Allgemein

- Datenbank und Benutzer anlegen (veraltet)
- Benutzertabelle als GRANT-Statements dumpen
- Datenbank und Benutzer löschen
- InnoDB - ideale Logfile Größe herausfinden
- Größe einer DB, Tabelle, aller DBs herausfinden
- Passwort setzen oder ändern
- Passwort vergessen
- MySQL-Datenbank umbenennen
- mysqltuner - MySQL Performance Tuning Check-Script
- Schnell Platz schaffen
- Ungenutzte Datenbanken finden

Datenbank und Benutzer anlegen (veraltet)

Einen neuen Benutzer (User superadmin kommend von IP-Adresse 123.123.123.123) mit vollen Rechten auf alles erzeugen inkl. dem Recht wiederum neue User zu erzeugen (GRANT OPTION):

```
GRANT ALL PRIVILEGES ON *.* TO 'superadmin'@'123.123.123.123' IDENTIFIED BY 'password' WITH
GRANT OPTION;
FLUSH PRIVILEGES;
```

Einen neuen Benutzer (User user1 auf Datenbank userdb1 kommend von IP-Adresse 123.123.123.123) mit vollen Rechten auf seine eigene Datenbank erzeugen:

```
GRANT ALL PRIVILEGES ON userdb1.* TO 'user1'@'123.123.123.123' IDENTIFIED BY 'password';
CREATE DATABASE userdb1;
FLUSH PRIVILEGES;
```

Benutzertabelle als GRANT-Statements dumpen

> Eine einfacherere Alternative wäre das Tool `pt-show-grants` aus dem Percona-Toolkit.

Bei der Einrichtung einer frischen Replikation kann es wichtig sein, dass vorhandene User nicht überschrieben werden. Mit dem folgenden Snippet lassen sich die Benutzer einer MySQL-DB als einfache GRANT-Statements inkl. gesonderten Rechten auf Datenbanken ausgeben und für den späteren Import passend bearbeiten.

```
mygrants()
{
  mysql -B -N $@ -e "SELECT DISTINCT CONCAT(
    'SHOW GRANTS FOR \'', user, '\'@\'', host, '\';'
    ) AS query FROM mysql.user" | \
  mysql $@ | \
  sed 's/\(GRANT .*\)/\1;/;s/^\(Grants for .*\)/## \1 ##/;/##/{x;p;x;}'
}
```

Ein Aufruf könnte dann so aussehen:

```
mygrants --host=prod-db1 --user=admin --password=secret
```

Datenbank und Benutzer löschen

Einen Benutzer inkl. Datenbank löschen (ab MySQL 5.0.2):

```
DROP USER user1;
DROP DATABASE userdb1;
```

Einen Benutzer inkl. Datenbank löschen (pre MySQL 4.1.1):

```
REVOKE ALL PRIVILEGES ON 'userdb1'.* FROM 'user1'@'123.123.123.123';

# und evtl.
REVOKE GRANT OPTION ON 'userdb1'.* FROM 'user1'@'123.123.123.123';

DELETE FROM mysql.user WHERE Host='123.123.123.123' AND User='root';
FLUSH PRIVILEGES;
DROP DATABASE userdb1;
```

InnoDB - ideale Logfile Größe herausfinden

von code.openark.com und MySQL Performance Blog

Um den idealen Wert für innodb_log_file_size für seine Datenbank herauszufinden, läßt sich sehr einfach dieser Code verwenden. Die Query erfasst über den Zeitraum von 60 Sekunden alle Schreiboperationen auf das InnoDB-Logfile. Idealer Zeitpunkt zur Ausführung ist während einer durchschittlichen Zugriffsmenge, nicht zur Peak-Zeit.

```
SELECT
  innodb_os_log_written_per_minute*60
    AS estimated_innodb_os_log_written_per_hour,
  CONCAT(ROUND(innodb_os_log_written_per_minute*60/1024/1024, 1), 'MB')
    AS estimated_innodb_os_log_written_per_hour_mb
FROM
  (SELECT SUM(value) AS innodb_os_log_written_per_minute FROM (
    SELECT -VARIABLE_VALUE AS value
      FROM INFORMATION_SCHEMA.GLOBAL_STATUS
      WHERE VARIABLE_NAME = 'innodb_os_log_written'
    UNION ALL
    SELECT SLEEP(60)
      FROM DUAL
    UNION ALL
    SELECT VARIABLE_VALUE
      FROM INFORMATION_SCHEMA.GLOBAL_STATUS
      WHERE VARIABLE_NAME = 'innodb_os_log_written'
  ) s1
) s2
;
```

Das sieht dann z.B. so aus:

```
+-----------------------------------------+----------------------------------------------+
| estimated_innodb_os_log_written_per_hour | estimated_innodb_os_log_written_per_hour_mb |
+-----------------------------------------+----------------------------------------------+
|                                58920960 | 56.2MB                                       |
+-----------------------------------------+----------------------------------------------+
```

Also sind es aufgerunden ca. 60 MB und da es standardmäßig 2 Logfiles gibt, lautet der ideale Wert:

```
innodb_log_file_size = 30M
```

Größe einer DB, Tabelle, aller DBs herausfinden

Ersetze 'YOUR_DATABASE_NAME' durch die gewünschte Datenbank.

Größe aller Tabellen einer Datenbank:

```
SELECT TABLE_SCHEMA AS 'Database_name', TABLE_NAME AS 'Table_Name',CONCAT(ROUND(((DATA_LENGTH + INDEX_LENGTH - DATA_FREE) / 1024 / 1024),2)," Mb") AS Size FROM INFORMATION_SCHEMA.TABLES;
```

Größe bestimmter Tabellen:

```
SELECT TABLE_SCHEMA AS 'Database_name', TABLE_NAME AS 'Table_Name',CONCAT(ROUND(((DATA_LENGTH + INDEX_LENGTH - DATA_FREE) / 1024 / 1024),2)," MB") AS Size FROM INFORMATION_SCHEMA.TABLES where TABLE_SCHEMA = 'YOUR_DATABASE_NAME';
```

Größe einer bestimmten Datenbank:

```
SELECT CONCAT(sum(ROUND(((DATA_LENGTH + INDEX_LENGTH - DATA_FREE) / 1024 / 1024),2))," MB") AS Size FROM INFORMATION_SCHEMA.TABLES where TABLE_SCHEMA =  'YOUR_DATABASE_NAME';
```

Größe aller Datenbanken:

```
SELECT table_schema AS "Database", sum( data_length + index_length ) / 1024 / 1024 AS "Size in MB" FROM information_schema.TABLES GROUP BY table_schema;
```

Größe aller Datenbanken und wieviel Platz durch ein "OPTIMIZE TABLE" gewonnen werden kann:

```
SELECT table_schema "database name", round( sum( data_length + index_length ) / 1024 / 1024) "database size in MB", round( sum( data_free )/ 1024 / 1024) "free space in MB" FROM information_schema.TABLES GROUP BY table_schema;
```

Die 10 größten Tabellen anzeigen und wie viel Platz gewonnen werden könnte:

```
SELECT table_schema AS database_name,
       table_name,
       round( (data_length + index_length) / 1024 / 1024, 2) AS total_size,
       round( (data_length) / 1024 / 1024, 2) AS data_size,
       round( (index_length) / 1024 / 1024, 2) AS index_size,
       round( (data_free) / 1024 / 1024) AS free_space
```

```
FROM information_schema.tables
WHERE table_schema NOT IN ('information_schema', 'mysql', 'performance_schema' ,'sys')
      AND table_type = 'BASE TABLE'
      -- AND table_schema = '$i'
ORDER BY total_size DESC
LIMIT 10;
```

wo kann am meisten Platz gewonnen werden (sortiert nach free_space):

```
SELECT table_schema AS database_name,
       table_name,
       round( (data_length + index_length) / 1024 / 1024, 2) AS total_size,
       round( (data_length) / 1024 / 1024, 2) AS data_size,
       round( (index_length) / 1024 / 1024, 2) AS index_size,
       round( (data_free) / 1024 / 1024) AS free_space
FROM information_schema.tables
WHERE table_schema NOT IN ('information_schema', 'mysql', 'performance_schema' ,'sys')
      AND table_type = 'BASE TABLE'
      -- AND table_schema = '$i'
ORDER BY free_space DESC
LIMIT 10;
```

Passwort setzen oder ändern

mit „mysqladmin“ ein Passwort setzen oder ändern:

```
mysqladmin -h 'hostname' -u root password 'neues-passwort'

für localhost:
mysqladmin -u root password 'neues-passwort'
```

mittels SQL ein neues Passwort vergeben:

```
mysql -u root
mysql> GRANT ALL PRIVILEGES ON userdb1.* TO 'user1'@'123.123.123.123' IDENTIFIED BY 'neues-
passwort';
```

Passwort vergessen

Wenn das Kennwort für 'root' verloren ging / vergessen wurde und sonst auch keine Adminuser existieren, lassen sich die Kennwörter noch über den folgenden Weg zurücksetzen (getestet unter Debian).

Ist euer MySQL-Server direkt und öffentlich über das Internet erreichbar solltet ihr vorher den Zugriff darauf per iptables-Regel einschränken, da sich anschließend alle User ohne Kennwort einloggen können.

1. MySQL Server stoppen: /etc/init.d/mysql stop

2. in /etc/mysql/my.cnf wird diese Zeile hinzugefügt, damit werden beim Start die Berechtigungstabellen ignoriert und jeder(!) User kann sich ohne Kennwort einloggen.

```
--skip-grant-tables
```

3. MySQL Server wieder starten: /etc/init.d/mysql start

4. MySQL Client starten und Kennwörter neu setzen

```
# mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('newrootpassword') WHERE User='root';
exit
```

5. MySQL-Server wieder stoppen

6. „–skip-grant-tables“ wieder aus der my.cnf löschen

7. MySQL-Server wieder starten. Ihr solltet Euch nun mit dem neuen Kennwort einloggen können.

MySQL-Datenbank umbenennen

Der ganz sichere Weg ist, die DB per mysqldump zu exportieren und wieder in die neue DB zu importieren:

```
mysqldump alteDb | mysql neueDB
```

Für kurze Zeit gab es das MySQL-Kommando „RENAME DATABASE“, das aber wegen verschiedenen Problemen schnell wieder entfernt wurde.

Per „RENAME TABLE“ lassen sich Tabellen umbenennen und dabei auch in ein anderes Schema verschieben. Da das für jede Tabelle einzeln gemacht werden muss, kann das über dieses Script auch automatisiert werden:

```
#!/bin/bash
#
# rename a mysql database - this works both for MyISAM and InnoDB
#
# usage: rename-mysql-db.sh old new
# dbname as in "SHOW DATABASES"
#

# mysql command and authorization info if needed
mysql="mysql"
#mysql="mysql -uroot -pblabla"
#mysql="mysql -uroot -pblabla -S /var/lib/mysql/mysql.sock -h localhost"

olddb=${1}
newdb=${2}

${mysql} -e "CREATE DATABASE ${newdb}"
tables=$(${mysql} -N -e "SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE
table_schema='${olddb}'")

for name in ${tables}; do
      ${mysql} -e "RENAME TABLE $olddb.$name to $newdb.$name";
done;

${mysql} -e "DROP DATABASE $olddb"
```

Oder einfach die notwendigen Kommandos für die manuelle Ausführung ausgeben lassen:

```
ALTE_DB="nextcloud_test"
NEUE_DB="bla_test"
for table in `mysql -s -N -e "USE $ALTE_DB; SHOW TABLES;"`; do echo mysql -s -N -e "USE
$ALTE_DB; RENAME TABLE $ALTE_DB.$table to $NEUE_DB.$table;"; done;
```

mysqltuner - MySQL Performance Tuning Check-Script

MySQLTuner is ein Perlscript, das versucht die MySQL Konfiguration zu prüfen und Vorschläge zum Tuning auf Basis der laufenden Installation zu machen.

Installation:

```
wget -O mysqltuner.pl http://mysqltuner.pl,
chmod u+x mysqltuner.pl
```

Ausführung:

```
./mysqltuner.pl

                                                                                Di 27.
Jul 2021 14:48:07
 >>  MySQLTuner 1.8.1 - Major Hayden <major@mhtx.net>
 >>  Bug reports, feature requests, and downloads at http://mysqltuner.pl/
 >>  Run with '--help' for additional options and output filtering

[--] Skipped version check for MySQLTuner script
[OK] Logged in using credentials from Debian maintenance account.
[OK] Currently running supported MySQL version 8.0.25-15
[OK] Operating on 64-bit architecture

-------- Log file Recommendations ------------------------------------------------------------------
[OK] Log file /var/log/mysql/error.log exists
[--] Log file: /var/log/mysql/error.log(0B)
[--] Log file /var/log/mysql/error.log is empty. Assuming log-rotation. Use --server-
log={file} for explicit file

-------- Storage Engine Statistics -----------------------------------------------------------------
[--] Status: +ARCHIVE +BLACKHOLE +CSV -FEDERATED +InnoDB +MEMORY +MRG_MYISAM +MyISAM
+PERFORMANCE_SCHEMA
[--] Data in InnoDB tables: 525.5M (Tables: 650)
[--] Data in MyISAM tables: 28.9M (Tables: 160)
```

```
[OK] Total fragmented tables: 0

-------- Analysis Performance Metrics
------------------------------------------------------------------
[--] innodb_stats_on_metadata: OFF
[OK] No stat updates during querying INFORMATION_SCHEMA.

-------- Security Recommendations
--------------------------------------------------------------------
[--] Skipped due to unsupported feature for MySQL 8

-------- CVE Security Recommendations
------------------------------------------------------------
[--] Skipped due to --cvefile option undefined

-------- Performance Metrics
------------------------------------------------------------------------
[--] Up for: 4d 12h 47m 32s (33M q [85.194 qps], 482K conn, TX: 155G, RX: 6G)
[--] Reads / Writes: 98% / 2%
[--] Binary logging is enabled (GTID MODE: OFF)
[--] Physical Memory     : 23.5G
[--] Max MySQL memory    : 9.9G
[--] Other process memory: 0B
[--] Total buffers: 3.4G global + 133.6M per thread (50 max threads)
[--] P_S Max memory usage: 72B
[--] Galera GCache Max memory usage: 0B
[OK] Maximum reached memory usage: 7.7G (32.72% of installed RAM)
[OK] Maximum possible memory usage: 9.9G (42.15% of installed RAM)
[OK] Overall possible memory usage with other process is compatible with memory available
[!!] Slow queries: 10% (3M/33M)
[OK] Highest usage of available connections: 66% (33/50)
[OK] Aborted connections: 0.27%  (1314/482261)
[--] Query cache have been removed in MySQL 8
[OK] Sorts requiring temporary tables: 0% (0 temp sorts / 435K sorts)
[!!] Joins performed without indexes: 16421
[OK] Temporary tables created on disk: 12% (78K on disk / 652K total)
[OK] Thread cache hit rate: 99% (49 created / 482K connections)
[OK] Table cache hit rate: 99% (37M hits / 37M requests)
[OK] table_definition_cache(2000) is upper than number of tables(1149)
[OK] Open file limit used: 0% (2K/1M)
```

```
[OK] Table locks acquired immediately: 100% (1M immediate / 1M locks)
[OK] Binlog cache memory access: 99.99% (463714 Memory / 463765 Total)

-------- Performance schema
--------------------------------------------------------------------------
[--] Memory used by P_S: 72B
[--] Sys schema is installed.

-------- ThreadPool Metrics
--------------------------------------------------------------------------
[--] ThreadPool stat is enabled.
[--] Thread Pool Size: 24 thread(s).
[OK] thread_pool_size between 16 and 36 when using InnoDB storage engine.

-------- MyISAM Metrics
---------------------------------------------------------------------------
[--] MyISAM Metrics are disabled on last MySQL versions.

-------- InnoDB Metrics
---------------------------------------------------------------------------
[--] InnoDB is enabled.
[--] InnoDB Thread Concurrency: 9
[OK] InnoDB File per table is activated
[OK] InnoDB buffer pool / data size: 3.0G/525.5M
[OK] Ratio InnoDB log file size / InnoDB Buffer pool size: 384.0M * 2/3.0G should be equal to
25%
[OK] InnoDB buffer pool instances: 3
[--] Number of InnoDB Buffer Pool Chunk : 24 for 3 Buffer Pool Instance(s)
[OK] Innodb_buffer_pool_size aligned with Innodb_buffer_pool_chunk_size &
Innodb_buffer_pool_instances
[OK] InnoDB Read buffer efficiency: 100.00% (3574568225 hits/ 3574619782 total)
[!!] InnoDB Write Log efficiency: 64.54% (3246059 hits/ 5029381 total)
[OK] InnoDB log waits: 0.00% (0 waits / 1783322 writes)

-------- Aria Metrics
-----------------------------------------------------------------------------
[--] Aria Storage Engine not available.

-------- TokuDB Metrics
---------------------------------------------------------------------------
```

```
[--] TokuDB is disabled.

-------- XtraDB Metrics
-----------------------------------------------------------------------------
[--] XtraDB is disabled.

-------- Galera Metrics
-----------------------------------------------------------------------------
[--] Galera is disabled.

-------- Replication Metrics
-------------------------------------------------------------------------
[--] Galera Synchronous replication: NO
[--] No replication slave(s) for this server.
[--] Binlog format: MIXED
[--] XA support enabled: ON
[--] Semi synchronous replication Master: Not Activated
[--] Semi synchronous replication Slave: Not Activated
[--] This is a standalone server

-------- Recommendations
-----------------------------------------------------------------------------
General recommendations:
    We will suggest raising the 'join_buffer_size' until JOINs not using indexes are found.
              See https://dev.mysql.com/doc/internals/en/join-buffer-size.html
              (specially the conclusions at the bottom of the page).
Variables to adjust:
    join_buffer_size (> 4.0M, or always use indexes with JOINs)
```

Schnell Platz schaffen

Wenn dein Server binlogs schreibt und die Disk voll läuft kann es helfen schnell Platz zu schaffen indem man nicht mehr benötigte binlogs löscht.

Hier gibts zwei Möglichkeiten.

Mit der Variablen expire_logs_days (akzeptiert Integer-Werte von 0-99) läßt sich der Zeitraum einstellen wie lange Binlogs in Tagen aufgehoben werden sollen. Wenn man den Wert auf 0 setzt werden die Binlogs gar nicht mehr gelöscht. Default-Wert ist 7.

Platz schaffen läßt sich also z.B. so:

```
mysql> SHOW VARIABLES LIKE 'expire_logs_days';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| expire_logs_days | 3     |
+------------------+-------+
1 row in set (0.01 sec)

mysql> SET GLOBAL expire_logs_days=3;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH LOGS;
Query OK, 0 rows affected (0.15 sec)
```

Der Wert wird nicht dauerhaft gespeichert und steht nach einem Restart der DB wieder auf Default, bzw. auf dem Wert, der in eurer Config gesetzt ist.

Man kann auch direkt alte Binlogs löschen, das läßt sich dann auch feiner steuern. In diesem Beispiel werden alle Binlogs älter als eine Stunde gelöscht:

```
mysql> FLUSH LOGS;
Query OK, 0 rows affected (0.15 sec)

mysql> PURGE BINARY LOGS BEFORE NOW() - INTERVAL 1 HOUR;
Query OK, 0 rows affected (2.23 sec)
```

> Falls man ein Replication-Setup hat muss man natürlich darauf achten, dass der Slave auf aktuellem Stand ist und man nicht zuviele Logs löscht!

Ungenutzte Datenbanken finden

Dieses Snippet listet die neueste Datei in jedem Datenbankverzeichnis unterhalb von /var/lib/mysql auf:

```
cd /var/lib/mysql
for i in `mysql --skip-column-names -B -e "show databases" | grep -v -E
"mysql|information_schema|sys|performance_schema|percona_schema"`; do echo -ne "\n--- $i ---
\nSize: "; du -hs $i; find $i -type f -print0 | xargs -0 stat --format '%Y :%y %n' | sort -nr
| cut -d: -f2- | head -n1; done
```

Beispiel:

```
--- database1 ---
Size: 121M      database1
2024-12-16 21:12:42.437939063 +0100 database1/Portal_PortalRequestLog.ibd

--- database2 ---
Size: 17G       database2
2024-12-10 06:24:49.902646396 +0100 database2/app_session.ibd

--- database3 ---
Size: 152M      database3
2024-10-10 14:20:10.301089103 +0200 database3/layout_file.frm
```