Workshop MariaDB Windows

Agenda

1. Grundsätzliches
   - Historie MySQL/MariaDB
   - Aufbau MariaDB
   - MySQL vs. MariaDB
   - Welche Logs gibt es?
2. Performance / Theorie - Aspekte der MariaDB - Architektur
   - Architektur Server (Schritte)
   - CPU oder io-Last klären
   - Storage Engines
   - InnoDB - Struktur
   - InnoDB - Optimierung
   - Query - Cache
   - 3-Phasen-Datengröße
3. Performance / Konfiguration
   - Slow query log
4. Administration / Logging
   - Standard storage engine bestimmen
   - Datenbank-Namen performant umbenennen
   - Show status
   - Server System Variablen - show variables
   - Arbeiten mit dem information_schema
   - User verwalten
   - Einstellungsmöglichkeiten für ErrorLogs
   - Prozesslist von mariadb nutzen
5. Backup und Restore
   - Wann binlog ?
   - Backup with mysqldump - best practices
   - PIT Exercise - point in time recovery
   - Mariabackup
6. Performance und Optimierung von SQL-Statements
   - Performance tmp_disk_tables problem
   - Explain verwenden
   - Do not use '*' whenever possible
   - Indexes
   - profiling-get-time-for-execution-of.query
   - Kein function in where verwenden
   - Optimizer-hints (and why you should not use them)
   - Query-Plans aka Explains
   - Query Pläne und die Key-Länge
   - Index und Likes
   - Index und Joins
   - Find out cardinality without index
   - Index and Functions
7. Tools

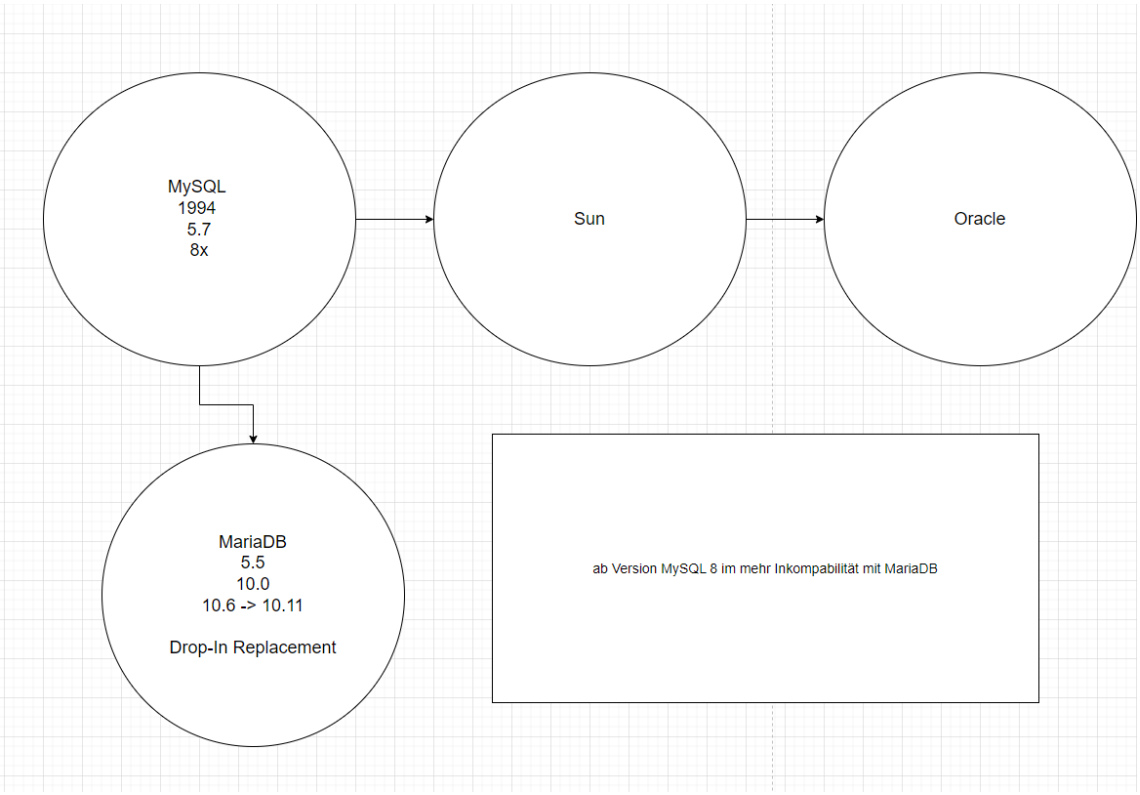
- Percona Toolkit - only pt-query-digest
- pt-query-digest - analyze slow logs
- pt-online-schema-change howto
- Example sys-schema and Reference

8. Beispieldaten
   - Verleihdatenbank - sakila
   - Setup training data "contributions"
9. Managing big tables
   - Using Partitions - Walkthrough
10. Replication
    - Aufbau Master/Slave - Replication
    - Replikation mit GTID
    - Replikation Read/Write - Split:
11. MariaDB (Galera Cluster) - Linux Only !!
    - Aufbau Galera Cluster
12. Fragen und Antworten
    - Fragen und Antworten
13. Projektarbeit/-optimierung
    - Praktisch Umsetzung in 3-Schritten
14. Monitoring
    - Was sollten wir monitoren ?
15. Dokumentation
    - MySQL - Performance - PDF
    - Effective MySQL
    - MariaDB Downloaden
    - MariaDB - Releases - including long - term releases
    - Effective MySQL

Grundsätzliches

Historie MySQL/MariaDB

Schaubild



Aufbau MariaDB

MySQL vs. MariaDB

Was ist gleich ?

- Gleiche CodeBasis weil Kopie
- gleiche Tools (weitesgehend)

Was ist anders (MariaDB)?

- Andere Storage Engine sind möglich
- physische Onlinebackup ist mit drin in der Community Version (mariabackup)
    - ein absolutes Muss für grosse Datenbestände (Geschwindigkeit ist wesentlich schneller beim zurückspielen
- Langsame Abfragen protokollieren lassen (hier habt ihr in MariaDB noch mehr Ausgabemöglichkeiten)
- Abweichende Datentypen für Json (anders implementiert als in MySQL)

mysql/mariadb - client

- datenbank die gerade ausgewählt ist, wird angezeigt im Prompt

Was ist anders ? (MySQL)

- ab MySQL 8 - die Server Konfiguration während der Laufzeit setzen und persistent ändern
- von Hause aus anderes Cluster-Technologie -> mysql group replication vs. MariaDB -> Galera Cluster

Welche Logs gibt es?

general_log

- Alle Anfrage gegen den Server (Abgesetzte SQL-Statements)

error_log

- Protokolliert nicht nur Fehler, sondern den kompletten Startup (auch warning und notes)
- Log - Level kann angepasst werden (Standard: 2, in der Regel ausreichend)
    - log_error_verbosity

Ereignisprotokoll (Windows)

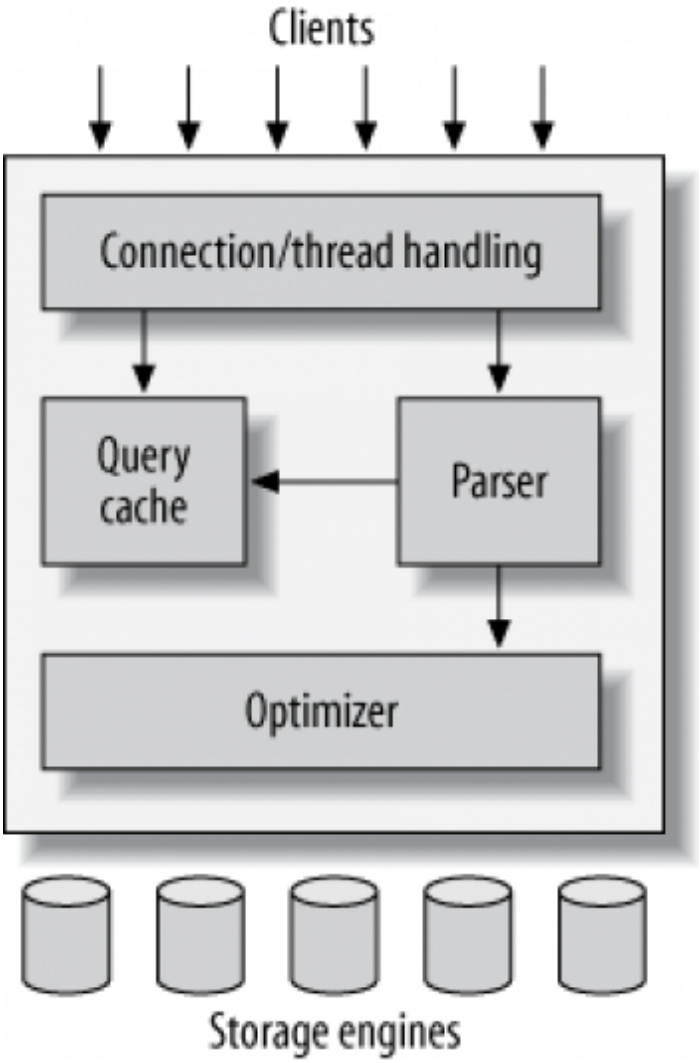
- Falls ich im error_log keinen ausreichenden Informationen finde, kann ich auch nochmal nachschauen.

slow_query_log

- Langsame Queries die Bedingung long_query_time erfüllen, werden mitgeloggt, wenn eingeschaltet
    - im Datenverzeichnis unter *-slow - Datei

Performance / Theorie - Aspekte der MariaDB - Architektur

Architektur Server (Schritte)



CPU oder io-Last klären

```
top - 07:29:09 up 19:14,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  69 total,   1 running,  68 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    989.9 total,    273.7 free,    155.3 used,    560.8 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    677.2 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0  104936  10296   7880 S   0.0   1.0   0:05.78 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-+
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0.0   0.0   0:00.80 ksoftirqd/0
   10 root      20   0       0      0      0 I   0.0   0.0   0:01.66 rcu_sched
   11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0.0   0.0   0:00.30 migration/0
   14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
   16 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
   17 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kauditd
   18 root      20   0       0      0      0 S   0.0   0.0   0:00.02 khungtaskd
   19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 oom_reaper
```

```
Fall 1
======
CPU-gebundene Last: in Zeile CPU:
nur 'sy' und 'us' ist hoch

Fall 2:
=======
IO-gebundene Last
(d.h. egal, ob man eine bessere hat, es bringt nicht mehr,
weil die Festplatte entscheidend ist (in diesem Fal))
Die Festplatte ist hier der begrenzende Faktor

sy und wa hoch (wa = waiting, cpu wartet auf das io-subsystem (Festplatte or Storage)
```

Storage Engines

Why ?

```
Let's you choose:
How your data is stored
```

What ?

- Performance, features and other characteristics you want

Where ?

- Theoretically you can use a different engine for every table
- But: For performance optimization and future, it is better to concentrate on one

What do they do ?

- In charge for: Responsible for storing and retrieving all data stored in MySQL
- Each storage engine has its:
  - Drawbacks and benefits
- Server communicates with them through the storage engine API
  - this interface hides differences
  - makes them largely transparent at query layer
  - api contains a couple of dozen low-level functions e.g. “begin a transaction”, “fetch the row that has this primary key”

Storage Engine do not ....

- Storage Engines do not parse SQL
- Storage Engines do not communicate with each other

They simply .....

- They simply respond to requests from the server

Which are the most important one ?

- InnoDB (currently default engine)
- MyISAM/Aria
- Memory
- CSV
- Blackhole (/dev/null)
- Archive
- Partition
- (Federated/FederatedX)

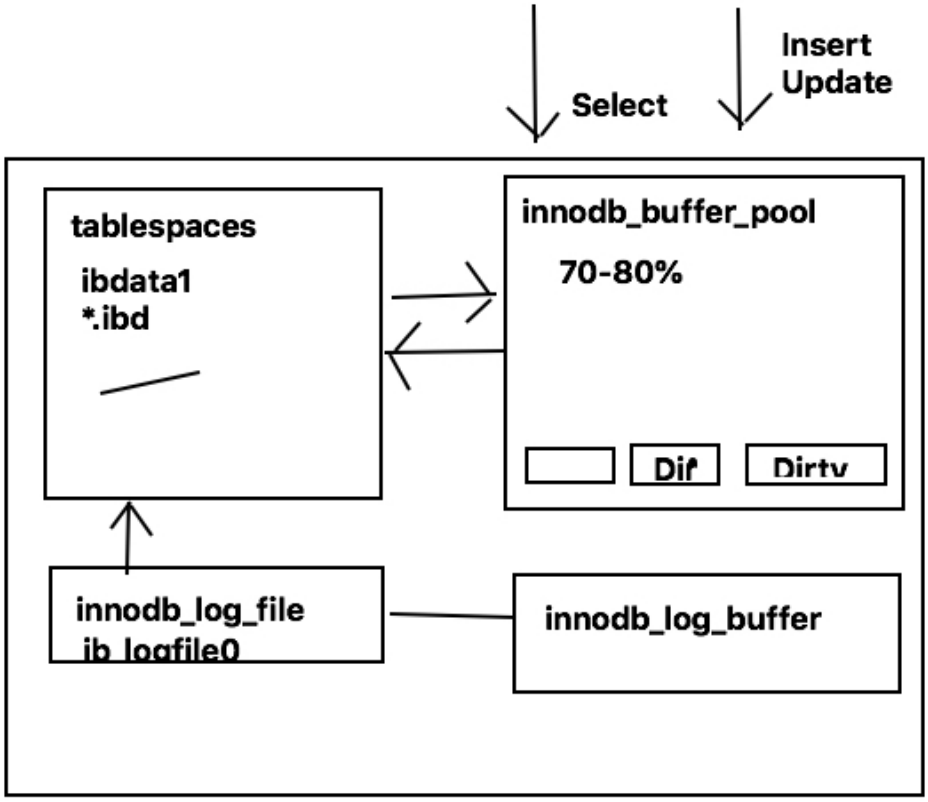
In Detail: MyISAM - Storage Engine

```
1. table locks → Locks are done table-wide
2. no automatic data-recovery (Aria hat das !)
3. you can loose more data on crashes than with e.g. InnoDB
4. no transactions
5. only indices are save in memory through MySQL
6. compact saving (data is saved really dense)
7. table scans are quick
```

In Detail: InnoDB - Storage Engine

```
1. support hot backups (because of transactions)
2. transactions are supported
3. foreign keys are supported
4. row-level locking (only single lines are locked)
5. multi-versioning
```

InnoDB - Struktur



InnoDB - Optimierung

Innodb buffer pool

- How much data fits into memory
- Free buffers = pages of 16 Kbytes
- Free buffer * 16Kbytes = free innodb buffer pool in KByte

How to find out ?

```
## OR:
MariaDB [(none)]> show status like '%free%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| Innodb_buffer_pool_pages_free | 48083   |
| Innodb_buffer_pool_wait_free  | 0       |
| Innodb_ibuf_free_list         | 0       |
| Qcache_free_blocks            | 1       |
| Qcache_free_memory            | 1031304 |
+------------------------------+---------+
5 rows in set (0.002 sec)
```

show engine innodb status

```
## please use command line
show engine innodb status \G
```

Overview innodb server variables / settings

- https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html

Change innodb_buffer_pool

```
## my.ini
## 70-80% of memory on dedicated mysql
[mysqld]
innodb-buffer-pool-size=6G

## Dienst neu starten

##
mysql
mysql>show variables like 'innodb%buffer%';
```

innodb_flush_method

```
Ideally O_DIRECT on Linux, but please test it, if it really works well.
- no changes needed in Windows as we are using unbuffered
```

innodb_flush_log_at_trx_commit

```
When is fliushing done from innodb_log_buffer to log.
Default: 1 : After every commit
-> best performance 2. -> once per second

## Good to use 2, if you are willing to loose 1 second of data on powerfail
```

innodb_flush_neighbors

```
## on ssd disks set this to off, because there is no performance improvement
innodb_flush_neighbors=0

## Default = 1
```

skip-name-resolv

```
## work only with ip's - better for performance
/etc/my.cnf
skip-name-resolve
```

- https://nixcp.com/skip-name-resolve/

Calculate innodb-log-file-size

```
-- Session 1: LSN abfragen
-- in mysql client
pager more;
-- Determine LSN from engine innodb status
--  Log sequence number 21879482
show engine innodb status \G
select sleep(60);
```

```
## Session 2: Import ausführen (als Beispiel für es finden Veränderungen stand)
## in command prompt (mariadb)
## ins backup verzeichnis wechseln
cd C:\Users\vgh-MariaDB\Desktop\Backups
## all-databases.sql einspielen
mysql -uroot -p<mein password> < all-databases.sql
```

```
## Wieder in Session 1
## Determine LSN #
## Log sequence number 22279482
pager more;
show engine innodb status \G
pager;
```

```
## letzter (2. Wert - 1.Wert)
mysql> select (3838334638 - 3836410803) / 1024 / 1024 as MB_per_min;
+-------------+
| MB_per_min |
+-------------+
| 1.83471203 |
+-------------+
```

- https://www.percona.com/blog/how-to-calculate-a-good-innodb-log-file-size/

Ref:

- https://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-pool-resize.html

Privilegs for show engine innodb status

```
 show engine innodb status \G
ERROR 1227 (42000): Access denied; you need (at least one of) the PROCESS privilege(s)
for this operation
```

Query - Cache

Defaults

- Default Value: OFF (>= MariaDB 10.1.7), ON (<= MariaDB 10.1.6)

Performance query cache

- Always try to optimize innodb with disabled query cache first (innodb_buffer_pool)
- If you use query_cache system can only use on CPU-Core. !!

How to enable query cache

```
## have_query_cache means compiled in mysql
## query_cache_type off means not enable by config
-- query cache is diabled
mysql> show variables like '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 1048576 |
| query_cache_type             | OFF     |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0.01 sec)

root@trn01:/etc/mysql/mysql.conf.d# tail mysqld.cnf
[mysqld]
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
datadir         = /var/lib/mysql
log-error       = /var/log/mysql/error.log
## By default we only accept connections from localhost
bind-address    = 0.0.0.0
## Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
query-cache-type=1

systemctl restart mysql

mysql> show variables like '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 1048576 |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0.01 sec)


mysql> show status like '%Qcache%';
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Qcache_free_blocks      | 1       |
```

```
| Qcache_free_memory      | 1031832 |
| Qcache_hits             | 0       |
| Qcache_inserts          | 0       |
| Qcache_lowmem_prunes    | 0       |
| Qcache_not_cached       | 0       |
| Qcache_queries_in_cache | 0       |
| Qcache_total_blocks     | 1       |
+-------------------------+---------+
8 rows in set (0.00 sec)

## status in session zurücksetzen.
mysql> flush status;
Query OK, 0 rows affected (0.00 sec)
```

Warum die Verwendung des Query Cache schlecht

```
TABELLE Mitarbeiter
Select * from Mitarbeiter -> query_cache
Nächste abfrage. Select * from Mitarbeiter
-> aus query_cache
Insert into Mitarbeiter
-> cache invalidiert -> kein Inhalt mehr
Select * from Mitarbeiter -> query_cache



Mutex:
-> bei Benutzung gesperrt

// dadurch können Schreibanfragen nur quasi sequentiell
A schreibt, B wartet bis a fertig ist, dann schreibt B

Nur Zeilensperrung
A schreibt, B schreibt auch, wenn nicht Genua die gleichen Zeile

Query cache verhindert, dass mehre Kerne der CPU von MySQL verwendet werden können.

-> lock-file im filesystem -> mutex -> mutual - exclusion.

Ich mache ein Lock-file damit du weisst, dass ich gerade
Dran arbeite.
```

3-Phasen-Datengröße

Phase 1: Table content is small (only some rows)

```
## table scan is quicker than index search
## e.g. 10 entries

## so eventually index is not needed
```

Phase 2: Index is good !!

```
## performance gain by using index
## Step 1: Obtaining id's from index (primary key id)
## Step 2: Retrieving data
```

Phase 3: Index is not improve performance / or would makes performance worse

```
Step 1: lookup in index:
1
70
1040
2100
35000
-> there is a lot of space (other rows) in between.

Step 2: Lookup data, but a lot lookups needed


-> random reads
-> So mysql might be better off to do a table scan.
```

Performance / Konfiguration

Slow query log

General

- Slow Query logs is activated with slow_query_log either in my.ini or as with set global
- But it only triggers when the time is given correctly
    - Default: 10 (only queries slower than 10 seconds are recorded)

Easiest way to activate during runtime

```
-- in mysql - client
set global slow_query_log = 1;
-- not set in session yet
show variables like '%slow%';
-- activate also for session OR: reconnect with session or new mysql-client session
set slow_query_log = 1
set global long_query_time=0.000001; -- 0,5 Sekunden. Alles was >= 0,5 sekunden
dauert, wird geloggt
set session long_query_time=0.000001;

-- Empfehlung für ein gutes Logging auch das auszugeben
set global log_slow_verbosity="query_plan,explain";
```

```

```

Logge alles wo kein Index verwendet werden kann (egal) wie langsam oder schnell

```
## damit er wirklich nur die queries logged, die keinen index haben, sollte. der
## long_query_time - Wert möglichst hoch sein.
set global long_query_time = 20
set session long_query_time = 20
set global slow_query_log = 1
set session slow_query_log = 1
set global log_queries_not_using_indexes = 1
set session log_queries_not_using_indexes = 1
```

Bitte slow_query_log bei der ausgabe geschätziger zu sein

```
set global log_slow_verbosity = 'query_plan,explain'
set session log_slow_verbosity = 'query_plan,explain'
```

Die Anzahl der Ausgabe reduzieren (nur jedes 5.)

```
### /etc/mysql/mariadb-conf.d/50-server.cnf und mysqld
log-slow-rate-limit=5;
```

Best - Practice - Phase 1

```
## Alle Logs analysieren, die kein Index verwendet
##/etc/mysql/mariadb.conf.d/50-server.cnf
## unter [mysqld]

## slow query log
slow-query-log
log-queries-not-using-indexes
log-slow-rate-limit=5
log-slow-verbosity = 'query_plan,explain'
```

Ref:

- https://mariadb.com/kb/en/slow-query-log-overview/

Administration / Logging

Standard storage engine bestimmen

```
Die Standard-Storage wird über die Server-System-Variable
default_storage_engine
festgelegt.

Wenn beim Erstellen einer Tabelle keine storage-engine angegeben
wird, wird diese verwendet .

(In Datenbanken/Schemas kann man KEINE Storage engine festlegen)

mysql>show variables like 'default_storage_engine';
```

Datenbank-Namen performant umbenennen

Walkthrough

```
## im mysql - client
create schema sakilanew
rename table sakila.actor to sakilanew.actor

## Attention, does not work views
## You have to dump and import views
## DOES NOT WORK BECAUSE OF VIEW
rename table sakila.actor_info to sakilanew.actor_info

## Also if there are triggers on a table it does not also not work.
## Eventually Delete triggers and set them again
rename table sakila.film to sakilanew.film;
ERROR 1435 (HY000): Trigger in wrong schema
```

Show status

with mysql -> show status

```
mysql> show status;
-- global status für den gesamten Server seit er läuft
mysql> show global status;
mysql> # setzt session status zurück
mysql> flush status;
mysql> show status;
```

Spezielle status variablen

```
show status like 'Com%';
show status like 'Com_select ';
```

Aus information_schema

```
select * from information_schema.global_status;
select * from information_schema.session_status;
```

Server System Variablen - show variables

```
show variables;
show global variables;
show variables like 'innodb%';
show global variables like 'innodb%';
```

```
## @@ steht für Server System Variable
select @@innodb_flush_method
```

Arbeiten mit dem information_schema

```
-- im mysql - client
use information_schema;
show tables;
select * from global_variables \G
-- show all buffer vars
select * from global_variables where variable_name like '%buffer%';
```

User verwalten

```
## bitte nur im Notfall von überall
## + passworr im klartext
mysql>create user training@'%' identified by 'meingeheimespasswort'
mysql>create user training@192.168.2.2; -- von einer bestimmten ip ausschliesslich //
ip des zugreifers

## Rechte vergeben *.* -> alle datenbanken.alle tabellen
## to -> für.
mysql>grant all on *.* to training@192.168.2.2
## Rechte entziehn
mysql>revoke select on *.* from training@192.168.2.2
## oder alle Rechte enziehen
mysql>revoke all on *.* from training@192.168.2.2

## Rechte eines Benutzers anschauen
mysql>show grants for training@192.168.2.2. // genaue Kombination muss angegeben
werden

## Eigentlich nicht notwendig, aber geht
mysql>select * from mysql.global_priv \G #  das geht nur im mysql-client und zeigt
Spalten in Zeilen an
mysql>select * from mysql.user;
```

Einstellungsmöglichkeiten für ErrorLogs

- https://mariadb.com/kb/en/error-log/

Prozesslist von mariadb nutzen

Prozesse (Threads)

Über show

```
show processlist;
```

Über information_schema

```
select * from information_schema.processlist;
-- oder
```

```
use information_schema;
select * from processlist;
```

Process beenden

```
-- kill <thread_id-die-wir-über-die-processliste-sehen>
-- z.B.
kill 314
```

Backup und Restore

Wann binlog ?

Auflistung

- Replication
- Point in Time Recovery

Backup with mysqldump - best practices

Dumping (best option) without active binary log

```
## MariaDB Command Prompt öffnen
## im command prompt
cd C:\Users\vgh-MariaDB\Desktop\Backups

mysqldump -uroot -p<password-for-root> --all-databases --single-transaction > all-
databases
## if you want to include procedures use --routines
## with event - scheduled tasks
mysqldump -uroot -p<password-for-root> --all-databases --single-transaction --routines
--events > all-databases.sql
```

Useful options for PIT

```
## —quick not needed, because included in —opt which is enabled by default

## on local systems using socket, there are no huge benefits concerning --compress
## when you dump over the network use it for sure

## MariaDB - Command Prompt öffnen
cd C:\Users\vgh-MariaDB\Desktop\Backups
mysqldump -uroot -p<dein-root-pw> --all-databases --single-transaction --gtid --
master-data=2 --routines --events --flush-logs  > all-databases.sql
mysqldump --user=root --password=<dein-root-pw> --all-databases --single-transaction -
-gtid --master-data=2 --routines --events --flush-logs  > all-databases.sql
```

With PIT_Recovery you can use --delete-master-logs

- All logs before flushing will be deleted

```
cd C:\Users\vgh-MariaDB\Desktop\Backups
mysqldump --all-databases --single-transaction --gtid --master-data=2 --routines --
events --flush-logs --delete-master-logs > all-databases.sql;
```

Flush binary logs from mysql

```
mysql -e "PURGE BINARY LOGS BEFORE '2013-04-22 09:55:22'";
```

Version with zipping

```
mysqldump —-all-databases —-single-transaction —-gtid —-master-data=2 —-routines
--events —-flush-logs --compress | gzip > /usr/src/all-databases.sql.gz
```

Performance Test mysqldump (1.7 Million rows in contributions)

```
date; mysqldump --all-databases --single-transaction --gtid --master-data=2 --routines
--events --flush-logs --compress > /usr/src/all-databases.sql; date
Mi 20. Jan 09:40:44 CET 2021
Mi 20. Jan 09:41:55 CET 2021
```

Seperated sql-structure files and data-txt files including master-data for a specific database

```
 # backups needs to be writeable for mysql
 mkdir /backups
 chmod 777 /backups
 chown mysql:mysql /backups
 mysqldump --tab=/backups contributions
 mysqldump --tab=/backups --master-data=2 contributions
 mysqldump --tab=/backups --master-data=2 contributions > /backups/master-data.tx
```

Create new database based on sakila database

```
## Backupverzeichnis
## Command Prompt in mnariadb
cd C:\Users\vgh-MariaDB\Desktop\Backups
mysqldump -uroot -p sakila > sakila-all.sql
mysql -uroot -p -e "create database mynewdb"
mysql -uroot -p mynewdb < sakila-all.sql
```

PIT Exercise - point in time recovery

Part 1: Problem coming up

```
## Step 1 : Create full backup (assuming 00:05 (5 minutes past) ) in the backup folder
cd C:\Users\vgh-MariaDB\Desktop\Backups
mysqldump -uroot -p --all-databases --single-transaction --gtid --master-data=2 --
routines --events --flush-logs --delete-master-logs > all-databases.sql

## Step 2: Working on data
```

```
mysql>use sakila;
mysql>insert into actor (first_name,last_name) values ('john','The Rock');
mysql>insert into actor (first_name,last_name) values ('johanne','Johannson');
mysql>select * from actor;
## Step 2.5
## Auf welcher Position steht das master - binlog
mysql>show master status;

## Optional: Step 3: Looking into binary to see this data
## im Datenverzeichnis
## last binlog
mysqlbinlog -vv mariadb-bin.000005

## Step 4: Somehow a guy deletes data
mysql>use sakila; delete from actor where actor_id > 200;
## now only 200 datasets
mysql>use sakila; select * from actor;
```

Part 2: Fixing the problem

```
## find out the last binlog
## Simply take the last binlog
```

```
## In command prompt (mariadb)
cd C:\Program Files\MariaDB 10.6\data

## IN THE DATA FOLDER
## Find the position where the problem occured
## mysqlbinlog -vv mysqld-bin.000005 | more
mysqlbinlog -vv mysqld-bin.000005
```

```
## and create a recover.sql - file (before apply full backup)
mysqlbinlog -vv --stop-position=857 mysqld-bin.000005 > recover.sql
move recover.sql C:\Users\vgh-MariaDB\Desktop\Backups\recover.sql
## in case of multiple binlog like so:
## Wenn es mehrere binary logs seit dem letzten vollen Backup gab:
## mysqlbinlog -vv --stop-position=857 mysqld-bin.000004 mysqld-bin.000005 >
recover.sql
```

```
## Step 1: Apply full backup
## Command Prompt (mariadb) aufrufen
## In das Backup-Verzeichnis wechseln
cd C:\Users\vgh-MariaDB\Desktop\Backups\
mysql -uroot -p < all-databases.sql
```

```
-- should be 200 or 202#
mysql -uroot -p -e "select * from actor;" sakila
```

```
## auf der Kommandozeile
mysql -uroot -p < recover.sql
```

```
-- im mysql-client durch eingeben des Befehls 'mysql'
-- should be 202
use sakila; select * from actor;
```

Mariabackup

Installation

```
Is done through MSI-Installer for MariaDB-Server
```

Backup Walkthrough (Windows)

Schritt 1: Backup erstellen

```
## Command Prompt (mariadb)
## Backupfolder C:\Users\vgh-MariaDB\Desktop\Backups
## Going to parent folder
cd C:\Users\vgh-MariaDB\Desktop\

 # target-dir needs to be empty or not present
mariabackup -uroot -p<passwort-for-root> --target-dir=Backups/20230321 --backup
```

Schritt 2: Prepare durchführen (Änderung für Tablespaces anwenden)

```
## apply ib_logfile0 to tablespaces
## after that ib_logfile0 ->  0 bytes
mariabackup --target-dir=Backups/20230321 --prepare
```

Schritt 3a: (Variante 1): Recover Walkhrough

```
1. Dienst mariadb stoppen
```

```
2. Im Datenverzeichnis - altes Datenverzeichnis verschieben
cd C:\Program Files\MariaDB 10.6\
rename data data.bkup
```

```
3. In das Elternverzeichnis von backup wechseln
cd C:\Users\vgh-MariaDB\Desktop
mariabackup --target-dir=Backups/20230321 --copy-back
3.5 my.ini in data - ordner reinkopieren (aus data.bkup ordner)
4. Rechte anpassen: Suchpfad ändern auf Server, auf dem ihr seid, dann (NT
Service\MariaDB) für den Ordner data -> Vollzugriff
5. Dienst mariadb starten
```

Schritt 3b: (Variante 2): Recover Walkhrough

```
1. Dienst mariadb stoppen
```

```
2. Im Datenverzeichnis - altes Datenverzeichnis verschieben
cd C:\Program Files\MariaDB 10.6\data
alle Datei in anderen Ordner (z.B. xy) kopierfen (beliebig, so dass der Ordner leer
ist
```

```
3. In das Elternverzeichnis von backup wechseln
cd C:\Users\vgh-MariaDB\Desktop
mariabackup --target-dir=Backups/20230321 --copy-back
4. my.ini in data - ordner reinkopieren (aus ordner xy )
5. Dienst mariadb starten
```

Performance und Optimierung von SQL-Statements

Performance tmp_disk_tables problem

Warum ?

```
Temporäre Tabellen die auf die Platte geschrieben
create_tmp_disk_tables sind generell für die Performance schlecht
```

Wie kann ich herausfinden, ob das bei mir auf meinem Server der Fall ist ?

```
show global status like '%Created_tmp_disk_tables%';
show global status like '%Created_tmp_files%';
```

Wie kann ich herausfinde, welche Queries das genau sind ?

- Information steht bei mariadb im slow_query_log

Explain verwenden

Einfacher Fall

```
explain select * from actor
```

Erweiterter Fall

```
explain extended select * from user
show warnings
```

Anzeigen der Partitions

```
explain partitions select * from actor
```

Ausgabe im JSON-Format

```
## Hier gibt es noch zusätzliche Informationen
explain format=json select * from actor
```

Do not use '*' whenever possible

Why ?

- You are adding .. to he server:
  - I/O
  - memory
  - CPU
- You are preventing covering indexes

Walkthrough. (Look at the time)

Using '*'

```
## using '* '
pager grep "rows in set";
select * from donors where last_name like 'Willia%'; select * from donors where
last_name like 'Willia%';
-- time between 0.02 and 0.04 secs
-- 2424 rows in set (0.02 sec)
-- reset pager
pager

## corresponding Explain (QEP)
explain select * from donors where last_name like 'Willia%';
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+-----------------------+
| id | select_type | table  | partitions | type  | possible_keys     | key
| key_len | ref  | rows | filtered | Extra                 |
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+-----------------------+
|  1 | SIMPLE      | donors | NULL       | range | donors_donor_info |
donors_donor_info | 213     | NULL | 4748 |   100.00 | Using index condition |
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)
```

using specific fields

```
pager grep 'rows in set'; select last_name,first_name from donors where last_name like
'Willia%'; pager;
PAGER set to 'grep 'rows in set''
2424 rows in set (0.01 sec)
```

```
 explain select last_name,first_name from donors where last_name like 'Willia%';
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+--------------------------+
| id | select_type | table  | partitions | type  | possible_keys     | key
| key_len | ref  | rows | filtered | Extra                    |
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+--------------------------+
|  1 | SIMPLE      | donors | NULL       | range | donors_donor_info |
```

```
donors_donor_info | 213     | NULL | 4748 |   100.00 | Using where; Using index |
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)
```

- Uses cover index (indicator in Extra: using index)

Ref:

- https://www.oreilly.com/library/view/high-performance-mysql/9780596101718/ch04.html

Indexes

Avoid ALL

- is the worst type : TABLE SCAN (Need to go through all rows)

```
mysql> create table actor4 as select * from actor;
mysql> explain select * from actor4 where actor_id > 10;
+----+-------------+--------+------------+------+---------------+------+---------+----
--+------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys | key  | key_len | ref
| rows | filtered | Extra       |
+----+-------------+--------+------------+------+---------------+------+---------+----
--+------+----------+-------------+
|  1 | SIMPLE      | actor4 | NULL       | ALL  | NULL          | NULL | NULL    |
NULL |  200 |    33.33 | Using where |
+----+-------------+--------+------------+------+---------------+------+---------+----
--+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

Cover Index.

- We can get all the necessary information from the index (no acces of filesystem necessary)

```
drop table if exists actor2;
create table actor2 as select * from actor;
create index idx_actor2_last_name on actor2 (last_name);

## using index
## <- indicates that a cover index is used

mysql> explain select last_name from actor2 where last_name like 'B%';
+----+-------------+--------+------------+-------+----------------------+-------------
---------+---------+------+------+----------+--------------------------+
| id | select_type | table  | partitions | type  | possible_keys        | key
| key_len | ref  | rows | filtered | Extra                    |
+----+-------------+--------+------------+-------+----------------------+-------------
---------+---------+------+------+----------+--------------------------+
|  1 | SIMPLE      | actor2 | NULL       | range | idx_actor2_last_name |
idx_actor2_last_name | 182     | NULL |   22 |   100.00 | Using where; Using index |
+----+-------------+--------+------------+-------+----------------------+-------------
---------+---------+------+------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)
```

Creating a primary index

```
create index primary key on actor2 (actor_id)
explain select actor_id from actor2 where actor_id > 2
```

Using an index for last_name

```
drop table if exists actor2;
create table actor2 as select * from actor;
create index idx_actor2_last_name on actor2 (last_name);
explain select * from actor2 where last_name like 'B%';
+----+-------------+--------+------------+-------+----------------------+-------------
---------+---------+------+------+----------+-----------------------+
| id | select_type | table  | partitions | type  | possible_keys        | key
| key_len | ref  | rows | filtered | Extra                 |
+----+-------------+--------+------------+-------+----------------------+-------------
---------+---------+------+------+----------+-----------------------+
|  1 | SIMPLE      | actor2 | NULL       | range | idx_actor2_last_name |
idx_actor2_last_name | 182     | NULL |   22 |   100.00 | Using index condition |
+----+-------------+--------+------------+-------+----------------------+-------------
---------+---------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)
```

Never use a function in where

Why ?

```
Step 1: MySQL needs to retrieve every row
Step 2: run function
--> so, no index can be used
```

Example

```
drop table if exists actor2;
create table actor2 as select * from actor;
create index idx_actor2_last_name on actor2 (last_name);
explain select * from actor2 where last_name like
concat(substring(first_name,1,1),'%');
```

Index is always read from left to right

```
## so the index cannot be used if we ask for last_name
drop table if exists actor2;
create table actor2 as select * from actor;
create index idx_actor2_first_name_last_name on actor2 (first_name,last_name);
explain select * from actor2 where last_name like 'B%';
```

```
##
explain select * from actor2 where first_name like 'B%';
```

profiling-get-time-for-execution-of.query

- Get better values, how long queries take

Example

```
set profiling = 1
-- Step 2 - Execute query
select last_name as gross from donors where last_name like lower('WILLI%')

## Step 3 - Show profiles
show profiles;
+----------+------------+-----------------------------------------------------------------
----------------------+
| Query_ID | Duration   | Query
|
+----------+------------+-----------------------------------------------------------------
----------------------+
|        1 | 0.01993525 | select last_name as gross from donors where last_name like
lower('WILLI%')        |
4 rows in set, 1 warning (0.00 sec)

## Step 4 - Show profile for a specific query
mysql> show profile for query 1;
+----------------------+----------+
| Status               | Duration |
+----------------------+----------+
| starting             | 0.000062 |
| checking permissions | 0.000006 |
| Opening tables       | 0.000021 |
| init                 | 0.000017 |
| System lock          | 0.000007 |
| optimizing           | 0.000007 |
| statistics           | 0.000083 |
| preparing            | 0.000012 |
| executing            | 0.000004 |
| Sending data         | 0.022251 |
| end                  | 0.000005 |
| query end            | 0.000008 |
| closing tables       | 0.000007 |
| freeing items        | 0.001792 |
| cleaning up          | 0.000016 |
+----------------------+----------+
15 rows in set, 1 warning (0.00 sec)
```

Kein function in where verwenden

1. No function in where (column_name)

```
## Never use a function for the column name in where
## e.g.
select * from donors where upper(last_name) like 'Willia%'
```

Why ?

- Not index can be used

```
## Not filtering possible by indx -> possible_keys -> NULL
 explain select last_name from donors where upper(last_name) like 'WILLI%';
+----+-------------+--------+------------+-------+---------------+-------------------
+---------+------+--------+----------+--------------------------+
| id | select_type | table  | partitions | type  | possible_keys | key               |
key_len | ref  | rows   | filtered | Extra                    |
+----+-------------+--------+------------+-------+---------------+-------------------
+---------+------+--------+----------+--------------------------+
|  1 | SIMPLE      | donors | NULL       | index | NULL          | donors_donor_info |
687     | NULL | 701948 |   100.00 | Using where; Using index |
+----+-------------+--------+------------+-------+---------------+-------------------
+---------+------+--------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)
```

Optimizer-hints (and why you should not use them)

Tell the optimizer what to do and what not to do

- https://dev.mysql.com/doc/refman/5.7/en/optimizer-hints.html#optimizer-hints-syntax

Query-Plans aka Explains

- Query Plans are the same as Query Execution Plans (QEP's)
- You will see the Query Plan's with explain

Example

```
mysql> explain select * from recipients where recipient_id > 1 and recipient_id < 5;
+----+-------------+------------+------------+-------+---------------+---------+------
---+------+------+----------+-------------+
| id | select_type | table      | partitions | type  | possible_keys | key     |
key_len | ref  | rows | filtered | Extra       |
+----+-------------+------------+------------+-------+---------------+---------+------
---+------+------+----------+-------------+
|  1 | SIMPLE      | recipients | NULL       | range | PRIMARY       | PRIMARY | 4
| NULL |    1 |   100.00 | Using where |
+----+-------------+------------+------------+-------+---------------+---------+------
---+------+------+----------+-------------+
1 row in set, 1 warning (0.01 sec)
```

Output-Format json

```
-- includes costs
EXPLAIN format=json SELECT * from audit_log WHERE yr in (2011,2012);
```

Select_Type

- simple = one table

Types (in order of performance

system

```
Only one row in table is present (only one insert)
```

const only one result

```
 EXPLAIN select contribution_id  from contributions where contribution_id = 262611;
+----+-------------+---------------+------------+-------+---------------+---------+---
------+-------+------+----------+-------------+
| id | select_type | table         | partitions | type  | possible_keys | key     |
key_len | ref   | rows | filtered | Extra       |
+----+-------------+---------------+------------+-------+---------------+---------+---
------+-------+------+----------+-------------+
|  1 | SIMPLE      | contributions | NULL       | const | PRIMARY       | PRIMARY | 4
| const |    1 |   100.00 | Using index |
+----+-------------+---------------+------------+-------+---------------+---------+---
------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

ALL - Full table scan. (slowest)

EXPLAIN select * from contributions where vendor_last_name like 'W%'; +----+-------------+---------------+------------+------+---------------+------+---------+------+---------+----------+-------------+ | id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra | +----+-------------+---------------+------------+------+---------------+------+---------+------+---------+----------+-------------+ | 1 | SIMPLE | contributions | NULL | ALL | NULL | NULL | NULL | NULL | 2028240 | 11.11 | Using where | +----+-------------+---------------+------------+------+---------------+------+---------+------+---------+----------+-------------+ 1 row in set, 1 warning (0.00 sec)

Extra

Using index - cover index is used

```
Looking data in index is sufficient
- no lookup of data on disk is necessary
```

```
mysql> EXPLAIN select contribution_id  from contributions where contribution_id =
262611;
+----+-------------+---------------+------------+-------+---------------+---------+---
------+-------+------+----------+-------------+
| id | select_type | table         | partitions | type  | possible_keys | key     |
key_len | ref   | rows | filtered | Extra       |
+----+-------------+---------------+------------+-------+---------------+---------+---
------+-------+------+----------+-------------+
|  1 | SIMPLE      | contributions | NULL       | const | PRIMARY       | PRIMARY | 4
| const |    1 |   100.00 | Using index |
+----+-------------+---------------+------------+-------+---------------+---------+---
------+-------+------+----------+-------------+
```

```
1 row in set, 1 warning (0.00 sec)

mysql> show warnings;
+-------+------+-----------------------------------------------------------------------------
----------------------------+
| Level | Code | Message
|
+-------+------+-----------------------------------------------------------------------------
----------------------------+
| Note  | 1003 | /* select#1 */ select '262611' AS `contribution_id` from
`contributions`.`contributions` where 1 |
+-------+------+-----------------------------------------------------------------------------
----------------------------+
1 row in set (0.00 sec)
```

Query Pläne und die Key-Länge

Index und Likes

1. like 'Will%' - Index works

explain select last_name from donors where last_name like 'Will%';

2. like '%iams' - Index does not work

```
-- because like starts with a wildcard
explain select last_name from donors where last_name like '%iams';
```

3. How to fix 3, if you are using this often ?

```
## Walkthrough
## Step 1: modify table
alter table donors add last_name_reversed varchar(70) GENERATED ALWAYS AS
(reverse(last_name));
create index idx_last_name_reversed on donors (last_name_reversed);

## besser - Variante 2 - untested
alter table donors add last_name_reversed varchar(70) GENERATED ALWAYS AS
(reverse(last_name)), add index idx_last_name_reversed on donors (last_name_reversed);

## Step 2: update table - this take a while
update donors set last_name_reversed = reversed(last_name)
## Step 3: work with it
select last_name,last_name_reversed from donor where last_name_reversed like
reverse('%iams');
```

```
## Version 2 with pt-online-schema-change
```

Index und Joins

Take a look which order the optimizer uses

With date

```
-- Using a date which has no index
-- Needs to do a table scan
explain select c.* from contributions c join donors d using (donor_id) join recipients
r using (recipient_id) where c.date_recieved > '1999-12-01' and c.date_recieved <
'2000-07-01';
+----+-------------+-------+------------+--------+-------------------------+---------
+---------+-------------------------------+---------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys           | key     |
key_len | ref                           | rows    | filtered | Extra       |
+----+-------------+-------+------------+--------+-------------------------+---------
+---------+-------------------------------+---------+----------+-------------+
|  1 | SIMPLE      | c     | NULL       | ALL    | donor_idx,recipient_idx | NULL    |
NULL    | NULL                          | 2028240 |    11.11 | Using where |
|  1 | SIMPLE      | r     | NULL       | eq_ref | PRIMARY                 | PRIMARY |
4       | contributions.c.recipient_id |       1 |   100.00 | Using index |
|  1 | SIMPLE      | d     | NULL       | eq_ref | PRIMARY                 | PRIMARY |
4       | contributions.c.donor_id     |       1 |   100.00 | Using index |
+----+-------------+-------+------------+--------+-------------------------+---------
+---------+-------------------------------+---------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

60626 rows in set (7.22 sec)
```

With date and filter on donor

```
 explain select c.*,d.last_name from contributions c join donors d using (donor_id)
join recipients r using (recipient_id)
where c.date_recieved > '1999-12-01' and c.date_recieved < '2000-07-01' and
d.last_name like 'A%';
+----+-------------+-------+------------+--------+-------------------------+--------
-----------+---------+-------------------------------+-------+----------+--------------
------------+
| id | select_type | table | partitions | type   | possible_keys           | key
| key_len | ref                           | rows  | filtered | Extra
|
+----+-------------+-------+------------+--------+-------------------------+--------
-----------+---------+-------------------------------+-------+----------+--------------
------------+
|  1 | SIMPLE      | d     | NULL       | range  | PRIMARY,donors_donor_info |
donors_donor_info | 213     | NULL                          | 65894 |   100.00 | Using
where; Using index |
|  1 | SIMPLE      | c     | NULL       | ref    | donor_idx,recipient_idx   |
donor_idx         | 5       | contributions.d.donor_id     |     2 |    11.11 | Using
where              |
|  1 | SIMPLE      | r     | NULL       | eq_ref | PRIMARY                   | PRIMARY
| 4       | contributions.c.recipient_id |     1 |   100.00 | Using index
|
+----+-------------+-------+------------+--------+-------------------------+--------
-----------+---------+-------------------------------+-------+----------+--------------
```

```
------------+
3 rows in set, 1 warning (0.00 sec)
```

With date and filter on donor, less specific

```
select c.*,d.* from contributions c join donors d using (donor_id) join recipients r
using (recipient_id) where c.date_recieved > '1999-12-01' and c.date_recieved < '2000-
07-01' and d.last_name like 'A%';
 explain select c.*,d.* from contributions c join donors d using (donor_id) join
recipients r using (recipient_id) where c.date_recieved > '1999-12-01' and
c.date_recieved < '2000-07-01' and d.last_name like 'A%';
+----+-------------+-------+------------+--------+---------------------------+--------
-----------+---------+-------------------------------+-------+----------+--------------
---------+
| id | select_type | table | partitions | type   | possible_keys             | key
| key_len | ref                           | rows  | filtered | Extra                 |
+----+-------------+-------+------------+--------+---------------------------+--------
-----------+---------+-------------------------------+-------+----------+--------------
---------+
|  1 | SIMPLE      | d     | NULL       | range  | PRIMARY,donors_donor_info |
donors_donor_info | 213     | NULL                          | 65894 |   100.00 | Using
index condition |
|  1 | SIMPLE      | c     | NULL       | ref    | donor_idx,recipient_idx   |
donor_idx         | 5       | contributions.d.donor_id     |     2 |    11.11 | Using
where           |
|  1 | SIMPLE      | r     | NULL       | eq_ref | PRIMARY                   | PRIMARY
| 4       | contributions.c.recipient_id |     1 |   100.00 | Using index           |
+----+-------------+-------+------------+--------+---------------------------+--------
-----------+---------+-------------------------------+-------+----------+--------------
---------+
3 rows in set, 1 warning (0.00 sec)
```

With date and filter on donor and filter on recipient

```
mysql> explain select c.*,d.last_name,r.* from contributions c join donors d using
(donor_id) join recipients r using (recipient_
id) where c.date_recieved > '1999-12-01' and c.date_recieved < '2000-07-01' and
d.last_name like 'A%' and r.name like 'Cit%';
+----+-------------+-------+------------+--------+---------------------------+--------
-------+---------+-------------------------------+------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys             | key
| key_len | ref                           | rows | filtered | Extra       |
+----+-------------+-------+------------+--------+---------------------------+--------
-------+---------+-------------------------------+------+----------+-------------+
|  1 | SIMPLE      | r     | NULL       | ALL    | PRIMARY                   | NULL
| NULL    | NULL                          | 6063 |    11.11 | Using where |
|  1 | SIMPLE      | c     | NULL       | ref    | donor_idx,recipient_idx   |
recipient_idx | 5       | contributions.r.recipient_id |  305 |    11.11 | Using where
|
|  1 | SIMPLE      | d     | NULL       | eq_ref | PRIMARY,donors_donor_info | PRIMARY
| 4       | contributions.c.donor_id     |    1 |     9.39 | Using where |
```

```
+----+-------------+-------+------------+--------+---------------------------+--------
-------+---------+-------------------------------+------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)
```

Find out cardinality without index

Find out cardinality without creating index

```
select count(distinct donor_id) from contributions;
```

```
select count(distinct(vendor_city)) from contributions;
+-------------------------------+
| count(distinct(vendor_city)) |
+-------------------------------+
|                          1772 |
+-------------------------------+
1 row in set (4.97 sec)
```

Index and Functions

No index can be used on an index:

```
explain select * from actor where upper(last_name) like 'A%';
+----+-------------+-------+------------+------+---------------+------+---------+-----
-+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref
| rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+-----
-+------+----------+-------------+
|  1 | SIMPLE      | actor | NULL       | ALL  | NULL          | NULL | NULL    | NULL
|  200 |   100.00 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+-----
-+------+----------+-------------+
```

Workaround with virtual columns (possible since mysql 5.7)

```
## 1. Create Virtual Column with upper
alter table sakila add idx_last_name_upper varchar(45) GENERATED ALWAYS AS
upper(last_name);
## 2. Create an index on that column
create index idx_last_name_upper on actor (last_name_upper);
```

Workaround with persistent/virtual columns (MariaDB)

```
mysql> alter table actor add column last_name_upper varchar(45) as (upper(last_name))
PERSISTENT ;
mysql> insert into actor (first_name,last_name,last_name_upper) values
('Max','Mustermann','MUSTERMANN');
mysql> select * from actor order by actor_id desc limit 1;
```

```
mysql> -- setting index
mysql> create index idx_last_name_upper on actor (last_name_upper);
Query OK, 0 rows affected (0.007 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> -- to use index we need to avoid the function in where
mysql> explain select * from actor where last_name_upper like 'WI%' \G
```

Reference:

- https://dev.mysql.com/doc/refman/5.6/en/innodb-online-ddl.html

Now we try to search the very same

```
explain select * from actor where last_name_upper like 'A%';
+----+-------------+-------+------------+-------+---------------------+---------------
------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys       | key
| key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------------+---------------
------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | actor | NULL       | range | idx_last_name_upper |
idx_last_name_upper | 183     | NULL |    7 |   100.00 | Using where |
+----+-------------+-------+------------+-------+---------------------+---------------
------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

Tools

Percona Toolkit - only pt-query-digest

Walkthrough (Windows)

```
## 1. Install strawberry perl
https://strawberryperl.com/

## 2. Download pt-query-digest and save with .pl suffix
https://www.percona.com/get/pt-query-digest

## 3. copy file to bin - folder of mariadb

## 4. Open mariadb Command Prompt
## Navigate to data - dir

## 5. as Admin: Execute once to get right connection to perl
pt-query-digest.pl <name-of-slow-query-log> > analyse.txt

## 6. once more  5.
pt-query-digest.pl <name-of-slow-query-log> > analyse.txt

## 7. Digest analyse.txt and be happy or not ;o)
```

```
## Referenz
## http://www.jonathanlevin.co.uk/2012/01/query-digest-on-windows.html
```

pt-query-digest - analyze slow logs

Requires

- Install percona-toolkit

Usage

```
## first enable slow_query_log
set global slow_query_log = on
set global long_query_time = 0.2
## to avoid, that i have to reconnect with new session
set session long_query_time = 0.2

## produce slow query - for testing
select * from contributions where vendor_last_name like 'W%';
mysql > quit

##
cd /var/lib/mysql
## look for awhile wih -slow.log - suffix
pt-query-digest mysql-slow.log > /usr/src/report-slow.txt
less report-slow.txt
```

pt-online-schema-change howto

Requirements

- Install percona-toolkit

Documentation

- https://www.percona.com/doc/percona-toolkit/3.0/pt-online-schema-change.html

What does it do ?

```
## Altering table without blocking them
## Do a dry-run beforehand
pt-online-schema-change --alter "ADD INDEX idx_city (city)" --dry-run
D=contributions,t=donors
##
pt-online-schema-change --alter "ADD INDEX idx_city (city)" --execute
D=contributions,t=donors
```

With foreign - keys

```
 # first try
 pt-online-schema-change --alter "add column remark varchar(150)" D=sakila,t=actor --
alter-foreign-keys-method=auto --dry-run
 # then run
```

```
 pt-online-schema-change --alter "add column remark varchar(150)" D=sakila,t=actor --
alter-foreign-keys-method=auto --execute
```

Example sys-schema and Reference

Install under mariadb 10.5

```
apt install git
cd /usr/src
git clone https://github.com/jmetzger/mariadb-sys.git
cd mariadb-sys
mysql < ./sys_10.sql
```

Examples

```
mysql> select * from sys.host_summary\G
*************************** 1. row ***************************
                   host: localhost
             statements: 1347
     statement_latency: 7.55 m
 statement_avg_latency: 336.50 ms
           table_scans: 15
              file_ios: 612857
       file_io_latency: 1.66 m
   current_connections: 1
     total_connections: 7
          unique_users: 1
        current_memory: 0 bytes
total_memory_allocated: 0 bytes
1 row in set (0.01 sec)
```

Ref:

- https://github.com/mysql/mysql-sys/blob/master/README.md

Beispieldaten

Verleihdatenbank - sakila

```
## Runterladen der Sakila DB
## von https://dev.mysql.com/doc/index-other.html
## die zip - Version

## Auf dem Desktop entpacken

## in das Verzeichnis reinwechseln durch kopieren des Pfades
cd C:\Users\vgh-MariaDB\Desktop\sakila-db
mysql -uroot -ppassword < sakila-db\sakila-schema.sql
mysql -uroot -ppassword < sakila-db\sakila-data.sql
```

Setup training data "contributions"

Walkthrough (Debian/Ubuntu)

- Complete process takes about 10 minutes

```
cd /usr/src;
apt update; apt install git;
git clone https://github.com/jmetzger/dedupe-examples.git;
cd dedupe-examples;
cd mysql_example;
## Eventually you need to enter (in mysql_example/mysql.cnf)
## Only necessary if you cannot connect to db by entering "mysql"
## password=<your_root_pw>
./setup.sh
```

Managing big tables

Using Partitions - Walkthrough

Walkthrough

```
##
## EXPLAIN PARTITIONS
##
DROP TABLE IF EXISTS audit_log;
CREATE TABLE audit_log (
  yr    YEAR NOT NULL,
  msg   VARCHAR(100) NOT NULL)
ENGINE=InnoDB
PARTITION BY RANGE (yr) (
  PARTITION p0 VALUES LESS THAN (2010),
  PARTITION p1 VALUES LESS THAN (2011),
  PARTITION p2 VALUES LESS THAN (2012),
  PARTITION p3 VALUES LESS THAN MAXVALUE);
INSERT INTO audit_log(yr,msg) VALUES (2005,'2005'),(2006,'2006'),(2011,'2011'),
(2020,'2020');
EXPLAIN PARTITIONS SELECT * from audit_log WHERE yr in (2011,2012)\G
```

Example with years

```
CREATE TABLE audit_log2 (   yr    YEAR NOT NULL,   msg   VARCHAR(100) NOT NULL)
ENGINE=InnoDB PARTITION BY RANGE (yr) (   PARTITION p2009 VALUES LESS THAN (2010),
PARTITION p2010 VALUES LESS THAN (2011),   PARTITION p2011 VALUES LESS THAN (2012),
PARTITION p_current VALUES LESS THAN MAXVALUE);
INSERT INTO audit_log2(yr,msg) VALUES (2005,'2005'),(2006,'2006'),(2011,'2011'),
(2012,'2012');

EXPLAIN PARTITIONS SELECT * from audit_log2 WHERE yr = 2012;

ALTER TABLE audit_log2 REORGANIZE PARTITION p_current INTO (
   PARTITION p2012 VALUES LESS THAN (2013),
```

```
   PARTITION p_current VALUES LESS THAN MAXVALUE);
)

--  Where is data now saved
EXPLAIN PARTITIONS SELECT * from audit_log2 WHERE yr = 2012;
```

Eine bestehende große Tabelle partitionieren (mariadb)

```
Variante 1:
## Wichtig vorher Daten sichern

ALTER TABLE `audit_log3` PARTITION BY RANGE (`yr`) ( PARTITION p2009 VALUES LESS THAN
(2010) ENGINE=InnoDB, PARTITION p2010 VALUES LESS THAN (2011) ENGINE=InnoDB, PARTITION
p2011 VALUES LESS THAN (2012) ENGINE=InnoDB, PARTITION p2012 VALUES LESS THAN (2013)
ENGINE=InnoDB, PARTITION p_current VALUES LESS THAN MAXVALUE ENGINE=InnoDB )

Variante 2:
Daten ausspielen ohne create (dump)  + evtl zur sicherheit Struktur-Dump
Tabelle löschen
Daten ohne Struktur einspielen
```

Ref:

- https://mariadb.com/kb/en/partition-maintenance/

Replication

Aufbau Master/Slave - Replication

Replikation mit GTID

Step 0.5a: Installation on ubuntu/debian

```
apt update
apt install mariadb-backup
## check if available
mariabackup --version

## prepare for mariabackup if you use it with root and with unix_socket
/root/.my.cnf
[mariabackup]
user=root
```

Step 1: mariabackup on master

```
mkdir /backups
## target-dir needs to be empty or not present
mariabackup --target-dir=/backups/20210121 --backup
## apply ib_logfile0 to tablespaces
## after that ib_logfile0 ->  0 bytes
mariabackup --target-dir=/backups/20210121 --prepare
```

Step 2: Transfer to new slave (from master)

```
## root@master:
rsync -e ssh -avP /backups/20210121 student@10.10.9.144:/home/student/
```

Step 3: Setup replication user on master

```
## as root@master
##mysql>
CREATE USER repl@'10.10.9.%' IDENTIFIED BY 'password';
GRANT REPLICATION SLAVE ON *.*  TO 'repl'@'10.10.9.%';
```

Step 3a (Optional): Test repl user (connect) from slave

```
## as root@slave
## you be able to connect to
mysql -urepl -p -h10.10.9.110
## test if grants are o.k.
show grants
```

Step 4a: Set server-id on master -> 1

```
[mysqld]
server-id=1

systemctl restart mariadb
###
```

Step 4b: Set server-id on slave -> 3 + same config as server 1 + log_slave_update

```
[mysqld]
server-id               = 3
## activate master bin log, if this slave might be a master later
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_format = ROW
log_slave_update = 1

systemctl restart mariadb
### auf dem master config mit rsync rüberschrieben
### root@master
rsync -e ssh -avP /etc/mysql/mariadb.conf.d/z_uniruhr.cnf kurs@10.10.9.144:/home/kurs/
```

Step 5: Restore Data on slave

```
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup
mariabackup --target-dir=/home/student/20210121 --copy-back
chown -R mysql:mysql /var/lib/mysql
systemctl start mariadb
```

Step 6: master.txt for change command

```
## root@slave
$ cat xtrabackup_binlog_info
mariadb-bin.000096 568 0-1-2

SET GLOBAL gtid_slave_pos = "0-1-2";
## /root/master.txt
## get information from master-databases.sql dump
CHANGE MASTER TO
   MASTER_HOST="192.168.56.102",
   MASTER_PORT=3306,
   MASTER_USER="repl",
   MASTER_PASSWORD="password",
   MASTER_USE_GTID=slave_pos;

mysql < master.txt
## or: copy paste into mysql>

## mysql>
start slave

## in mysql -> show slave status
mysql>show slave status
## Looking for
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

Walkthrough

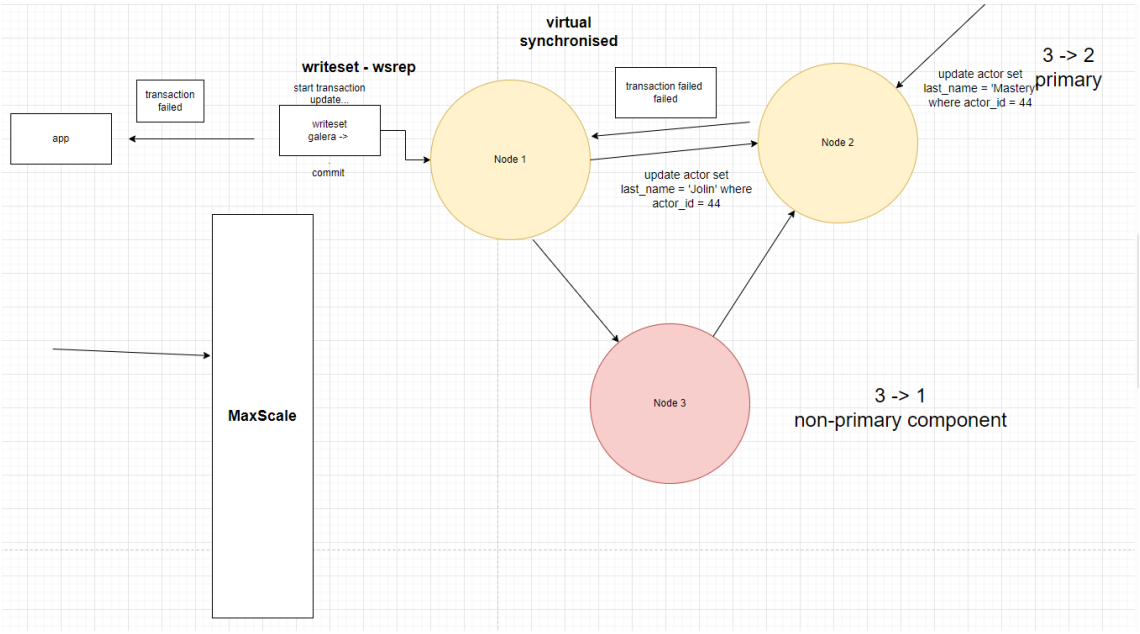
https://mariadb.com/kb/en/setting-up-a-replication-slave-with-mariabackup/

Replikation Read/Write - Split:

- https://proxysql.com/blog/configure-read-write-split/

MariaDB (Galera Cluster) - Linux Only !!

Aufbau Galera Cluster



Fragen und Antworten

Fragen und Antworten

1. Archive Data

```
https://www.percona.com/doc/percona-toolkit/LATEST/pt-archiver.html
```

2. Does innodb do defragmentation by itself ?

```
## Some background while doing research.
## Nil performance benefits of defragmentation in index.
https://stackoverflow.com/questions/48569979/mariadb-table-defragmentation-using-
optimize
```

3. Defragmentation

```
## Optimize table
ALTER TABLE contributions engine = InnoDB  # Das gleiche wie OPTIMIZE TABLE


## mariadb has a patch for defragmentation
https://mariadb.org/defragmenting-unused-space-on-innodb-tablespace/

## alter table xyz engine=InnoDB - defragements
## but is also invasive.
## with ibdata1 innodb_file_per_table it lets the size grow
```

4. Is it possible to do select, update, deletes without using innodb_buffer in specific

```
No, this is not possible
```

8. MariaDB (Features/Vorteile)

- flashback
- Verschlüsselung von Tabellen // mariabackup
- Einige Storage Engine (Aria -> MyISAM - crash-recovery)
- JSON anders implementiert
- galera
- feature: defragementation

```
MysqL 8 does not:
decode
set profiling (still available but deprecated )
```

9. Select without locking

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED ;
BEGIN ;
SELECT * FROM TABLE_NAME ;
COMMIT ;
```

Projektarbeit/-optimierung

Praktisch Umsetzung in 3-Schritten

Schritt 1: Hardware

```
1) Arbeitsspeicher erhöhen/nachkaufen und mindestens 50% der Nutzdaten
2) Extra Maschine für Applikation und für Datenbank (möglichst gute Anbindung
untereinander)
   d.h. Maschinen stecken am besten gleichen Serverschrank
```

Schritt 2: Konfiguration

1. Optimierung des InnoDB Buffers - Größe
2. innodb_flush_log_at_trx_commit auf 0 setzen (jede Sekunde statt bei jedem Commit)

Schritt 3: Optimierung der Anfragen

1. Vorbereitung Ausgabe Slow Log für die Analyse
2. Installation percona-toolkit
3. Analyse slow-log-file mit pt-query-digest
4. Analyse langsamer Queries mit explain und Index setzen
    - Explain inkl. JSON-Format
    - Index setzten - Teil 1
    - Index und Joins
    - Function in Wheres vermeiden
    - Workaround für Funktionen - Virtual Column

Extra: Der Ausweg bei großen Tabellen

1. Falls es keine andere Lösung gibt, könnte u.U. Partitionierung helfen. Hier

Monitoring

Was sollten wir monitoren ?

What to monitor

System

- Last auf dem System
- Festplatte (z.B. 85% voll ?)

Erreichbarkeit

- Server per ping erreichen (mysqladmin ping -h ziel-ip)
- Einlogbar ? (myadmin ping -h ziel-ip -u control_user)

Platte aka IO-Subsystem (iostats)

- http://schulung.t3isp.de/documents/pdfs/mysql/mysql-performance.pdf

--	--	--
Read/Write requests	IOPS (Input/Output operations per second)	--
Average IO wait	Time that queue operations have to wait for disk access	--
Average Read/Write time	Time it takes to finish disk access operations (latency)	--
Read/Write bandwidth	Data transfer from and towards your disk	--

General mysql metrics

```
mysql -E -e "select variable_value from information_schema.session_status where
variable_name = 'uptime'";

# max connections
MariaDB [(none)]> show status like 'max_used_connections';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Max_used_connections | 1     |
+----------------------+-------+
1 row in set (0.001 sec)

MariaDB [(none)]> show variables like 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 151   |
+-----------------+-------+
1 row in set (0.001 sec)
```

```
mysqladmin status
## you will find uptime here in seconds
```

Metric	Comments	Suggested Alert
Uptime	Seconds since the server was started. We can use this to detect respawns.	When uptime is < 180. (seconds)
Threads_connected	Number of clients currently connected. If none or too high, something is wrong.	None
Max_used_connections	Max number of connections at a time since server started. (max_used_connections / max_connections) indicates if you could run out soon of connection slots.	When connections usage is > 85%.
Aborted_connects	Number of failed connection attempts. When growing over a period of time either some credentials are wrong or we are being attacked. show status like 'Aborted_connects'	When aborted connects/min > 3.

InnoDB (show status like ...)

Metric	Coments	Suggested Alert
Innodb_row_lock_waits	Number of times InnoDB had to wait before locking a row.	None
Innodb_buffer_pool_wait_free	Number of times InnoDB had to wait for memory pages to be flushed. If too high, innodb_buffer_pool_size is too small for current write load.	None

Query tracking

Metric	Comments	Suggested Alert
Slow_queries	Number of queries that took more than long_query_time seconds to execute. Slow queries generate excessive disk reads, memory and CPU usage. Check slow_query_log to find them.	None
Select_full_join	Number of full joins needed to answer queries. If too high, improve your indexing or database schema.	None
Created_tmp_disk_tables	Number of temporary tables (typically for joins) stored on slow spinning disks, instead of faster RAM.	None
(Full table scans) Handler_read% Number of times the system reads the first row of a table		

index. (if 0 a table scan is done - because no key was read). Sequential reads might indicate a faulty index. None		

Track Error

- error_log mariadb

Dokumentation

MySQL - Performance - PDF

- http://schulung.t3isp.de/documents/pdfs/mysql/mysql-performance.pdf

Effective MySQL

- https://www.amazon.com/Effective-MySQL-Optimizing-Statements-Oracle/dp/0071782796

MariaDB Downloaden

- https://mariadb.org/download/

MariaDB - Releases - including long - term releases

- https://mariadb.com/kb/en/mariadb-server-release-dates/

Effective MySQL

- https://www.amazon.com/Effective-MySQL-Optimizing-Statements-Oracle/dp/0071782796