Relationale Datenbanken und SQL

Ein paar wichtige Begriffe:

Relationale Datenbank: Eine strukturierte Sammlung von Daten, deren Struktur auf dem **Relationalen Datenmodell** beruht.

Datenmodell: Ein Paradigma, das der Struktur von großen Datensammlungen zugrunde liegt.

Gebräuchliche Datenmodelle:

- hierarchisch
- objektorientiert
- deduktiv
- relational
- netzwerkorientiert

Das Relationale Datenmodell beruht auf **relationaler Algebra**, einem Zweig der Mengenlehre.

Relationales Datenbank-Management-System (RDBMS): Oracle, MS SQL Server, IBM Db2 oder MySQL können mehrere relatioale Datenbanken verwalten.

Client-Server-Architektur: ist ein Prinzip, nach dem in einem Computer-Netzwerk Aufgaben zwischen den verbundenen Rechnern (hosts) verteilt werden. Ein Rechner (**host**) im Netzwerk stellt einen Dienst oder bestimmte Resourcen bereit (WWW, Email, Dateien etc.), und andere Rechner im Netzwerk können als Clients den Dienst in Anspruch nehmen. Der Client sendet einen **Request** (Anforderung), der Server eine **Response** (Antwort). Ein Server kann gleichzeitig eine große Anzahl von Requests bearbeiten, bzw. eine große Anzahl von Verbindungen verwalten.

(Auch das WWW ist eine Client-Server-Architektur, mit dem Unterschied, dass http ein statusloses Protokoll ist, d. h. Server und Client haben keine dauerhafte Verbindung; wenn die Response erledigt ist, dann „vergisst“ der Server den Client.)

Host: ein Rechner, der mit einem Rechnernetzwerk verbunden ist. Dabei ist egal, was der Rechner für eine Funktion innerhalb des Netzwerks hat.

localhost bzw. **127.0.0.1** (Ipv4) bzw. **::1** (Ipv6): Die Loopback-Adresse für jeden Rechner. Die Datenpakete, die ein Rechner sendet, werden an denselben Rechner addressiert. (127.0.0.1 ist nicht die einzige Loopback-Adresse, aber die gebräuchlichste)

Im Falle von MySQL (und anderen Datenbankservern) kann der Server jeden Request einem bestimmten Client zuordnen. Jeder Verbindung ist eine eindeutige Id zugeordnet, die für die ganze Dauer der Session gilt.

Bei einer MySQL-Installation ist **mysqld.exe** der Datenbankserver und **mysql.exe** (der MySQL-Monitor) der Standard-Client. Wenn man den MySQL-Monitor, zeigt er in der Begrüßung den Hinweis auf die Verbindungs-Id an:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
```

Erst, wenn die Session mit **quit** am MySQL-Prompt beendet wird, wird die Session zerstört:

```
mysql> quit
Bye
```

MySQL als Beispiel für relationale Datenbank-Management-Systeme

MySQL ist seit über zwanzig Jahren ein sehr populäres RDBMS. das ursprünglich einmal als Open-Source-Projekt begann. Inzwischen gibt es eine kommerzielle Lizenz, aber in der Version Community Server ist es weiterhin mit einer Public License verfügbar.
Wenn man MySQL installiert hat, dann heißt das Installationsverzeichnis **mysql**. (Bei einer Installation von xampp ist das unter Windows der Pfad **X:\xampp\mysql** (wobei X für einen Laufwerksbuchstaben steht). Im Installationsverzeichnis gibt es einen Unterordner **bin**. Darin befinden sich eine ganze Reihe ausführbarer Dateien, unter anderem eine Datei **mysqld.exe** und eine Datei **mysql.exe**.

mysqld.exe ist der Datenbankserver.

mysql.exe ist der Standard-Client für den MySQL-Server. (mysql-monitor)

Die meisten anderen Programme (z. B. **mysqladmin.exe**, **mysqldump.exe**) sind ebenfalls Clientprogramme für ganz bestimmte Zwecke.

An sich startet man das Programm mit der Eingabe

mysql

am Prompt, z. B. am Prompt der Windows-Eingabeaufforderung:

C:\Users\USER>

oder am Prompt der XAMPP-Shell:

#

da noch kein Nutzer außer **root** existiert, kann man sich auch nur als **root** anmelden:

C:\Users\USER> mysql -u root

wenn alles geklappt hat, sieht man eine Begrüßung und dann den MySQL-Prompt:

```
MariaDB [(none)]>
```

Der MySQL-Prompt kann je nach Version oder je nach Plattform untershiedlich aussehen. In diesem Fall kann man erkennen, dass es sich um den Community-Server handelt (**`MariaDB`**), und dass noch keine Datenbank ausgewählt wurde (**`none`**)

Beenden kann man den MySQL-Monitor mit dem Befehl **`quit`**:

```
MariaDB [(none)]> quit
```

Wenn man als root eingeloggt ist, kann man mit den bereits vorhandenen Datenbanken arbeiten oder neue Datenbanken anlegen.

Andere häufig eingesetzte Clients für MySQL sind: MySQL Workbench (ein grafischer Client) oder phpMyAdmin (ein PHP-Skript und ein Web-Client)

SQL

Die Verkehrssprache von MySQL und anderen RDBMSs ist **S**tructured **Q**uery **L**anguage (SQL)

Mit SQL kann man drei große Aufgabenkomplexe bearbeiten:

- **Datendefinition** (Datenbanken anlegen, Tabellenstruktur definieren, Tabellenstruktur ändern)
- **Datenmanipulation** (Datensätze einfügen, Datensätze löschen, Werte in Datensätzen aktualisieren)
- **Administration** (Nutzer anlegen und löschen, Nutzerrechte vergeben und widerrufen, Speicherengine, Kodierung und Sortierung festlegen)

SQL-Statements (Befehle, Anweisungen) sind eine Kombination von SQL-Keywords (**`CREATE`**, **`WHERE`**, **`HAVING`** etc.), Operatoren (**`=`**, **`&&`**, **`||`**, **`LIKE`** etc.), Funktionsnamen (**`CURRENT_DATE()`**, **`MAX()`**, **`AVG()`** etc.) und eigenen Bezeichnern.

Bezeichner sind Namen für Tabellen, Datenbanken, Spalten, Prozeduren.

Für Bezeichner gibt es einige Regeln:

Bezeichner ohne Begrenzer (Backticks) können bestehen aus:

- a – z und A – Z
- $-Zeichen und _ (Unterstrich)
- die Ziffern 0 - 9
- Zeichen des erweiterten Unicode-Zeichensatzes von U+80 (dezimal 128) aufwärts

Bezeichner können mit jedem legalen Zeichen beginnen. Ein Bezeichner kann aber nicht ausschließlich aus Ziffern bestehen, da er sonst nicht von einer Zahl zu unterscheiden ist. Problematisch sind hier auch

„e“ und „E“, da beide in Zahlen in wissenschaftlicher Schreibweise vorkommen können.

Bezeichner mit Begrenzern (Backticks) können alle Unicode-Zeichen außer NUL und Zeichen von U+10000 aufwärts enthalten.

Als Begrenzer für Bezeichner dürfen nur Backticks verwendet werden, nicht einfache oder doppelte Anführungszeichen. Anführungszeichen sind nur als Begrenzer für Zeichenketten (Strings) erlaubt.

MySQL ist, was Schlüsselwörter und Bezeichner betrifft, case-insensitive. Zu bedenken ist, dass Datenbanken als Dateiordner im Dateisystem abgebildet werden und Tabellen als Dateien. Windows ist da case-insensitive, somit ist das unproblematisch, aber Unix/Linux ist case-sensitive.

Per Konvention schreibt man SQL-Schlüsselworte durchgängig groß und Bezeichner durchgängig klein. Wenn ein Bezeichner aus mehreren Worten zusammenesetzt ist, fügt man vor jedem neuen Wortstamm einen Unterstrich ein:

```
umsatzsteuer_satz
```

```
umsatzsteuer_betrag
```

Beispiele für gültige Bezeichner ohne Begrenzer:

```
i18n
```

```
first_name
```

```
_id
```

ungültig:

```
first-name
```

Beispiele für gültige Bezeichner mit Begrenzer:

```
`letzter zugriff`
```

```
`überhang`
```

```
`first-name`
```

Datendefinition

Um eine Datenbank anzulegen benutzt man (im einfachsten Fall) ein Statement nach dem Muster

```
CREATE DATABASE <bezeichner>;
```

Beispiel:

```
MariaDB [(none)]> CREATE DATABASE kunden;
```

Mit

```
SHOW DATABASES;
```

kann man die vorhandenen Datanbanken anzeigen lassen.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ajax               |
| dilettante         |
| krunden            |
| lilypond           |
| mysql              |
| ocp                |
| performance_schema |
| phpmyadmin         |
| sakila             |
| test               |
| wordpress          |
+--------------------+
11 rows in set (0.02 sec)
```

Je nach Version sind mehrere Datenbanken bereits nach der Installation vorhanden. Eine Datenbank **`mysql`** ist immer vorhanden, ebenso eine Datenbank **`test`**.

Mit

```
USE <datenbankname>;
```

kann man eine bestimmte Datenbank auswählen.

Jede Datenbank wird unter Windows im Dateisystem im Pfad **`X:\xampp\mysql\data`** (wenn es sich um eine XAMPP-Installation handelt) oder im Pfad **`X:\Program Files (x86)\mysql\data`** (wenn MySQL alleine installiert wurde) als Dateiordner abgebildet. **`X`** ist dabei ein Laufwerksbuchstabe.

Die Datenbank **`mysql`** verwaltet Nutzerrechte für den betreffenden MySQL-Server. Normalerweise existiert ein Nutzer root,manchmal auch ein anonymer Nutzer. In der Tabelle user in mysql

```
MariaDB [mysql]> SELECT user, host, password FROM user;
+------+-----------+----------+
| User | Host      | Password |
+------+-----------+----------+
| root | localhost |          |
| root | 127.0.0.1 |          |
| pma  | localhost |          |
+------+-----------+----------+
4 rows in set (0.007 sec)
```

Löschen einer Datenbank:

```
MariaDB [(none)]> DROP DATABASE my_test;
```

Ein SQL-Statement wird mit einem Semikolon beendet:

```
MariaDB [(none)]> CREATE
    -> DATABASE
    -> test_20200302
    -> ;
Query OK, 1 row affected (0.001 sec)
```

Man kann mehrere Statements in eine Zeile schreiben, man kann aber auch ein-und dasselbe Statement auf mehrere Zeilen verteilen.

Die Datenbank selbst speichert keine Daten. Das übernehmen Strukturen, die man **Tabelle** oder **Relation** nennt. Eine Tabelle, bzw. eine Relation ist definiert durch Ihre **Attribute**. Eine Tabelle **`kontakt`** könnte demnach die Attribute **`vorname`**, **`nachname`** und **`email`** haben.

Zur Definition eines Attributs (einer Spalte) sind mindestens ein Bezeichner und der Datentyp nötig. Andere Egenschaften der Spalte wie **`NOT NULL`** oder **`UNIQUE`** können hinzugefügt werden.

Eine Zeile bezeihnet man auch als **Datensatz** (record), bzw. **Tupel**.

Die Attribute sind in der Tabelle die Spaltennamen und in den Zeilen bezeichnet man eine Spalte als **Feld**.

In der ersten Zeile hätte demnach beispelsweise das Feld **`vorname`** den wert „Paula“ und das Feld email den Wert „paula@paulsen.de“, in der zweiten Zeile den Wert „Hans“ und den Wert „hans@hansen.de“.

So gesehen repräsentiert eine Tabelle eine **Entity/Entität** (ein eindeutig bestimmbares Objekt) und ein Datensatz eine **Instanz** bzw. eine **Ausprägung** dieser Entity.

Um Daten in einer Tabelle zu speichern, muss die Tabelle erst einmal definiert werden.

Dazu gehört die Angabe wie die Tabelle heißen soll, wie jede Spalte heißen soll, aber auch was jede Spalte für ein Datentyp sein soll.

MySQL kennt verschiedene Gruppen von Datentypen:

- Numerische Datentypen
- Zeichenketten-Datentypen
- Datums- und Uhrzeit-Typen
- Ein Datentyp für Geometrische Objekte
- JSON (**J**ava**S**cript **O**bject **N**otation, eine Datenstruktur, die an JavaScript-Objekte angelehnt ist)

Zeichenkettendatenypen

- **CHAR** (Abkürzung von "character")
- **VARCHAR** (zusammengezogen aus "variable character")
- **TEXT**
- **TINYTEXT**
- **MEDIUMTEXT**
- **LONGTEXT**

CHAR bis **LONGTEXT** sind bei Zeichenkettenvergleichen case-insensitive. MySQL-Zeichenkettenvergleiche sind generell case-insensitive:

```
mysql> select "Hugo" = "hugo";
+-----------------+
| "Hugo" = "hugo" |
+-----------------+
|               1 |
+-----------------+
1 row in set (0.00 sec)
```

Andere Zeichenkettentypen werden wie Binärstrings bzw. Character-Sequenzen behandelt:

- **BINARY**
- **VARBINARY**
- **BLOB** (Binary Large OBject)
- **TINYBLOB**
- **MEDIUMBLOB**
- **LONGBLOB**

Daraus ergibt sich, dass hier Charactersets und Kollation keine Rolle spielen und beim Vergleich von zwei Zeichenketten Groß- und Kleinschreibung unterschieden wird.

SET und **ENUM** sind Zeichenkettentypen, die als Wert einen aus einer vordefinierten Liste (**ENUM**) oder mehrere aus einer vordefinierten Liste (**SET**) erlauben.

Eine mögliche Tabellendefinition mit zwei Spalten:

```
MariaDB [string_test]> CREATE TABLE names (
    -> vorname CHAR(50),
    -> nachname VARCHAR(1000));
```

```
Query OK, 0 rows affected (0.021 sec)
```

Wenn man eine Tabelle hat, kann man sich die Struktur mit
`EXPLAIN <tabellenname>`
anzeigen lassen.
und man kann nun Datensätze eingeben:

```
MariaDB [string_test]> INSERT INTO names (vorname, nachname) VALUES
    -> ("hans", 'Hansen');
```

Query OK, 1 row affected (0.004 sec)

```
MariaDB [string_test]> EXPLAIN names;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| vorname  | char(50)      | YES  |     | NULL    |       |
| nachname | varchar(1000) | YES  |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
2 rows in set (0.003 sec)

MariaDB [string_test]> SELECT * FROM names;
+---------+----------+
| vorname | nachname |
+---------+----------+
| hans    | Hansen   |
| Paula   | Paulsen  |
+---------+----------+
2 rows in set (0.001 sec)
```

enum und set

```
MariaDB [string_test]> CREATE TABLE pizza (
    -> size ENUM('normal', 'medium', 'large'),
    -> topping SET('mozzarella', 'tomate', 'salami', 'schinken',
'oliven'));
Query OK, 0 rows affected (0.015 sec)
```

MariaDB [mysql]> USE new_database
Database changed
MariaDB [new_database]> CREATE TABLE new_table (
-> column_1 VARCHAR(255),
-> column_2 CHAR(100),
-> column_3 SET('Englisch', "Französisch", "Arabisch", "Spanisch", 'Italienisch'),
-> column_4 ENUM('Grundkenntnisse', 'fließend', 'verhandlungssicher'));
tabellenstruktur ändern:
MariaDB [new_database]> ALTER TABLE new_table ADD COLUMN
-> column_5 INT;

Query OK, 0 rows affected (0.014 sec)

```
MariaDB [string_test]> CREATE TABLE numeric_test (
    -> anzahl INTEGER,
    -> faktor INTEGER UNSIGNED,
    -> preis DECIMAL(5,2) UNSIGNED,
    -> ratio FLOAT);
Query OK, 0 rows affected (0.016 sec)

MariaDB [string_test]> INSERT INTO numeric_test VALUES
    -> (-4, 19, 34.56, 3.14);
Query OK, 1 row affected (0.006 sec)
```

```
MariaDB [new_database]> INSERT INTO new_table VALUES
  -> ("Hänsel und Gretel", "Hexenhaus", "Französisch,Italienisch",
  -> "fließend", 3, 456.34, 345.43958);
Query OK, 1 row affected (0.005 sec)

MariaDB [new_database]> INSERT INTO new_table (column_2,column_6)
  -> VALUES
  -> ("Murder, My Sweet", 4.67);
Query OK, 1 row affected (0.005 sec)
```

Datum und Uhrzeit

```
MariaDB [new_database]> CREATE TABLE date_time_table (
  -> date_col DATE,
  -> time_col TIME,
  -> date_time_col DATETIME,
  -> timestamp_col TIMESTAMP);
Query OK, 0 rows affected (0.022 sec)
EXPLAIN date_time_table;

MariaDB [new_database]> SELECT * FROM date_time_col;
ERROR 1146 (42S02): Table 'new_database.date_time_col' doesn't exist
MariaDB [new_database]> SELECT * FROM date_time_table;
+------------+----------+---------------+--------------------+
| date_col   | time_col | date_time_col | timestamp_col      |
+------------+----------+---------------+--------------------+
| 2020-03-03 | NULL     | NULL          | 2020-03-03 17:22:43 |
+------------+----------+---------------+--------------------+
1 row in set (0.000 sec)

MariaDB [new_database]> UPDATE date_time_table
```

```
   -> SET date_col = "2020-03-04";
Query OK, 1 row affected (0.004 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [new_database]> SELECT * FROM date_time_table;
+------------+----------+--------------+--------------------+
| date_col   | time_col | date_time_col | timestamp_col      |
+------------+----------+--------------+--------------------+
| 2020-03-04 | NULL     | NULL         | 2020-03-03 17:25:25 |
+------------+----------+--------------+--------------------+

MariaDB [new_database]> CREATE TABLE test_table (
   -> text_col VARCHAR(100),
   -> int_col INTEGER);
Query OK, 0 rows affected (0.022 sec)

MariaDB [new_database]> ALTER TABLE test_table CHANGE
   -> int_col integer_col INTEGER;
Query OK, 0 rows affected (0.012 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Where-Klausel und relationale Operatoren
Where-Klausel
Ausdrücke, die true oder false zurückgeben

MySQL im Batch-Modus

```
USER@DESKTOP-V8C2S2E c:\xampp

# mysql -u root < c:\xampp\sample-dbs\sakila-schema.sql
```

Normalisierung

Normalisierung hat den Zweck bestimmte Fehlerquellen zu vermeiden.
Mögliche Fehlerquellen nennt man Anomalien. Z. B.:
UPDATE-Anomalie oder DELETE-Anomalie (Wenn Daten redundant sind, dann müssen mehrere Datensätze bearbeitet werden anstatt nur einem bei einer normalisierten Tabelle.
Normalisierung erfolgt in ganz bestimmten Schritten und jeder Schritt hat ganz bestimmte Regeln.

1. Normalform
2. Normalform
3. Normalform
   Boyce-Codd-Normalform
4. Normalform
5. Normalform

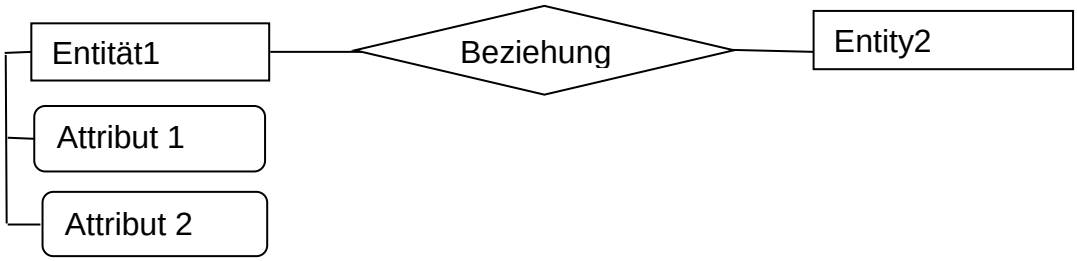
Die erste Normalform ist erfüllt, wenn die Werte der Felder der Datensätze **atomar** sind und wenn es **keine Wiederholungsspalten** gibt.
So gesehen ist die Folgende Tabelle problematisch: Die Spalten Lieferadresse, Rechnungsanschrift und Kontakt enthalten nicht-atomare Werte und Lieferadresse und Rechnungsanschrift sind Wiederholungsspalten.

Kundennummer	Kunde	Lieferadresse	Rechnungsanschrift	Kontakt
324098	Firma Meier	Bartnigalle 50, 10997 Berlin	Lausitzer Straße 30, 10965 Berlin	Lieferadresse: Müller, Rechnung: Meier

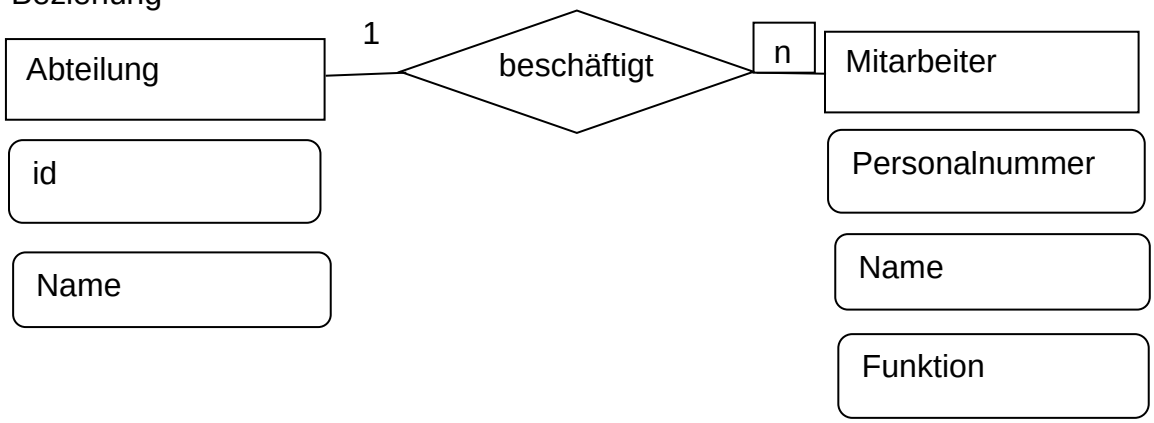
Kundennummer	Kunde	Straße	Hausnumer	Plz	Ort	Kontakt	
324098	Firma Meier	Bartningallee	50	10997	Berlin	Müller	Lieferanschrift
324098	Firma Meier	Lausitzer Straße	30	10965	Berlin	Meier	Rechnungsanschrift

Für die zweite Normalform müssen alle Nicht-Schlüssel-Attribute einer Tabelle vom gleichen Schlüssel abhängen.
Demnach ist die folgende Tabelle problematisch,

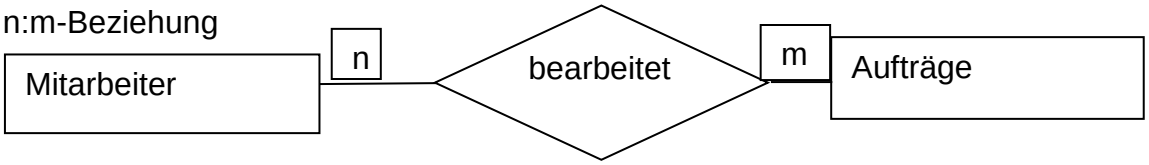
Kundennummer	Kunde	Rechnungsnummer	Rechnungsdatum	Betrag	Kontakt
324098	Firma Meier	3458320	20.3.2019	1500.-	Müller
324098	Firma Meier	3458321	30.4.2019	2000.-	Müller



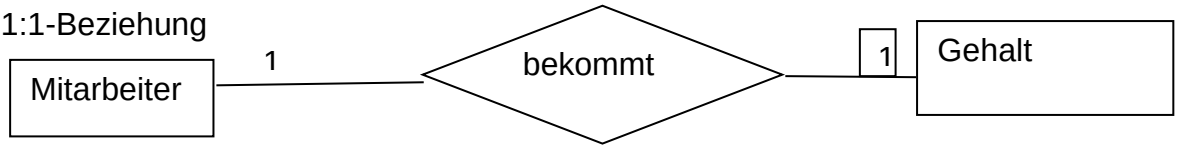
1:n-Beziehung



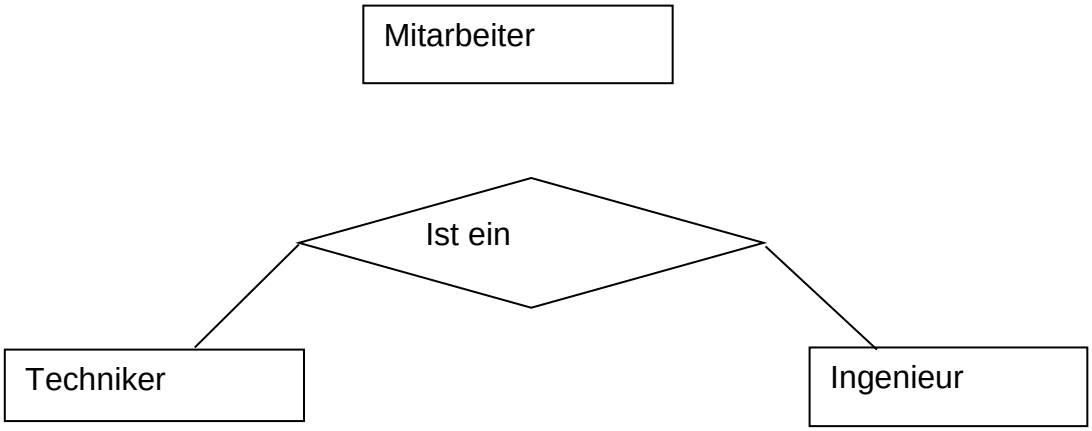
n:m-Beziehung



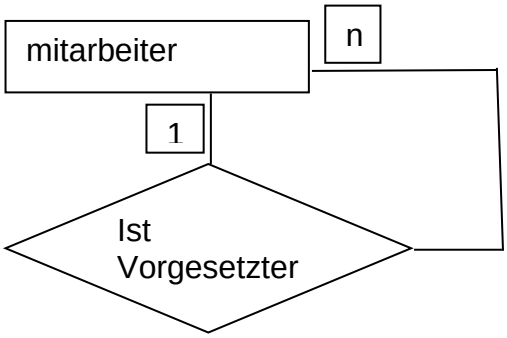
1:1-Beziehung



Is-A-Beziehung



Rekursive Beziehung



USER@DESKTOP-V8C2S2E c:\xampp

mysql -u root < c:\xampp\sample-dbs\sakila-data.sql

Backup mit mysqldump

USER@DESKTOP-V8C2S2E c:\xampp

mysqldump -u root --opt --all-databases > c:\xampp\sample-dbs\dump.dmp

Indizes

MariaDB [kontakte]> CREATE TABLE index_test (

-> vorname VARCHAR(100),

-> nachname VARCHAR(100),

-> email VARCHAR(100),

-> INDEX(nachname(10)));

Query OK, 0 rows affected (0.021 sec)

```
MariaDB [kontakte]> explain index_test;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| vorname  | varchar(100) | YES  |     | NULL    |       |
| nachname | varchar(100) | YES  | MUL | NULL    |       |
| email    | varchar(100) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.007 sec)
```

MariaDB [kontakte]> CREATE TABLE unique_test (

-> vorname VARCHAR(100),

-> nachname VARCHAR(100) UNIQUE,

-> email VARCHAR(100));

Query OK, 0 rows affected (0.019 sec)

```
MariaDB [kontakte]> explain unique_test;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| vorname  | varchar(100) | YES  |     | NULL    |       |
| nachname | varchar(100) | YES  | UNI | NULL    |       |
| email    | varchar(100) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.008 sec)
```

```
MariaDB [kontakte]> INSERT INTO unique_test VALUES
   -> ("Peter", "Petersen", "peter@pertersen.de");
Query OK, 1 row affected (0.002 sec)

MariaDB [kontakte]> INSERT INTO unique_test VALUES
   -> ("Paula", "Petersen", "paula@petersen.de");
ERROR 1062 (23000): Duplicate entry 'Petersen' for key 'nachname'
MariaDB [kontakte]> select * from unique_test;
+---------+----------+-------------------+
| vorname | nachname | email             |
+---------+----------+-------------------+
| Peter   | Petersen | peter@pertersen.de |
+---------+----------+-------------------+
1 row in set (0.001 sec)
```

```
MariaDB [kontakte]> CREATE TABLE primary_key_test (
    -> id INTEGER AUTO_INCREMENT PRIMARY KEY,
    -> vorname VARCHAR(100),
    -> nachname VARCHAR(100),
    -> email VARCHAR(100));
Query OK, 0 rows affected (0.026 sec)

MariaDB [kontakte]> INSERT INTO primary_key_test
    -> (vorname, nachname, email)
    -> VALUES
    -> ("Ludwig", "Zankl", "ludwig@zankl.de");
Query OK, 1 row affected (0.007 sec)

MariaDB [kontakte]> INSERT INTO primary_key_test
    -> VALUES
    -> (0,"Elfriede", "Meier", "elfriede@meier.de");
Query OK, 1 row affected (0.005 sec)

MariaDB [kontakte]> SELECT * FROM primary_key_test;
+----+----------+----------+-------------------+
| id | vorname  | nachname | email             |
+----+----------+----------+-------------------+
|  1 | Ludwig   | Zankl    | ludwig@zankl.de   |
|  2 | Elfriede | Meier    | elfriede@meier.de |
+----+----------+----------+-------------------+
2 rows in set (0.000 sec)
```

JOIN

Setting environment for using XAMPP for Windows.

USER@DESKTOP-V8C2S2E c:\xampp

mysql -u root

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MariaDB connection id is 8

Server version: 10.4.11-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
MariaDB [(none)]> CREATE DATABASE normalization;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> CREATE TABLE person (
    -> name VARCHAR(100),
    -> tel VARCHAR(100),
    -> tel2 VARCHAR(100));
ERROR 1046 (3D000): No database selected
MariaDB [(none)]> use normalization
Database changed
MariaDB [normalization]> CREATE TABLE person (
    -> name VARCHAR(100),
```

```
   -> tel VARCHAR(100),
   -> tel2 VARCHAR(100));
Query OK, 0 rows affected (0.030 sec)

MariaDB [normalization]> ALTER TABLE ADD vorname VARCHAR(100);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near 'ADD vorname VARCHAR(100)' at line 1
MariaDB [normalization]> ALTER TABLE normalization ADD vorname VARCHAR(100);
ERROR 1146 (42S02): Table 'normalization.normalization' doesn't exist
MariaDB [normalization]> ALTER TABLE person ADD vorname VARCHAR(100);
Query OK, 0 rows affected (0.015 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> explain person;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| name    | varchar(100) | YES  |     | NULL    |       |
| tel     | varchar(100) | YES  |     | NULL    |       |
| tel2    | varchar(100) | YES  |     | NULL    |       |
| vorname | varchar(100) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
4 rows in set (0.010 sec)

MariaDB [normalization]> ALTER TABLE person DROP vorname;
Query OK, 0 rows affected (0.011 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> explain person;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| name  | varchar(100) | YES  |     | NULL    |       |
| tel   | varchar(100) | YES  |     | NULL    |       |
| tel2  | varchar(100) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
3 rows in set (0.003 sec)

MariaDB [normalization]> ALTER TABLE person ADD vorname VARCHAR(100)
    -> AFTER name;
Query OK, 0 rows affected (0.013 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> explain person;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| name    | varchar(100) | YES  |     | NULL    |       |
| vorname | varchar(100) | YES  |     | NULL    |       |
| tel     | varchar(100) | YES  |     | NULL    |       |
| tel2    | varchar(100) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

```
4 rows in set (0.003 sec)

MariaDB [normalization]> ALTER TABLE person DROP tel2;
Query OK, 0 rows affected (0.010 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> explain person;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| name    | varchar(100) | YES  |     | NULL    |       |
| vorname | varchar(100) | YES  |     | NULL    |       |
| tel     | varchar(100) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
3 rows in set (0.003 sec)

MariaDB [normalization]> ALTER TABLE person ADD
    -> id INTEGER AUTO_INCREMENT PRIMARY KEY FIRST;
Query OK, 0 rows affected (0.056 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> CEATE TABLE daten (
    -> tel VARCHAR(50));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near 'CEATE TABLE daten (
tel VARCHAR(50))' at line 1
```

```
MariaDB [normalization]> CREATE TABLE daten (
   -> tel VARCHAR(50));
Query OK, 0 rows affected (0.025 sec)

MariaDB [normalization]> INSERT INTO person VALUES
   -> ("Schmitz", "Charly");
ERROR 1136 (21S01): Column count doesn't match value count at row 1
MariaDB [normalization]> INSERT INTO person VALUES
   -> ("Schmitz", "Charly", "");
ERROR 1136 (21S01): Column count doesn't match value count at row 1
MariaDB [normalization]> explain person;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| id      | int(11)      | NO   | PRI | NULL    | auto_increment |
| name    | varchar(100) | YES  |     | NULL    |                |
| vorname | varchar(100) | YES  |     | NULL    |                |
| tel     | varchar(100) | YES  |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
4 rows in set (0.003 sec)

MariaDB [normalization]> INSERT INTO person VALUES
   -> (0,"Schmitz", "Charly", "");
Query OK, 1 row affected (0.006 sec)

MariaDB [normalization]> INSERT INTO daten VALUES
```

```
   -> ("+49 (0) 30 80 34 39 01");
Query OK, 1 row affected (0.004 sec)

MariaDB [normalization]> INSERT INTO daten VALUES
   -> ("+49 (0) 30 80 34 39 02");
Query OK, 1 row affected (0.005 sec)

MariaDB [normalization]>
MariaDB [normalization]> ALTER TABLE daten ADD person_id INTEGER NOT NULL;
Query OK, 0 rows affected (0.011 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> ALTER TABLE daten DROP person_id;
Query OK, 0 rows affected (0.009 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> ALTER TABLE daten ADD person_id INTEGER NOT NULL FIRST;
Query OK, 0 rows affected (0.014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [normalization]> explain daten;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| person_id | int(11)     | NO   |     | NULL    |       |
| tel       | varchar(50) | YES  |     | NULL    |       |
```

```
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.003 sec)

MariaDB [normalization]> SELECT * FROM person;
+----+---------+---------+------+
| id | name    | vorname | tel  |
+----+---------+---------+------+
|  1 | Schmitz | Charly  |      |
+----+---------+---------+------+
1 row in set (0.001 sec)

MariaDB [normalization]> UPDATE daten SET person_id = 1
    -> ;
Query OK, 2 rows affected (0.004 sec)
Rows matched: 2  Changed: 2  Warnings: 0

MariaDB [normalization]> SELECT * FROM person;
+----+---------+---------+------+
| id | name    | vorname | tel  |
+----+---------+---------+------+
|  1 | Schmitz | Charly  |      |
+----+---------+---------+------+
1 row in set (0.000 sec)

MariaDB [normalization]> SELECT * FROM daten;
+-----------+-----------------------+
```

```
| person_id | tel                  |
+-----------+----------------------+
|         1 | +49 (0) 30 80 34 39 01 |
|         1 | +49 (0) 30 80 34 39 02 |
+-----------+----------------------+
2 rows in set (0.001 sec)

MariaDB [normalization]> SELECT * FROM person, daten
    -> WHERE person.id = daten.person_id;
+----+---------+---------+------+-----------+----------------------+
| id | name    | vorname | tel  | person_id | tel                  |
+----+---------+---------+------+-----------+----------------------+
|  1 | Schmitz | Charly  |      |         1 | +49 (0) 30 80 34 39 01 |
|  1 | Schmitz | Charly  |      |         1 | +49 (0) 30 80 34 39 02 |
+----+---------+---------+------+-----------+----------------------+
2 rows in set (0.004 sec)

MariaDB [normalization]> SELECT name, vorname, tel FROM person, daten
    -> WHERE person.id = daten.person_id;
ERROR 1052 (23000): Column 'tel' in field list is ambiguous
MariaDB [normalization]> SELECT name, vorname, daten.tel FROM person, daten
    -> WHERE person.id = daten.person_id;
+---------+---------+----------------------+
| name    | vorname | tel                  |
+---------+---------+----------------------+
| Schmitz | Charly  | +49 (0) 30 80 34 39 01 |
```

| Schmitz | Charly | +49 (0) 30 80 34 39 02 |

+---------+---------+------------------------+

2 rows in set (0.000 sec)

MariaDB [normalization]>

```
MariaDB [normalization]> SELECT name, vorname, daten.tel
    -> FROM person
    -> JOIN daten ON person.id = daten.person_id;
+---------+---------+------------------------+
| name    | vorname | tel                    |
+---------+---------+------------------------+
| Schmitz | Charly  | +49 (0) 30 80 34 39 01 |
| Schmitz | Charly  | +49 (0) 30 80 34 39 02 |
+---------+---------+------------------------+
```

SELECT vorname, nachname, name, beginn, ende FROM mitarbeiter, projekt, projekt_mitarbeiter
WHERE mitarbeiter.id = mitarbeiter_id && projekt.id = projekt_id

SELECT title FROM film WHERE length > 90 && length < 130

SELECT title FROM film WHERE length BETWEEN 90 AND 120

SELECT name, nachname, qualifikation

FROM abteilung

JOIN mitarbeiter ON id = abteilung_id

SELECT name, nachname, qualifikation

FROM abteilung

JOIN mitarbeiter ON id = abteilung_id

WHERE qualifikation = "Techniker"

Abfrage über 3 Tabellen

SELECT * FROM auftrag, mitarbeiter, auftrag_mitarbeiter

WHERE

mitarbeiter.personalnummer = auftrag_mitarbeiter.personalnummer

&&

auftrag.auftragsnummer = auftrag_mitarbeiter.personalnummer

Abfrage über 3 Tabellen (M:N-Beziehung)

SELECT *

FROM auftrag

JOIN (auftrag_mitarbeiter

JOIN mitarbeiter

ON auftrag_mitarbeiter.auftragsnummer = auftragsnummer)

ON mitarbeiter.personalnummer = auftrag_mitarbeiter.personalnummer