MySQL Einsteiger

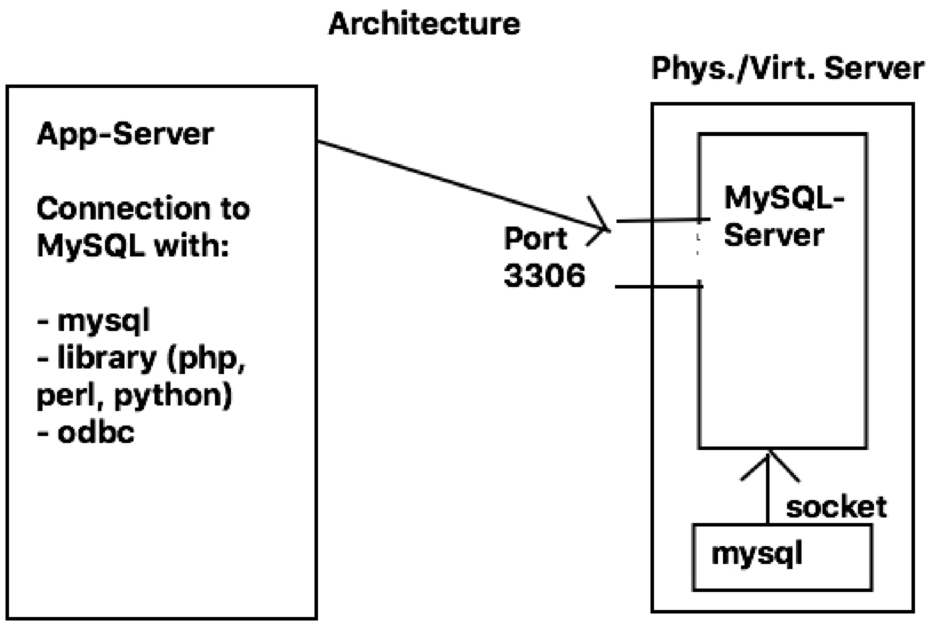
Agenda

Backlog

Backlog

Technical Background

Technical Structure



Was ist SQL

```
SQL - Structured Query Language -> Strukturierte Abfragesprache
Es ist eine standardisierte Programmiersprache
die zur Verwaltung relationale Datenbanken
und zum Durchführen verschiedener mit den darin
enthaltenen Daten verwendet wird.
```

```
Will ich Daten aus eine SQL-Datenbank brauche ich die Abfrage
sprache SQL
```

Grundlagen

Was sind Datenobjekte und welche gibt es ?

Was sind Datenbankobjekte

```
Bilden eine Strutkur um Daten zu speichern
```

Welche gibt es ? (Klassiker)

Ebene 1 (oberste Ebene) Datenbank (databases/schemas)

```
eine Organisationseinheit: die Datenbank
wie ein Behältnis
unter der Haube: Verzeichnis

z.B. Datenbank sakila
```

Ebene 2 - Tabellen

```
Eine Datenbank kann mehrere Tabellen haben.
Ähnlich eines Vorratsschrankes.
Im Filesystem finden man diese unterhalb der Datenbank mit
dem entsprechenden der Tabelle:
z.B. sakila\actor.ibd

Jede Tabelle hat eine Struktur -> Columns (Felder)
```

Ebene 3 - Felder / Columns

```
Die Tabelle hat eine Struktur, die bestimmt wird durch die Columns (Felder)
Jedes Feld (Column) hat einen Datentyp, der bestimmt welcher Daten dort rein dürfen.
z.B. Strings (varchar), oder Zahlen (z.B. INT -> Integer)
```

Daten -> Zeilen (ROW)

```
Daten werden zeilenweise in Tabellen geschrieben
Jede Zeile hat ein eindeutiges Merkmale (eine eindeutige Nummer) -> Primärschlüssel
(Primary Key)
```

Weitere Datenbankobjekte

```
Views - Schadpotenzial -> 0%
Trigger - Schadpotential -> 50%

-> hier unkritisch, da nur neu eintrag in der Datenbank
CREATE TRIGGER before_employee_update
    BEFORE UPDATE ON employees
    FOR EACH ROW
 INSERT INTO employees_audit
 SET action = 'update',
     employeeNumber = OLD.employeeNumber,
     lastname = OLD.lastname,
     changedat = NOW();

Procedures -
CALL film_in_stock(1,1,@ausgabe);
select @ausgabe;
```

```
Function (wie systemfunktionen, nur selbst erstellt)
use sakila;
select get_customer_balance(1,'2015-06-01 12:55:12');

Events (zeitgesteuerte Ereignisse) -
Schadcodepotenzial when aktiviert -> Procedures
https://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/

Wie weiss ich, dass events generell ausgeführt auf meinem System, wenn vorhanden
```

- Werden events ausgeführt

SQL-Syntax

Kommentare

```
-- Das ist ein Kommentar, der Text beschreibt etwas aber wir nicht vom SQL-Server
ausgeführt
```

SQL-Schlüsselworte für Operationen

```
DDL - Daten Definition Language
DML - Data Modelling Language

DDL ändern die Struktur
=======================
CREATE DATABASE #
CREATE SCHEMA  # Datenbank
CREATE TABLE
ALTER TABLE # VEründere die Struktur
DROP

CREATE - erstellen
DROP - Löschen (komplettes Objekt)
ALTER - verändern

DML - Ändert die Inhalte und macht Abfragen
====
SELECT
DELETE - löscht Daten
INSERT - einfügen vpn Daten
UPDATE - Updaten von Daten
```

Datenbank-und-Tabellen-verwenden

```
-- Datenbanken anzeigen
show databases;
-- or:
show schemas;

-- Datenbank verwenden / wechseln
use sakila;
```

```
-- Tabellen für ausgewählte Datenbanken anzeigen
show tables;

### Struktur und Indizes von Tabellen auslesen
```

show fields from actor; show create table actor; show indexes from actor;

-- Empfehlung, was ir können solltet describe actor; show indexes from actor;

```
## SELECT's

### Alle Datensätze abfragen und alias für Spalte setzen
```

select first_name as Vorname,last_name as Nachname from actor;

```
### Rechnen mit SELECT - Beispiele
```

-- erlaubt sind + - / * DIV (Integer Division) -- Integer-Division: 15/7 = 2 (nur Ganzzahl wird als Ergebnis zurückgegeben)

-- + - * / DIV(Integer) SELECT 15 DIV 2 as Ergebnis;

```
### Beispiel und Übung

### Beispiel mit Select where


### Example (Simple)
```

SELECT * FROM actor WHERE last_name = 'AKROYD' or last_name = 'GABLE';

```
### Example (mit Klammern)
```

SELECT * FROM actor WHERE (last_name = 'Akroyd' and first_name = 'Christian') or (last_name = 'Gable' and first_name = 'Christian');

```
### Übung
```

- Alle Datensätze aus actor anzeigen bei dennen der Nachname Akroyd oder der Vorname Christian ist.

```
### Beispiel mit select where in
### Example

SELECT * FROM actor WHERE first_name IN ('JOE','ED','JENNIFER');

### Übung
```

- Sucht Euch 3 Citys und last die Datensätze aus city dazu ausgeben
- Frage: Welche Feld verwenden ?
- mit where ... IN umsetzen

```

### Beispiel mit select und like


### Example
```

SELECT * FROM actor WHERE last_name like 'D%';

```

### Exercise
```

Lass Euch alle actor ausgeben, deren Nachname auf N endet.

```

## ORDER BY (Sortierung)

### Beispiel und Übung order by


### Beispiel
```

SELECT first_name,last_name FROM actor ORDER BY last_name DESC, first_name;

```

### Übung (schlechtes Beispiel, anderes wäre besser)
```

1. Wir sortieren alle Einträge film - Tabelle
2. rental_rate absteigend, Titel aufsteigend, release_year absteigend

LIMIT

Kombiniertes Beispiel mit Order By und Limit + Übung

What does it do ?

```
From your results of your query only shows a subset
```

Example

```
SELECT * from actor ORDER BY last_name DESC limit 3;
```

Übung

```
1. Zeige von den Filmen (sakila.film) die teuersten (rental_rate) zuerst, davon

Variante 1:
- Zeige die ersten 20

Variante 2: Zeige 10 ab dem 11. Film
```

```


## ÜBUNGEN

### Uebung - Berechnung aus Feld

### Übung order by


### Beispiel
```

SELECT first_name,last_name FROM actor ORDER BY last_name DESC, first_name;

```

### Übung (schlechtes Beispiel, anderes wäre besser)
```

1. Wir sortieren alle Einträge film - Tabelle
2. rental_rate absteigend, Titel aufsteigend, release_year absteigend

   ```
   ```

Übung mit Order By und Limit

What does it do ?

```
From your results of your query only shows a subset
```

Example

```
SELECT * from actor ORDER BY last_name DESC limit 3;
```

Übung

```
1. Zeige von den Filmen (sakila.film) die teuersten (rental_rate) zuerst, davon

Variante 1:
- Zeige die ersten 20

Variante 2: Zeige 10 ab dem 11. Film
```

```

### Übung mit Select where


```

```
### Example (Simple)
```

SELECT * FROM actor WHERE last_name = 'AKROYD' or last_name = 'GABLE';

```
### Example (mit Klammern)
```

SELECT * FROM actor WHERE (last_name = 'Akroyd' and first_name = 'Christian') or (last_name = 'Gable' and first_name = 'Christian');

```
### Übung
```

- Alle Datensätze aus actor anzeigen bei dennen der Nachname Akroyd oder der Vorname Christian ist.

```
### Übung mit select where in


### Example

SELECT * FROM actor WHERE first_name IN ('JOE','ED','JENNIFER');

### Übung
```

- Sucht Euch 3 Citys und last die Datensätze aus city dazu ausgeben
- Frage: Welche Feld verwenden ?
- mit where ... IN umsetzen

```
### Übung mit select und like


### Example
```

SELECT * FROM actor WHERE last_name like 'D%';

```
### Exercise
```

Lass Euch alle actor ausgeben, deren Nachname auf N endet.

```
### Übung - Refreshes Tag 2 - morgens - SELECT

  * https://github.com/jmetzger/training-mysql-einsteiger/blob/main/select/uebung-tag-2-morgens.md

## Beispieldaten / Testserver
```

```
### 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

### Spendenliste

### Walkthrough (Debian/Ubuntu 18.04. with mysql 5.7.)

  * Complete process takes about 10 minutes

```bash
cd /usr/src;
apt update; apt 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.sh
```

### Walkthrough (Debian/Ubuntu 20.04. with mysql 8)

- 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;
## otherwise script does not work
echo "set local_infile=1" | mysql
## 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
```

Server Vagrant

```
## Put this in a Vagrantfile
## 1. mkdir project; cd project
## 2. Put this in a Vagrantfile
## 3. vagrant up
## 4. vagrant ssh

Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/focal64"
  config.vm.provision "shell", inline: <<-SHELL

    apt-get update
    apt-get install -y mysql-server-8.0 wget
    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

    echo "-- Setting up external user"
    echo "CREATE USER ext@'%' identified by 'student'" | mysql
    echo "GRANT ALL PRIVILEGES ON *.* TO ext@'%'" | mysql

    echo "-- Setting mysql up for external access"
    echo "bind-address = 0.0.0.0" >> /etc/mysql/mysql.conf.d/mysqld.cnf
    systemctl restart mysql

    echo "-- Setting up contributions database - big data"
    cd /usr/src;
    apt-get install -y git
    git clone https://github.com/jmetzger/dedupe-examples.git;
    cd dedupe-examples;
    cd mysql_example;
    echo "set global local_infile = 1" | mysql
    echo "# Eventually you need to enter (in mysql_example/mysql.cnf)"
    echo '# Only necessary if you cannot connect to db by entering "mysql"'
    echo '# password=<your_root_pw>'
    ./setup.sh
    date
    echo "-- Done -  Setting up contributions database - big data"
  SHELL
end
```

Client-Tools

mysql - client

Aufrufen unter Windows

- Programme -> Mariadb -> Mariadb Client
- Passwrot eingeben

Hilfe

```
help
```

Wichtige Datenbank - Objekte anzeigen

```
show databases;
```

Basics

```
mysql
mysql>

## Wie kommen wie raus ?
exit;
```

Delimiter

```
Normalerweise ";"

Ist zum Trennen von Befehlen
```

Use user and password automatically

```
nano /root/.my.cnf
## BE CAREFUL EVERYBODY CAN LOGIN AS ROOT TO MYSQL NOW
## in there
[mysql]
user=root
password=root-password-on-your-system
```

Datentypen

Übersicht Datentypen

- https://mariadb.com/kb/en/data-types/

Integer Datentypen

```
TINYINT
  unsigned: 0-255
  signed: -127 bis +128

SMALLINT
A small integer. The signed range is -32768 to 32767. The unsigned range is 0 to
65535.

MEDIUMINT
A medium-sized integer. The signed range is -8388608 to 8388607. The unsigned range is
0 to 16777215.

INT

BIGINT
```

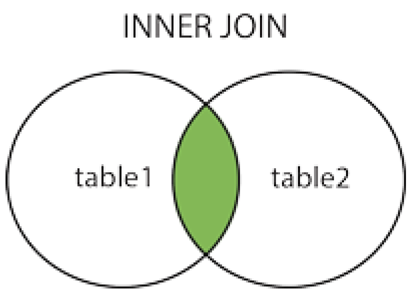
JOINS

Basics of Joins

What is a JOIN for ?

- combines rows from two or more tables
- based on a related column between them.

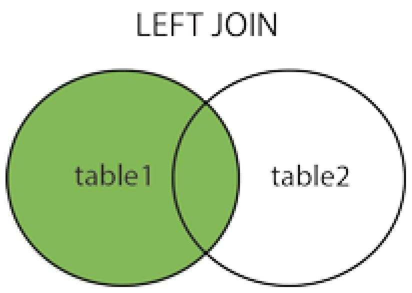
MySQL/MariaDB (Inner) Join



MySQL/MariaDB (Inner) Join (explained)

- Inner Join and Join are the same
- Returns records that have matching values in both tables
- Inner Join, Cross Join and Join
  - are the same in MySQL

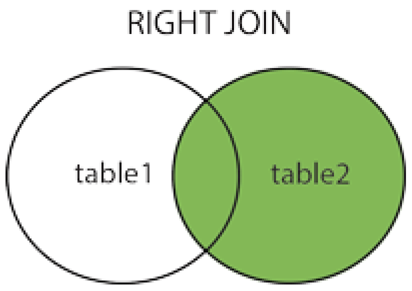
MySQL/MariaDB Left Join



MySQL/MariaDB Left (outer) Join (explained)

- Return all records from the left table
- *AND* the matched records from the right table
- The result is NULL on the right side
  - if there are no matched columns on the right
- Left Join and Left Outer Join are the same

MySQL Right Join



MySQL Right Join (explained)

- Return all records from the right table
    - *AND* the matched records from the left table
- Right Join and Right Outer Join are the same

MySQL Straight Join

- MySQL (inner) Join and Straight Join are the same
- **Difference:**
    - The left column is always read first
- **Downside:**
    - Bad optimization through mysql (query optimizer)
- **Recommendation:**
    - Avoid straight join if possible
    - use join instead

Type of Joins

- [inner] join
    - **inner join** and **join** are the same
- left [outer] join
- right [outer] join
- full [outer] join
- straight join < equals > join
- cross join = join (in mysql)
- natural join <= equals => join (but syntax is different)

In Detail: [INNER] JOIN

- Return rows when there
    - is a match in both tables
- Example

```
SELECT actor.first_name, actor.last_name, film.title
FROM film_actor
INNER JOIN actor ON film_actor.actor_id = actor.actor_id
INNER JOIN film ON film_actor.film_id = film.film_id;
```

In Detail: Joining without JOIN - Keyword ====

- Explanation: Will have the same query execution plan as [INNER] JOIN

```
SELECT actor.first_name, actor.last_name, film.title
FROM film_actor,actor,film
where film_actor.actor_id = actor.actor_id
and film_actor.film_id = film.film_id;
```

In Detail: Left Join

- Return all rows from the left side
  - even if there is not result on the right side
- Example

```
SELECT
c.customer_id,
c.first_name,
c.last_name,
a.actor_id,
a.first_name,
a.last_name
FROM customer c
LEFT JOIN actor a
ON c.last_name = a.last_name
ORDER BY c.last_name;
```

In Detail: Right Join

- Return all rows from the right side
  - even if there are no results on the left side
- Example

```
SELECT
c.customer_id,
c.first_name,
c.last_name,
a.actor_id,
a.first_name,
a.last_name
FROM customer c
RIGHT JOIN actor a
ON c.last_name = a.last_name
ORDER BY a.last_name;
```

In Detail: Having

- Simple: WHERE for GroupBy (because where does not work here)
- Example

```
SELECT last_name, COUNT(*)
FROM sakila.actor
GROUP BY last_name
HAVING count(last_name) > 2
```

Internal (type of joins) - NLJ

- NLJ - (Nested Loop Join)

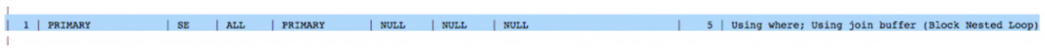
  ```
  for each row in t1 matching range {
  for each row in t2 matching reference key {
  for each row in t3 {
    if row satisfies join conditions, send to client
  }
  }
  }
  ```

Internal (type of joins) - BNL

- BNL - (Block Nested Loop)
  - in explain: -> using join buffer
  - columns of interest to a join are stored in join buffer
    - --> not whole rows.
  - join_buffer_size system variable
    - -> determines the size of each join buffer used to process a query.
- https://dev.mysql.com/doc/refman/5.7/en/nested-loop-joins.html

BNL - Who can I see, if it is used ?

- Can be seen in explain



```
|  1 | PRIMARY            | SE    | ALL    | PRIMARY        | NULL    | NULL    | NULL                         |    5 | Using where; Using join buffer (Block Nested Loop)
```

```

explain SELECT a.* FROM actor a  INNER JOIN actor b where a.actor_id > 20 and
b.actor_id < 20
```

```
When using a Block Nested-Loop Join, MySQL will, instead of automatically joining t2,
insert as many rows from t1 that it can into a join buffer
and then scan the appropriate range of t2 once,
matching each record in t2 to the join buffer.
From here, each matched row is then sent to the next join,
which, as previously discussed, may be another table,
t3, or, if t2 is the last table in the query,
the rows may be sent to the network.
```

BNL's - Refs:

- https://www.burnison.ca/notes/fun-mysql-fact-of-the-day-block-nested-loop-joins

Working with LEFT JOIN

General

- Show all entries from the left table and only from the right if available
- Examples are based on sakila database.

Example - new language / not in language table

```
SELECT @@foreign_key_checks;
SET FOREIGN_KEY_CHECKS=0
UPDATE film  SET language_id = 99 WHERE film_id >= 800 and film_id <= 899
SELECT f.film_id,f.title,f.description,l.name FROM film f LEFT JOIN language l ON
f.language_id = l.language_id;
SET FOREIGN_KEY_CHECKS=1
```

Example

```
use sakila
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    a.actor_id,
    a.first_name,
    a.last_name
FROM customer c
LEFT JOIN actor a
ON c.last_name = a.last_name
ORDER BY c.last_name;
```

Join examples

```
## Work - step by step.
SELECT * FROM film f;
SELECT f.title FROM film f;
SELECT f.language_id FROM film f JOIN language l ON f.language_id = l.language_id;
-- aus film-tabelle alle felder - f.*
SELECT f.*,l.name FROM film f JOIN language l ON f.language_id = l.language_id;
SELECT f.film_id,f.title,l.name,f.description FROM film f JOIN language l ON
f.language_id = l.language_id;
```

GROUP BY

Simple Group By Example

```
## Variante 1
SELECT last_name,COUNT(last_name) as cnt FROM actor GROUP BY last_name

## Variante 2: ohne Group - akroyd zählen -> geht nur bei einem Namen
SELECT last_name,COUNT(last_name) as cnt FROM actor WHERE last_name = 'AKROYD'

## Das ist falsch - weil mehrere Namen, Ausgabe nur eine Zeile
SELECT last_name,COUNT(last_name) as cnt FROM actor WHERE last_name = 'AKROYD' or
last_name = 'ALLEN'

## Variante 2a (Erst Daten holen - 6 Datensätze, dann aggregieren (group by)
## Für grosse Datenmengen besser !
SELECT last_name,COUNT(last_name) as cnt FROM actor WHERE last_name = 'AKROYD' or
```

```
last_name = 'ALLEN' GROUP BY last_name
## Variante 2b (Alle Daten holen - 200 Datensätze, dann alles aggregieren)
SELECT last_name,COUNT(last_name) as cnt FROM actor GROUP BY last_name HAVING
last_name = 'AKROYD' or last_name = 'ALLEN'
```

Join and group - example

```
## fake 100 itilian movies
UPDATE film SET language_id = 2 WHERE film_id > 900;
## Group by to show number of itilian and english movies
## Interesting for reporting
SELECT l.name,count(l.name) FROM film f JOIN language l ON f.language_id =
l.language_id GROUP BY l.name;
```

CONSTRAINTS

Foreign Key Constraints - Example

Walkthrough

Step 1: Sample Table and data

```
create table gadget_types(
    type_id int auto_increment,
    name varchar(100) not null,
    primary key(type_id)
);

insert into gadget_types(name)
values
    ('Entertainment'),
    ('Computing'),
    ('Communication'),
    ('Lifestyle'),
    ('Cameras');

create table gadgets(
    gadget_id int auto_increment,
    gadget_name varchar(100) not null,
    type_id int,
    primary key(gadget_id),
    constraint fk_type
    foreign key(type_id)
        references gadget_types(type_id)
);
```

Step 2: Insert data to gadgets

```
insert into
    gadgets(gadget_name,type_id)
values
    ('Amazon Kindle',1),
    ('Apple iPod',1),
    ('Audio Highway Listen Up',1),
    ('Apple iPad',2),
    ('MicroSD',2),
    ('Apple iPhone',3),
    ('BlackBerry 6210',3),
    ('Pager',3),
    ('Air Taser Model 34000',4),
    ('Credit Card',4),
    ('Zippo',4),
    ('Casio G-Shock DW-5000C',4),
    ('Nikon F',5),
    ('Canon EOS 5D Mark II',5);
```

Step 3 - Try to delete

```
delete from gadget_types
where type_id = 1;



SQL Error (1451): Cannot delete or update a parent row: a foreign key constraint fails
(`nation`.`gadgets`, CONSTRAINT `fk_type` FOREIGN KEY (`type_id`) REFERENCES
`gadget_types` (`type_id`))
-- --> To delete a row from the gadget_types table, you need to remove all the
referencing rows from the gadgets table first.
```

Step 4 - Drop Contrains and set NULL Reference

```
alter table gadgets
drop constraint fk_type;

alter table gadgets
add constraint fk_type
foreign key(type_id)
    references gadget_types(type_id)
    on delete set null
    on update set null;

delete from gadget_types
where type_id = 1;

select * from gadgets;
-- --> As shown clearly from the output, the values in the type_id column of rows with
type_id 1 from the gadgets table were set to null because of the on -- delete set null
option.
```

Step 5 - change id in gadget_types

```
update gadget_types
set type_id = 20
where type_id = 2;
```

```
select * from gadgets;

--> The values in the type_id column of rows with type_id 2 from the gadgets table
were set to null because of the on update set null option.
```

Step 6 - remove orphans

```
delete from gadgets
where type_id is null;
```

Step 7 - cascade reference option

```
-- Drop constraint
alter table gadgets
drop constraint fk_type;

alter table gadgets
add constraint fk_type
foreign key(type_id)
    references gadget_types(type_id)
    on delete cascade
    on update cascade;

delete from gadget_types
where type_id = 3;
-- > MariaDB automatically deleted rows from the gadgets table whose type_id is 3
because of the on delete cascade action.

select * from gadgets;
```

Step 8 - Update gadget_type id 4 to 40:

```
update gadget_types
set type_id = 40
where type_id = 4

--> MariaDB automatically updated rows from the gadgets table whose type_id is 4 to 40
because of the on update cascade action:
```

```
select * from gadgets;
--> Updated all references
```

Ref:

- https://www.mariadbtutorial.com/mariadb-basics/mariadb-foreign-key/

Check constraint with example

Column constraints

```
create table classes(
    class_id int auto_increment,
    class_name varchar(255) not null,
    student_count int
        constraint positive_student_count
        check(student_count >0),
    primary key(class_id)
);
```

Table Check Constraint

```
create table classes(
    class_id int auto_increment,
    class_name varchar(255) not null,
    student_count int,
    constraint positive_student_count
        check(student_count >0),
    primary key(class_id)
);

insert into classes(class_name, student_count)
values('MariaDB for Developers',0);

-- SQL Error (4025): CONSTRAINT `positive_student_count` failed for `nation`.`classes`

insert into classes(class_name, student_count)
values('MariaDB for Developers',100);
```

Multi Column Constraint

```
create table classes(
    class_id int auto_increment,
    class_name varchar(100) not null,
    begin_date date not null,
    end_date date not null,
    student_count int,
    constraint positive_student_count
        check(student_count >0),
    constraint valid_date
        check(end_date >= begin_date),
    primary key(class_id)
);

-- Change structure like so
ALTER TABLE classes ADD (begin_date date NOT NULL, end_date date NOT NULL,
CONSTRAINT valid_date CHECK (end_date > begin_date))

--
```

Ref:

- https://www.mariadbtutorial.com/mariadb-basics/mariadb-check-constraint/

UPDATE

Sophisticated Update

Aufgabe

```
Erhöhe bei allen Datensätzen in film die rental_rate um 1 Euro,
für die Schauspieler "actor", deren Nachname mit D anfängt
```

Lösung

```
SELECT DISTINCT fc.film_id FROM filmcopy fc JOIN film_actor fa ON fc.film_id =
fa.film_id JOIN actor a ON fa.actor_id = a.actor_id where a.last_name like 'D%';

-- Version 1
UPDATE filmcopy fc JOIN film_actor fa ON fc.film_id = fa.film_id JOIN actor a ON
fa.actor_id = a.actor_id SET rental_rate = rental_rate + 1 WHERE a.last_name LIKE 'D%'

-- Version 2
UPDATE actor a  JOIN film_actor fa ON a.actor_id = fa.actor_id JOIN filmcopy f on
fa.film_id = f.film_id  SET f.rental_rate = f.rental_rate + 1 WHERE a.last_name LIKE
'D%'


Query OK, 432 rows affected (0.02 sec)
Rows matched: 432  Changed: 432  Warnings: 0
```

DELETE

Delete mit Transaktion

Beispiel

```
Variante 1: Andere sehen es erst nach commit (andere Sessions/Verbindungen)

BEGIN;
DELETE FROM actor where actor_id = 200;
COMMIT;

Variante 2: Aktion mir nicht, ich rolle sie zurück, mache sie rückgängig
BEGIN;
DELETE FROM actor where actor_id = 200;
ROLLBOCK;
```

INDEX HINTS

Force use of an Index

```
EXPLAIN SELECT Name,CountryCode FROM City FORCE INDEX (Name)
WHERE name>="A" and CountryCode >="A";
```

```
EXPLAIN SELECT Name,CountryCode FROM City USE INDEX (Name)
WHERE name>="A" and CountryCode >="A";
```

Do not use an index if indexes are present

```
SELECT * FROM actor USE INDEX() WHERE last_name LIKE 'D%';
-- Identify if really no index is used
EXPLAIN SELECT * FROM actor USE INDEX() WHERE last_name LIKE 'D%';
```

PREPARED STATEMENTS

Prepared Statements with examples

Setup

```
CREATE TABLE test (id int auto_increment, data varchar(40) NOT NULL DEFAULT '',
PRIMARY KEY(id));
```

Single Line (Insert)

```
USE sakila;
PREPARE st1 FROM 'INSERT INTO actor (first_name,last_name) VALUES (?,?)';
SET @first_name = 'Johan';
SET @last_name = 'Muster';
EXECUTE st1 USING @first_name,@last_name;
DEALLOCATE PREPARE st1;

SELECT * FROM actor ORDER BY actor_id DESC;
```

Multiline Prepared Statement (Insert)

```
-- Statement vorbereiten
PREPARE stmt1 FROM 'INSERT INTO test (data) VALUES (?), (?), (?)';
SET @d1 = 'Line1';
SET @d2 = 'Line2';
SET @d3 = 'Line3';
EXECUTE stmt1 USING @d1, @d2, @d3;

-- If you do not want to use it anymore DEALLOCATE IT
DEALLOCATE PREPARE stmt1;

SELECT * from test;
```

Using it with select

```
create table t1 (a int,b char(10));
insert into t1 values (1,"one"),(2, "two"),(3,"three");
```

```
prepare test from "select * from t1 where a=?";
set @param=2;
execute test using @param;
+------+------+
| a    | b    |
+------+------+
|    2 | two  |
+------+------+
set @param=3;
execute test using @param;
+------+-------+
| a    | b     |
+------+-------+
|    3 | three |
+------+-------+
deallocate prepare test;
```

Finding out number of rows (for select)

```
-- If this function is executed immediately after execute.
SELECT FOUND_ROWS();
```

Prepared Statements and transactions

```
use sakila;
create table test if not exists (id int auto_increment, data varchar(30), primary
key(id));

START TRANSACTION;
PREPARE stmt2 FROM 'INSERT INTO test (`data`) VALUES (?)';
SET @d1 = 'Line1';
EXECUTE stmt2 USING @d1;
SET @d1 = 'Line2';
EXECUTE stmt2 USING @d1;
COMMIT;

SELECT * from test;


BEGIN;
SET @d1 = 'Line3 -uncommited';
EXECUTE stmt2 USING @d1;
ROLLBACK;

DEALLOCATE PREPARE stmt2;
SELECT * FROM test;
```

References

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

TRIGGERS

Triggers

Create the structure

```
create table countries (
    country_id int auto_increment,
    name varchar(50) not null,
    primary key(country_id)
);

INSERT INTO countries (name) values ('Germany'), ('Austria');

create table country_stats(
    country_id int,
    year int,
    population int,
    primary key (country_id, year),
    foreign key(country_id)
    references countries(country_id)
);

INSERT INTO country_stats (country_id, year, population) values (1,2020,100000);


create table population_logs(
    log_id int auto_increment,
    country_id int not null,
    year int not null,
    old_population int not null,
    new_population int not null,
    updated_at timestamp default current_timestamp,
    primary key(log_id)
);
```

Create the trigger

```
create trigger before_country_stats_update
    before update on country_stats
    for each row
    insert into population_logs(
        country_id,
        year,
        old_population,
        new_population
    )
    values(
        old.country_id,
        old.year,
        old.population,
        new.population
    );
```

Create trigger (the same) but with BEGIN/END - Block

```
delimiter //
create trigger before_country_stats_update
    before update on country_stats
    for each row

    BEGIN
    SET @anfang = 1;
    insert into population_logs(
        country_id,
        year,
        old_population,
        new_population
    )
    values(
        old.country_id,
        old.year,
        old.population,
        new.population
    );
    END//
```

Run a test

```
update
    country_stats
set
    population = 1352617399
where
    country_id = 1 and
    year = 2020;

-- what's the new result

select * from population_logs;
```

Ref:

- https://mariadb.com/kb/en/trigger-overview/

EVENTS

Events

Preparation

```
-- scheduler is not there
SHOW PROCESSLIST;
```

```
-- Prüfen ob scheduler läuft
show variables like '%event%';
set GLOBAL event_scheduler = on;

-- scheduler appears
SHOW PROCESSLIST;

-- Events anzeigen
show events;
```

preparation

```
USE schulung;
CREATE TABLE messages (
    id INT PRIMARY KEY AUTO_INCREMENT,
    message VARCHAR(255) NOT NULL,
    created_at DATETIME NOT NULL
);
```

One time event

```
USE schulung;
CREATE EVENT IF NOT EXISTS test_event_01
ON SCHEDULE AT CURRENT_TIMESTAMP
DO
  INSERT INTO messages(message,created_at)
  VALUES('Test MariaDB Event 1',NOW());

SELECT * FROM messages;
```

Show all events from a specific database

```
SHOW EVENTS FROM schulung;
```

Show all events in active database

```
USE schulung;
SHOW EVENTS;
```

One time event but preserved (so runs once every minute)

```
-- To keep the event after it is expired, you use the  ON COMPLETION PRESERVE clause.

CREATE EVENT test_event_02
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 MINUTE
ON COMPLETION PRESERVE
DO
```

```
   INSERT INTO messages(message,created_at)
   VALUES('Test MariaDB Event 2',NOW());
```

Same version, but with begin end block

```
DELIMITER /
CREATE EVENT test_event_03
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 MINUTE
ON COMPLETION PRESERVE
DO
   BEGIN
   INSERT INTO messages(message,created_at)
   VALUES('Test MariaDB Event 3',NOW());
   END /
DELIMITER ;

SELECT * FROM messages;
```

Recurring Example

```
CREATE EVENT test_event_04
ON SCHEDULE EVERY 1 MINUTE
STARTS CURRENT_TIMESTAMP
ENDS CURRENT_TIMESTAMP + INTERVAL 1 HOUR
DO
   INSERT INTO messages(message,created_at)
   VALUES('Test MariaDB recurring Event',NOW());

SELECT * FROM messages;

// after 1 minute
SELECT * FROM messages;
```

Drop an event

```
DROP EVENT IF EXISTS test_event_03;
```

Set event-scheduler in config / my.cnf / my.ini

```
[mysqld]
event-scheduler

## after that restawrt
systemctl restart mariadb
```

Fix timezone problem Linux (when time is displayed wrong)

```
## 09:32 UTC should be 11:32 CEST
## also root ausführen
timedatectl list-timezones  | grep 'Europe/Berlin';
timedatectl set-timezone Europe/Berlin
timedatectl
date
systemctl restart mariadb
mysql
mysql>select now();
mysql>--- time should ok now
```

FUNCTIONS

Functions with example

Example 1

```
use sakila;
CREATE FUNCTION hello (s CHAR(20))
    RETURNS CHAR(50) DETERMINISTIC
    RETURN CONCAT('Hello, ',s,'!');

select hello();
## weniger performant
select first_name,last_name,hello(first_name) from actor where actor_id = 2;
## mehr performant
select first_name,last_name,concat('Hello, ',first_name,'!') from actor where actor_id
= 2;
```

Example 2

```
DELIMITER //

CREATE FUNCTION CalcValue ( starting_value INT )
RETURNS INT DETERMINISTIC

BEGIN

   DECLARE total_value INT;

   SET total_value = 0;

   label1: WHILE total_value <= 3000 DO
     SET total_value = total_value + starting_value;
   END WHILE label1;

   RETURN total_value;

END; //

DELIMITER ;
```

```
-- Use it
SELECT CalcValue (1000);
```

Example 3 (Mit Variable anzahl vorher setzen)

```
DELIMITER /

CREATE DEFINER=`ext`@`%` FUNCTION `Anzahl`(
    `starting_value` INT
)
RETURNS INT

BEGIN

   DECLARE total_value INT;
   DECLARE anzahl INT;
   -- SET total_value = 0;
   SELECT COUNT(*) INTO anzahl FROM actor;

   -- WHILE total_value <= 3000 DO
   --  SET total_value = total_value + starting_value;
   -- END WHILE;

   -- RETURN total_value;
   RETURN anzahl;

END/
```

Example 3 (Direkt in @anzahl schreiben)

```
DELIMITER /

CREATE DEFINER=`ext`@`%` FUNCTION `schaupieler`(
    `starting_value` INT
)
RETURNS INT

BEGIN

   DECLARE total_value INT;
   SELECT COUNT(*) INTO @anzahl FROM actor;
   RETURN @anzahl;

END/
```

Ref:

- https://mariadb.com/kb/en/create-function/

STORED PROCEDURES

Create Procedure

Example

```
USE sakila;
DELIMITER //

CREATE PROCEDURE simpleproc (OUT param1 INT)
 BEGIN
  SELECT COUNT(*) INTO param1 FROM actor;
 END;
//

DELIMITER ;

CALL simpleproc(@a);

SELECT @a;
+-------+
| @a    |
+-------+
|     1 |
+-------+
```

Reference

- https://mariadb.com/kb/en/create-procedure/

If

Example 1

```
-- Gibt 1 aus
-- SELECT werden auf dem Bildschirm angezeigt ;o)

DELIMITER $$
USE sakila $$
DROP PROCEDURE IF EXISTS my_sproc $$
CREATE PROCEDURE
my_sproc ()
BEGIN
   IF 1=1
   THEN
     SELECT 1;
   END IF;
END $$

CALL my_sproc
```

Example 2

```
DELIMITER $
CREATE PROCEDURE my_pr()
BEGIN
IF 2 = 2 THEN
SELECT 'TRUE';
ELSE
SELECT 'FALSE';
END IF;
END $
DELIMITER ;
CALL my_pr;
```

Example 3

```
DELIMITER /
CREATE OR REPLACE PROCEDURE addActor (IN startdate DATE, IN enddate DATE)
main: BEGIN
   IF startdate > enddate
   THEN
      SELECT 'Das Startdaum liegt nach dem Enddatum';
      LEAVE main;
   END IF;

    SELECT 'das passt';

END/
DELIMITER ;
```

Reference

https://mariadb.com/kb/en/if/

Cursors

Example 1:

```
CREATE DATABASE IF NOT EXISTS training;
USE training;


CREATE TABLE c1(i INT);

CREATE TABLE c2(i INT);

CREATE TABLE c3(i INT);

DELIMITER //

CREATE PROCEDURE p1()
BEGIN
```

```
  DECLARE done INT DEFAULT FALSE;
  DECLARE x, y INT;
  DECLARE cur1 CURSOR FOR SELECT i FROM c1;
  DECLARE cur2 CURSOR FOR SELECT i FROM c2;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN cur1;
  OPEN cur2;

  read_loop: LOOP
    FETCH cur1 INTO x;
    FETCH cur2 INTO y;
    IF done THEN
      LEAVE read_loop;
    END IF;
    IF x < y THEN
      INSERT INTO c3 VALUES (x);
    ELSE
      INSERT INTO c3 VALUES (y);
    END IF;
  END LOOP;

  CLOSE cur1;
  CLOSE cur2;
END; //

DELIMITER ;

INSERT INTO c1 VALUES(5),(50),(500);
INSERT INTO c2 VALUES(10),(20),(30);

CALL p1;
SELECT * FROM c3;
```

Example 2

```
CREATE OR REPLACE PROCEDURE getActorNames(p_ab CHAR(1))
BEGIN
  DECLARE d_full_name VARCHAR(90);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur1 CURSOR FOR SELECT CONCAT(last_name,',',first_name) FROM actor where
last_name LIKE CONCAT(p_ab,'%');
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


  OPEN cur1;
  read_loop: LOOP
    FETCH cur1 INTO d_full_name;

    IF done THEN
      LEAVE read_loop;
    ELSE
```

```
        INSERT INTO actorlog (full_name) values (d_full_name);
      END IF;
  END LOOP;

  CLOSE cur1;
END; //

DELIMITER ;

CALL getActorNames('B');
SELECT * FROM actorlog;
```

Example 3

```
USE sakila;
-- DROP TABLE IF EXISTS actor_stats;
CREATE TABLE IF NOT EXISTS actor_stats(id INT auto_increment, last_name VARCHAR (90),
howmany TINYINT, UNIQUE (last_name), primary key(id));

CREATE OR REPLACE PROCEDURE writeActorStats()
BEGIN
  DECLARE d_last_name VARCHAR(45);
  DECLARE done INT DEFAULT FALSE;
  DECLARE c_actors CURSOR FOR SELECT last_name FROM actor;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  DECLARE CONTINUE HANDLER FOR 1062
  BEGIN
    UPDATE actor_stats SET howmany = howmany + 1 WHERE last_name = d_last_name;
  END;

  TRUNCATE actor_stats;

  OPEN c_actors;
  read_loop: LOOP
    FETCH c_actors INTO d_last_name;

    IF done THEN
      LEAVE read_loop;
    ELSE
        INSERT INTO actor_stats (last_name,howmany) values (d_last_name,1);
    END IF;
  END LOOP;

  CLOSE c_actors;
END; //

DELIMITER ;

CALL writeActorStats();
SELECT * FROM actor_stats;
SELECT * FROM actorlog;
```

Reference

- https://mariadb.com/kb/en/cursor-overview/

Cursor with Params

Example 1:

```
USE sakila;
DROP TABLE IF EXISTS actorlog;
CREATE TABLE actorlog(id INT auto_increment, full_name VARCHAR (90), primary key(id));
DELIMITER //

CREATE OR REPLACE PROCEDURE getActorName(p_id INT)
BEGIN
  DECLARE d_full_name VARCHAR(90);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur1 CURSOR(p_actor_id INT) FOR SELECT CONCAT(last_name,',',first_name) FROM
actor where actor_id = p_actor_id;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN cur1(p_id);
  read_loop: LOOP
    FETCH cur1 INTO d_full_name;

    IF done THEN
      LEAVE read_loop;
    ELSE
        INSERT INTO actorlog (full_name) values (d_full_name);
      END IF;
  END LOOP;

  CLOSE cur1;
END; //

DELIMITER ;

CALL getActorName(1);
SELECT * FROM actorlog;
```

Loop

Example

```
DELIMITER //

CREATE or REPLACE PROCEDURE CalcValue ( starting_value INT )


BEGIN

   DECLARE total_value INT;

   SET total_value = 0;

   label1: LOOP
     SET total_value = total_value + starting_value;
     IF total_value < 850 THEN
       ITERATE label1;
     END IF;
     LEAVE label1;
   END LOOP label1;

   SELECT total_value;

END; //

DELIMITER ;

CALL CalcValue(200);
```

Reference

- https://mariadb.com/kb/en/loop/
- https://www.mysqltutorial.org/stored-procedures-loop.aspx

Case

Example with database insert

```
use sakila;
ALTER TABLE actor ADD COLUMN (startdate DATE, enddate DATE, star_type CHAR(3));



DELIMITER /
CREATE OR REPLACE PROCEDURE addActor (IN startdate DATE,
                                      IN enddate DATE,
                                      IN first_name VARCHAR(45),
                      IN last_name VARCHAR(45),
                      IN fame VARCHAR(9))
main: BEGIN

   DECLARE star_type CHAR(2);

   IF startdate > enddate
   THEN
      SELECT 'Das Startdaum liegt nach dem Enddatum';
      LEAVE main;
   END IF;

   IF first_name = ''
   THEN
      SELECT 'Bitte gebe einen First Name ein: ';
      LEAVE main;
   END IF;
   IF last_name = ''
   THEN
      SELECT 'Bitte gebe einen Last Name ein: ';
      LEAVE main;
   END IF;

   CASE fame

     WHEN 'superstar' THEN
       SET star_type='ST';
     WHEN 'megastar' THEN
       SET star_type='MS';
     WHEN 'star' THEN
       SET star_type='S';
     ELSE
       SELECT 'Als Star ist nur superstar,megastart oder star erlaubt';
       LEAVE main;
   END CASE;

   INSERT INTO actor (startdate,enddate,first_name,last_name,star_type)
   VALUES (startdate,enddate,first_name,last_name,star_type);

   SELECT CONCAT ('Schauspieler ',first_name,' ',last_name,' ',fame);

END/
DELIMITER ;


CALL addActor('2021-12-22','2021-12-31','Peter','Lausitz','megastar');
```

Reference

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

Continue Handler Example

Example 1 (Handler without begin end)

```
-- In heidisql -> you will have multiple tabs as output

DELIMITER /
CREATE OR REPLACE PROCEDURE handlertest()
BEGIN
  DECLARE CONTINUE HANDLER FOR 1146
    SELECT 'Sorry mate, wrong table';

  SELECT actor_id FROM wrong_table_name;
  SELECT 'continue';
  SELECT * FROM actor WHERE actor_id = 1;

END /
```

```
-- in mariadb client
MariaDB [sakila]> CALL handlertest;
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
1 row in set (0.001 sec)

+----------+
| continue |
+----------+
| continue |
+----------+
1 row in set (0.001 sec)

+----------+------------+-----------+
| actor_id | first_name | last_name |
+----------+------------+-----------+
|        1 | PENELOPE   | GUINESS   |
+----------+------------+-----------+
1 row in set (0.003 sec)

Query OK, 0 rows affected (0.003 sec)

MariaDB [sakila]>
```

Example 2: with begin and end (handler)

```
-- Important: At the end of th BEGIN END; block for DECLARE CONTINUE ..
-- There has to be an ";" at then end of END -> END;

DELIMITER /
CREATE OR REPLACE PROCEDURE handlertest()
BEGIN
  DECLARE CONTINUE HANDLER FOR 1146
  BEGIN
     SELECT 'Sorry mate, wrong table';
  END;

  SELECT actor_id FROM wrong_table_name;
  SELECT 'continue';
  SELECT * FROM actor WHERE actor_id = 1;

END /
```

```
-- Execute the CALL in an mysql - client to really see, what is going on
-- In heidisql and other guis you will probably only see the output of the first
select
MariaDB [sakila]> CALL handlertest;
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
1 row in set (0.001 sec)

+----------+
| continue |
+----------+
| continue |
+----------+
1 row in set (0.001 sec)

+----------+------------+-----------+
| actor_id | first_name | last_name |
+----------+------------+-----------+
|        1 | PENELOPE   | GUINESS   |
+----------+------------+-----------+
1 row in set (0.003 sec)

Query OK, 0 rows affected (0.003 sec)

MariaDB [sakila]>
```

Exit Handler Example

Example 1 (Handler without begin end)

```
-- In heidisql

DELIMITER /
CREATE OR REPLACE PROCEDURE handlertest()
BEGIN
  DECLARE EXIT HANDLER FOR 1146
    SELECT 'Sorry mate, wrong table';

  SELECT actor_id FROM wrong_table_name;
  SELECT 'continue';
  SELECT * FROM actor WHERE actor_id = 1;

END /
```

```
-- Execute the CALL in an mysql - client to really see, what is going on
-- In heidisql and other guis you will probably only see the output of the first
select
MariaDB [sakila]();
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
1 row in set (0.001 sec)
```

```
Query OK, 0 rows affected (0.002 sec)
```

Example 2: with begin and end (handler)

```
-- Important: At the end of th BEGIN END; block for DECLARE CONTINUE ..
-- There has to be an ";" at then end of END -> END;

DELIMITER /
CREATE OR REPLACE PROCEDURE handlertest()
BEGIN
  DECLARE EXIT HANDLER FOR 1146
  BEGIN
     SELECT 'Sorry mate, wrong table';
  END;

  SELECT actor_id FROM wrong_table_name;
  SELECT 'continue';
  SELECT * FROM actor WHERE actor_id = 1;

END /
```

```
-- Execute the CALL in an mysql - client to really see, what is going on
-- In heidisql and other guis you will probably only see the output of the first
select
MariaDB [sakila]();
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
| Sorry mate, wrong table |
+-------------------------+
1 row in set (0.001 sec)

Query OK, 0 rows affected (0.002 sec)
```

Custom Error message

Variante 1

```
USE `sakila`;
DROP PROCEDURE IF EXISTS actortest;
DELIMITER /
CREATE OR REPLACE PROCEDURE actortest(OUT n_actor_id INT)
BEGIN


    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
         -- set local vars within diagnostics
     GET DIAGNOSTICS CONDITION 1 @sqlstate = RETURNED_SQLSTATE,
     @errno = MYSQL_ERRNO, @mytext = MESSAGE_TEXT;
     SET @full_error = CONCAT("ERROR ", @errno, " (", @sqlstate, "):", @mytext);
     SELECT @full_error;
    END;

    SELECT actor_id INTO @n_actor_id FROM NOT_actor WHERE actor_id = 1;



END; /

DELIMITER ;
```

Variante 2: Mit Schreiben in log-tabelle

```
USE `sakila`;
CREATE TABLE IF NOT EXISTS applogs (id INT AUTO_INCREMENT, message MEDIUMTEXT, PRIMARY
KEY(id));

DROP PROCEDURE IF EXISTS actortest;
DELIMITER /
CREATE OR REPLACE PROCEDURE actortest(OUT n_actor_id INT)
BEGIN


    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
    GET DIAGNOSTICS CONDITION 1 @SQLSTATE = RETURNED_SQLSTATE,
     @errno = MYSQL_ERRNO, @TEXT = MESSAGE_TEXT;
     SET @full_error = CONCAT("ERROR ", @errno, " (", @SQLSTATE, "):", @TEXT);
     INSERT INTO applogs (message) VALUES (@full_error);
     SELECT @full_error;
     END;

    SELECT actor_id INTO n_actor_id FROM NOT_actor_ctor WHERE actor_id = 1;
     -- SET n_actor_id = 55;



END; /

DELIMITER ;
```

ERRORS

Error Codes List

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

LOCKS

Table wide locks

```
## Session 1
use sakila;
-- nobody else can read or write
LOCK TABLES actor WRITE;
```

```
## Session 2
show open tables where in_use like 1;
+----------+------------+--------+-------------+
| Database | Table      | In_use | Name_locked |
+----------+------------+--------+-------------+
| sakila   | actor      |      1 |           0 |
| sakila   | film_actor |      1 |           0 |
+----------+------------+--------+-------------+
2 rows in set (0.002 sec)

-- will hang till locks are unlocked
INSERT INTO actor (first_name,last_name) values ('Josy','McCosy');
```

```
## Session 1
mysql> UNLOCK TABLES
```

```
## Session 2
-- INSERT is written now

-- no output for show open tables if only these are locked
show open tables where in_use like 1;
```

InnoDB Implicit Locks

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).
```

SELECT FOR UPDATE

Important

- It only locks (X) the rows needed, not the complete table

When is it used ?

- You want to be sure, a specific dataset will not be changed

Example

```
create database if not exists training;
use training;
create table rooms (room_id tinyint auto_increment, room varchar(20), primary
key(room_id));
insert into rooms (room) values ('Honeymoon');
insert into rooms (room) values ('Sunset');

## Session 1:
BEGIN;
```

```
select room_id from rooms where room_id = 1 for update;

## Session 2:
BEGIN;
use training;
delete from rooms where room_id = 2;
delete from rooms where room_id = 1;
-- transaction waiting

## Session 3:
SELECT   waiting_trx_id,   waiting_pid,   waiting_query,   blocking_trx_id,
blocking_pid,   blocking_query FROM sys.innodb_lock_waits;

## Session 1:
COMMIT;

## Session 2:
## See what happens
```

sys (Database included since MariaDB 10.6 AFAIK)

Sys-Schema instalieren mariadb <= 10.6

ab 10.6

- Hier ist das bereits im Server beinhaltet

vor 10.6

- https://github.com/FromDual/mariadb-sys

show innodb locks with sys

```
SELECT waiting_trx_id, waiting_pid, waiting_query, blocking_trx_id, blocking_pid,
blocking_query
FROM sys.innodb_lock_waits;
```

Formatierung Ausgaben / Funktionen

Datumsausgabe formatieren

- https://mariadb.com/kb/en/date_format/

Strings zusammenfügen

- https://mariadb.com/kb/en/concat/

Partitions

Partitions - Why and Howto?

Pre-Requisites

```
## partition als engine muss aktiviert sein
## ist eigentlich immer der fall
SHOW PLUGINS;
...
| Aria                          | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
| FEEDBACK                      | DISABLED | INFORMATION SCHEMA | NULL    | GPL     |
| partition                     | ACTIVE   | STORAGE ENGINE     | NULL    | GPL     |
+-------------------------------+----------+--------------------+---------+---------+
```

Refs:

- https://mariadb.com/kb/en/partitioning-overview/

Maintain Partitions and Explain

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 pless2010 VALUES LESS THAN (2010),
  PARTITION pless2011 VALUES LESS THAN (2011),
  PARTITION pless2012 VALUES LESS THAN (2012),
  PARTITION p2x 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 (Reorganizing Partitions)

```
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 Partition als ganzes löschen

- Vorteil: Schneller als ein Delete (delete from audit_log2 where yr <= 2009; (langsamer)

```
ALTER TABLE audit_log2 DROP PARTITION p2009;
```

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
```

Partitionierung (komplett) entfernen

```
## Partitionierung entfernen, aber Daten sind noch da
## Nur nicht mehr in einzelne Zellen partitioniert
ALTER TABLE audit_log  REMOVE PARTITIONING;
```

Ref:

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

Performance

Performance - Konfiguration von InnoDB Buffer Pool Size

Innodb buffer pool

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

Find out size of innodb_buffer_pool_size

```
## in bytes but free pages in pages (=16KBytes per page)
SHOW VARIABLES LIKE 'innodb_buffer_pool_size';
```

Find free buffer pool pages (linux style)

```
## grep befehl geht nur unter linux
pager grep -i 'free buffers'
```

```
show engine innodb status \G
Free buffers       7905
1 row in set (0.00 sec)
```

Find free buffer pool pages (Windows and Linux)

```
## 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)
```

Change innodb_buffer_pool_size

```
## Windows change in my.ini
## Linux my.cnf 50-server.cnf etc.
innodb_buffer_pool_size=3200M

## Server neu starten
```

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
```

Performance - Unterschied where between und <= and >=

Example

```
select * from actor where actor_id between 10 and 50;
select * from actor where actor_id >= 10 and actor_id <= 50;
```

No performance difference

- https://stackoverflow.com/questions/2692593/between-operator-vs-and-is-there-a-performance-difference

* vs. specific field in field list - select

```
## You need to set up contributions database to test that
use contributions

## Variant 1: ALL Fields

mysql> select * from contributions limit 0,100000
100000 rows in set (0.25 sec)
```

```
## Variant 2: Specific field
## Try this multiple times, because the first time it is not
## in the innodb buffer (cache)

mysql> select vendor_last_name from contributions limit 0,100000;
100000 rows in set (0.04 sec)

## Result: Variant 2 wins over Variant 1
## The difference between these 2 is factor 5x in my case
```

Möglichst keine Funktion in where (spalte) verwenden

```
## Bad for performance

## Bad
## Because, system has to read each row from db, to use function first
## and then compare -> no index used
select * from actor WHERE UPPER(last_name) like ('B%');

MariaDB [sakila](last_name) like 'A%';
+------+-------------+-------+------+---------------+------+---------+------+------+--
-----------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows |
Extra       |
+------+-------------+-------+------+---------------+------+---------+------+------+--
-----------+
|    1 | SIMPLE      | actor | ALL  | NULL          | NULL | NULL    | NULL | 202  |
Using where |
+------+-------------+-------+------+---------------+------+---------+------+------+--
-----------+
1 row in set (0.000 sec)


## Good
select * from actor WHERE last_name like UPPER('b%')
```

How to solve it, if you want to use upper

```
-- Step 1: Create a virtual column
ALTER TABLE actor ADD COLUMN last_name_upper VARCHAR(45) AS (UPPER(last_name))
PERSISTENT;
ALTER TABLE actor ADD COLUMN last_name_lower VARCHAR(45) AS (LOWER(last_name))
PERSISTENT;

-- Step 2: Create Index on last_name_lower
CREATE INDEX idx_actor_last_name_lower ON actor (last_name_lower);

-- Step 3: use new field instead of function
explain extended SELECT last_name,first_name FROM actor WHERE last_name_lower LIKE
```

```
'c%';
```

Referenz:

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

SQL-Rewrite Pager- Subselect

```
explain SELECT film_id, description FROM sakila.film ORDER BY title LIMIT 50, 5;
+----+-------------+-------+------------+------+---------------+------+---------+-----
-+------+----------+----------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref
| rows | filtered | Extra          |
+----+-------------+-------+------------+------+---------------+------+---------+-----
-+------+----------+----------------+
|  1 | SIMPLE      | film  | NULL       | ALL  | NULL          | NULL | NULL    | NULL
| 1000 |   100.00 | Using filesort |
+----+-------------+-------+------------+------+---------------+------+---------+-----
-+------+----------+----------------+
1 row in set, 1 warning (0.00 sec)

## Achtung auf title ist ein index sonst geht es nicht
SELECT film.film_id, film.description
FROM sakila.film
INNER JOIN (
    SELECT film_id FROM sakila.film
    ORDER BY title LIMIT 50, 5
) AS lim USING(film_id);

SELECT film.film_id, film.description FROM sakila.film INNER JOIN (     SELECT film_id
FROM sakila.film     ORDER BY title
LIMIT 50, 5 ) AS lim USING(film_id);

explain SELECT film.film_id, film.description FROM sakila.film INNER JOIN (     SELECT
film_id FROM sakila.film     ORDER B
Y title LIMIT 50, 5 ) AS lim USING(film_id);

explain SELECT film.film_id, film.description FROM sakila.film INNER JOIN (     SELECT
film_id FROM sakila.film     ORDER BY title LIMIT 50, 5 ) AS lim USING(film_id);
+----+-------------+------------+------------+--------+---------------+-----------+---
------+-------------+------+----------+-------------+
| id | select_type | table      | partitions | type   | possible_keys | key       |
key_len | ref         | rows | filtered | Extra       |
+----+-------------+------------+------------+--------+---------------+-----------+---
------+-------------+------+----------+-------------+
|  1 | PRIMARY     | <derived2> | NULL       | ALL    | NULL          | NULL      |
NULL    | NULL        |   55 |   100.00 | NULL        |
|  1 | PRIMARY     | film       | NULL       | eq_ref | PRIMARY       | PRIMARY   | 2
| lim.film_id |    1 |   100.00 | NULL        |
|  2 | DERIVED     | film       | NULL       | index  | NULL          | idx_title |
514     | NULL        |   55 |   100.00 | Using index |
+----+-------------+------------+------------+--------+---------------+-----------+---
------+-------------+------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)
```

Slow Queries (Logging)

Log slow queries

Schritt 1: slow_query_log permanent aktivieren

```
## Minimal
## my.ini /my.cnf
slow-query-log



## Server neu starten.
```

Schritt 2: was ist eine langsame Query ? (long_query_time)

```
## jeder query die >= long_query_time ist, wird geloggt.
## z.B. 0.000001 Sekunden oder langsamer
## Setzen das während der Laufzeit (serverweit)
SET GLOBAL lonq_query_time = 0.000001
## und zusätzlich für die aktuelle Session
SET lonq_query_time = 0.000001
```

Wo ist das slow query log zu finden ?

```
Datenverzeichnis von MariaDB
{HOSTNAME}-slow.log
z.B.
ITSLAB101-slow.log
## Datei lässt sich mit Editor öffnen
```

Wie kann ich mehr Ausgaben in jeden Eintrag bekommen ?

```
## oder direkt in my.ini / my.cnf
SET GLOBAL log_slow_verbosity='query_plan,explain';
SET log_slow_verbosity='query_plan,explain';
```

Log slow queries that do not use indexes

```
## Windows my.ini
[mysqld]

slow_query_log
## standard ist o.k.
long_query_time=10
log_queries_not_using_indexes=ON

```

Analyzing Slow Queries / Indexes

Prinzipien/Grundlagen von Indizies by MariaDB

Prinzipien

Prinzip 1: Nur ein Index pro Tabelle kann verwendet werden (i.d.R.)

- Bei mehreren Indizes die für eine Tabelle gesetzt sind, kann i.d.R. (99 %) nur ein Index verwendet
- Stehen mehrere zur Auswahl, entscheidet sich der Optimierer für den bestmöglichen (aus seiner Sicht)

Prinzip 2: Ein Index kann nur von links nach rechts gelesen werden

```
## Kann einen Index verwenden
select first_name,last_name from actor where last_name like 'B%';



## Kann keinen Index verwenden
## Weil er nichts weiss, bei welchem Buchstaben er anfangen soll
## Es kann A-Z sein.
select first_name,last_name from actor where last_name like '%B%';
```

Indizes - Typen

Primary

- eindeutig, darf nur 1x pro Tabelle 1x erstellet
- Darf nur in den Daten in einer Zeile vorkommen, d.h. es gibt nur eine Zeile mit der actor_id = 1

Unique

- wenn ich einen weiteren eindeutigen Schlüssel haben möchte, kann ich auf eine Spalte einen Unique-Key setzen
- Dann der Wert auch nur in einer Datenzeile für diese Spalte vorkommen

Index

- Ganz normaler Index
- Werte dürfen mehrmals vorkommen

Create unique index

```
CREATE UNIQUE INDEX HomePhone ON Employees(Home_Phone);
```

Find indexes

Show index from table

```
create database showindex;
use showindex;
CREATE TABLE `people` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(25) DEFAULT NULL,
  `last_name` varchar(25) DEFAULT NULL,
  `passcode` mediumint(8) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_passcode` (`passcode`),
  KEY `idx_first_name_last_name` (`first_name`,`last_name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
show index from people
```

Show create table

```
show create table peple
```

show index from

```
show index from contributions
```

Create Index/Delete/Drop Index

```
create index idx_vendor_state on contributions (vendor_state);
```

Drop/Delete Index

```
drop index idx_last_name_first_name on customer;
```

Indizes (Table Scan / Cover Index)

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%';
```

Explain

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
```

```
explain format=json  SELECT a.first_name, a.last_name, fa.film_id FROM film_actor2 fa
INNER JOIN actor2 a ON fa.actor_id = a.actor_id
```

What does type say ?

- https://mariadb.com/kb/en/explain/

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

This one is good for debugging / do not use index at all

```

explain select vendor_city from contributions use index() where vendor_city like 'S%'
```

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)
```

Query Pläne und die Key-Länge

Example

```
 select table_schema,table_name,character_set_name,column_name from
information_schema.columns where table_name = 'donors'
and column_name = 'city';
+---------------+------------+--------------------+-------------+
| table_schema  | table_name | character_set_name | column_name |
+---------------+------------+--------------------+-------------+
| contributions | donors     | utf8               | city        |
+---------------+------------+--------------------+-------------+
1 row in set (0.00 sec)
```

```
mysql> describe donors;
+--------------------+-------------+------+-----+---------+----------------+
| Field              | Type        | Null | Key | Default | Extra          |
+--------------------+-------------+------+-----+---------+----------------+
| donor_id           | int(11)     | NO   | PRI | NULL    | auto_increment |
| last_name          | varchar(70) | YES  | MUL | NULL    |                |
| first_name         | varchar(35) | YES  |     | NULL    |                |
| address_1          | varchar(35) | YES  |     | NULL    |                |
| address_2          | varchar(36) | YES  |     | NULL    |                |
| city               | varchar(20) | YES  | MUL | NULL    |                |
| state              | varchar(15) | YES  |     | NULL    |                |
| zip                | varchar(11) | YES  |     | NULL    |                |
| employer           | varchar(70) | YES  |     | NULL    |                |
| occupation         | varchar(40) | YES  |     | NULL    |                |
| last_name_reversed | varchar(70) | YES  | MUL | NULL    |                |
+--------------------+-------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

```
## only the first part of the combined index is used, because 213 bytes is a
varchar(70) for utf8 - characters in index
## utf8 takes up to 3 bytes per character.
mysql> explain select first_name,last_name from donors where last_name like 'Wi%';
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+-------+----------+--------------------------+
| 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 | 18722 |   100.00 | Using where; Using index |
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+-------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)

## both last_name and first_name are used to filter
## 3 bytes + 70x3 (varchar(70)) + 3Bytes + 35x3 (varchar(35)) = 321
mysql> explain select first_name,last_name from donors where last_name like 'Williams'
and first_name like 'A%';
```

```
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+--------------------------+
| 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 | 321     | NULL |   46 |   100.00 | Using where; Using index |
+----+-------------+--------+------------+-------+-------------------+----------------
---+---------+------+------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

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
```

Index neu aufbauen ?

```
## Aus meiner Sicht sollte das auch so für die aktuelle MariaDB Version 10.6 gelten

For basic cleanup and re-analyzing you can run "OPTIMIZE TABLE ...", it will compact
out the overhead in the indexes and run ANALYZE TABLE too, but it's not going to re-
sort them and make them as small & efficient as they could be.

https://dev.mysql.com/doc/refman/8.0/en/optimize-table.html

However, if you want the indexes completely rebuilt for best performance, you can:

drop / re-add indexes (obviously)
dump / reload the table
ALTER TABLE and "change" using the same storage engine
REPAIR TABLE (only works for MyISAM, ARCHIVE, and CSV)
https://dev.mysql.com/doc/refman/8.0/en/rebuilding-tables.html

If you do an ALTER TABLE on a field (that is part of an index) and change its type,
then it will also fully rebuild the related index(es).
```

Index Stats

General

```
Index stats where introduced in MariaDB 10.4
(These can be used by the query optimizer)

Before that only indexes were used,
now it is also possible these stats into account

The stats are saved in the mysql-database
mysql.column_stats
mysql.table_stats
mysql.index_stats

These are histogramm stats,
they are used automatically from 10.4.1 on,
but they are not created automatically.

You have to perform an operation which is
cost intensive to create them, because a full
table scan is done.
```

```
These stats are
Engine-independent Statistics
```

Notes

- Stats are currently used (MariaDB 10.6) by default because of setting
  - show variables like 'use_stat_tables'
  - -> preferably_for_queries
- But: They are not created automatically (see Howto how to do that)
- Also: They are not deleted automatically

Howto

```
use mysql;
select * from column_stats;
select * from index_stats;
select * from table_stats;
use contributions;
ANALYZE TABLE contributions PERSISTENT FOR ALL;

use mysqL;
select * from column_stats;
select * from index_stats;
select * from table_stats;
```

When will it be used ?

```
2021-08-11: Looks like it is currently used for range-scan for the optimizer,
which table to start with

An excessive example can be found here:

```

Refs:

- https://mariadb.com/kb/en/histogram-based-statistics/
- https://mariadb.com/kb/en/server-system-variables/#use_stat_tables
- Includes Example for how to index specific columns and indexes or exclude them
  - https://mariadb.com/kb/en/engine-independent-table-statistics/

Tools

Percona Toolkit

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

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-query-digest - Windows/MariaDB

Walkthrough

```
## Step 1: install strawberry perl as msi - 64Bits
## https://strawberryperl.com/

## Step 2: Open Browser in RDP and open this Page
## https://www.percona.com/get/pt-query-digest
## Step 3: Rechte Maustaste -> Speichern unter auf Desktop: pt-query-digest.pl

## Step 4: Verschieben in bin - ordner von MariaDB-Server

## Step 5: Console mit MariaDB - Umgebung öffnen.

## Step 6: Analysieren der slow-query-log mit pt-query-digest.pl
## z.B. report.txt
## hier öffnet sich ein 2. Fenster mit der Frage, wo es ausgeführt -> Strawberry Perl
pt-query-digest.pl <pfad-und-name-des-slow-query-logs> > <pfad-zur-neuen-
datei>/report.txt
## z.B.
pt-query-digest.pl ITSLAB101-slow.log > C:\Users\Admin\Desktop\backup\report.txt
```

Ref:

- Achtung: URL (Domain) von percona ist falsch
- http://www.jonathanlevin.co.uk/2012/01/query-digest-on-windows.html

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

```
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

Profiling

Example 1

```
MariaDB [(none)]> SET profiling = 1;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> select * from actor where last_name like 'D%';
ERROR 1046 (3D000): No database selected
MariaDB [(none)]> use sakila;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [sakila]> select * from actor where last_name like 'D%';
+----------+------------+-----------+---------------------+
| actor_id | first_name | last_name | last_update         |
+----------+------------+-----------+---------------------+
|        4 | JENNIFER   | DAVIS     | 2006-02-15 04:34:33 |
|       35 | JUDY       | DEAN      | 2006-02-15 04:34:33 |
|       36 | BURT       | DUKAKIS   | 2006-02-15 04:34:33 |
|       41 | JODIE      | DEGENERES | 2006-02-15 04:34:33 |
|       48 | FRANCES    | DAY-LEWIS | 2006-02-15 04:34:33 |
|       81 | SCARLETT   | DAMON     | 2006-02-15 04:34:33 |
|       89 | CHARLIZE   | DENCH     | 2006-02-15 04:34:33 |
|      100 | SPENCER    | DEPP      | 2006-02-15 04:34:33 |
|      101 | SUSAN      | DAVIS     | 2006-02-15 04:34:33 |
|      106 | GROUCHO    | DUNST     | 2006-02-15 04:34:33 |
|      107 | GINA       | DEGENERES | 2006-02-15 04:34:33 |
|      109 | SYLVESTER  | DERN      | 2006-02-15 04:34:33 |
|      110 | SUSAN      | DAVIS     | 2006-02-15 04:34:33 |
|      123 | JULIANNE   | DENCH     | 2006-02-15 04:34:33 |
|      138 | LUCILLE    | DEE       | 2006-02-15 04:34:33 |
|      143 | RIVER      | DEAN      | 2006-02-15 04:34:33 |
|      148 | EMILY      | DEE       | 2006-02-15 04:34:33 |
|      160 | CHRIS      | DEPP      | 2006-02-15 04:34:33 |
|      166 | NICK       | DEGENERES | 2006-02-15 04:34:33 |
|      173 | ALAN       | DREYFUSS  | 2006-02-15 04:34:33 |
|      188 | ROCK       | DUKAKIS   | 2006-02-15 04:34:33 |
+----------+------------+-----------+---------------------+
21 rows in set (0.001 sec)

MariaDB [sakila]> show profiles;
+----------+------------+----------------------------------------------+
| Query_ID | Duration   | Query                                        |
+----------+------------+----------------------------------------------+
|        1 | 0.00078507 | select * from actor where last_name like 'D%' |
|        2 | 0.00073284 | SELECT DATABASE()                            |
|        3 | 0.00074666 | show databases                               |
|        4 | 0.00117272 | show tables                                  |
|        5 | 0.00190385 | select * from actor where last_name like 'D%' |
+----------+------------+----------------------------------------------+
5 rows in set (0.000 sec)

MariaDB [sakila]> show profile all for query  5 \G
*************************** 1. row ***************************
             Status: Starting
           Duration: 0.000239
           CPU_user: 0.000000
         CPU_system: 0.000238
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: NULL
        Source_file: NULL
        Source_line: NULL
*************************** 2. row ***************************
             Status: checking permissions
           Duration: 0.000185
           CPU_user: 0.000000
         CPU_system: 0.000185
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_parse.cc
        Source_line: 6703
*************************** 3. row ***************************
             Status: Opening tables
           Duration: 0.000039
           CPU_user: 0.000000
         CPU_system: 0.000038
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_base.cc
        Source_line: 4222
*************************** 4. row ***************************
             Status: After opening tables
           Duration: 0.000012
           CPU_user: 0.000000
         CPU_system: 0.000012
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_base.cc
        Source_line: 4505
*************************** 5. row ***************************
             Status: System lock
           Duration: 0.000009
           CPU_user: 0.000000
         CPU_system: 0.000009
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: lock.cc
        Source_line: 337
```

```
*************************** 6. row ***************************
             Status: table lock
           Duration: 0.000016
           CPU_user: 0.000000
         CPU_system: 0.000016
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: lock.cc
        Source_line: 342
*************************** 7. row ***************************
             Status: init
           Duration: 0.000050
           CPU_user: 0.000000
         CPU_system: 0.000050
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_select.cc
        Source_line: 4967
*************************** 8. row ***************************
             Status: Optimizing
           Duration: 0.000022
           CPU_user: 0.000000
         CPU_system: 0.000022
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_select.cc
        Source_line: 1973
*************************** 9. row ***************************
```

```
             Status: Statistics
           Duration: 0.000088
           CPU_user: 0.000000
         CPU_system: 0.000090
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_select.cc
        Source_line: 2451
*************************** 10. row ***************************
             Status: Preparing
           Duration: 0.000038
           CPU_user: 0.000000
         CPU_system: 0.000037
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_select.cc
        Source_line: 2526
*************************** 11. row ***************************
             Status: Executing
           Duration: 0.000009
           CPU_user: 0.000000
         CPU_system: 0.000009
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_select.cc
        Source_line: 4533
*************************** 12. row ***************************
             Status: Sending data
           Duration: 0.000299
           CPU_user: 0.000000
         CPU_system: 0.000301
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_select.cc
        Source_line: 4731
*************************** 13. row ***************************
             Status: End of update loop
           Duration: 0.000016
           CPU_user: 0.000000
         CPU_system: 0.000013
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_select.cc
        Source_line: 5011
*************************** 14. row ***************************
             Status: Query end
           Duration: 0.000007
           CPU_user: 0.000000
         CPU_system: 0.000008
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_parse.cc
        Source_line: 6009
*************************** 15. row ***************************
             Status: Commit
           Duration: 0.000010
           CPU_user: 0.000000
         CPU_system: 0.000010
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_parse.cc
        Source_line: 6055
*************************** 16. row ***************************
             Status: closing tables
           Duration: 0.000008
           CPU_user: 0.000000
         CPU_system: 0.000008
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_base.cc
        Source_line: 786
*************************** 17. row ***************************
             Status: Unlocking tables
           Duration: 0.000006
           CPU_user: 0.000000
         CPU_system: 0.000006
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: lock.cc
        Source_line: 429
*************************** 18. row ***************************
             Status: closing tables
           Duration: 0.000015
           CPU_user: 0.000000
         CPU_system: 0.000016
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: lock.cc
        Source_line: 442
*************************** 19. row ***************************
             Status: Starting cleanup
           Duration: 0.000007
           CPU_user: 0.000000
         CPU_system: 0.000007
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_parse.cc
        Source_line: 6121
*************************** 20. row ***************************
             Status: Freeing items
           Duration: 0.000011
           CPU_user: 0.000000
         CPU_system: 0.000011
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_parse.cc
        Source_line: 8047
*************************** 21. row ***************************
             Status: Updating status
           Duration: 0.000803
           CPU_user: 0.000000
         CPU_system: 0.000040
  Context_voluntary: 0
Context_involuntary: 1
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_parse.cc
        Source_line: 2400
*************************** 22. row ***************************
             Status: Reset for next command
           Duration: 0.000012
           CPU_user: 0.000000
         CPU_system: 0.000011
  Context_voluntary: 0
Context_involuntary: 0
       Block_ops_in: 0
      Block_ops_out: 0
      Messages_sent: 0
  Messages_received: 0
  Page_faults_major: 0
  Page_faults_minor: 0
              Swaps: 0
    Source_function: <unknown>
        Source_file: sql_parse.cc
        Source_line: 2432
22 rows in set (0.001 sec)

MariaDB [sakila]>
```

Example 2:

```
set profiling = 1;
-- please do it twice
SELECT a.first_name,a.last_name,f.* FROM actor a JOIN film_actor fa ON a.actor_id =
fa.actor_id JOIN film f ON fa.film_id = f.film_id WHERE f.description LIKE 'A%';
SELECT a.first_name,a.last_name,f.* FROM actor a JOIN film_actor fa ON a.actor_id =
fa.actor_id JOIN film f ON fa.film_id = f.film_id WHERE f.title LIKE 'A%';
SELECT a.first_name,a.last_name,f.* FROM actor a JOIN film_actor fa ON a.actor_id =
fa.actor_id JOIN film f ON fa.film_id = f.film_id WHERE f.description LIKE 'A%';
SELECT a.first_name,a.last_name,f.* FROM actor a JOIN film_actor fa ON a.actor_id =
fa.actor_id JOIN film f ON fa.film_id = f.film_id WHERE f.title LIKE 'A%';
show profiles;
```

Backup und Restore

Backup und Restore

Unter Linux

```
mysqldump sakila > /usr/src/sakila.sql
mysql sakila < /usr/src/sakila.sql
echo "show tables;" | mysql sakila;


##echo "create schema verleih;" | mysql
## oder
mysql -e 'create schema verleih'
mysql verleih < /usr/src/sakila.sql
```

Unter Windows

```
## mysqldump muss entweder in der %PATH% variablen stehen oder wir müssen
## bin verzeichnis von mysql, sein,
## vorher cmd.exe ausführen über Windows ausführen (oder Suchfeld cmd)
mysqldump -uext -p -h 127.0.0.1 sakila  > C:\Users\Jochen Metzger\Documents\sakila.sql
```

Alle Daten und Struktur sichern

```
## Wichtig --events --routines -> sonst werden diese nicht gesichert. !!!!
mysqldump --all-databases --events --routines > all-structure.sql
```

Nur Struktur sichern

```
mysqldump --no-data --all-databases --events --routines > all-structure.sql
```

Nur daten pro Tabelle

```
mysqldump --no-create-info sakila actor > sakila-actor-data.sql
```

Datenbank sichern und in andere Datenbank (sakila2) zurückspielen (restore)

```
mysqldump -uroot -p --events --routines sakila > sakila.sql

## Schritt 1: Datenbank erstellen
## -e -> execute
mysql -uroot -p -e "CREATE SCHEMA sakila2;"
mysql -uroot -p sakila2 < sakila.sql
```

User und Berechtigungen

Nutzer erstellen/Berechtigungen setzen/entfernen

Creating user first

- The user needs to be created, otherwice it is not working

```
##
CREATE USER training@localhost identfied by '123xy6aT';
```

Login as user training on commandline

```
mysql -utraining -p
```

Wie sind unsere Berechtigungen

```
## Welcher Benutzer bin ich ?
select user();

## Welche Rechte habe ich
show grants;
```

Rights on the different levels

```
GLOBAL: for all databases -> mysql.user
DATBASES: mysql.db
TABLES: mysql.tables_priv
FIELDS: mysql.columns_priv
```

Grant Privileges (all)

```
mysql> grant all on *.* to training@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for training@localhost;
+---------------------------------------------------------+
| Grants for training@localhost                           |
+---------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'training'@'localhost' |
+---------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Grant privileges on a specific database

```
revoke all on *.* to training@localhost
grant all on training.* to training@localhost;
```

Tipps & Tricks

Dummy table DUAL

- https://mariadb.com/kb/en/dual/

Wie kann ich verwendete Storage Engine rausfinden - Table

```
use sakila;
show create table actor;
```

SHOW VARIABLES WITH WHERE

```
SHOW VARIABLES WHERE VARIABLE_NAME like '%slow%' OR VARIABLE_NAME LIKE '%long%'
```

Schnellster Import von Daten mit csv

Example

```
LOAD DATA INFILE '/tmp/test.txt' INTO TABLE test
  FIELDS TERMINATED BY ','  LINES STARTING BY 'xxx';

LOAD DATA INFILE 'data.txt' INTO TABLE tbl_name
  FIELDS TERMINATED BY ',' ENCLOSED BY '"'
  LINES TERMINATED BY '\r\n'
  IGNORE 1 LINES;
```

General/Ref

- Is the quickest way
- Performance Ref: https://jynus.com/dbahire/testing-the-fastest-way-to-import-a-table-into-mysql-and-some-interesting-5-7-performance-results/

Queries in Datenbank (mysql) loggen, die keine Indizes verwenden

```

-- long_query_time can auf 10 sec bleiben
SET GLOBAL log_queries_not_using_indexes = 1;
SET GLOBAL log_output = 'TABLE';
-- last setting
SET slow_query_log = 1;
```

Workaround Materialized View

```
CREATE EVENT `tr_sakila_aggregate`
    ON SCHEDULE
        EVERY 1 MONTH STARTS '2021-08-12 11:20:00'
    ON COMPLETION PRESERVE
    ENABLE
    COMMENT 'Simulation Materialized View (Light)'
    DO BEGIN
   DROP TABLE IF EXISTS sakila_aggregate;
   CREATE TABLE sakila_aggregate AS SELECT actor_id,last_name,first_name FROM actor
WHERE actor_id > 200;
END
```

Zeichensatz umstellen

Example (if it works it is great)

```
-- Do this for every table
ALTER TABLE Tabellenname CONVERT TO CHARACTER SET utf8 COLLATE utf8_general_ci
ALTER DATABASE Datenbankname DEFAULT CHARACTER SET  utf8 COLLATE utf8_general_ci
```

Settings des und des Clients herausfinden

```
## im mysql
mysql>status;

## mit sql z.B. heidisql
show variables like '%char%';
```

Ref. with problems (specific project)

- https://fromdual.com/mariadb-and-mysql-character-set-conversion

Hat InnoDB genug Speicher - Pages

Variante 1

```
SHOW STATUS LIKE '%pages_free%';
```

Variante 2:

```

MariaDB [sakila]> pager grep "Free buffer";
PAGER set to 'grep "Free buffer"'
MariaDB [sakila]> show engine innodb status;
Free buffers       0
1 row in set (0.001 sec)
```

Storage Engines

MyISAM Key Buffer

- http://www.mysqlab.net/knowledge/kb/detail/topic/myisam/id/7200

References/Documentation

Server System Variables

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

MySQL/MariaDB Performance Document

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

MariaDB - Changes in 10.6

- https://mariadb.com/kb/en/changes-improvements-in-mariadb-106/#comment_5088

MariaDB - Installation Linux mit Repos

- https://downloads.mariadb.org/mariadb/repositories/#distro=Ubuntu&distro_release=bionic--ubuntu_bionic&mirror=agdsn&version=10.6

JDBC-Treiber

- https://mariadb.com/kb/en/about-mariadb-connector-j/

Working with database objects

Working with databases

Explanations

```
## open a connection to the mysql-server by entering
mysql
## then you will get
mysql>
```

```
## Comments within mysql-client
## three - in a row
---
```

Create database

```
create database training
create schema training2
```

Show databases

```
mysql
mysql>
;; from here i leave out mysql>
;; so you can easily copy & paste the lines hereafter
show databases

--
-- or
--

show schemas

--
-- or by using information_schema
--

select * from information_schema.schemata;
```

Working with tables

Show tables

```
## within mysql>
## so on the command-line enter:
## mysql (as root)
USE sakila
SHOW TABLES

-- or --

select * from information_schema.TABLES
```

Create table

```
-- only if you want to create table in a completely new database
create schema training;
USE training
CREATE TABLE people (id INT NOT NULL AUTO_INCREMENT, name VARCHAR(20), PRIMARY
KEY(id));
```

Find out the structure of the table

```
## you have to connect to db first with
## mysql
## within mysql>
DESCRIBE people
SHOW CREATE TABLE people
-- or : if you want to know more --
SELECT * from INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='people' AND
TABLE_SCHEMA='training' \G
```

Show indexes

```
SHOW INDEX FROM actor
SHOW INDEXES FROM ACTOR
```

Change table (Add field)

```
--- We want to add a field before name
--- IMPORTANT: BEFORE does not exist
ALTER TABLE people ADD first_name VARCHAR(10) AFTER id;

ALTER TABLE schulungen ADD seats TINYINT unsigned DEFAULT 1, ADD price DECIMAL(6,2);
ALTER TABLE schulungen ADD (room TINYINT unsigned DEFAULT 1,  discount DECIMAL(6,2));
```

Modify a field in table (Change property)

```
ALTER TABLE people
    MODIFY COLUMN first_name VARCHAR(20);
```

Drop a field from the table

```
ALTER TABLE people ADD middle_name VARCHAR(25) BEFORE name;
DESCRIBE people;
ALTER TABLE people DROP COLUMN middle_name;
```

```
## More Examples
--
ALTER TABLE actor ADD in_rente BOOLEAN default true
INSERT INTO actor (first_name,last_name,in_rente) values ('Jochen','Metzger',false)
-- Wieder loswerden
ALTER TABLE actor DROP in_rente;
## add and drop in once command
ALTER TABLE actor ADD in_rente2 BOOLEAN default true, DROP in_rente;
```

Deleting table data (truncate)

```
USE sakila
-- Create table based on other table
CREATE TABLE actorcopy as SELECT * FROM actor;
-- Fields ?
SELECT * FROM actorcopy;
-- Empty it
TRUNCATE TABLE actorcopy;
-- Emptry ?
SELECT COUNT(*) FROM actorcopy;
```

Delete table data (with delete)

Explanation

- Do not use delete when you want to use data of complete table
  - truncate is quicker in this case.
- DELETE FROM ... WHERE ... does a SELECT first

Example

```
USE sakila
CREATE TABLE actorbackup AS SELECT * FROM actor;
SELECT COUNT(*) FROM actorbackup;
DELETE FROM actorbackup WHERE actor_id > 100;
SELECT COUNT(*) FROM actorbackup;
```

Delete complete table

```
USE sakila
DROP TABLE actorbackup;
```

Teststellung - Feld verkleinern

```
ALTER TABLE actor ADD filme tinyint unsigned default 255;
-- Does not work
-- UPDATE actor SET filme = 256  WHERE actor_id = 1;
-- ALTER TABLE actor MODIFY filme smallint unsigned default 256;
INSERT INTO actor (first_name,last_name) values ('Gaucho','Poncho');
INSERT INTO actor (first_name,last_name,filme) values ('Gauchina','Ponchina',32000);

-- Does not work
ALTER TABLE actor MODIFY filme tinyint unsigned;
```

SELECT

Select Beispiele

Einfaches Beispiel (bestimmte Felder)

```
## Zeige alle diese felder aus dieser Tabelle
SELECT <welche_feld_mit_komma_getrennt> FROM <welche tabelle>

-- bitte datenbank sakila verwenden
use sakila
-- zeige actor_id, last_name
SELECT actor_id, last_name FROM actor
```

Einfaches Beispiel (alle Felder)

```


-- bitte datenbank sakila verwenden
use sakila
-- zeige actor_id, last_name
SELECT * FROM actor
```

Einfaches Beispiel mit Bedingung

```
-- SYNTAX
use sakila;
-- SELECT <welche_felder> FROM <welche_tabelle> WHERE
<wo_welches_feld_welchen_wert_hat>
-- Beispiel 1
SELECT actor_id, last_name FROM actor where actor_id = 5;
-- Beispiel 2
SELECT * FROM actor where actor_id = 5;
```

Einfache Bedingung mit Bereich(en)

```
-- SYNTAX
use sakila;
-- SELECT <welche_felder> FROM <welche_tabelle> WHERE
<wo_welches_feld_welchen_wert_hat>
-- Beispiel 1
SELECT actor_id, last_name FROM actor where actor_id > 8;
-- Beispiel 2
SELECT * FROM actor where actor_id > 8;
-- Beispiel 3
SELECT * FROM actor where actor_id > 8 and actor_id < 50;
```

Zwei Bereiche abfragen

```
## Brackets are not necessary, works the same
select * from actor where (actor_id >= 8  and actor_id <=50) or (actor_id >= 100 and
actor_id <= 150)
```

Select Beispiele mit Like

Hintergrund Platzhalter '%' und '_'

- https://elearn.inf.tu-dresden.de/sqlkurs/lektion01/01_07_03_einschr_like.html

Name fängt mit J an

```
## Alle Datensätze, die mit J anfangen
select first_name,last_name from actor where last_name like 'j%'
## mit ausgabe zusätzlichem String
select first_name,last_name,' ist der/die Beste' as bewertung from actor where
last_name like 'j%'
```

Name hört mit n auf

```
select first_name,last_name from actor where last_name like '%n'
```

Name beinhaltet 'q'

```
select first_name,last_name from actor where last_name like '%q%'
```

Platzhalter für genau ein Zeichen (Linux/Windows -> ?)

```
## Alle Zeilen mit Last name McQ, dann genau einem beliebigen Zeichen und dann 'een'
SELECT * FROM sakila.actor where last_name like 'McQ_een'
```

SELECT ORDER BY

Syntax

```

-- Variante 1: (ohne where)
-- SELECT * FROM <welche_tabelle> ORDER BY <welches_feld>


-- Variante 2: (mit where)
-- SELECT * FROM <welche_tabelle> WHERE <welche_bedingung> ORDER BY <welches_feld>
```

Beispiel ohne where

```
## feld aufsteigend
SELECT last_name,first_name,actor_id FROM sakila.actor ORDER BY last_name

## feld absteigend
```

```
SELECT last_name,first_name,actor_id FROM sakila.actor ORDER BY last_name desc

## erstes feld aufsteigend, zweites feld absteigend
SELECT last_name,first_name,actor_id FROM sakila.actor ORDER BY last_name,first_name
DESC
```

Beispiel mit where

```
SELECT last_name,first_name,actor_id FROM sakila.actor WHERE last_name like 'J%' ORDER
BY last_name ASC,first_name DESC
```

Unterschiede einfache und doppelte Hochkommas bei Oracle/MySQL

MySQL

```
Hochkommas werden nur bei der Abfrage des Wertes verwendet, z.B

select * from actor where last_name like 'A%'
## das ist das gleiche wie:
select * from actor where last_name like "A%"
```

MySQL - Wann nehme ich einfache, wann doppelte ?

```
## z.B. wenn ich ein Hochkomma in der Abfrage brauche, z.B
## damit besser lesbar für Admin/Entwickler
select * from actor where last_name like "O'Connor"
## Alternativ geht auch:
## Verfahren: Escapen
select * from actor where last_name like 'O\'Connor'
```

Oracle:

```
## Für Feldname und Ausgabe Feldname (Alias) Doppelte hochkommas.
## z.B
## Hier immer das doppelte Hochkomma, weil es um den Identifier/Bezeichner geht
SELECT  'Hello, world!'    AS "My Greeting"

### Für String und Date - Werte einfache Hochkommas
select * from actor where last_name like 'A%'
```

INSERT/UPDATE

Praktisches Beispiel und erweitertes insert - INSERT

Beispiel

```
INSERT INTO actor (first_name,last_name) values ('Joe','Manchos');
```

Nicht-optimales Beispiel

```
## Problem die Spaltenzahl und die Spaltennamen können sich ändern
## dann geht das insert nicht mehr

### 2022-01-01
### Folgende Felder
### first_name
### last_name

### 2022-03-01
### Folgender Felder nach Strukturänderung
### first_name
### middle_name
### last_name

INSERT INTO actor values ('Joe','Manchos');
```

Erweitertes Insert

```
## Mehrere Wertepaare einfügen - geht schneller als einzelne Inserts
INSERT INTO actor (first_name,last_name) values ('Joe','Metzgeros'),
('Hans','Mustermann');
```

Referenz

- https://www.w3schools.com/sql/sql_insert.asp

Praktisches Beispiel - Update

Einfaches Beispiel ein Datensatz ändern

```
## Variante 1
UPDATE actor SET last_name = 'GUINESSA' WHERE actor_id = 1
## Variante 2 - 2 Felder ändern
UPDATE actor SET first_name='PENELOPEZ',last_name = 'GUINESS' WHERE actor_id = 1
```

Referenz

- https://www.w3schools.com/sql/sql_update.asp

DELETE

Einfaches Delete Beispiel

Example

```
DELETE FROM actor WHERE id = 200
```

Reference

- https://www.mysqltutorial.org/mysql-delete-statement.aspx

Datentypen

Integer - INT - Datentypen

Reference

- https://dev.mysql.com/doc/refman/8.0/en/integer-types.html

Basics

Connection to DB + exit

General Explanation

- Step 1: connection to db
- Step 2: using specific database
- 1. ▪ 2. can be done in one step

Connection as Root (without using -u/--user and db)

```
## If you are root and are connecting locally (socket), you do not need to enter a
passwort on root
## Why ?
mysql> use mysql
mysql> select user,host,plugin from user where user = 'root' and host = 'localhost';
```

```
root@mysql-server:~#whoami
root
## this work, because we are connecting locally
## by default mysql uses the user we are logged in with
mysql
```

Connection with credentials

```
mysql -uroot -p
## passwort auf der Kommandozeile eingeben
```

Connection to remote mysql - server

```
mysql -u root -p -h 10.10.9.117
```

mysql-client

Aufrufen unter Windows

- Programme -> Mariadb -> Mariadb Client
- Passwrot eingeben

Hilfe

```
help
```

Wichtige Datenbank - Objekte anzeigen

```
show databases;
```

Basics

```
mysql
mysql>

## Wie kommen wie raus ?
exit;
```

Delimiter

```
Normalerweise ";"

Ist zum Trennen von Befehlen
```

Use user and password automatically

```
nano /root/.my.cnf
## BE CAREFUL EVERYBODY CAN LOGIN AS ROOT TO MYSQL NOW
## in there
[mysql]
user=root
password=root-password-on-your-system
```

Charset-Collations

server system variablen abfragen

```
mysql> show session variables like '%hostname%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| hostname      | trn01 |
+---------------+-------+
1 row in set (0.00 sec)

mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| trn01      |
+------------+
1 row in set (0.00 sec)
```

Working with the data modelling language (DML's)

Working with INSERT

```
## Always use the field-names
INSERT INTO actor (first_name,last_name) values ('John','Smith');

## Extended inserts are quick (better than single inserts)
INSERT INTO actor (first_name,last_name) values ('John','Peters'),
('Mandy','Johnsson');
```

Tipps & Tricks / Do Not

Dump/SQL-File einspielen auf der Kommandozeile - Windows

```
mysql -u root -p < C:\Users\Admin\Downloads\test.sql
```

Tipps & Tricks

Best Practice DBAL - Kochrezept

Ausgangsituation

```
DBAL erstellt query
Query ist langsam
Was tun ?
```

Steps 1+2

```
### An das SQL-Statement rankommen.



### 1. Analyse, warum ist Query langsam (kann ich von aussen etwas tun, um die Query
zu beeinflussen ohne sie zu ändern
explain <sql-statement>

### Sichtprüfung, fehlen Keys ??
##
##explain select vendor_city,vendor_state from contributions where vendor_first_name
like 'D%';
##+----+-------------+---------------+------------+------+---------------+------+-----
----+------+---------+----------+-------------+
##| id | select_type | table         | partitions | type | possible_keys | key  |
key_len | ref  | rows    | filtered | Extra       |
##+----+-------------+---------------+------------+------+---------------+------+-----
----+------+---------+----------+-------------+
##|  1 | SIMPLE      | contributions | NULL       | ALL  | NULL          | NULL | NULL
| NULL | 2028402 |    11.11 | Using where |
##+----+-------------+---------------+------------+------+---------------+------+-----
----+------+---------+----------+-------------+
## Gibt es bei dem type einen Eintrag ALL.

### Oder da ist ein key drin, aber der wird garnicht verändert.
### Index setzen
```

Step 3.

```
Query analysieren -> Schritt debuggen.
1) Links -> Rechts: Entweder von Links (sprich: was sind die ersten Daten, die ich
brauche, welche dann, welche dann usw.)

2) Rechts -> Links -> immer mehr wegnehmen und gucken, ob es schneller wird.
```

Step 4:

```
index hint.
```

Step 5: Customized Query

References

Examples Left Join

- https://www.quackit.com/mysql/examples/mysql_left_join.cfm

Notes on Specific MySQL Knowledge

- https://www.burnison.ca/notes

Many Sakila Example Queries

- https://github.com/ashok-bidani/MySQL-Sakila-queries-and-joins

Helpful Examples

- https://www.quackit.com/mysql/examples/mysql_group_by_clause.cfm

Übungen

Übung Update/Insert

```
1. In einer Anweisung alle Datensätze ändern in denen der erste Name JUDY und der
Nachname DEAN ist -> dort Vorname in JAMES ändern.

UPDATE actor SET first_name = 'JAMES' where last_name='DEAN' and first_name='JUDY';



2. In einem SQL-Statement 2 neue Datensätze einfügnen. 1. Mann, Josef  2. Mannheim,
Martha

INSERT INTO actor (first_name, last_name) values ('Mann','Josef'),
('Mannheim','Martha');
```