MariaDB Performance Training (deutsch)

Agenda

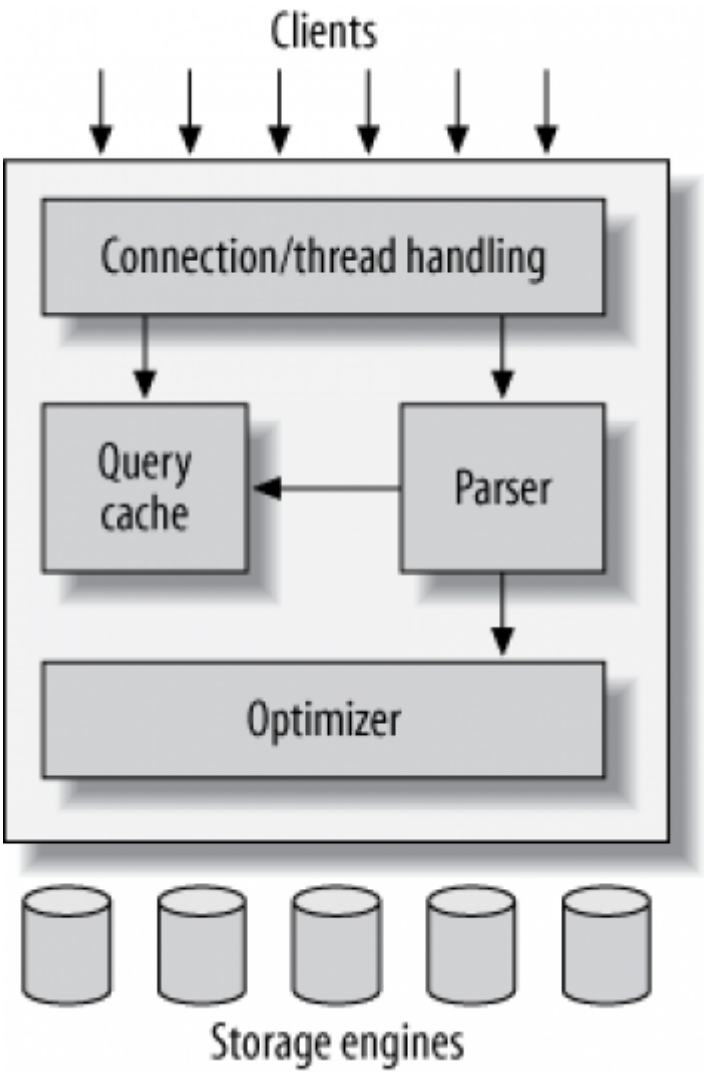
1. Performance / Theorie - Aspekte der MariaDB - Architektur
    - Architektur Server (Schritte)
    - CPU oder io-Last klären
    - Storage Engines
    - InnoDB - Struktur
    - InnoDB - Optimierung
    - InnoDB - innodb_log_file_size berechnen
    - Query - Cache
    - 3-Phasen-Datengröße
2. Installation
    - Installation (Rocky/Redhat) - 10.6
    - Open Mariadb to the outside world - port and firewall
3. Konfiguration
    - Slow query log
4. Administration
    - Standard storage engine bestimmen
    - Show status
    - Server System Variablen - show variables
    - systemctl/jorunalctl - Server starten,stoppen/Logs
    - User verwalten
5. Joins
    - Joins - Overview
6. Performance und Optimierung von SQL-Statements
    - 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
    - index and group by
    - forcing good index
    - forcing bad index
7. Performance - Views
    - Performance - Views
8. Performance Schema
    - Performance Schema - slow queries
9. Performance Metrics from status

- Performance Metrics from status

10. Locks
    - How does innodb implicit locking work
    - Exercise - identify deadlocks
11. Tools
    - Percona Toolkit
    - pt-query-digest - analyze slow logs
    - pt-online-schema-change howto
    - Example sys-schema and Reference
12. Beispieldaten
    - Verleihdatenbank - sakila
    - Setup training data "contributions"
13. Managing big tables
    - Using Partitions - Walkthrough
14. Replication
    - Replikation mit GTID
    - Replikation Read/Write - Split:
15. Fragen und Antworten
    - Fragen und Antworten
16. Projektarbeit/-optimierung
    - Praktisch Umsetzung in 3-Schritten
17. Dokumentation (Performance)
    - MySQL - Performance - PDF
    - Effective MySQL
    - Analyze statt Explain
18. Dokumentation (Releases)
    - Identify Long-Term Support Releases
19. Dokumentation (Server System Variables / Alter inplace)
    - Server System Variables
    - Inplace Alter
20. Dokumentation (Indexes) * Recipes for settings indexes
21. Dokumenation (Virtual Columns / Persistentn)
    - Persistent / Virtual Columns
22. Backup/Restore
    - Mariabackup

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 MariaDB
- 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 / PartitionX
- (Federated/FederatedX)

Comparison MyISAM vs. InnoDB

On Detail: MyISAM - Storage Engine

Features

- table locks
- Locks are done table-wide
- no automatic data-recovery
- you can loose more data on crashes than with e.g. InnoDB
- no transactions
- only indices are save in memory through MySQL
- compact saving (data is saved really dense)
- table scans are quick

In Detail: InnoDB - Storage Engine

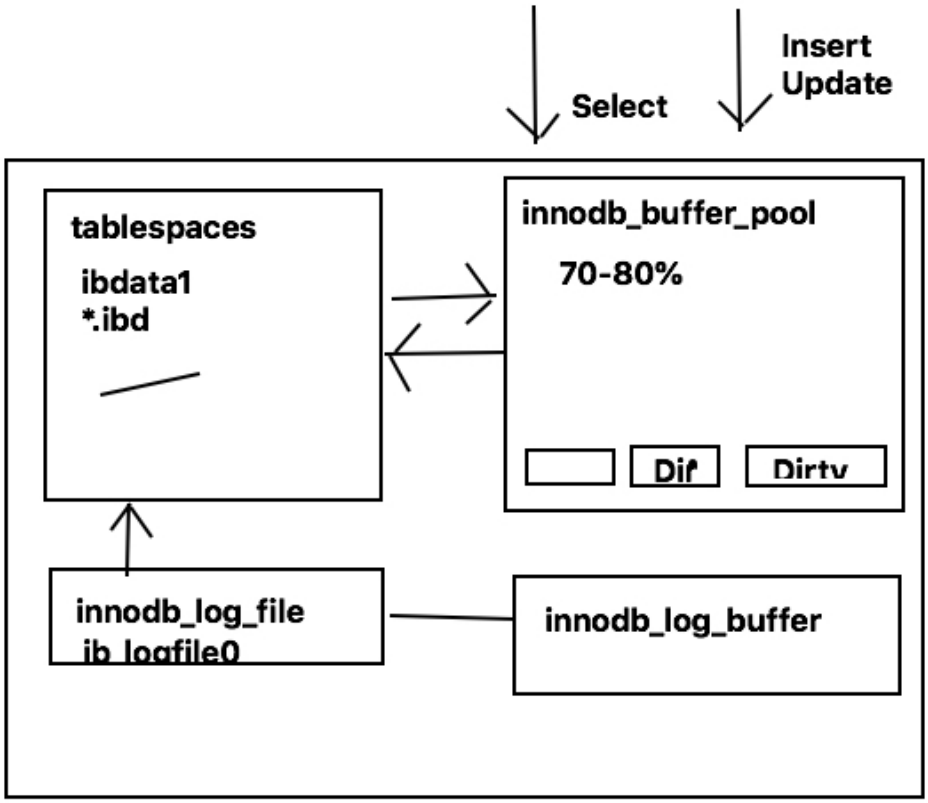
Features

- support hot backups (because of transactions)
- transactions are supported
- foreign keys are supported
- row-level locking
- multi-versioning
- indexes refer to the data through primary keys
- indexes can quickly get huge in size
  - if size of primary index is not small

Difference MyISAM / Aria

- Crash Recovery (only difference)

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

```
pager grep -i 'free buffers'
show engine innodb status \G
Free buffers       7905
1 row in set (0.00 sec)
```

```
## 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)
```

Overview innodb server variables / settings

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

Change innodb_buffer_pool

```
## /etc/mysql/mysql.conf.d/mysqld.cnf
## 70-80% of memory on dedicated mysql
[mysqld]
innodb-buffer-pool-size=6G

##
systemctl restart mysql

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

innodb_flush_method

```
Ideally O_DIRECT on Linux, but please test it, if it really works well.
```

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.conf

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

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

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
```

InnoDB - innodb_log_file_size berechnen

Exercise (simple)

```
pager grep sequence;
show engine innodb status;
select sleep(60);
show engine innodb status;
-- pager zuruecksetzen
pager;
```

```
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/

Exercise 2: with sakila

```
Open 2 sessions
Session 1: mysql to measure
Session 2: import sakila
```

Step 1: In Session 1 (measure)

```
mysql
```

```
pager grep sequence;
show engine innodb status;
select sleep(120);
show engine innodb status;
```

Step 2: In Session 2 (import sakila)

```
cd /usr/src/sakila-db
mysql < sakila-schema.sql; mysql < sakila-data.sql;
```

Step 3: In Session 1 (measure) - Calculate values

```
-- Calculate values
-- second-value-from-step1 - first-value-from-step1
-- select (second_value - first_value) / 1024 / 1024 as MB_per_2min


pager;
## eg. 312MB
select (3838334638 - 3836410803) / 1024 / 1024 as MB_per_2min;
exit
```

Step 4: Adjust config accordingly

```
## Ubuntu / Debian
cd /etc/mysql/mariadb.conf.d/
nano 50-server.cnf
```

```
## mysqld section
[mysqld]
## other settings
innodb-log-file-size=312M
```

```
systemctl restart mariadb
mysql
```

```
select @@innodb_log_file_size
## oder
show variables like '%log_file%';
exit
```

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.
```

Installation

Installation (Rocky/Redhat) - 10.6

Setup repo and install

- https://downloads.mariadb.org/mariadb/repositories/

```
sudo su -
cd /etc/yum.repos.d
nano mariadb.repo
```

```
## MariaDB 10.6 RedHatEnterpriseLinux repository list - created 2025-01-08 10:52 UTC
## https://mariadb.org/download/
[mariadb]
name = MariaDB
## rpm.mariadb.org is a dynamic mirror if your preferred mirror goes offline. See
https://mariadb.org/mirrorbits/ for details.
## baseurl = https://rpm.mariadb.org/10.6/rhel/$releasever/$basearch
baseurl = https://ftp.agdsn.de/pub/mirrors/mariadb/yum/10.6/rhel/$releasever/$basearch
## gpgkey = https://rpm.mariadb.org/RPM-GPG-KEY-MariaDB
gpgkey = https://ftp.agdsn.de/pub/mirrors/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck = 1
```

```
dnf install -y MariaDB-server MariaDB-client
```

Check if running and enabled

```
systemctl status mariadb
systemctl start mariadb
systemctl enable mariadb
systemctl status mariadb
## enabled, wenn in Zeile 2 mariadb.service;enabled; auftaucht
```

Secure installation

```
mariadb-secure-installation
```

OR: if not present before 10.4

mysql_secure_installation

```
### Open Mariadb to the outside world - port and firewall


### Step 1: Is services listening to the outside world (Ubuntu, Redhat, Rocky) ?
```

lsof -i | grep mariadb

localhost means it does NOT listen to the outside now

mariadbd 5208 mysql 19u IPv4 56942 0t0 TCP localhost:mysql (LISTEN)

netstat -tupel

or

netstat -an

```
### Step 2: is firewall open on port 3306 or service mysql (Rocky / RHEL)
```

firewall-cmd --list-all

services: cockpit dhcpv6-client ssh

firewall-cmd --add-service=mysql --zone=public firewall-cmd --runtime-to-permanent

```
### How to fix (Ubuntu -> Mariadb Foundation)
```

nano /etc/mysql/mariadb.conf.d/50-server.cnf

```

```

In Section [mysqld]

Change bind-address to -> bind-address = 0.0.0.0

[mysqld] bind-address = 0.0.0.0

```

```

systemctl restart mariadb systemctl status mariadb lsof -i

```
### Can I bind to multiple addresses  ?
```

from 10.11 it is possible, before not, you have to use 0.0.0.0

Starting with MariaDB 10.11, a comma-separated list of addresses to bind to can be given.

```
 * https://mariadb.com/kb/en/server-system-variables/#bind_address

## Konfiguration

### Slow query log


### Variante 1: Langsame Queries loggen

#### Step 1: Config vorbereiten
```

nano /etc/my.cnf.d/server.cnf

```

```

unterhalb von [mysqld]

guter richtwert

long_query_time=0.5

nur fürs Training um Daten zu bekommen

long_query_time=0.000001

```

```

systemctl restart mariadb

```
#### Step 2: Während der Laufzeit global aktivieren
```

mysql

```

```

set global slow_query_log = 1;

```

```

Wichtig: Ist jetzt erst beim nächsten Connection mit einer Session für diese Session aktiv

```
#### Step 3: Zusätzliche sinnvolle Einstellungen vornehmen für das slow_query_log


##### bis Version 10.4 (inkl.)
```

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'

```
##### ab Version 10.6. (auf Rocky / RHEL)
```

Empfehlung in der config

nano /etc/my.cnf.d/server.cnf

```

```

alle queries loggen die kein index haben

log-queries-not-using-indexes

das nur für production, wenn wirklich dort getracked werden soll

nur jede 20. mitloggen

log-slow-rate-limit=20

hier nimmt er alle optionen als Zusatzinformationen

auch neu (ab. 10.6: engine, Innodb

log-slow-verbosity=All

```

```

systemctl restart mariadb

```

```

```
### Variante 1: Aktivieren (minimum)
```

auch direkt in 50-server.cnf möglich

mysql>set global long_query_time=0.5 # 0,5 Sekunden. Alles was >= 0,5 sekunden dauert, wird geloggt mysql>set session long_query_time=0.5 mysql>set global slow_query_log=1 mysql>set session slow_query_log=1

```
### 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;

```
### Ref:
 * https://mariadb.com/kb/en/slow-query-log-overview/
## Administration
### 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'

```
### 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

```
### systemctl/jorunalctl - Server starten,stoppen/Logs
```

systemctl TAB TAB -> zeigt alle Unterbefehle an systemctl status mariadb.service systemctl start mariadb # .service darf man weglassen bei start/status/stop/restart systemctl stop mariadb systemctl restart mariadb

```
```

journalctl -u mariadb.service # Zeigt alle Logs an seit dem letzten Serverstart (Debian 10)

```
### 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;

```
## Joins

### Joins - Overview

## Performance und Optimierung von SQL-Statements

### 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

```
### Knoten im Taschentuch

#### bei Extras

##### Using Index - Cover Index
```

z.B. select actor_id, last_name from actor;

Using index - nur den Index verwendet und kein ansonsten holt

```
##### Using Index Condition = Index Condition Pushdown (ICP)
```

explain extended select first_name, actor_id from actor where last_name like 'A%';

```
![image](https://github.com/user-attachments/assets/c9efa8ea-de10-4f3d-ac9e-
825e92464129)

 * https://mariadb.com/kb/en/index-condition-pushdown/


### 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)

```
### index and group by


![image](https://github.com/user-attachments/assets/e45c829f-8456-47bc-8f33-05cce6e05b88)


![image](https://github.com/user-attachments/assets/0d03e530-7e2e-4bcb-8167-ae5600fe3287)
```

Variante 1: explain SELECT last_name, COUNT(*) AS 'count' FROM actor WHERE first_name LIKE 'S%' GROUP BY last_name

Variante 2: explain SELECT first_name, COUNT(*) AS 'count' FROM actor WHERE last_name LIKE 'S%' GROUP BY first_name

```
### forcing good index


### In Order By an index is often not used .

  * https://mariadb.com/kb/en/index-hints-how-to-force-query-plans/#force-index-forcing-an-index

### Use data from contributions

#### Walkthrough
```

mysql contributions

```

```

Set an index

create index idx_contributions_date_recieved on contributions (date_recieved);

```

```

order by without index (decides to use table scan)

explain select * from contributions order by date_recieved desc limit 100000;

```
![image](https://github.com/user-attachments/assets/f5947f26-d2be-4c68-9b80-8ef14933327e)
```

explain select * from contributions force index (idx_contributions_date_recieved) order by date_recieved desc limit 100000;

```
![image](https://github.com/user-attachments/assets/28d251b3-4268-441c-8409-32509a63ca71)
```

how long does it take ? without index

select * from contributions order by date_recieved desc limit 100000;

```
![image](https://github.com/user-attachments/assets/289b679c-280f-492c-9d19-ace3d4f20b85)
```

Now forcing the index

select * from contributions force index (idx_contributions_date_recieved) order by date_recieved desc limit 100000;

```
![image](https://github.com/user-attachments/assets/c2396575-e153-4415-84a6-11a37e782497)

### forcing bad index


   * Sometimes you force the wrong index
   * You will have penalty on your time
   * Here is an example

### Using contributions
```

if not created in example before,please create

create index if not exists idx_contributions_date_recieved on contributions (date_recieved);

```

```

Do what the optimizer wants

explain select * from contributions where contribution_id < 400000 and date_recieved < '2000-12-31'; select * from contributions where contribution_id < 400000 and date_recieved < '2000-12-31';

```
![image](https://github.com/user-attachments/assets/804c1be1-52f5-454d-a5cb-c38b2bace278)
```

Forcing the index, performance is worse

explain select * from contributions force index (idx_contributions_date_recieved) where contribution_id < 400000 and date_recieved < '2000-12-31'; select * from contributions force index (idx_contributions_date_recieved) where contribution_id < 400000 and date_recieved < '2000-12-31';

```
![image](https://github.com/user-attachments/assets/23b3f9de-0dd0-47d7-9364-
815ddd2439d6)

## Performance - Views

### Performance - Views


### General
   * SHOW CREATE VIEW
   * Views can use 3 algorithms:
     * merge
     * simple rewrites (translates the query)
   * temptable
     * Creates a temptable to retrieve information
     * In this case no indexes can be used
     * Shows up explain with <derived>:
```

+----+-------------+-----------+------+---------------+------+---------+------+------+-------+ | id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra | +----+-------------+-----------+------+---------------+------+---------+------+------+-------+ | 1 | PRIMARY | | ALL | NULL | NULL | NULL | NULL | 33 | NULL | | 2 | DERIVED | task | ALL | NULL | NULL | NULL | NULL | 33 | NULL | +----+-------------+-----------+------+---------------+------+---------+------+------+-------+

```
   * undefined
     * MySQL chooses, if to use merge or temptable
     * prefers merge over temptable if possible



### Handling (best practice)

   * You can define the algorithm when creating the view
   * If you define merge and mysql cannot handle it
     * you will get a warning
```

mysql> CREATE ALGORITHM=MERGE VIEW priority_counts AS SELECT priority_id, COUNT(1) AS quanity FROM task GROUP BY priority_id; Query OK, 0 rows affected, 1 warning (0.12 sec)

mysql> SHOW WARNINGS; +---------+------+--+ | Level | Code | Message | +---------+------+--+ | Warning | 1354 | View merge algorithm can't be used here for now (assumed undefined algorithm) | +---------+------+--+ 1 row in set (0.08 sec)

```
   * Ref: https://dba.stackexchange.com/questions/54481/determining-what-algorithm-mysql-view-is-using


## Performance Schema

### Performance Schema - slow queries


### Shortcut (but activation to be done .. consumers / actors)
```

select * from sys.statement_analysis

```
### Step 1: Enable performance schema
```

cd /etc/my.cnf.d nano server.cnf

```
[mysqld] performance_schema=on

```

systemctl restart mariadb

```
mysql -e 'select @@performance_schema'

```
### Step 2: Activate specific settings to enable it
```

Instrumente aktivieren

update performance_schema.setup_instruments set enabled = 'yes', timed = 'yes';

consumers aktiviert

performance_schema.setup_consumers

```
![image](https://github.com/user-attachments/assets/cca55ed0-738f-45f3-8c99-8d3687eb7ec4)

```
-- Query ausgeführt: use contributions; select * from contributions where contribution_id < 400000 and date_recieved < '2000-12-31';';

```

## uns in digest angeschaut

use performance_schema; select * from events_statements_summary_by_digest \G

## Die Laufzeit LAST_SEEN - FIRST_SEEN (Laúfzeit der Query)

## Danach könnten man sortieren

```
Referenz:

  * https://fromdual.com/mariadb-and-mysql-performance-schema-hints#top-long-running-queries

Performance Metrics from status

Performance Metrics from status
```

Description: Number of joins which used a full scan of the first table.

->

Select_scan

No index at all

Select_full_join

Using index scan - 2nd worst

Select_full_range_join

Select_range_check

```
## Locks

### How does innodb implicit locking work


### How do the work in general

  * Implicit locks are done by InnoDB itself
  * We can only partly influence them.

### Who wants what ?
```

<who?, what?, how?, granted?>

```
### Explanation (a bit clumsy)

  * IS and IX (intended share an intended write lock)
  * IS and IX can be trigged on SQL
  * IX -> SUFFIX -> FOR UPDATE (this triggers a IX lock)
  * IX and IS are the first step (on table layer)
  * After that IX -> tries to get an write lock on row-level -> X
  * Works unless there is another X
  * IX and IS is not retrieved on TABLE spaced operations (construction --- alter)

### Lock Type compability matrix
```

```
X           IX          S           IS
```

X Conflict Conflict Conflict Conflict IX Conflict Compatible Conflict Compatible S Conflict Conflict Compatible Compatible IS Conflict Compatible Compatible Compatible

```
### The best explanation across the internet ;o)

  * http://stackoverflow.com/questions/25903764/why-is-an-ix-lock-compatible-with-another-ix-lock-in-innodb|IX_and_IS-locks
```

Many people, both visitors and curators, enter the museum. The visitors want to view paintings, so they wear a badge labeled "IS". The curators may replace paintings, so they wear a badge labeled "IX". There can be many people in the museum at the same time, with both types of badges. They don't block each other.

During their visit, the serious art fans will get as close to the painting as they can, and study it for lengthy periods.

They're happy to let other art fans stand next to them before the same painting. They therefore are doing SELECT ... LOCK IN SHARE MODE and they have "S" lock, because they at least don't want the painting to be replaced while they're studying it.

The curators can replace a painting, but they are courteous to the serious art fans, and they'll wait until these viewers are done and move on. So they are trying to do SELECT ... FOR UPDATE (or else simply UPDATE or DELETE). They will acquire "X" locks at this time, by hanging a little sign up saying "exhibit being redesigned." The serious art fans want to see the art presented in a proper manner, with nice lighting and some descriptive placque. They'll wait for the redesign to be done before they approach (they get a lock wait if they try).

```
### Exercise - identify deadlocks


### Prerequisite
```

2 sessions (connected to same server): Session 1 Session 2

sakila database is installed

```
### Session 1:
```

Start transaction and lock row by updating it

mysql>use sakila; mysql>begin; mysql>update actor set last_name='Johnsson' where actor_id = 200;

Attention: not commit yet please, leave transaction open

```
### Session 2:
```

Start transactio and try to update same row

mysql>use sakila; mysql>begin; mysql>update actor set last_name='John' where actor_id = 200;

Now update cannot be done, because of lock from session one

```
### Session 1: / or new Session 3
```

find out who blocks session 2

mysql>use information_schema;

find out trx_id of session 2

mysql>select * from innodb_trx;

assuming we have trx_id 1468;

now we find out what is blocking this transaction

mysql>select * from innodb_lock_waits; MariaDB [information_schema]> select * from innodb_lock_waits; +----------------+-------------------+-----------------+-------------------+ | requesting_trx_id | requested_lock_id | blocking_trx_id | blocking_lock_id | +-------------------+-------------------+-----------------+-------------------+ | 1469 | 1469:66:3:201 | 1468 | 1468:66:3:201 | +-------------------+-------------------+-----------------+-------------------+ 1 row in set (0.001 sec)

either additional infos

select * from innodb_trx where trx_id = 1468;

get thread_id -> e.g. 50

or directly kill this transaction

show processlist; kill 50;

```
### Easier way to find locks with sys - database
```

\G nur in mysql/mariadbclient

select * from sys.innodb_lock_waits \G

```
### With this command you can also see pending locks
```

show engine innodb status \G

```
### There is more (activate: do not only show last deadlock)
```

by setting this, deadlocks are written to error log

set global innodb_print_all_deadlocks = ON

```
### Refs:

  * https://fromdual.com/mariadb-deadlocks

### Refs ( 3 important tables )

  * https://mariadb.com/kb/en/information-schema-innodb_lock_waits-table/ (most
important one)
  * https://mariadb.com/kb/en/information-schema-innodb_locks-table/
  * https://mariadb.com/kb/en/information-schema-innodb_trx-table/

## Tools

### Percona Toolkit


### Walkthrough Rocky 9 / RHEL 9
```

wget https://downloads.percona.com/downloads/percona-toolkit/3.7.0/binary/redhat/9/x86_64/percona-toolkit-3.7.0-1.el9.x86_64.rpm

dnf localinstall percona-toolkit-3.7.0-1.el9.x86_64.rpm

```
### Walkthrough (Ubuntu 20.04)
```

Howto

https://www.percona.com/doc/percona-toolkit/LATEST/installation.html

Step 1: repo installieren mit deb -paket

wget https://repo.percona.com/apt/percona-release_latest.focal_all.deb apt update apt install -y curl dpkg -i percona-release_latest.focal_all.deb apt update apt install -y percona-toolkit

```
### Walkthrough (Debian 10)
```

sudo apt update sudo apt install -y wget gnupg2 lsb-release curl cd /usr/src wget https://repo.percona.com/apt/percona-release_latest.generic_all.deb dpkg -i percona-release_latest.generic_all.deb apt update apt install -y percona-toolkit

```

```

sudo apt update; sudo apt install -y wget gnupg2 lsb-release curl; cd /usr/src; wget https://repo.percona.com/apt/percona-release_latest.generic_all.deb; dpkg -i percona-release_latest.generic_all.deb; apt update; apt install -y percona-toolkit

```
### pt-query-digest - analyze slow logs


### Requires

  * Install percona-toolkit

### Prepare: Activate slow quer log
```

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

```

### Analyze: slow_query_log
```

cd /var/lib/mysql

look for awhile wih -slow.log - suffix

pt-query-digest mysql-slow.log > /usr/src/report-slow.txt cd /usr/src 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
```

cd /usr/src wget https://downloads.mysql.com/docs/sakila-db.tar.gz tar xzvf sakila-db.tar.gz

cd sakila-db mysql < sakila-schema.sql mysql < sakila-data.sql

```
### Setup training data "contributions"


### Walkthrough (Debian/Ubuntu)

  * Complete process takes about 10 minutes

```bash
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-debian.sh
```

Walkthrough (Redhat/Rocky)

```
cd /usr/src;
dnf install -y 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-rhel.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

Replikation mit GTID

- https://www.admin-magazin.de/Das-Heft/2017/02/MySQL-Replikation-mit-GTIDs

Replikation Read/Write - Split:

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

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

Dokumentation (Performance)

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

Analyze statt Explain

- https://mariadb.com/kb/en/analyze-statement/

Dokumentation (Releases)

Identify Long-Term Support Releases

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

Dokumentation (Server System Variables / Alter inplace)

Server System Variables

- https://mariadb.com/kb/en/server-system-variables/

Inplace Alter

- https://mariadb.com/kb/en/innodb-online-ddl-operations-with-the-inplace-alter-algorithm/

Dokumentation (Indexes)

Dokumenation (Virtual Columns / Persistentn)

Persistent / Virtual Columns

- https://mariadb.com/kb/en/generated-columns/

Backup/Restore

Mariabackup

Installation

dnf (using mariadb from mariadb.org - repo)

```
dnf install MariaDB-backup
```

Installation von Distri (Centos/Rocky/RHEL)

```
## Rocky 8
dnf install mariadb-backup
```

Installation deb (Ubuntu/Debian)

```
apt search mariadb-backup
apt install -y mariadb-backup
```

Walkthrough (Ubuntu/Debian)

Schritt 1: Grundkonfiguration

```
## user eintrag in /root/.my.cnf
[mariabackup]
user=root
## pass is not needed here, because we have the user root with unix_socket - auth
## or generic
## /etc/mysql/mariadb.conf.d/mariabackup.cnf
[mariabackup]
user=root
```

Schritt 2: Backup erstellen

```
mkdir /backups
## target-dir needs to be empty or not present
mariabackup --target-dir=/backups/2023091901 --backup
```

Schritt 3: Prepare durchführen

```
## apply ib_logfile0 to tablespaces
## after that ib_logfile0 ->  0 bytes
mariabackup --target-dir=/backups/2023091901 --prepare
```

Schritt 4: Recover

```
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup
mariabackup --target-dir=/backups/2023091901 --copy-back
chown -R mysql:mysql /var/lib/mysql
chmod -R 755 /var/lib/mysql # otherwice socket for unprivileged user does not work
systemctl start mariadb
systemctl status mariadb
```

Walkthrough (Redhat/Centos/Rocky Linux)

Schritt 1: Grundkonfiguration

```
## user eintrag in /root/.my.cnf
[mariabackup]
user=root
## pass is not needed here, because we have the user root with unix_socket - auth
## or generic
## /etc/my.cnf.d/mariabackup.cnf
[mariabackup]
user=root
```

Schritt 2: Backup erstellen

```
mkdir /backups
## target-dir needs to be empty or not present
mariabackup --target-dir=/backups/2023091901 --backup
```

Schritt 3: Prepare durchführen

```
## apply ib_logfile0 to tablespaces
## after that ib_logfile0 ->  0 bytes
mariabackup --target-dir=/backups/2023091901 --prepare
```

Schritt 4: Recover

```
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.bkup
mariabackup --target-dir=/backups/2023091901 --copy-back
chown -R mysql:mysql /var/lib/mysql
chmod -R 755 /var/lib/mysql # otherwice socket for unprivileged user does not work
## Does not work !!! Because of selinux // does not start
## ls -laZ /var/lib
systemctl start mariadb

### important for selinux if it does not work
### mariadb 10.6 from mariadb does not have problems here !
### does not start
restorecon -vr /var/lib/mysql
systemctl start mariadb

### Cleanup if everything works
rm -fR /var/lib/mysql/mysql.bkup
```

Ref.

https://mariadb.com/kb/en/full-backup-and-restore-with-mariabackup/