**Inhalt**

[**Datenbankmodelle** 4](#_Toc159917107)

[Hierarchische Datenbanken 4](#_Toc159917108)

[Relationale Datenbanken: 4](#_Toc159917109)

[Objektrelationale Datenbanken: 5](#_Toc159917110)

[Objektorientierte Datenbanken: 5](#_Toc159917111)

[**Datenbanktheorie zu relationalen Datenbanken** 6](#_Toc159917112)

[Strukturierte Datenorganisation 6](#_Toc159917113)

[ACID-Eigenschaften 6](#_Toc159917114)

[Datenintegrität und -konsistenz 6](#_Toc159917115)

[Datenbanksprache SQL 6](#_Toc159917116)

[Datenabstraktion durch Views 6](#_Toc159917117)

[Normalisierung 7](#_Toc159917118)

[Skalierbarkeit und Flexibilität 7](#_Toc159917119)

[Erweiterte Funktionen durch objektrelationale DBMS: 7](#_Toc159917120)

[**Datenbanksprache SQL** 7](#_Toc159917121)

[DDL (Data Definition Language) 7](#_Toc159917122)

[DML (Data Manipulation Language) 8](#_Toc159917123)

[DRL (Data Retrieval Language) 8](#_Toc159917124)

[TCL (Transaction Control Language) 8](#_Toc159917125)

[DCL (Data Control Language) 8](#_Toc159917126)

[**Datensätze** 9](#_Toc159917127)

[**Allgemeine Repetitionsfragen** 9](#_Toc159917128)

[**Normalform - Wie unterscheiden sich die erste, zweite und dritte Normalform?** 11](#_Toc159917129)

[**Standartbefehle in MariaDB** 12](#_Toc159917130)

[Hilfe anzeigen 12](#_Toc159917131)

[Liste aller Datenbanken anzeigen 12](#_Toc159917132)

[Mit einer bestimmten Datenbank verbinden 12](#_Toc159917133)

[Alle Tabellen anzeigen 12](#_Toc159917134)

[Beschreibung (Struktur) einer Tabelle anzeigen 12](#_Toc159917135)

[Bezeichnungen 13](#_Toc159917136)

[ERD / ERM 13](#_Toc159917137)

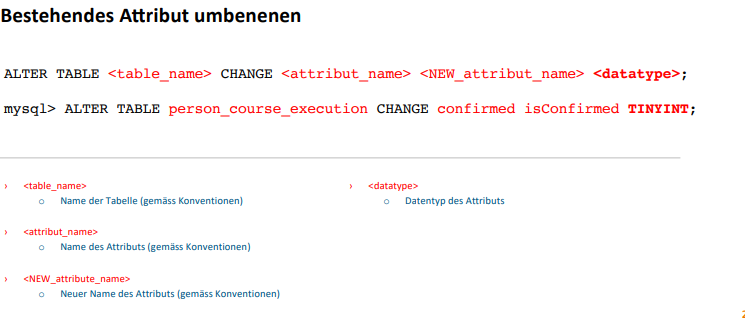
[DBM 13](#_Toc159917138)

[**Commands in SQL** 14](#_Toc159917139)

[Neue Tabelle erstellen 14](#_Toc159917140)

[Neues Attribut hinzufügen 15](#_Toc159917141)

[Bestehendes Attribut umbenennen 15](#_Toc159917142)

[ 15](#_Toc159917143)

[Bestehendes Attribut löschen 16](#_Toc159917144)

[Foreign Key löschen 16](#_Toc159917145)

[Daten in die Tabelle einfügen 17](#_Toc159917146)

[COMMIT 17](#_Toc159917147)

[ROLLBACK 17](#_Toc159917148)

[Daten abfragen (SELECT) 19](#_Toc159917149)

[SELECT 19](#_Toc159917150)

[FROM 19](#_Toc159917151)

[WHERE 19](#_Toc159917152)

[GROUP 20](#_Toc159917153)

[HAVING 20](#_Toc159917154)

[ORDER 20](#_Toc159917155)

[LIMIT 20](#_Toc159917156)

[Daten abfragen (Attribute) 20](#_Toc159917157)

[Daten abfragen (WHERE, Operatoren, AND/OR, Funktionen) 21](#_Toc159917158)

[Daten abfragen (ORDER BY) 22](#_Toc159917159)

[Daten abfragen (Funktionen) 23](#_Toc159917160)

[Daten abfragen (LIKE Patterns) 23](#_Toc159917161)

[% 23](#_Toc159917162)

[\_ 23](#_Toc159917163)

[Daten abfragen (LIMIT) 24](#_Toc159917164)

[**Funktionen in MariaDB SQL** 24](#_Toc159917165)

[DESC 24](#_Toc159917166)

[ASC 24](#_Toc159917167)

[**Gruppenfunktionen** 24](#_Toc159917168)

[COUNT 24](#_Toc159917169)

[COUNT(\*) 25](#_Toc159917170)

[MIN 25](#_Toc159917171)

[MAX 25](#_Toc159917172)

[AVG 25](#_Toc159917173)

[SUM 25](#_Toc159917174)

[Modulo Operator (%) 25](#_Toc159917175)

[Runden 25](#_Toc159917176)

[Quadratwurzel 25](#_Toc159917177)

[Initialen des Namens 26](#_Toc159917178)

[Datensatz erstellen: 26](#_Toc159917179)

[Datensatz löschen: 26](#_Toc159917180)

[Indexes 26](#_Toc159917181)

[-- Vor dem Index 26](#_Toc159917182)

[-- Index erstellen 27](#_Toc159917183)

[-- Nach dem Index 27](#_Toc159917184)

[Subqueries (Table) 27](#_Toc159917185)

[Subqueries (Value) 27](#_Toc159917186)

[Tabellenstruktur analysieren (DESCRIBE / DESC) 28](#_Toc159917187)

[**Aufgabe 5.2: SQL-Abfragen machen** 28](#_Toc159917188)

[**Daten importieren** 29](#_Toc159917189)

[**Daten exportieren** 30](#_Toc159917190)

[**DFB Dumb – Export** 30](#_Toc159917191)

[Exportieren einer bestimmten Datenbank: 30](#_Toc159917192)

[ mysqldump.exe -u [Benutzername] -p [Datenbankname] > [Zielordner]/[Dateiname].sql 30](#_Toc159917193)

[Exportieren aller Datenbanken: 30](#_Toc159917194)

[**DB Dumb – Import** 30](#_Toc159917195)

[Erstellen einer neuen Datenbank (optional): 30](#_Toc159917196)

[Importieren eines Datenbankdumps: 30](#_Toc159917197)

[**Benutzer und Zugriffe / Berechtigungen** 31](#_Toc159917198)

[Benutzer erstellen und Passwort setzten 31](#_Toc159917199)

[Benutzer umbenennen 31](#_Toc159917200)

[Berechtigungen zuweisen 31](#_Toc159917201)

[Berechtigungen entziehen 31](#_Toc159917202)

[Berechtigungen anzeigen 31](#_Toc159917203)

[Benutzer löschen 31](#_Toc159917204)

***Wichtig für DBM!***

***DBM immer auf Englisch, klein, singular, Identifikationsschlüssel mit "id", Fremdschlüssel mit dem Prefix "fk\_"***

***WICHTIG FÜR DATENBANK!***

***DROP DATABASE IF EXISTS carwebshop;***

***CREATE DATABASE carwebshop;***

***USE carwebshop;***

# **Datenbankmodelle**

Was ist eine Datenbank?: Eine Datenbank ist eine Sammlung von Tabellen

## Hierarchische Datenbanken

*Funktionsweise*:

Hierarchische Datenbanken organisieren Daten in einer Baumstruktur, in der jedes Element eine Eltern-Kind-Beziehung zu einem übergeordneten Element hat. Ein Element kann mehreren untergeordneten Elementen zugeordnet sein, aber nur einem übergeordneten Element.

*Merkmale*:

Einfache Struktur: Daten sind in einer Baumstruktur organisiert.

Effizient für bestimmte Anwendungen wie die Organisation von Dateisystemen.

Die Beziehung zwischen Daten ist streng hierarchisch.

***Zusammenfassung:***

* Hierarchische Speicherung der Daten in einem sequenziellen File (Land, Kanton, Ort/Ort, Bezirk, Gemeinde, Strasse, Hausnummer, Name, Vorname, Geburtsdatum
* Neue Daten erfordern ein umkopieren, sortieren der Daten/File

## Relationale Datenbanken:

*Funktionsweise*:

Relationale Datenbanken speichern Daten in Tabellen, wobei jede Tabelle Datensätze (Zeilen) enthält, die aus Attributen (Spalten) bestehen. Beziehungen zwischen Tabellen werden durch Primärschlüssel und Fremdschlüssel definiert.

*Merkmale*:

Tabellenstruktur: Daten werden in Tabellen mit Zeilen und Spalten gespeichert.

Deutlich definierte Beziehungen zwischen Daten.

Verwendung von Abfragesprachen wie SQL (Structured Query Language) zur Datenmanipulation und Abfrage.

## Objektrelationale Datenbanken:

*Funktionsweise*:

Objektrelationale Datenbanken erweitern das relationale Datenbankmodell, indem sie objektorientierte Eigenschaften wie Vererbung, Polymorphismus und Kapselung unterstützen. Sie ermöglichen die Speicherung komplexer Datenstrukturen wie Objekte und komplexe Datentypen in der Datenbank.

*Merkmale:*

Kombiniert Eigenschaften von relationalen und objektorientierten Datenbanken.

Unterstützt objektorientierte Konzepte wie Klassen, Vererbung und Methoden.

Bietet Flexibilität bei der Speicherung und Abfrage komplexer Datenstrukturen.

***Zusammenfassung über Relationale und Objektrelationale Datenbanken***

* Objekt(-relationale) Datenbanken erlauben benutzerdefinierte Datentypen und Objekte - Daten werden in Themenkreisen (Entitäten) in Form von Tabellen gespeichert
* Flexibler als Hierachische Datenbanken
* Daten verschiedener Tabellen sind unabhängiger
* Einfach erweiterbar
* Unübersichtlicher und schwerfälliger

## Objektorientierte Datenbanken:

*Funktionsweise*:

Objektorientierte Datenbanken speichern Daten als Objekte, die sowohl Daten (Attribute) als auch Methoden zur Datenmanipulation enthalten können. Sie unterstützen Konzepte wie Vererbung, Polymorphismus und Kapselung.

*Merkmale*:

Daten werden als Objekte gespeichert, wodurch komplexe Datenstrukturen und Beziehungen leichter abgebildet werden können.

Unterstützt objektorientierte Prinzipien wie Klassen, Vererbung, Polymorphismus.

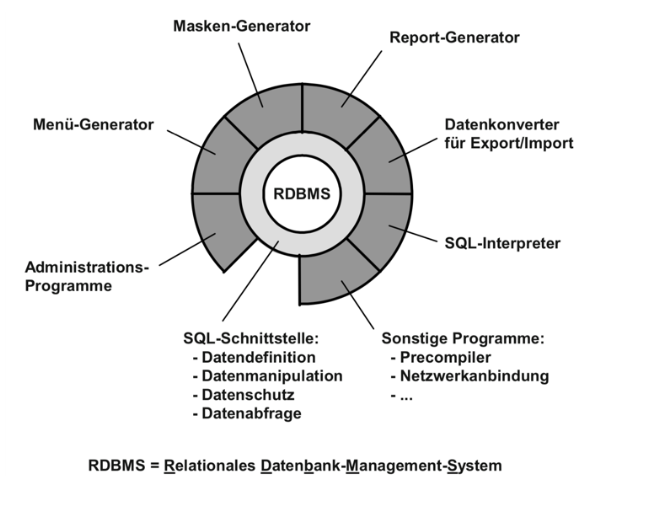
Geeignet für Anwendungen, bei denen die Objektorientierung im Vordergrund steht, wie zum Beispiel in der Softwareentwicklung.

**Zusammenfassung:**

* Objekt steht im Zentrum nicht die Tabelle
* Beinhaltent auf Methoden, um Daten zu ändern
* Immer noch im Nischendasein

# **Datenbanktheorie zu relationalen Datenbanken**

Eine relationale Datenbank:

* verwaltet Daten strukturiert und überschaubar
* regelt den Zugriff auf Datensätze
* bietet Schnittellen zur flexiblen Interaktion durch externen Applikationen
* speichert Daten nicht redundant
* gewährleistet die Datenintegrität
* gewährleistet die Referenzielle Integrität

## Strukturierte Datenorganisation

Relationale Datenbanken organisieren Daten in Tabellen, wobei jede Tabelle aus Zeilen (Datensätzen) und Spalten (Attributen) besteht. Dies ermöglicht eine klare und effiziente Datenstrukturierung

ACID-Eigenschaften Zur Sicherstellung der Datenintegrität und des zuverlässigen Datenbankbetriebs erfüllen relationale Datenbanken die ACID-Eigenschaften: Atomarität, Konsistenz, Isolation und Dauerhaftigkeit

Datenintegrität und -konsistenz: Durch Integritätsbedingungen (wie z.B. eindeutige Schlüssel) und Transaktionsmanagement wird gewährleistet, dass Daten stets korrekt und konsistent sind

Datenbanksprache SQL: Für den Zugriff und die Manipulation der Daten wird SQL (Structured Query Language) verwendet. SQL basiert auf der relationalen Algebra und ermöglicht das Abfragen, Einfügen, Aktualisieren und Löschen von Daten

Datenabstraktion durch Views: Relationale Datenbankmanagementsysteme (RDBMS) unterstützen das Erstellen von Views, die als virtuelle Tabellen dienen. Diese ermöglichen es, Daten in spezifischer Form darzustellen, ohne die zugrunde liegenden physischen Tabellen zu verändern

Normalisierung: Um Redundanzen zu vermeiden und Anomalien zu verhindern, wird das Datenbankschema mittels Normalisierung optimiert. Dies trägt zur Vereinfachung der Wartung und zur Gewährleistung der Datenkonsistenz bei

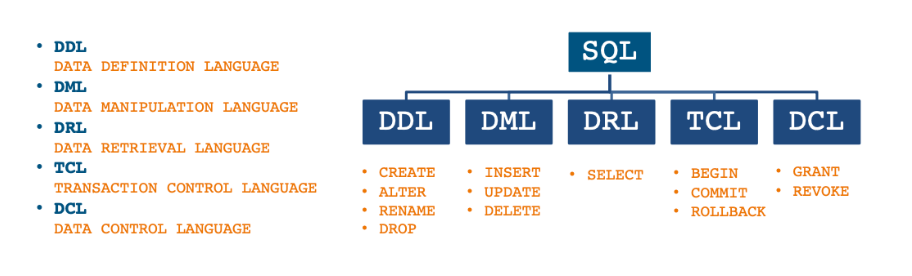
Skalierbarkeit und Flexibilität: Relationale Datenbanken sind aufgrund ihres Designs in der Lage, sich anwachsende Anforderungen anzupassen. Die Verwendung von SQL und die Unterstützung von Transaktionen ermöglichen flexible und skalierbare Datenmanagementlösungen

Erweiterte Funktionen durch objektrelationale DBMS: Einige relationale Datenbanken erweitern ihre Funktionalität um objektorientierte Eigenschaften, wie die Vererbung bei der Definition und Abfrage von Tabellen, um die Handhabung zu vereinfachen

# **Datenbanksprache SQL**

SQL = Structured Query Language

Die Datenbanksprach besitzt vier Elemente:

* Datendefinition (DDL, Data Definition Language)
* Datenmanipulation (DML, Data Manipulation Language)
* Datenabfrage (DRL, Data Retrieval Language)
* Datenschutz (DSL, Data Security Language) —> Besteht aus TCL (Transaction Control Language) und DCL (Data Control Language)

## DDL (Data Definition Language)

Eine Sprache, die verwendet wird, um die Struktur von Datenbanken zu definieren oder zu ändern, z. B. Tabellen, Indizes und Beziehungen.

* CREATE Erzeugt eine neue Datenbank, Tabelle, usw.
* ALTER Ändert die Tabellenstruktur (z.B Spalten hinzufügen, löschen oder ssssssssssssssssumbenennen)
* RENAME Ändert den Namen eines vorhandenen Datenbankobjekts cccccccccccccccccc dddddddddddd(z. B. Tabelle, Spalte) ohne die Struktur zu beeinflussen
* DROP Löscht den Datenbankinhalt und die Protokolldatei für die Datenbank

## DML (Data Manipulation Language)

Eine Sprache, die verwendet wird, um Daten in einer Datenbank zu manipulieren, z. B. zum Einfügen, Aktualisieren oder Löschen von Datensätzen.

* INSERT Fügt neue Datensätze in eine Tabelle ein.
* UPDATE Ändert vorhandene Daten in einer Tabelle.
* DELETE Löscht Daten aus einer Tabelle.

## DRL (Data Retrieval Language)

Eine Sprache, die verwendet wird, um Daten aus einer Datenbank abzurufen, typischerweise durch die Ausführung von Abfragen, z. B. SELECT-Anweisungen.

* SELECT Wählt aus die Daten aus der Datenbank aus (z.b: SELECT \* FROM user;)

## TCL (Transaction Control Language)

Eine Sprache, die verwendet wird, um Transaktionen in einer Datenbank zu verwalten, einschließlich des Starts, Abschlusses oder Rollbacks von Transaktionen, um Datenkonsistenz sicherzustellen.

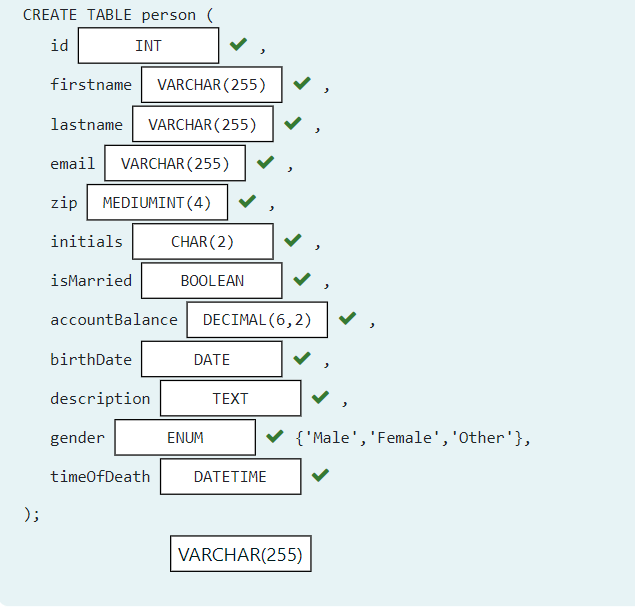
* BEGIN Startet eine Aktion.
* COMMIT Speichert die Änderungen ab
* ROLLBACK Beendet eine Aktion und macht Änderungen rückgängig.

## DCL (Data Control Language)

Eine Sprache, die verwendet wird, um die Zugriffsrechte auf Datenbankobjekte zu steuern, einschließlich der Gewährung oder Entziehung von Berechtigungen für Benutzer oder Rollen.

* GRANT Gewährt Zugriffsrechte auf Datenbankobjekte.
* REWOKE Entzieht Zugriffsrechte auf Datenbankobjekte.

# **Datensätze**



# **Allgemeine Repetitionsfragen**

**Was ist eine Relationale Datenbank?**

Eine relationale Datenbank….

* verwaltet Daten strukturiert und überschaubar
* regelt den Zugriff auf Datensätze
* muss flexibel sein um bestehende Daten in anderen Applikationen nutzen zu können
* speichert Daten nicht redundant
* gewährleistet die Datenintegrität

**Nennen Sie drei verschiedene relationale Datenbank Management Systeme DBMS?**

* MSSQL
* Oracle
* MySQL

**Welche Aufgaben erfüllt ein DBMS für Sie als Entwickler?**

DBMS stellt sicher, dass die Eigenschaften einer Datenbank (Datenkonsistenz, Datenintegrität, …) sichergestellt ist, Verwaltet die Daten strukturiert und ermöglicht mir die Manipulation dieser über eine SQL-Schnittstelle.

**Wie greifen Benutzer (keine Entwickler) auf eine Datenbank zu?**

Nicht Entwickler greifen mittels Applikationen auf die Datenbank zu, welche im Hintergrund die SQL Schnittstelle konsumieren resp. ansteuert.

**Handelt es sich bei SQL um eine Programmiersprache?**

Nein, es handelt sich bei SQL um eine strukturierte Abfragesprache (Structured Query Language).

**Ist die Sprache SQL auf allen Datenbanken gleich?**

Der SQL Standart ist auf allen Datenbanken gleich. Nur existieren je nach Produkt (MSSQL, …) gewisse Dialekte, welche diese um spezifische Funktionen erweitern.

**Was versteht man unter NoSQL Datenbank?**

NoSQL bezeichnet Datenbanken, die einen nicht-relationalen Ansatz verfolgen und damit mit der langen Geschichte relationaler Datenbanken brechen. Diese Datenspeicher benötigen keine festgelegten Tabellenschemata und versuchen Joins zu vermeiden. Sie skalieren dabei horizontal. Im akademischen Umfeld werden sie häufig als „strukturierte Datenspeicher“ bezeichnet.

**Nennen Sie drei NoSQL Datenbanksysteme?**

CouchDB, MongoDB, Cassandra

**Was versteht man unter Datenbank Schema?**

Ein Datenbankschema ist eine Skizze einer geplanten Datenbank als Modell dargestellt. Es enthält selbst keine Daten.

**Wozu dient ein Entity Relationship Model (ERM)?**

Ein ERM dient die Entitäten, dessen Attribute und Relationen in einem Modell einfach und standarisiert darzustellen

**Wie sieht ein Enhanced Entity Relationship (EER) Model aus?**

Das erweiterte Entity-Relationship-Modell in der Informatik ist ein übergeordnetes oder konzeptionelles Datenmodell, das Erweiterungen des ursprünglichen Entity-Relationship-Modells enthält, das beim Entwurf von Datenbanken verwendet wird.

**Was bedeuten die Begriffe: Entität, Attribut und Tupel?**

*Entität*: Ist eine logische Sammlung von Attributen gemäss dem gegeben Kontext. Die Entität wird in relationalen Datenbanken mit der Tabelle gleichgestellt.

*Attribut*: Attribute beschreiben die verschiedenen Felder/Spalten einer Entität resp. Einer Tabelle.

*Tupel*: Ein Tupel beschreibt ein Datensatz in einer Tabelle über sämtliche Spalter

**Was versteht man unter Datenkonsistenz?**

Die „Datenkonsistenz“ beschreibt die Widerspruchsfreiheit innerhalb einer Datenbank bzw. die Korrektheit der dort gespeicherten Daten, z.B. dass alle Replikas eines Datenfelds identisch sind. In relationalen Datenbanken versteht man unter Konsistenz die Integrität von Daten.

**Was bedeutet Redundanz im Zusammenhang mit einer Datenbank?**

Eine Information ist dann Redundant, wenn Sie mehrfach vorkommt. Redundanz im Zusammenhang mit Datenbanken bedeutet, dass gewisse Information demnach mehrfach vorhanden sind

**Was versteht man unter Normalform?**

Es gibt verschiedene Ausmasse, in denen ein Datenbankschema gegen Anomalien gefeit sein kann. Je nachdem spricht man davon, dass es in erster, zweiter, dritter usw. Normalform vorliege. Diese Normalformen sind durch bestimmte formale Anforderungen an das Schema definiert.

# **Normalform - Wie unterscheiden sich die erste, zweite und dritte Normalform?**

*1. Normalform*: ist dann gegeben, wenn alle Attributwerte atomar sind. Dabei ist ein Attribut dann atomar, wenn es weder aus kleineren Einheiten zusammengesetzt ist, noch mehrwertig ist.

*2. Normalform*: ist dann gegeben, wenn sie in der ersten Normalform ist und jedes Nichtschlüsselattribut von jedem Schlüssel voll funktional abhängig ist. Konkret heisst das, dass Nichtschlüsselattribute vom kompletten Schlüssel und nicht nur von einem Teil des Schlüssels abhängen müssen

*3. Normalform*: Ist dann gegeben, wenn sie in der ersten und zweiten Normalform vorliegt und kein Nichtschlüsselattribut von (irgend)einem Schlüssel transitiv abhängt. Um eine transitive Abhängigkeit also auszuschließen, dürfen alle Nichtschlüsselattribute nicht von anderen Nichtschlüsselattributen abhängig sein

**Was versteht man unter Primär- und Fremdschlüssel?**

Primärschlüssel identifiziert eindeutig ein Tupel in einer Tabelle. Der Fremdschlüssel identifiziert einen Primärschlüssel in einer andere Tabelle.

**Was ist referentielle Integrität?**

Unter referentieller Integrität (RI) versteht man die Bedingungen, die zur Sicherung der Datenintegrität bei Nutzung relationaler Datenbanken beitragen können. Nach der RI-Regel dürfen Datensätze (über ihre Fremdschlüssel) nur auf existierende Datensätze verweisen.

**Was bringt Object-Relational Mapping (ORM)?**

ORM ermöglicht es einem Anwendungsprogramm, Objekte in einer relationalen Datenbank abzulegen, sodass die Datenbank für das Programm wie eine objektorientierte Datenbank erscheint.

**Wie unterscheiden sich die Datentypen Char und Varchar?**

Char behält seine fix angegeben Grösse, während Varchar sich auf den effektiv verwendeten Speicherplatz reduziert. Somit is Varchar in jedem Fall sparsamer mit Speicher.

**Was ist ein Character Set und wie unterscheiden sich ASCII, LATIN1 und UTF-8**

Ein Character Set definiert welch Zeichen wie in die Daten geschrieben werden müssen. Im Ursprünglichen ASCII Character Set waren keine Umlaute, Kyrillische, Griechiesche, … - Zeichen vorgehsen. Latin1 unterstützt alle Zeichen von West Europa. Mit ‘SHOW CHARACTER SET;’ können alle installierten Character Sets angezeigt werden.

**Wie unterscheiden sich die Datentypen Integer und Float?**

Decimal speichert exakte Werte, während Float nur annähernde Werte speichert. Decimal verliert bei einer Ganzzahl-Multiplikation und Division an Genauigkeit, Float hingegen nicht.

**Wozu verwendet man den Datentyp Decimal?**

Zur Darstellung von Fliesskommazahlen. Hierbei müssen die Anzahl von Nachkommastellen und die gesamte Anzahl von Stellen angegeben werden.

**Wie unterscheiden sich die Datentypen Timestamp und Datetime?**

Timestamp speichert Daten von '1970-01-01 00:00:01' UTC bis '2038-01-19 03:14:07' UTC und konvertiert automatisch in die aktuelle Zeitzone. Datetime hat einen größeren Bereich von '1000-01-01 00:00:00' bis '9999-12-31 23:59:59' und konvertiert nicht automatisch in die Zeitzone.

**Wie unterscheiden sich 0 und NULL?**

NULL steht für ein nicht definierter Wert, während 0 für eine natürliche Zahl (zero) gehalten wird.

# **Standartbefehle in MariaDB**

## Hilfe anzeigen

mysql> \?

## Liste aller Datenbanken anzeigen

mysql> SHOW DATABASES;

## Mit einer bestimmten Datenbank verbinden

mysql> USE mysql;

## Alle Tabellen anzeigen

mysql> SHOW TABLES;

## Beschreibung (Struktur) einer Tabelle anzeigen

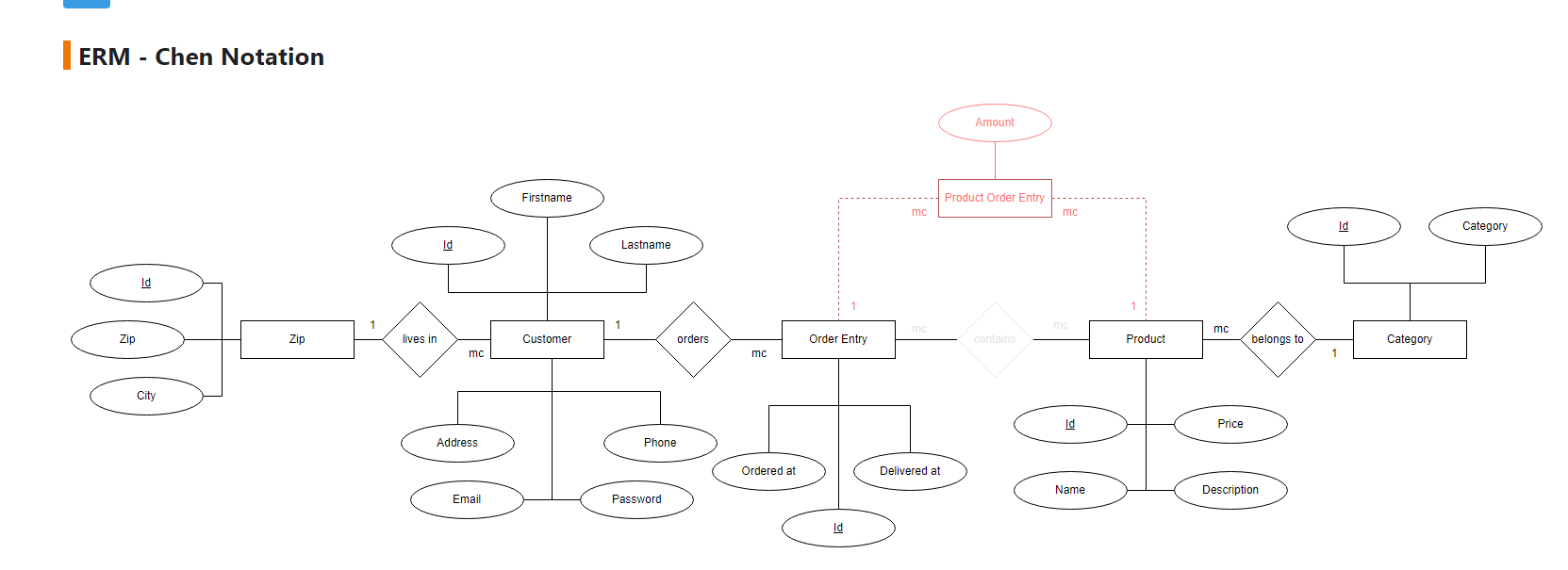
mysql> DESCRIBE user;  
mysql> DESC user;

# Bezeichnungen

## ERD / ERM

"Entity-Relationship Diagram / Model"

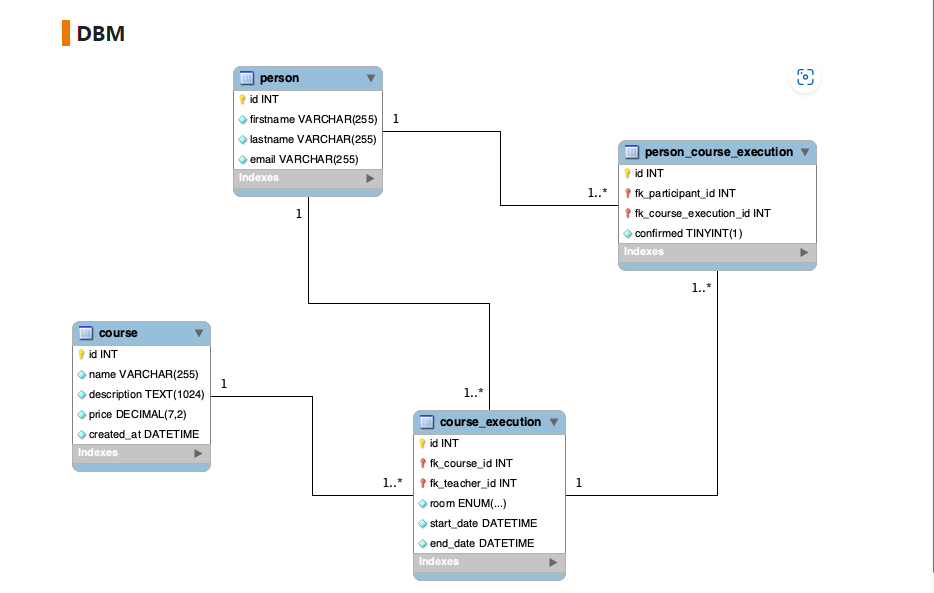
ERD (Entity-Relationship Diagram) und ERM (Entity-Relationship Model) sind Konzepte zur Datenmodellierung, die die Struktur von Datenbanken darstellen. ERD ist eine grafische Darstellung von Entitäten, Attributen und Beziehungen, während ERM eine abstraktere Beschreibung dieser Struktur ist.



## DBM

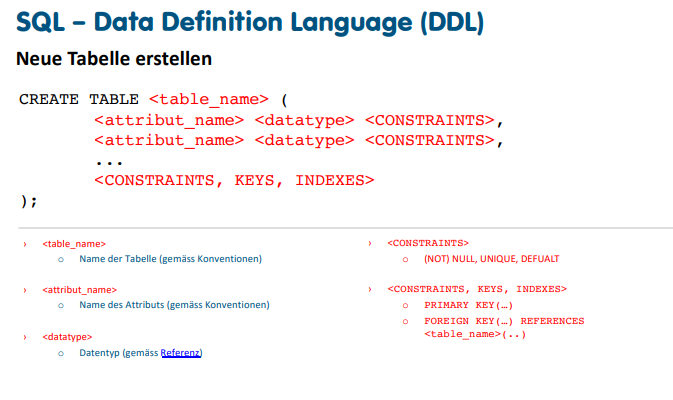
"Database Management"

Database Management umfasst die effiziente Organisation, Speicherung, Sicherung und Zugriffssteuerung von Datenbanken, einschließlich Strukturdesign, Schema-Definition, Abfrageverarbeitung und Leistungsoptimierung.

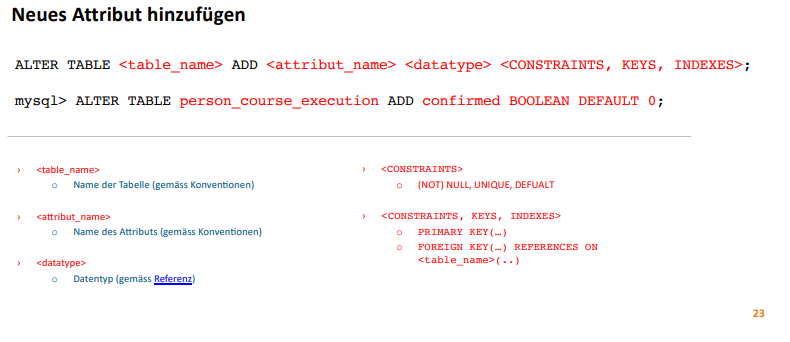
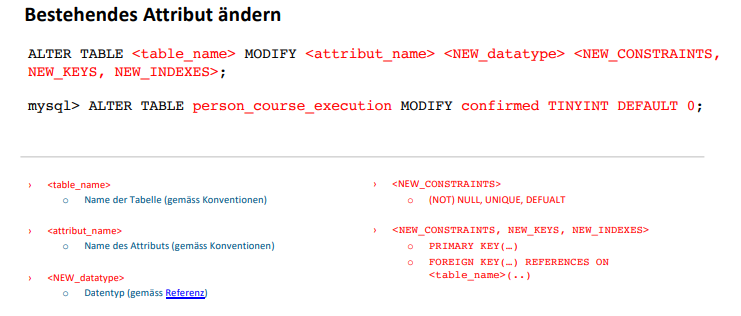


# **Commands in SQL**

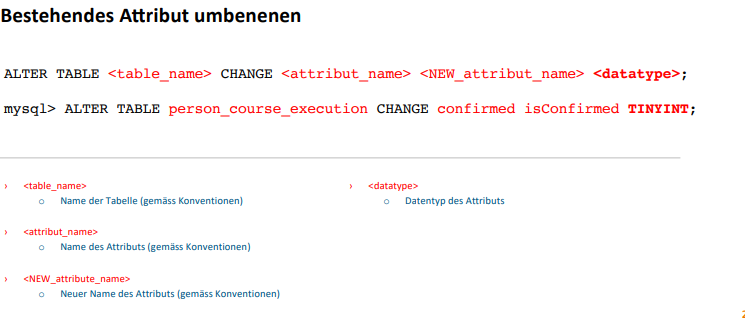
## Neue Tabelle erstellen - CREATE TABLE



## Neues Attribut hinzufügen – ALTER TABLE

Bestehendes Attribut ändern – MODIFY

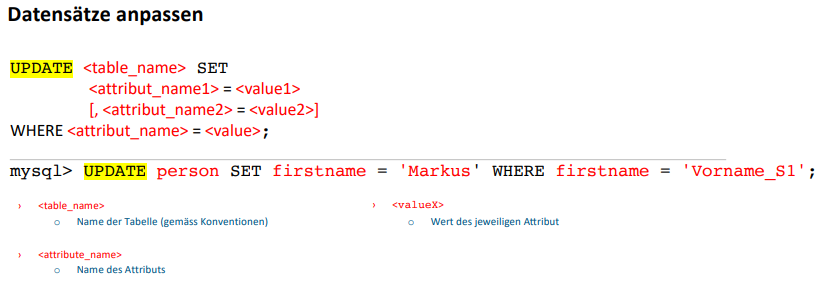
Bestehendes Attribut umbenennen – CHANGE



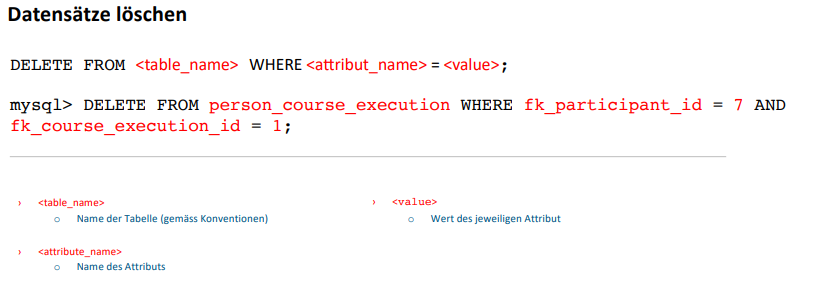
## Bestehendes Attribut löschen



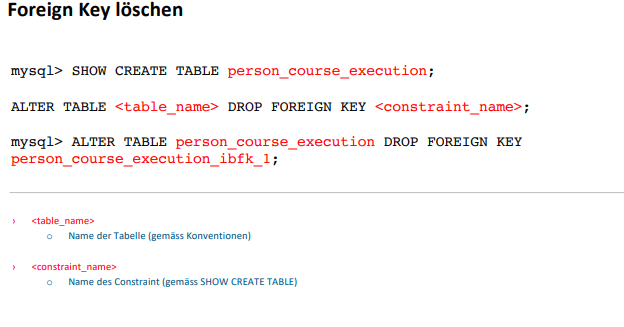
## Datensätze anpassen – UPDATE



## Datensätze löschen



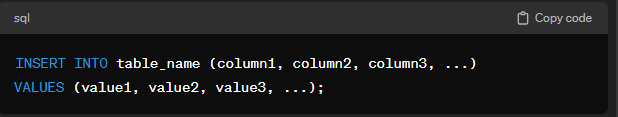
## Foreign Key löschen

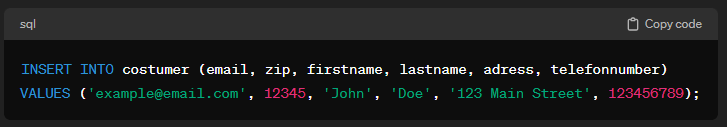


## Foreign Key ändern



## Daten in die Tabelle einfügen





## COMMIT

Speichert die Änderungen

* Beginnt Transaktion *IMMER*mit einem **BEGIN**; und endet mit einem **COMMIT**;

z.B: ***BEGIN*;**

**UPDATE employees SET salary = salary \* 1.1;**

***COMMIT*;**

## ROLLBACK

Verwirft die Änderungen

* Beginnt Transaktion *IMMER*mit einem **BEGIN**; und endet mit einem **ROLLBACK**;

z.B: ***BEGIN*;**

**UPDATE employees SET salary = salary \* 0.5;**

***ROLLBACK*;**

DROP DATABASE IF EXISTS carwebshop;

CREATE DATABASE carwebshop;

USE carwebshop;

CREATE TABLE costumer (

id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(255) NOT NULL,

zip MEDIUMINT(4) NOT NULL,

firstname VARCHAR(255) NOT NULL,

lastname VARCHAR(255) NOT NULL,

adress TEXT NOT NULL,

telefonnumber INT NOT NULL

);

CREATE TABLE productcategory (

id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

light VARCHAR(255) NOT NULL,

engine VARCHAR(255) NOT NULL,

wheel VARCHAR(255) NOT NULL,

suspension VARCHAR(255) NOT NULL,

brake VARCHAR(255) NOT NULL,

liquid VARCHAR(255) NOT NULL

);

CREATE TABLE product (

id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

fk\_orders\_id INT UNSIGNED NOT NULL,

fk\_productcategory\_id INT UNSIGNED NOT NULL,

description TEXT NOT NULL,

stock ENUM('in stock', 'out of stock'),

price DECIMAL(6,2),

FOREIGN KEY(fk\_orders\_id) REFERENCES orders(id),

FOREIGN KEY(fk\_productcategory\_id) REFERENCES productcategory(id)

);

CREATE TABLE order (

id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

fk\_costumer\_id INT UNSIGNED NOT NULL,

delivery\_date DATETIME NOT NULL DEFAULT NOW(),

quantity INT UNSIGNED NOT NULL,

orderdate DATETIME NOT NULL DEFAULT NOW(),

FOREIGN KEY(fk\_costumer\_id) REFERENCES costumer(id)

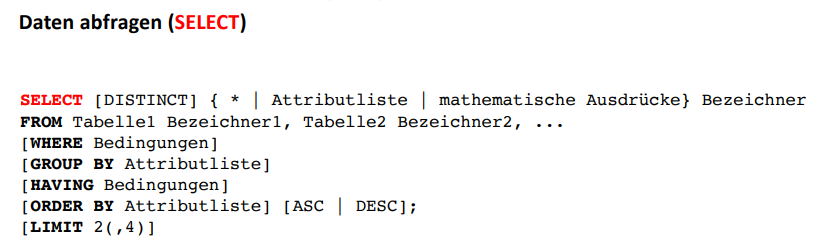
);

**Tabelle schule hat ein Attribut schule\_id**

**NOW() = jetzige default Zeit**

**SHOW CREATE TABLE ….. = zeigt den FORGEIN KEE an**

# Daten abfragen (SELECT)



SELECT: Verwendet, um Daten abzurufen.

* SELECT first\_name, last\_name, age FROM employees;

FROM: Gibt an, aus welcher Tabelle(n) Daten abgerufen werden sollen.

* SELECT \* FROM customers;
* SELECT

email,

SUBSTRING\_INDEX(email, '@', 1) AS Account,

SUBSTRING\_INDEX(email, '@', -1) AS Domain

FROM customer;

WHERE: Wird verwendet, um die abgerufenen Datensätze zu filtern.

* SELECT \* FROM products WHERE price > 50

GROUPBY: Wird verwendet, um Datensätze zu gruppieren, normalerweise mit Aggregatfunktionen. Also in Diesem Beispiel gruppiert die Mitarbeiter nach Abteilung und gibt nur die Abteilungen aus, in denen mehr als 5 Mitarbeiter arbeiten.

* SELECT department, COUNT(\*) AS Anzahl\_Mitarbeiter FROM employees GROUP BY department;

HAVING: Ähnlich wie WHERE, wird aber auf Gruppen von Datensätzen angewendet.

* SELECT department, COUNT(\*) AS Anzahl\_Mitarbeiter FROM employees GROUP BY department HAVING COUNT(\*) > 5;

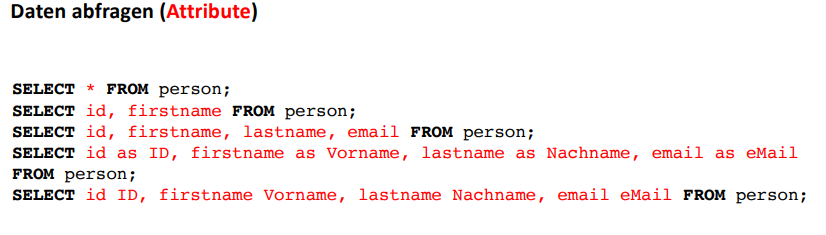
ORDER BY: Wird verwendet, um die Reihenfolge der Ergebnisse zu sortieren.

* SELECT \* FROM products ORDER BY price DESC;

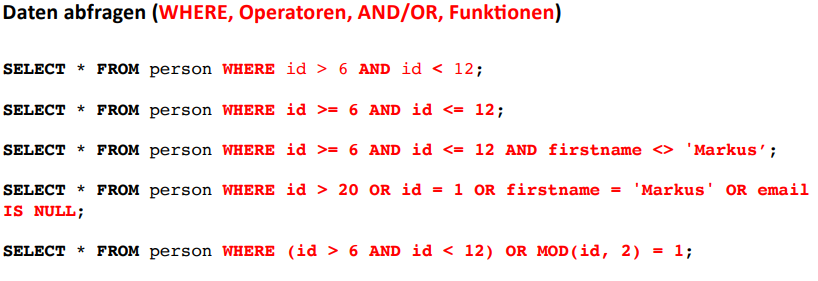
LIMIT: Wird verwendet, um die Anzahl der zurückgegebenen Datensätze zu

* SELECT \* FROM orders LIMIT 10;
* SELECT \* FROM orders LIMIT (3,10); *Es beginnt erst bei 3 zu zählen und dann von 3, 10 orders.*

# Daten abfragen (Attribute)



# Daten abfragen (WHERE, Operatoren, AND/OR, Funktionen)



**id > 6 AND id < 12;**

Wähle Datensätze mit einer ID zwischen 7 und 11 aus.

**id >= 6 AND id <= 12;**

Wähle Datensätze mit einer ID zwischen 6 und 12 aus, einschließlich 6 und 12.

**id >= 6 AND id <= 12 AND firstname <> 'Markus';**

Wähle Datensätze mit einer ID zwischen 6 und 12 aus, bei denen der Vorname nicht "Markus" ist.

**id > 20 OR id = 1 OR firstname = 'Markus' OR email IS NULL;**

Wähle Datensätze mit einer ID größer als 20 oder einer ID von 1 oder einem Vorname "Markus" oder einer leeren E-Mail-Adresse aus.

**(id > 6 AND id < 12) OR MOD(id, 2) = 1;**

Wähle Datensätze mit einer ID zwischen 7 und 11 oder Datensätze mit einer ungeraden ID aus.

**AUS DEN AUFGABEN:**

SELECT \* FROM product WHERE id = 5;

SELECT \*

FROM customer

WHERE id < 3;

SELECT \*

FROM customer

WHERE id < 3 OR id > 8;

SELECT \*

FROM order\_entry

WHERE id BETWEEN 3 AND 7;

SELECT \*

FROM customer

WHERE id IN (1, 3, 5, 6);

SELECT \*

FROM order\_entry

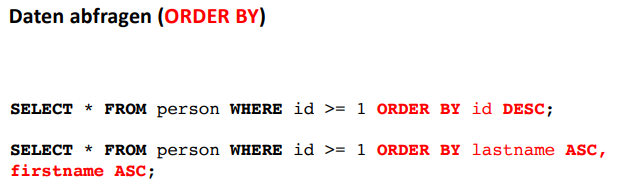
WHERE delivered\_at IS NULL;

SELECT \*

FROM product

WHERE price > (SELECT AVG(price) FROM product);

# Daten abfragen (ORDER BY)



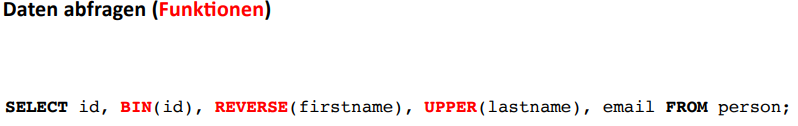
**SELECT \* FROM person WHERE id >= 1 ORDER BY id DESC;**

Wählt alle Datensätze aus der Tabelle "person" aus, bei denen die ID größer oder gleich 1 ist, und sortiert sie absteigend nach der ID.

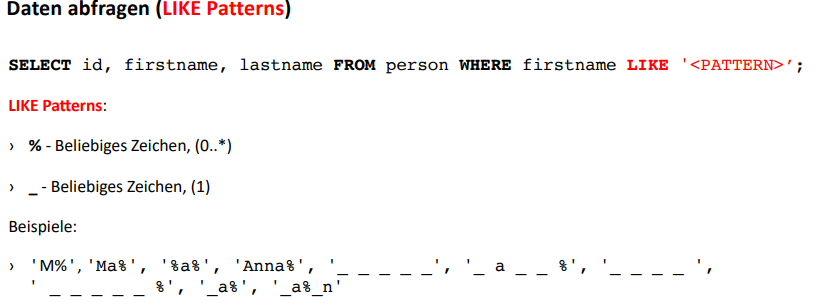
**SELECT \* FROM person WHERE id >= 1 ORDER BY lastname ASC, firstname ASC;**

Wählt alle Datensätze aus der Tabelle "person" aus, bei denen die ID größer oder gleich 1 ist, und sortiert sie aufsteigend nach dem Nachnamen und dann nach dem Vornamen.

# Daten abfragen (Funktionen)



# Daten abfragen (LIKE Patterns)



Die LIKE-Bedingung ermöglicht die Verwendung von Platzhaltern:

% = entspricht einer beliebigen Anzahl von Zeichen (auch keiner).

\_ = entspricht einem einzelnen Zeichen.

*Beispiele für Muster:*

**'M%'** - Alle Vornamen, die mit 'M' beginnen.

**'Ma%'** - Alle Vornamen, die mit 'Ma' beginnen.

**'%a%'** - Alle Vornamen, die den Buchstaben 'a' enthalten.

**'Anna%'** - Alle Vornamen, die mit 'Anna' beginnen.

**'\_\_\_\_\_ '** - Alle Vornamen, die genau fünf Zeichen lang sind.

**'\_ a \_ \_ %'** - Alle Vornamen, die mit einem 'a' als drittem Buchstaben beginnen und mindestens vier Buchstaben lang sind.

**'\_a%'** - Alle Vornamen, die mit 'a' als zweitem Buchstaben beginnen.

**'\_a%\_n'** - Alle Vornamen, die mit 'a' beginnen und mit 'n' enden.

# DATE\_ADD

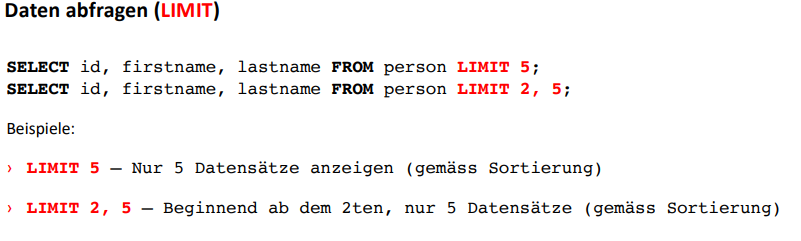
SELECT INTERVAL xDay + '2008-12-31';

# DATEDIFF

DATEDIFF(2008-12-31, 2007-12-31);

Gibt differenz aus

# Daten abfragen (LIMIT)



**LIMIT 5**: Zeigt nur die ersten 5 Datensätze aus der Tabelle "person" an, gemäß der aktuellen Sortierreihenfolge.

**LIMIT 2, 5**: Zeigt 5 Datensätze aus der Tabelle "person" an, beginnend ab dem zweiten Datensatz, gemäß der aktuellen Sortierreihenfolge.

# **Funktionen in MariaDB SQL**

wwww **= Muss durch Attributname ersetzt werden z.B *lastname*.**

Wwww **= Muss durch Tabellenname ersetz werden z.B *customer***

## DESC

Sortiert die Werte absteigend

**z.B: SELECT ordered\_at ORDER BY order\_entry DESC; /**

## ASC

Sortiert die Werte aufsteigend

# **Gruppenfunktionen**

* Gruppenfunktionen geben genau einen Wert zurück. Darum dürfen diese (für den Moment) nicht mit normalen Feldern vermischt werden, welche pro Datensatz einen Wert zurückgeben

## COUNT

Zeigt Anzahl der Datensätze an

**z.B: SELECT COUNT(id) FROM course; / SELECT COUNT(firstname) AS AJRI FROM customer WHERE firstname = 'John';**

## COUNT(\*)

Zählt die Anzahl der Zeilen in einer Tabelle.

**z.B: SELECT COUNT(\*) AS Anzahl\_Kunden FROM customers;**

## MIN

Das zeigt den minimalsten Wert an

**SELECT MIN(age) FROM students;**

## MAX

Das zeigt den maximalen Wert an

**SELECT MAX(salary) FROM employees;**

## AVG

Das berechnet den Durchschnitt der Werte

**SELECT AVG(age) FROM students;**

## SUM

Berechnet die Summe der Werte in einer Spalte

**SELECT SUM(salary) FROM employees;**

## Modulo Operator (%)

Der Modulo-Operator (%) in SQL gibt den Rest einer Division zweier Zahlen zurück.

*Modulo*: 'Zahl' % 'Zahl'

**ODER**

MOD(15, 4)

**SELECT 15 % 4 AS Rest\_der\_Division;**

## Runden

SELECT ROUND(price \* 0.08, 2) AS mwst

FROM product;

## Quadratwurzel

SELECT SQRT(price) AS PRICE\_SQRT FROM product;

## Initialen des Namens

SELECT

CONCAT(LEFT(firstname, 1), LEFT(lastname, 1)) AS Initialen

FROM customer;

## Datensatz erstellen:

Um einen Datensatz zu erstellen, verwenden Sie die INSERT INTO Anweisung, gefolgt vom Tabellennamen und den Werten, die Sie für die neuen Datensätze einfügen möchten.

**INSERT INTO employees (first\_name, last\_name, salary) VALUES ('John', 'Doe', 50000);**

## Datensatz löschen:

Um einen Datensatz zu löschen, verwenden Sie die DELETE FROM Anweisung, gefolgt vom Tabellennamen und einer Bedingung, die angibt, welche Datensätze gelöscht werden sollen.

**DELETE FROM employees WHERE employee\_id = 1001;**

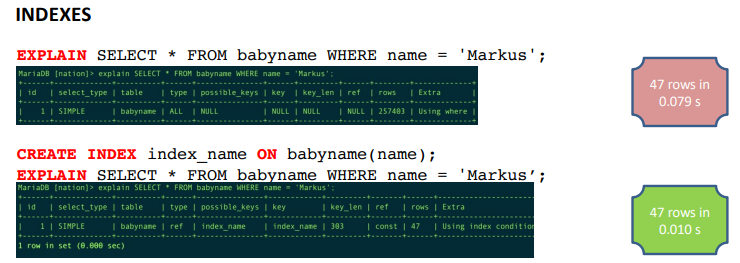
**WICHTIG**

**Z.b SELECT price + 5 from course;**

**Um den Preis aller Daten in der Tabelle zu ändern Bsp: update course set price = price + 5;**

# Indexes

Ein Index in MariaDB SQL beschleunigt den Zugriff auf Daten in einer Tabelle. Durch Erstellen eines Index auf einer häufig verwendeten Spalte können Abfragen schneller ausgeführt werden.



## -- Vor dem Index

EXPLAIN SELECT \* FROM babyname WHERE name = 'Markus';

## -- Index erstellen

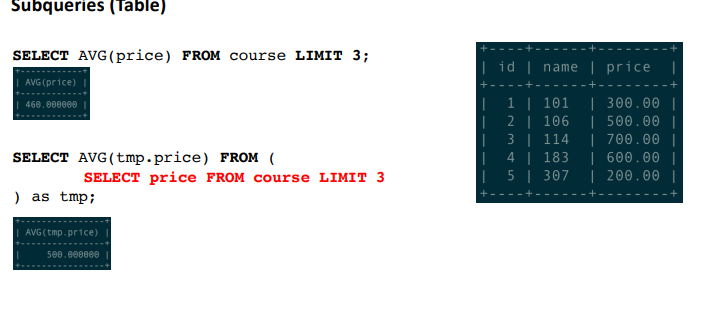
CREATE INDEX index\_name ON babyname(name);

## -- Nach dem Index

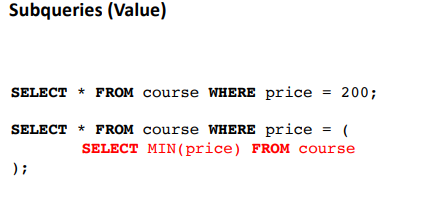
EXPLAIN SELECT \* FROM babyname WHERE name = 'Markus';

Das **EXPLAIN** zeigt die Ausführung der Abfrage an. Nach dem Index sollte die Abfrage schneller sein, da der Index verwendet wird, um die passenden Zeilen zu finden.

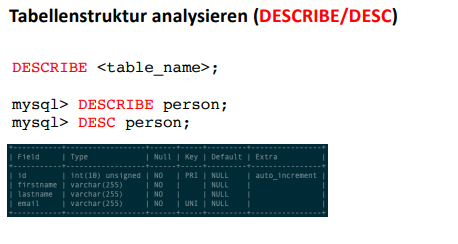
## Subqueries (Table)



## Subqueries (Value)



## Tabellenstruktur analysieren (DESCRIBE / DESC)



# **Aufgabe 5.2: SQL-Abfragen machen**

1. Anzahl Datensätze in der Tabelle payment

SELECT COUNT(\*) AS Anzahl\_Datensätze FROM payment;

1. Anzahl Schauspieler mit dem Namen Julia

SELECT COUNT(\*) AS Anzahl\_Julia\_Schauspieler FROM actor WHERE first\_name = 'Julia';

1. Anzahl inaktive Kunden

SELECT COUNT(\*) AS Anzahl\_inaktive\_Kunden FROM customer WHERE active = 0;

1. Durchschnittliche Länge der Filme mit Rating "PG"

SELECT AVG(length) AS Durchschnittliche\_Länge FROM film WHERE rating = 'PG';

1. Ausleihen, die noch nicht zurück gebracht wurden, sortiert nach Ausleihdatum (Format: dd.mm.YYYY)

SELECT rental\_date FROM rental WHERE return\_date IS NULL ORDER BY rental\_date;

1. Berechnen Sie die durchschnittliche Ausleihdauer in Tagen

SELECT AVG(DATEDIFF(return\_date, rental\_date)) AS Durchschnittliche\_Ausleihdauer FROM rental WHERE return\_date IS NOT NULL;

1. Liste der Vornamen von Schauspielern, deren Vorname nur 3 Buchstaben lang ist. Zeigen Sie keine doppelten Vornamen an

SELECT DISTINCT first\_name FROM actor WHERE LENGTH(first\_name) = 3;

1. Erstellen Sie eine Liste der Schauspieler, deren Nachname mit 'B' beginnt und an zweitletzter Stelle ein 'e' haben. Zeigen Sie keine doppelten Namen an

SELECT DISTINCT first\_name, last\_name FROM actor WHERE last\_name LIKE 'B%e\_';

1. Zählen Sie bei allen Datensätzen der Tabelle "rental" 12 Jahre zu Ausleih- und Rückgabedatum dazu

UPDATE rental SET rental\_date = DATE\_ADD(rental\_date, INTERVAL 12 YEAR), return\_date = DATE\_ADD(return\_date, INTERVAL 12 YEAR);

1. Erstellen Sie eine neue Kategorie namens "Art"

INSERT INTO category (name) VALUES ('Art');

# **Daten importieren**

Befüllen einer Datenbank mit Daten aus einem File (Zip, usw….):

mysql> LOAD DATA INFILE './baby-names.csv' INTO TABLE babyname FIELDS TERMINATED BY ',';

mysql> SELECT \* FROM babyname;

**ODER**

LOAD DATA INFILE '[Pfad zur Datei mit normalen /]'   
INTO TABLE tablename

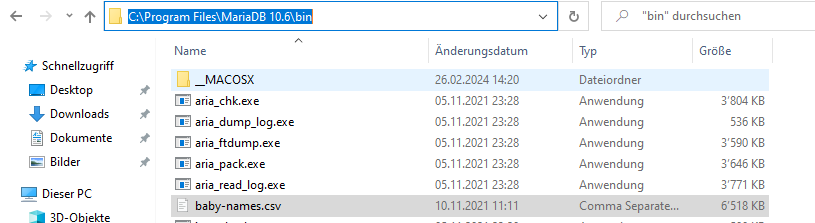
CHARACTER SET 'latin1'

FIELDS

TERMINATED BY ','

ENCLOSED BY '"' LINES

TERMINATED BY '\r\n';

* **BEI DATEIPFAD NUR DEN GEWÜNSCHTEN DATEINAME EINGEBEN UND AUSFÜHREN!! DANACH IM VERZEICHNIS VON DER COMMAND LINE DAS FILE SUCHEN (z.B BEI MARIA DB)**
* **DAS FILE IST IN "**C:\Program Files\MariaDB 10.6\data\sakila**" GESPEICHERT**
* **Wichtig: Das File nicht in "Users" machen, da es keine Berechtigung hat. Deshalb in einen Folder wie z.b das von MariaDB hineintun**

# **Daten exportieren**

SELECT  
  fieldname1, fieldname2, fieldname3, fieldname4, fieldname5  
FROM  
  tablename  
WHERE  
  fieldname6 = 'VALUE'  
INTO OUTFILE '[Pfad zur Datei mit normalen /]'  
FIELDS   
 TERMINATED BY ','  
 ENCLOSED BY '"'  
 LINES TERMINATED BY '\r\n';



# **DB Dumb – Export**

## Exportieren einer bestimmten Datenbank:

## mysqldump.exe -u [Benutzername] -p [Datenbankname] > [Zielordner]/[Dateiname].sql

**Datenbank sakila export command:** "C:\Program Files\MariaDB 10.6\bin\mysqldump.exe" -u root -p sakila > "C:/Program Files/MariaDB 10.6/data/sakila/sakila.sql"

**In TERMNAL machen und nicht in COMMANDLINE VON MARIADB**

**Und TERMINAL als ADMINISTRATOR ausführen**

## Exportieren aller Datenbanken:

* mysqldump.exe -u [Benutzername] -p --all-databases > [Zielordner]/[Dateiname].sql

Alles exportieren command, man muss noch **Dateinamen** und eventuell **Pfad** ändern: "C:\Program Files\MariaDB 10.6\bin\mysqldump.exe" -u root -p --all-databases > "C:/Program Files/MariaDB 10.6/data/alle\_datenbanken.sql"

# **DB Dumb – Import**

### Erstellen einer neuen Datenbank (optional):

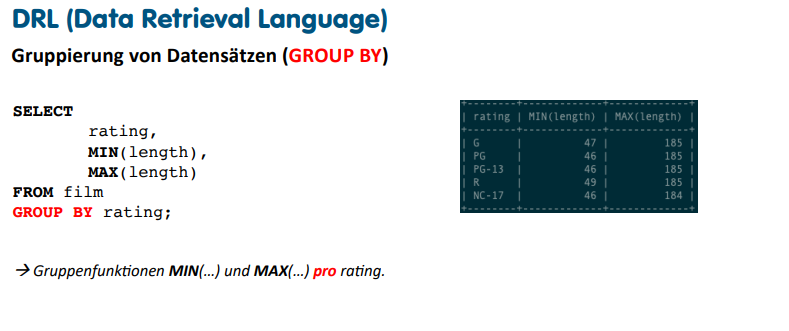
* mysql -u [Benutzername] -p -e "CREATE DATABASE [NeueDatenbank];

## Importieren eines Datenbankdumps:

* mysql.exe -u [Benutzername] -p [ZielDatenbank] < [Quellordner]/[Dateiname].sql

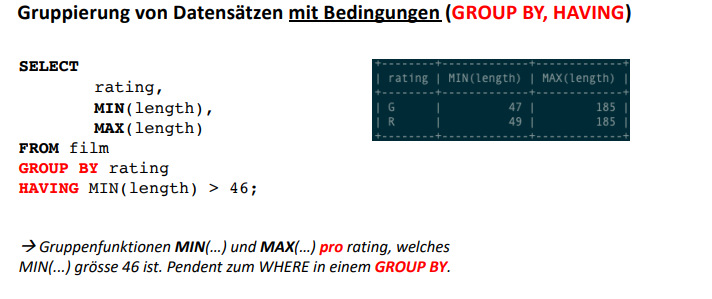
**Beispiel, jedoch muss Pfad der Datei und Datenbank geändert werden:** "C:\Program Files\MariaDB 10.6\bin\mysql.exe" -u root -p sakila\_copy < "C:/Program Files/MariaDB 10.6/data/alle\_datenbanken.sql"

# **Gruppierung von Datensätzen (GROUP BY)**

****

SELECT rating, MIN(length), MAX(length) FROM film GROUP BY rating;

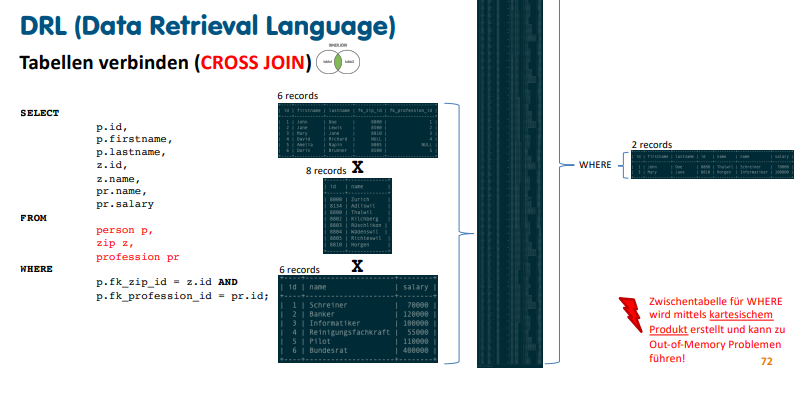
# **Grupperung von Datensätzen mit Bedingungen (GROUP BY, HAVING)**



SELECT rating, MIN(length), MAX(length) FROM film GROUP BY rating HAVING MIN(length) > 46

# **Tabelle verbinden (JOINs)**

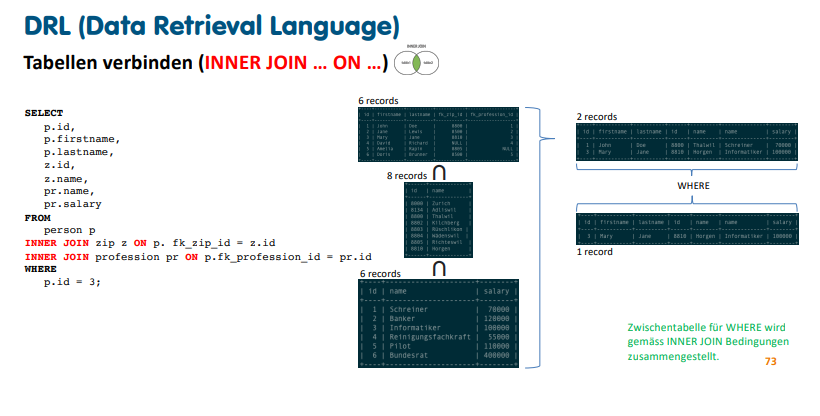
## CROSS JOIN



**SELECT p.id, p.firstname, p.lastname, z.id, z.name, pr.name, pr.salary FROM person p, zip z, profession pr WHERE p.fk\_zip\_id = z.id AND p.fk\_profession\_id = pr.id;**

## INNER JOIN … ON …

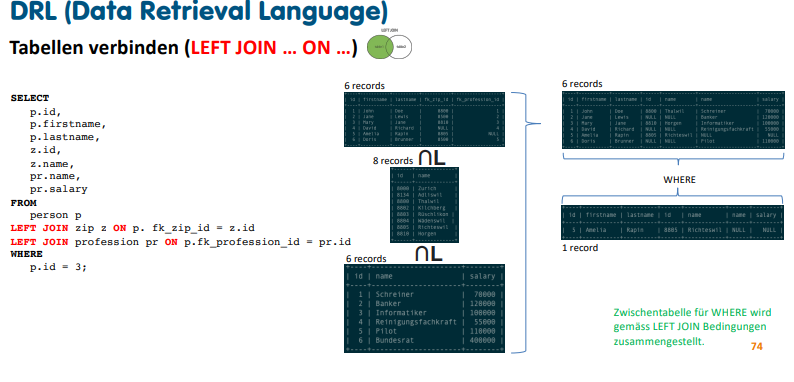
Nimmt immer die Tabellen als Ausgangslage, die in den ausgewählten Tabellen vorhanden sind



**SELECT p.id, p.firstname, p.lastname, z.id, z.name, pr.name, pr.salary FROM person p INNER JOIN zip z ON p. fk\_zip\_id = z.id INNER JOIN profession pr ON p.fk\_profession\_id = pr.id WHERE p.id = 3;**

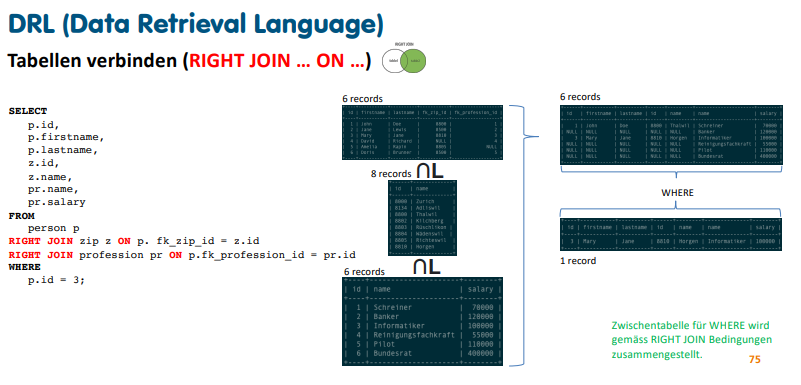
## LEFT JOIN … ON …

Nimmt immer die ganz **Linke** Tabelle als Ausgangslage

**SELECT p.id, p.firstname, p.lastname, z.id, z.name, pr.name, pr.salary FROM person p LEFT JOIN zip z ON p. fk\_zip\_id = z.id LEFT JOIN profession pr ON p.fk\_profession\_id = pr.id WHERE p.id = 3;**

## RIGHT JOIN … ON …

Nimmt immer die ganz **Rechten** Tabelle als Ausgangslage



**SELECT p.id, p.firstname, p.lastname, z.id, z.name, pr.name, pr.salary FROM person p RIGHT JOIN zip z ON p. fk\_zip\_id = z.id RIGHT JOIN profession pr ON p.fk\_profession\_id = pr.id WHERE p.id = 3;**

# **Benutzer und Zugriffe / Berechtigungen**

# Benutzer erstellen und Passwort setzten

# Benutzer umbenennen

# Berechtigungen zuweisen

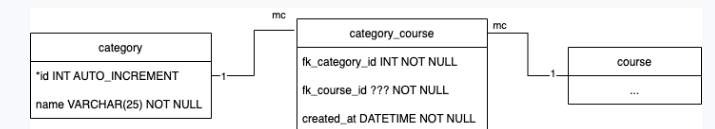
# Berechtigungen entziehen

# Berechtigungen anzeigen

# Benutzer löschen

# **Prüfungsvorbereitung für 28.02.2024**

1. Erweitern Sie die Datenbank um die Tabellen **category** und **category\_course** und stellen Sie Daten- sowie die referenzielle Integrität sicher.



CREATE TABLE category (

id INT AUTO\_INCREMENT

name VARCHAR(25) NOT NULL

PRIMARY KEY(id)

);

CREATE TABLE category\_course (

fk\_category\_id INT NOT NULL

fk\_course\_id INT NOT NULL

created\_at DATETIME NOT NULL

FOREIGN KEY(fk\_category\_id) REFERENCES category(id),

FOREIGN KEY(fk\_course\_id) REFERENCES course(id)

);

1. Ändern Sie die Tabelle **category\_course** so, dass **created\_at** jeweils immer den aktuellen Zeitpunkt speichert.

ALTER TABLE category\_course MODIFY created\_at NOT NULL DATETIME DEFAULT NOW();

1. Fügen Sie Daten in die **category** Tabelle ein, damit diese wie folgt aussehen und weisen Sie den Kurs 106 der "Datenbank" Kategorie zu:



INSERT INTO category (name)

VALUES ('Applikationsentwickler'),('Datanbanken'), ('MANAGEMENT'), ('Netzwerk');

INSERT INTO category\_course (fk\_category\_id, fk\_course\_id) VALUES (2,2);

1. Der Kursleiter **Patrick Wirz** hat geheiratet und seinen Nachnamen auf **Nuesch** geändert. Tragen Sie diese Änderung nach.

UPDATE person SET lastname = 'Nuesch' WHERE fistname = 'Patrick' lastname = 'Wirz';

1. Listen Sie alle Schüler mit Vornamen und Nachnamen sowie den dazugehörigen Kursnamen auf, welche sich für die Kursdurchführung im Kursraum Red angemeldet haben

SELECT p.firstname, p.lastname, c.name

FROM

person p

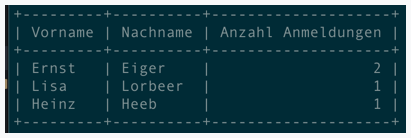
INNER JOIN person\_course\_execution pce ON p.id = pce.fk\_participant\_id

INNER JOIN course\_execution ce ON pce.fk\_course\_execution\_id = ce.id

INNER JOIN course c ON ce.fk\_course\_id = c.id

WHERE ce.room = 'Red';

1. Schreiben Sie den SQL Query, welchen folgende Ausgabe erzielt:



SELECT p.firstname as "Vorname", p.lastname as "Nachname", COUNT(pce.fk\_participant\_id) as "Anzahl Anmeldungen"

FROM

person p

INNER JOIN person\_course\_execution pce ON p.id = pce.fk\_participant\_id

GROUP BY p.id

HAVING COUNT(pce.fk\_participant\_id) <= 2

ORDER BY COUNT(pce.fk\_participant\_id) DESC;

1. Löschen Sie den Schüler **Heinz Heeb** aus der Datenbank

DELETE FROM person WHERE id = 14;

**Tabelle für ChatGPT**

+- n