TRUNCATE TABLE t;                        # Struktur+Partitionen bleiben, Daten löschen
  ALTER TABLE t DROP PARTITION p0;         # Part. löschen (inkl. Daten, effizient)
  ALTER TABLE t TRUNCATE PARTITION p2;     # Partition löschen (MY!5.5)
  ALTER TABLE t TRUNCATE PARTITION ALL;    # Alle Partitionen löschen (MY!5.5)
```

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
```

LIST + RANGE Partitionen mergen/splitten (REORGANIZE):

```
  # 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.
```

Partitionstyp von RANGE auf KEY ändern (geht nicht per REORGANIZE):

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

Partitionen optimieren und pflegen (ab MY!5.6):

```
  ALTER TABLE t ANALYZE PARTITION p2;     # Key-Verteilung ermitteln (Statistik)
  ALTER TABLE t CHECK PARTITION p3;       # Fehler erkennen
  ALTER TABLE t OPTIMIZE PARTITION p1;    # Defragmentierung + Leerraum freigeb.
                                          # (CHECK + ANALYZE + REPAIR)
  ALTER TABLE t REPAIR PARTITION p3;      # Fehler beheben
```

Partitionen einer Tabelle auflisten:

```
  SHOW CREATE TABLE t1;
  SHOW TABLE STATUS t1;                    #
```

Nutzung von Partitionen in einer SQL-Anweisung erklären lassen:

```
  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 zu Partition
  DATA DIRECTORY [=] <DirPath>       # Abs. Pfad für Datenablage der Partition
  INDEX DIRECTORY [=] <DirPath>      # Abs. Pfad für Indexablage der Partition
  MAX_ROWS [=] <N>                   # Max. Anzahl Datensätze (nur Hinweis!)
  MIN_ROWS [=] <N>                   # Min. Anzahl Datensätze (nur Hinweis!)
  TABLESPACE [=] (<TblSp>)           #
  NODEGROUP [=] <N>                  #
```

19) MySQL Hilfsprogramme

19a) MySQL Utilities

MySQL bietet ab MY!5.5 die sogenannten "MySQL Utilities" an, die in Python programmiert sind und eine Reihe von MySQL-Zusatzfunktionalitäten auf der Kommandozeile nachrüsten (Kmdo-Präfix "mysql..." wird jeweils weggelassen):

Kommando	Bedeutung
uc	Utilities Commandline (Kommandozeilen-Client)
serverinfo	Informationen zu Server anzeigen
diskusage	Plattenplatzverbrauch von Datenbanken ermitteln
frm	CREATE TABLE Anweisung aus "*.frm"-Datei generieren
dbcompare	Zwei Datenbanken vergleichen auf Konsistenz
diff	Zwei Datenbank-Objekte db1.o1 und db2.o2 vergleichen
serverclone	Weitere Instanz eines laufenden Servers starten
dbcopy	Datenbank kopieren (Server A --> Server B)
dbexport	Datenbank exportieren (Daten + Metadaten)
dbimport	Datenbank importieren (Daten + Metadaten)
userclone	Benutzer inkl. Rechte mit neuem Passwort kopieren
indexcheck	Auf doppelte/redundante Indices prüfen
metagrep	Nach Metadaten suchen
procgrep	Nach Prozessinformationen suchen
failover	Replikation ständig monitoren + Failover
replicate	Replikation mit einem Master einrichten
rpladmin	Replikation Administrationstool
rplcheck	Replikation überprüfen
rplms	Multi-Source Replikation einrichten
rplshow	An einen Master angebundenen Slaves anzeigen
rplsync	Replikation Synchronisationsstatus überprüfen
auditadmin	Audit Log verwalten (nur kommerzielle Version)
auditgrep	Audit Log durchsuchen (nur kommerzielle Version)

Die Anmeldung der MySQL Utilities an Datenbanken erfolgt über folgende Syntax:

```
USER[:PASSWORD]@HOST[:PORT][:SOCKET]
```

Mit den Utilities ist es möglich, zwei verschiedene Datenbanken eines MySQL Servers oder je eine Datenbank zweier verschiedener MySQL-Server anzusprechen.

Im Kommandozeilen-Client "mysqluc" kann der Präfix "mysql" der Utilities weggelassen werden, auf der Kommandozeile ist es anzugeben. Eine allgemeine Hilfe bzw. eine Liste der Utilities bzw. Hilfe zu einem Utility erhält man so:

```
mysqluc> help              # Allgemeine Hilfe
mysqluc> help utilities    # Liste aller Utilities
mysqluc> help userclone    # Utility zum Benutzer inkl. Rechte kopieren
```

Beispiel für das Kopieren eines Benutzer mit neuem Passwort:

```
# Direkt auf der Kommandozeile
mysqluserclone --source=root:geheim@localhost \   # Quell-Server + Anmeldung
               --source=root:geheim@localhost \   # Ziel-Server + Anmeldung
               --force \                          # Neuen Ben. überschreiben
               root@127.0.0.1 \                   # Zu kopierender Benutzer
               new:secret@%                       # Neuer Ben. mit Passwort
# Im Kommandozeilen-Client "mysqluc"
mysqluc> userclone --source=root:geheim@localhost ...   # Parameter analog
```

19b) Percona Toolkit

Die Firma "Percona" bietet das sogenannte "Percona Toolkit" (früher "Maatkit") an, das in Perl programmiert ist und MySQL-Zusatzfunktionalitäten auf der Kommandozeile nachrüstet (Kmdo-Präfix "pt-..." wird jeweils weggelassen):

Tool "pt-..."	Bedeutung
align	Align output from other tools to columns
archiver	Archive table rows into another table or a file
config-diff	Diff config files and server variables
deadlock-logger	Extract and log deadlock information
diskstats	Interactive I/O monitoring tool for GNU/Linux
duplicate-key-checker	Find dup. indexes and foreign keys on tables
fifo-split	Split files and pipe lines to a fifo without really splitting
find	Find tables and execute actions, like GNU find
fingerprint	Convert queries into fingerprints
fk-error-logger	Extract and log foreign key errors
heartbeat	Monitor replication delay
index-usage	Read queries from log and analyze index usage
ioprofile	Watch process IO and print file and I/O activity
kill	Kill queries that match certain criteria
log-player	Replay query logs
mext	Compare "SHOW GLOBAL STATUS" lists side-by-side
mysql-summary	Summarize MySQL information nicely
online-schema-change	ALTER tables without locking them
pmp	Aggregate GDB stack traces for a selected program
query-advisor	Analyze queries and advise on possible problems
query-digest	Analyze query execution logs and generate a query report, filter, replay, or transform queries for MySQL, PostgreSQL, memcached, and more
show-grants	Canonicalize + print grants to effectively replicate, compare and version-control them
sift	Browses files created by pt-collect
slave-delay	Make a slave server lag behind its master
slave-find	Find+print replication hierarchy tree of slaves
slave-restart	Watch and restart replication after errors
stalk	Gather forensic data about when a problem occurs
summary	Summarize system information nicely
table-checksum	Verify replication integrity
table-sync	Synchronize table data efficiently
table-usage	Analyze how queries use tables
tcp-model	Transform tcpdump into metrics that permit performance and scalability modeling
trend	Compute statistics on time-series data points set
upgrade	Execute queries on multiple servers and check for differences
variable-advisor	Analyze variables and advise on possible problems
visual-explain	Format EXPLAIN output as a tree

Konfigurations-Dateien des Percona Toolkit:

1. /etc/percona-toolkit/percona-toolkit.conf
2. /etc/percona-toolkit/TOOL.conf
3. $HOME/.percona-toolkit.conf.
4. $HOME/.TOOL.conf

Die Percona DSN (Data Source Name) Spezifikation zum Erstellen einer Datenbank-Verbindung ist ein kommagetrennte Liste von "key=value" Paaren, z.B.:

```
h=host1,P=3306,u=bob   # Server, Port, User
```

Einige der Tools unterstützen auch die Standard MySQL-Optionen wie "--host", "--user" und "--password". Sie erzeugen im Hintergrund automatisch eine DSN.

Einige der Tools unterstützen sowohl DSN als auch die Standard MySQL-Optionen. Die DSN wird automatisch um die Einstellungen der MySQL-Optionen erweitert, wenn der entsprechende Teil in der DSN fehlt.

20) MySQL Cluster

MySQL Cluster beruht auf der NDB (Network Database) von Ericsson, die ursprünglich für Anwendungen im Bereich der Telekommunikation

entwickelt wurde (Verbindungsdaten = viele Schreiboperationen,
geringe Datenmenge die vollständig in den Hauptspeicher passt,
Daten sind kurzlebig). Für einen sinnvollen Einsatz von MySQL
Cluster sollten daher folgende Punkte gegeben sein:

  * Daten + Indices müssen vollständig in den Hauptspeicher (RAM)
    passen (geteilt durch Anzahl Knotengruppen)
  * Viele Schreiboperationen sind OK
  * Wenig Bereichsabfragen BETWEEN ... AND ..., sondern Abfrage auf
    Einzelwerte per "==" oder "IN" (da Index-Hash-Verfahren genutzt)
  * Wenig Sortieroperationen auf großen Datenmengen (Grund: analog)
  * Alle Daten müssen einen Primärschlüssel PK (Primary Key) haben
  * Referenzielle Integrität ("Foreign Keys") irrelevant
  * Daten sind in Zeilen fester Länge gespeichert (VARCHAR --> CHAR)
  * Die maximale Datensatzlänge beträgt 8052 Byte
  * Anzahl Datenobjekte <= 20320 (Anzahl Datenbanken + Tabellen +
    Spalten + Indices + Views + Triggers + Stored Procedures + ...)
  * Löschen von Datensätzen gibt Speicher nur pro Tabelle zurück,
    tabellenübergreifende Neunutzung erfordert "Rolling Restart"
  * Wenig grosse Transaktionen (viele kleine TA sind OK)
  * Schreiben auf Platte erfolgt per REDO LOG und Checkpoints
  * Backups sind online und nicht-blockierend
  * Partitionierung nur per HASH/KEY (RANGE/LIST nicht sinnvoll)

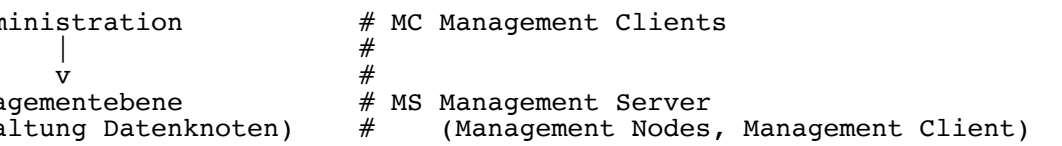
MySQL selbst benutzt NDB als Engine ("NDB" oder "NDBCLUSTER"), das Clustering
wird automatisch durch die NDB-Daemonen ("ndb_mgmd" + "ndbd") realisiert.

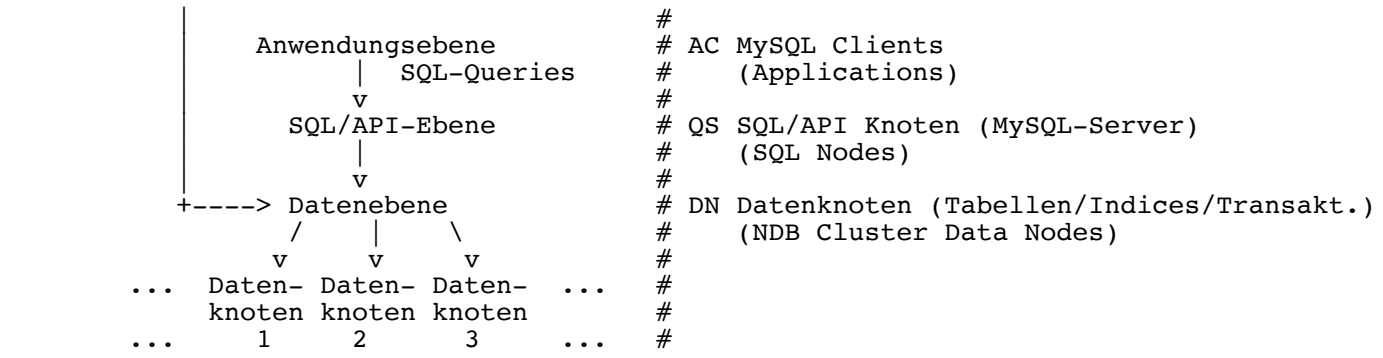
Eigenschaften:
  * Bietet 99.999% ("Five Nines") Verfügbarkeit --> Ausfall 5 Minuten pro Jahr
  * Über mehrere Rechner verteilte Datenbanken ("distributed")
  * "Shared-nothing" Architektur (statt Shared-Disk wie die meisten anderen)
    --> Einfachere Hardware verwendbar --> günstiger
    --> Höherer Aufwand für HA und Synchronisation (Netzwerk-Last)
  * ACID Transaktionen
    + Max. Isolationslevel: READ-COMMITTED
  * Row-level locking
  * Eigentlich eine "In-Memory" Datenbank
    + Nichtindizierte Daten können aber auch auf Platte liegen
    + Plattendaten für Checkpoints (reduzierter Sync.aufwand beim Hochfahren)
  * Index-Typen: Unique Hash Index + Ordered T-Tree
  * Online Backup eingebaut
  * ALTER TABLE Anweisungen auch mit ungelockten Tabellen möglich
    (die meisten anderen Engines brauchen Table Locks)
  * Two-Phase Commit (2PC) + Transaktionen (TA)

Vorteile:
  * Auto-sharding for Write-scalability: Partitioniert Tabellen
    automatisch über die Datenknoten (horizontale Skalierung).
  * Real-time Performance
    Replikation über geografisch verteilte Systeme:
    + Disaster Recovery
    + Skalierbar
  * Datenknoten und Schema-Änderungen sind online einfügbar
  * GUI-Tool "MySQL Cluster Manager" verfügbar

Nachteile:
  * Jede Tabelle muss einen Primärschlüssel PK (Primary Key) haben
  * AUTO_INCREMENT-Spalte muss Primary Key sein
  * Referenzielle Integrität nicht unterstützt
    ("Foreign Keys" ignoriert)
  * Datenbank befindet sich vollständig im Speicher
    ("in-memory" Tablespaces ab MY!5.1)
  * Temporäre Tabellen sind nicht erlaubt
  * Volltext-Suche ist nicht erlaubt
  * Transaktionen sollten möglichst klein sein
  * Transaktions-Isolationslevel max. READ COMMITTED (STD. REPEATABLE READ nicht!)
  * INSERT IGNORE/UPDATE IGNORE/REPLACE nur für Primärschlüssel erlaubt,
    nicht für Unique Keys (bis MY!5.0.19)

Architektur:





Datenpartitionierung ("sharded" = verteilt):
  * Jede Tabelle wird in DN Partitionen zerlegt (etwa gleichverteilt
    per Hash auf dem PK) und auf alle DN Datenknoten verteilt
  * Jeder Datenknoten enthält ein "Primary Replica" einer Partition (Fragment)
    und eine KOPIE einer anderen Partition namens "Secondary Replica"
  * Die Knoten gehören paarweise zusammen, je 2 Knoten bilden eine
    "Node Group" (Knotengruppe)
    --> Bei 2 Knoten enthält jeder Knoten die gesamten Daten
    --> Bei 4 Knoten enthält jeder Knoten die Hälfte der Daten
    --> Bei 6 Knoten enthält jeder Knoten ein Drittel der Daten
    --> Bei 8 Knoten enthält jeder Knoten ein Viertel der Daten
  * MySQL-Cluster fasst die Knoten anhand ihrer Nummer aufsteigend
    automatisch zu diese "Knotengruppen" zusammen (triv. Beispiel):
      1,2 --> Knotengruppe 1
      3,4 --> Knotengruppe 2
      5,6 --> Knotengruppe 3
      ... --> ...

Datenknoten-Fehler erkennen:
  * Alle Knoten müssen immer wissen, welche Knoten im Cluster vorhanden sind
  * Alle Knoten bilden einen (logischen) Ring,
    jeder Knoten sendet an seinen Nachfolger "Heartbeats" (Ping)
  * Erhält ein Knoten 3x nicht den Heartbeat seines Vorgängers, dann:
    + Leitet er ein Network-Partitioning-Protokoll ein
    + Wird die Arbeit des Clusters komplett gestoppt und jeder Knoten
      entscheidet für sich, ob er weitermachen kann oder nicht
    + Der nicht antwortende Knoten wird aus dem Ring ausgeschlossen
    + Die restlichen Knoten bauen einen neuen Ring auf
    + Der Partner des verschwundenen Knotens enthält nun 2 "Primary Replica"
      (seine "Secondary Replica" wird zur "Primary Replica") --> doppelte Last

"Split-Brain" vermeiden:
  * Problem: Netzwerkfehler NICHT unterscheidbar von Knotenfehler
    + Netzwerkfehler = Funktionsfähige Knoten-Teilmengen erreichen sich nicht
    + Knotenfehler   = Funktionsunfähige Knoten-Teilmenge nicht erreichbar
  * Netzwerkfehler vorhanden --> Ring zerfällt in 2 Teilmengen (oder mehr)
  * Jede Knoten-Teilmenge trifft nach folgenden Regeln eine Entscheidung,
    so dass garantiert nur EIN laufender Cluster übrigbleiben kann
    + Mind. ein Knoten jeder Gruppe in Teilmenge vorhanden
      --> Shutdown wenn nicht
    + Alle Knoten jeder Gruppe vorhanden (kurzeitiges Problem)
      --> Wir machen als (vollständiger) Cluster weiter
    + Jede prinzipiell funktionsfähige Teilmenge fragt "Schiedsrichter" an
      --> Shutdown wenn nicht erreichbar
    + Der "Schiedsrichter" entscheidet, welche Teilmenge als Cluster arbeitet
      - Einfache Regel: Die erste anfragende Teilmenge --> JA
                        Alle anderen Teilmengen   --> NEIN

Dauerhaftigkeit der Daten sichern ("Durability"):
  * In-memory Datenbank --> Cluster-Shutdown soll trotzdem möglich sein
  * REDO LOG in jedem Knoten: Transaktionen im REDO Log Buffer
    (periodisch Flush auf Platte) --> würde beliebig wachsen
  * Local Checkpoint (LCP) in jedem Knoten:
    --> Daten-Snapshot auf Platte speichern
    --> REDO LOG bis dahin verwerfen
  * Auf jedem Knoten befinden sich immer zwei LCPs
    --> Notwendig ist daher ein REDO LOG vom Start des ersten LPC bis zum
       Start des dritten LPC
  * Die LPCs werden im Kreis verwendet: 1 + 2, 3 --> 1, 2 --> 3 --> 2, 3 --> 1, ...
  * Global Checkpoints (GCP) aller Knoten
    --> Rebuild aller Knoten ganz alleine bis dorthin möglich
       Rest von anderen Knoten holen (--> Netzwerktraffic)

  * Shutdown --> GCP für jeden Knoten --> Rebuild vollständig alleine möglich

Transaktionen (TA) werden per Two-Phase Commit (2PC) durchgeführt:
  * SYNCHRONE Replikation zw. Datenknoten (aus Clientsicht)
  * 2PC-Protokoll --> TA = A) Prepare + B) Commit (2 Phasen)
  * Jeder Datenknoten enthält einen "Transaction Coordinator" (TC)
    + Client schickt TA --> TC
    A) Prepare-Phase (erst Primary, dann Secondary)
      - TC sucht DN mit Primary Partition für Datensatz
      - Dieser DN merkt vor --> Auf DN mit Secondary Partition gl. Op. auslösen
      - Antwort Secondary DN --> Primary DN wenn OK
      - Antwort Primary DN   --> TC wenn OK
    B) Commit-Phase (umgekehrt, erst Secondary, dann Primary)
      - TC sucht DN mit Secondary Partition für Datensatz
      - Dieser DN merkt vor --> Auf DN mit Primary Partition gl. Op. auslösen
      - Antwort Primary DN   --> Secondary DN wenn OK
      - Antwort Secondary DN --> TC wenn OK
    + TC schickt Bestätigung --> Client
    + Gesamt: 6 Nachrichten zw. Daten-Knoten DN und
              2 Nachrichten zw. Client und TC

Programme und Tools:

Programmm	Bedeutung
ndb_mgm	Management-Client
ndb_mgmd	Management-Daemon
ndbd	Datenknoten-Daemon
ndbmtd	Datenknoten-Daemon (multithreaded Version)
ndb_waiter	Wartet auf bestimmten Zustand eines NDB-Cluster
ndb_size.pl	NDB-Speicherplatz einer MySQL-Datenbank schätzen
ndb_config	Cluster-Konfiguration ausgeben
ndb_cpcd	Automatischer Test eines NDB-Clusters
ndb_delete_all	Daten einer NDB-Tabelle löschen
ndb_desc	NDB-Tabellen + -Datensätze ausgeben
ndb_drop_index	Index einer NDB-Tabelle löschen
ndb_drop_table	NDB-Tabelle löschen
ndb_error_reporter	NDB-Logdateien in Archiv für Fehlerbericht sammeln
ndb_index_stat	Indexstatus einer NDB-Tabelle ausgeben
ndb_print_backup_file	NDB-Backupdatei ausgeben (auf "ndbd")
ndb_print_file	
ndb_print_schema_file	NDB-Schema ausgeben (auf "ndbd")
ndb_print_sys_file	NDB-Systemdaten ausgeben (auf "ndbd")
ndb_redo_log_reader	
ndb_restore	Backup eines NDB-Clusters zurückspielen
ndb_select_all	Daten einer NDB-Tabelle ausgeben
ndb_select_count	Anzahl der Datensätze einer NDB-Tabelle ausgeben
ndb_show_tables	Liste der Tabellen eines NDB-Clusters ausgeben
ndbinfo_select_all	

Konfigurations-Dateien:

```
/etc/my.cnf
/var/lib/mysql/config.ini
```

MySQL-Cluster 7.3:
* Fremdschlüssel-Beziehungen definierbar
  + Vorgaben CASCADE, RESTRICT, NO ACTION und SET NULL
  + Zur Laufzeit ohne Betriebsunterbrechung
* Freier JavaScript-Server "node.js" integriert (experimentell)
* JSON-Datentyp

21) 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

  * Continuity during migration: Transform queries to support legacy application, migrate applications and database asynchronously without service disruption
  * Security: Block queries conditionally based on user and query syntax
  * Monitor Query Performance: Log and identify bottleneck queries
  * Cross technology processing: Query Duplication to another database, storage engine or application. Send data to transactional storage engine suchs as InnoDB as well as to NoSQL database

Eigenschaften:
  * Klassische "Middleware"-Komponente
  * Kann MySQL-Netzwerkprotokoll (Client und Server merken nichts vom Proxy)
  * Ermöglich Kommunikation zwischen mehreren Clients und mehreren Server
  * Eingebaute Scriptsprache "Lua" (8-()
    + Fertige Skripte für obige Zwecke erhältlich

==========