CODERS.BAY

DATABASE

16.6.2020 Basics of databases

Relational databases

e.g. MySQL, Oracle, Microsoft SQL

Other databases:

- Document-based databases
- hierarchical database
- nosql-databases (e.g. Graph Ql) → new stuff

⏩ **We are learning MySQL**.
When googling for help you need to check if the answer refers to the language MySQL.

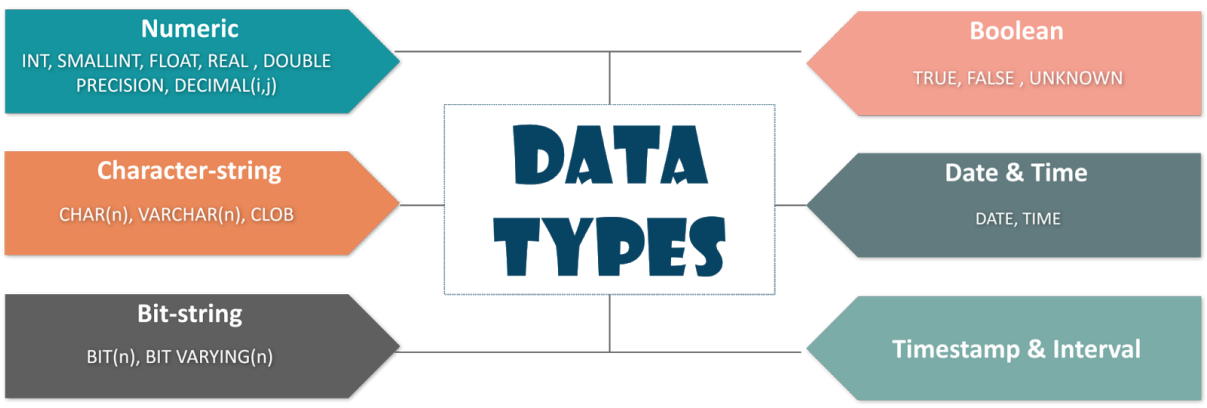
Data modelling

Sketch of a use case for a database. Has to be made for all databases and differs later in the implementation (which DB and language is used), when we setup the database.

1. Conceptual schema
2. Logic schema (readable by machines)
3. Implementation

⏩ For DB we are using the **ER-model with the CHEN-notation.**
UML is not used for DB, more for programming use cases.

Data Types



1. Numerics (Zahlen)

INT - for whole numbers, no decimals
Used for e.g. IDs, to calculate

Numeric data Types have 2 options.
signed: allows negative numbers
unsigned: does not allow negative numbers

TINYINT: max. 255
SMALLINT: max. 65535
MEDIUMINT: ...
INT
BIGINT

DECIMAL - if you need perfect numbers

DECIMAL	**999**	**,**	**99**
naming	precision		scale

other names for decimal:
DEC
NUMERIC
FIXED

FLOAT 4 bytes (about 7 digits)
DOUBLE 8 bytes (about 15 digits)

Data types that can store a lot of data and decimal numbers, but they do not maintain precision. Better use DOUBLE, because it's more precise.

If precision does not have to be perfect, minor differences are ok at the end of a decimal.

2. Binary (Binärdaten)

consists of 0 and 1, nothing else
BINARY
VARBINARY

3. Strings (Zeichenketten)

CHAR - fixed length, for strings

e.g. CHAR (10) → The size parameter specifies the column length in characters.
Which character set? → unicode → encoded with UTF8
Default for text data type text: anything

What values are allowed? The default number is 1.
0 → null
255 → max. number for 8 bit number

Note:
If you do not need the foll space then you waste space.
If you export data with spaces SQL strips those spaces away.
→ PAD_CHAR_TO_FULL_LENGTH

VARCHAR - variable length

A VARIABLE length string (can contain letters, numbers, and special characters).
Attention: every string has different bytes, the encoding has also different bytes (e.g. chinese has 3 bytes)
Max. 65535 bytes (if a character takes 1 byte, the it´s 65535 characters)
Row-limit: 65535

TEXT:
Use depends on the size of the text.
No row limit like char because the text data is stored elsewhere in MySql.
If you need more text in your rows then use text data.
Default for text data type text: NULL

“CLOB”: character large object

TEXT: max. 65535 bytes
TINYTEXT: max. 255 bytes
MEDIUMTEXT: 16 mio. bytes
LONGTEXT: 4 billion. bytes

ENUM
list of values
Max. 3000 options
allows you to pick one of many options
everything is stored as a number
1 very good
2 ok
3 bad → #3 is stored as an empty string, no data is stored, just the number (save space)
0 is invalid

ENUM(“very good”, “ok”, “bad”)

→ entering #2 = “ok”
The string has to match the string.

Using enum can be done with database design. It might be easier.

SET
Similar to ENUM
allows you to pick many of many options - a set of data
Max. 64 options possible

Note: Do not use SET for M:N relations, solve this with database design (intermediate table)

BLOB - For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
TINYBLOB - For BLOBs (Binary Large OBjects). Max length: 255 bytes
MEDIUMBLOB - For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGBLOB - For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data

4. Booleans (Boolesche Werte)

true or false
If a boolean is compared to “null” or “unknown”, then the result is always “unknown”.

5. Datetimes (Datum-/ Zeit-Werte)

DATE
TIME
DATETIME
TIMESTAMP

DATE	TIME (6)	DATETIME
2020-01-22	22:54:30.123456 Seconds up to 6 decimals!	Date and time in one!

Date conventions: YY MM TT yy mm tt dd hh ss,... attention to detail!

Check documentation: [MySQL 8.0 Reference Manual :: 11.2 Date and Time Data Types]

Einheit	Kürzel
year	yyyy, yy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	dw
hour	hh
minute	mi, n
second	ss, s
millisecond	ms

6. Intervals (Intervalle)

Similar to datetime, difference between 2 datetimes

7. XML

Format for plattform neutral exchange of data

SQL Data Types for MySQL, SQL Server, and MS Access

https://www.youtube.com/watch?v=UGu9unCW4PA&list=PL_c9BZzLwBRKn20DFbNeLAAbw4ZMTlZPH

How to use and write date formats:

Student (Matrnr: INTEGER, Name: VARCHAR, Semester: TINYINT)

→ Der Student (Objekt) hat die Attribute Matrnr, Name und Semester. Es gibt immer ein Schlüsselattribut.
→ Objekte stehen in Beziehung zueinander. Beziehungen können auch Attribute haben.

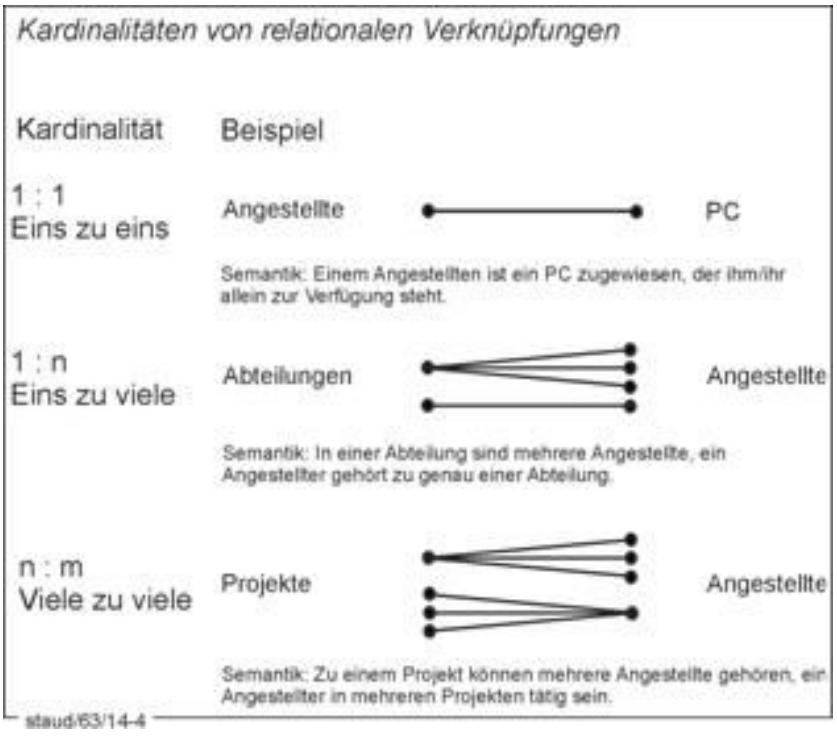
Relations

Different kinds of relations

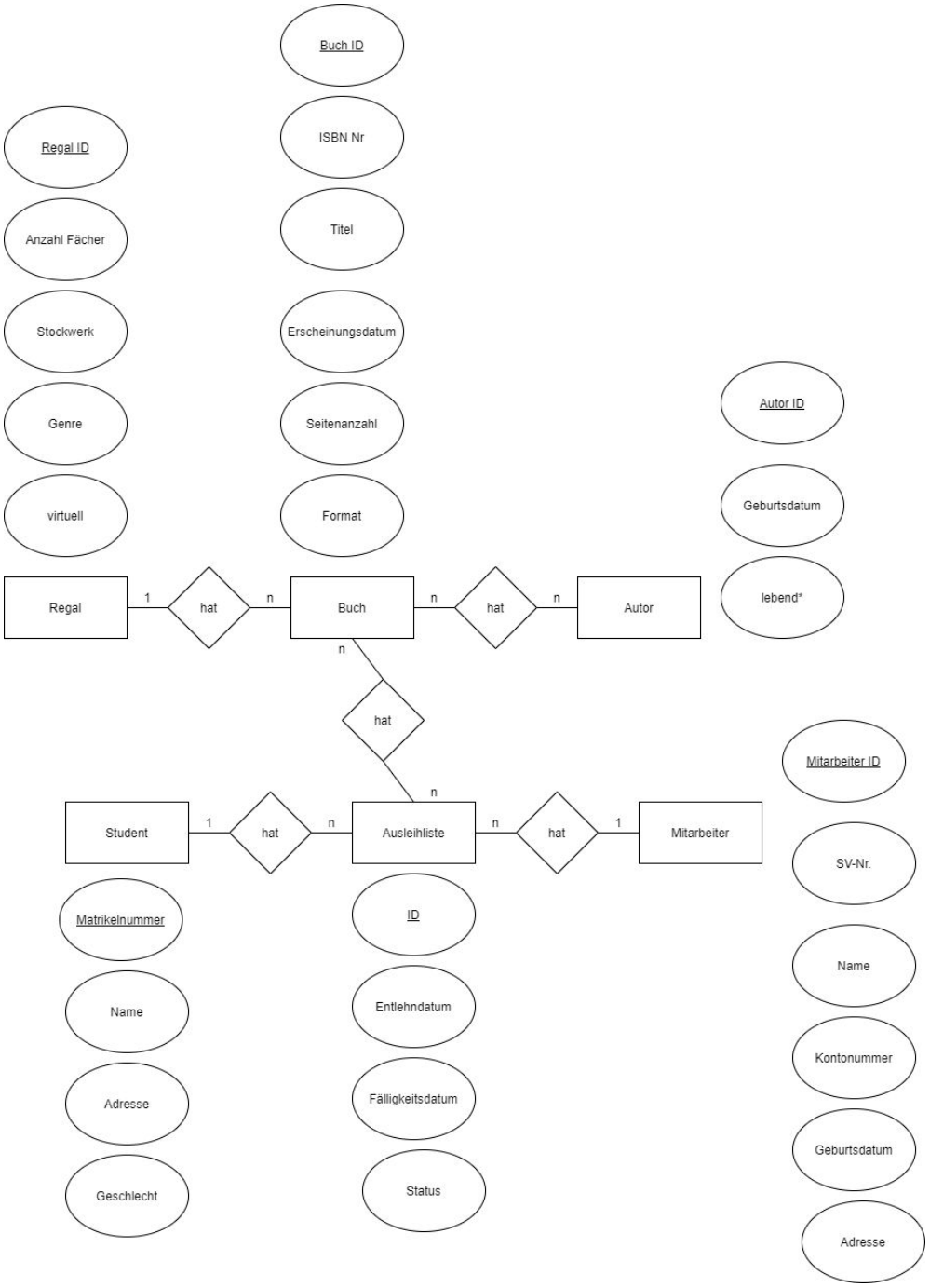
1. binär
2. unär
3. ternär

Describing relations with cardinalities

1. 1:1
2. 1:n
3. n:n



First draft of ER-diagram - a library:



MIN, MAX-Notation

Bei der (min,max)-Notation wird für jeden an einer Beziehung beteiligten Entitätstyp ein geordnetes Paar mit einem **Minimal- und einem Maximalwert** angegeben. Diese Werte geben

an, an wie vielen Beziehungsausprägungen die Entitätausprägungen mindestens teilnehmen müssen und an wie vielen sie höchstens teilnehmen dürfen.

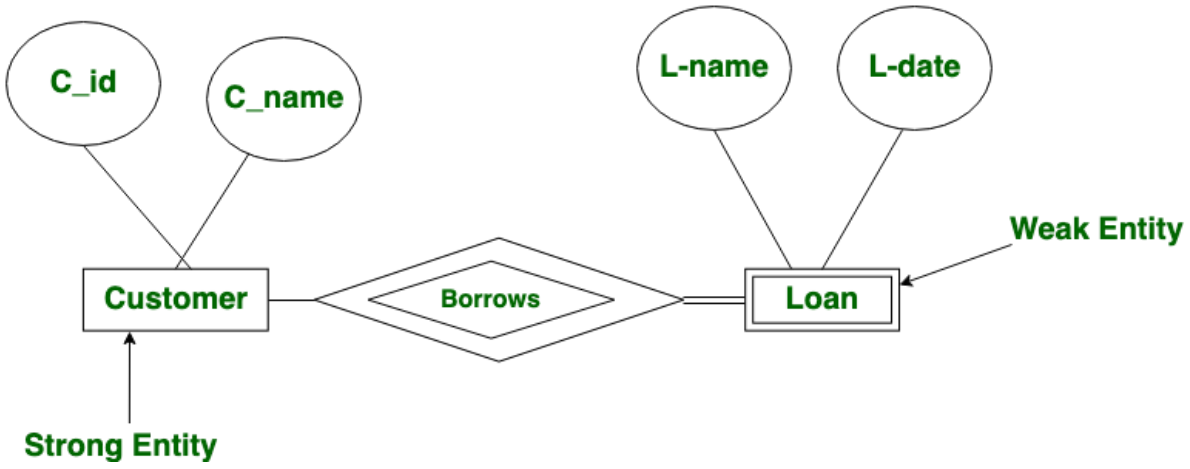
Spieler-Entität (1,1) → da jeder Spieler zu genau einer Mannschaft gehören soll

Mannschafts-Entität (11,11) → da es für jede Mannschaft genau 11 Spieler geben soll, die zu ihr gehören

Other possible options:
0,0
0,1
0,*
1,*

Starke und schwache Entitäten

A **weak entity** is **dependent on a strong entity** to ensure the its existence. Unlike a strong entity, a weak entity does not have any primary key. It instead has a partial discriminator key.



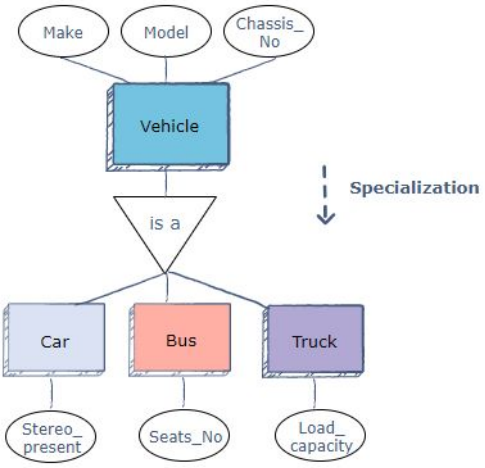
E.g. A bank loan does not exist without a customer. A Customer can have a loan.

A weak entity is represented by a double rectangle.
The relation between one strong and one weak entity is represented by a double diamond.
https://www.geeksforgeeks.org/weak-entity-set-in-er-diagrams/

Generalisierung und Spezialisierung

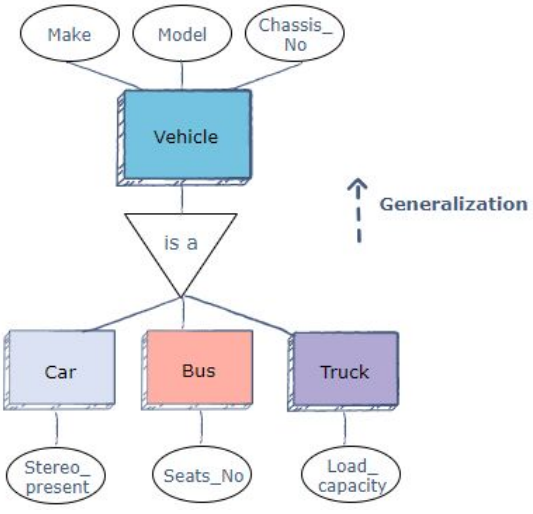
Specialisation

Top-down approach where a higher-level entity is divided into multiple specialised lower-level entities.



Generalisation

Bottom-up approach: lower-level entities are combined to a higher-level entifies.



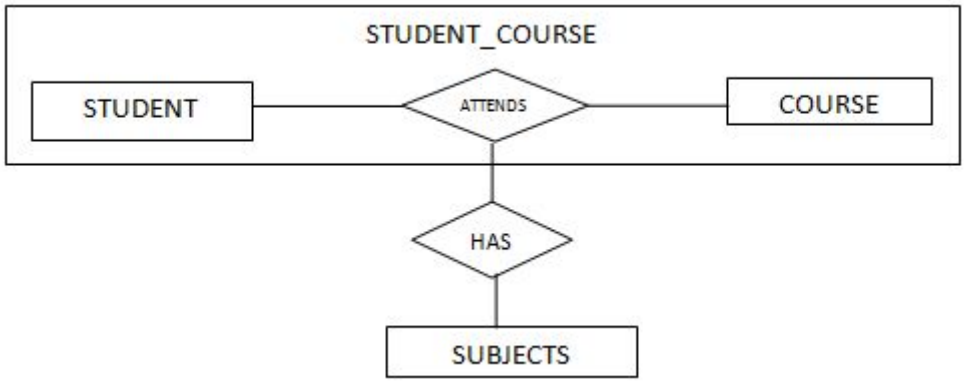
NOTE: In both cases Make, Model, and Chassis_No are shared attributes among the entities Car, Bus, Truck. These entities have their own attributes as well.

is a-Beziehung

Type A is a subtype of type B when A's specification implies B's specification.

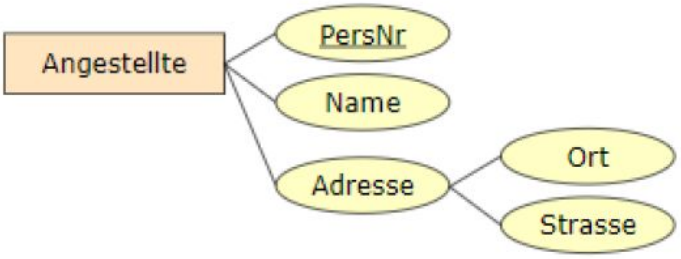
Aggregation

Relations with corresponding entities are aggregated.



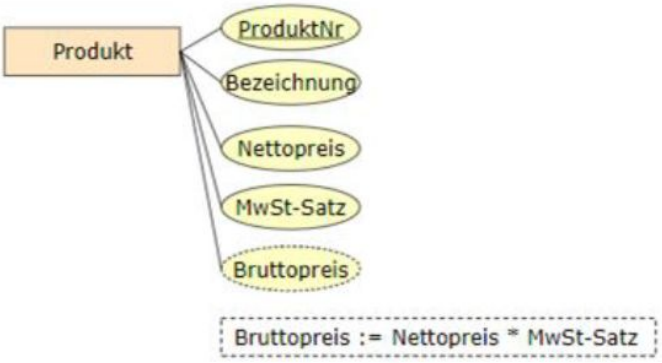
Strukturierte Attribute

Groups of attributes that share similar attributes.



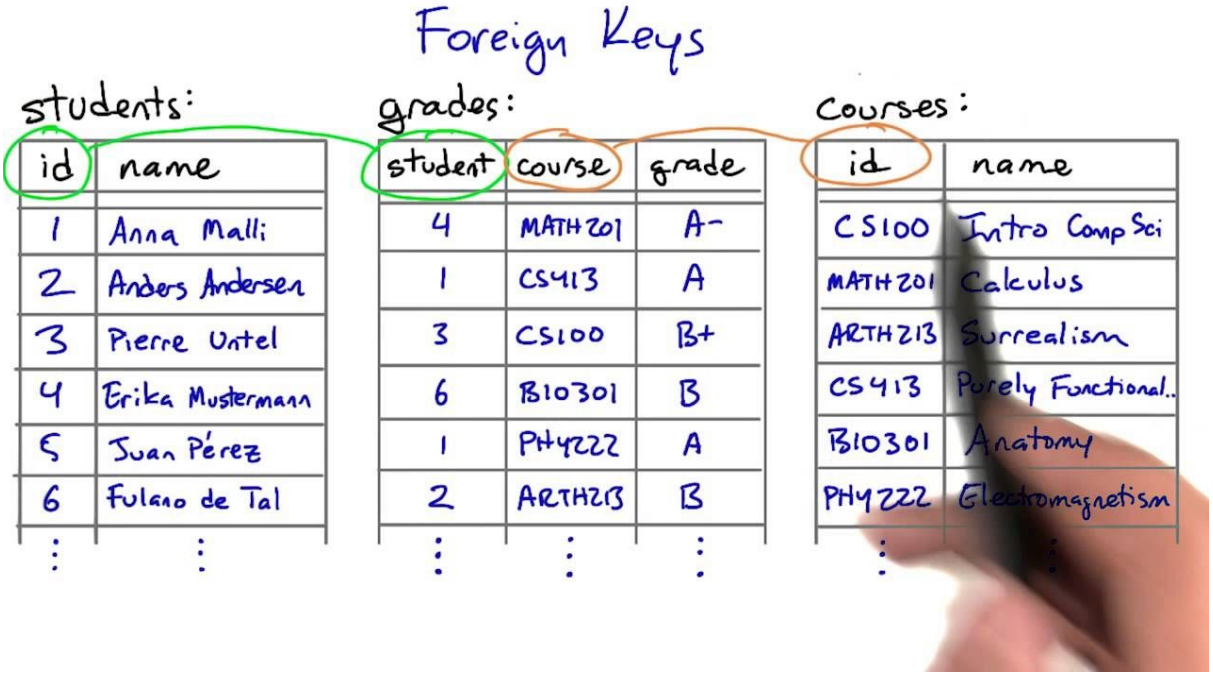
Abgeleitete Attribute (berechnet sich aus anderen Attributen)

“Calculated attributes” who´s values can´t be stored in an attribute, but calculated via a DB query.



Fremdschlüssel

Primary keys from other entities used as referential key in another entity.



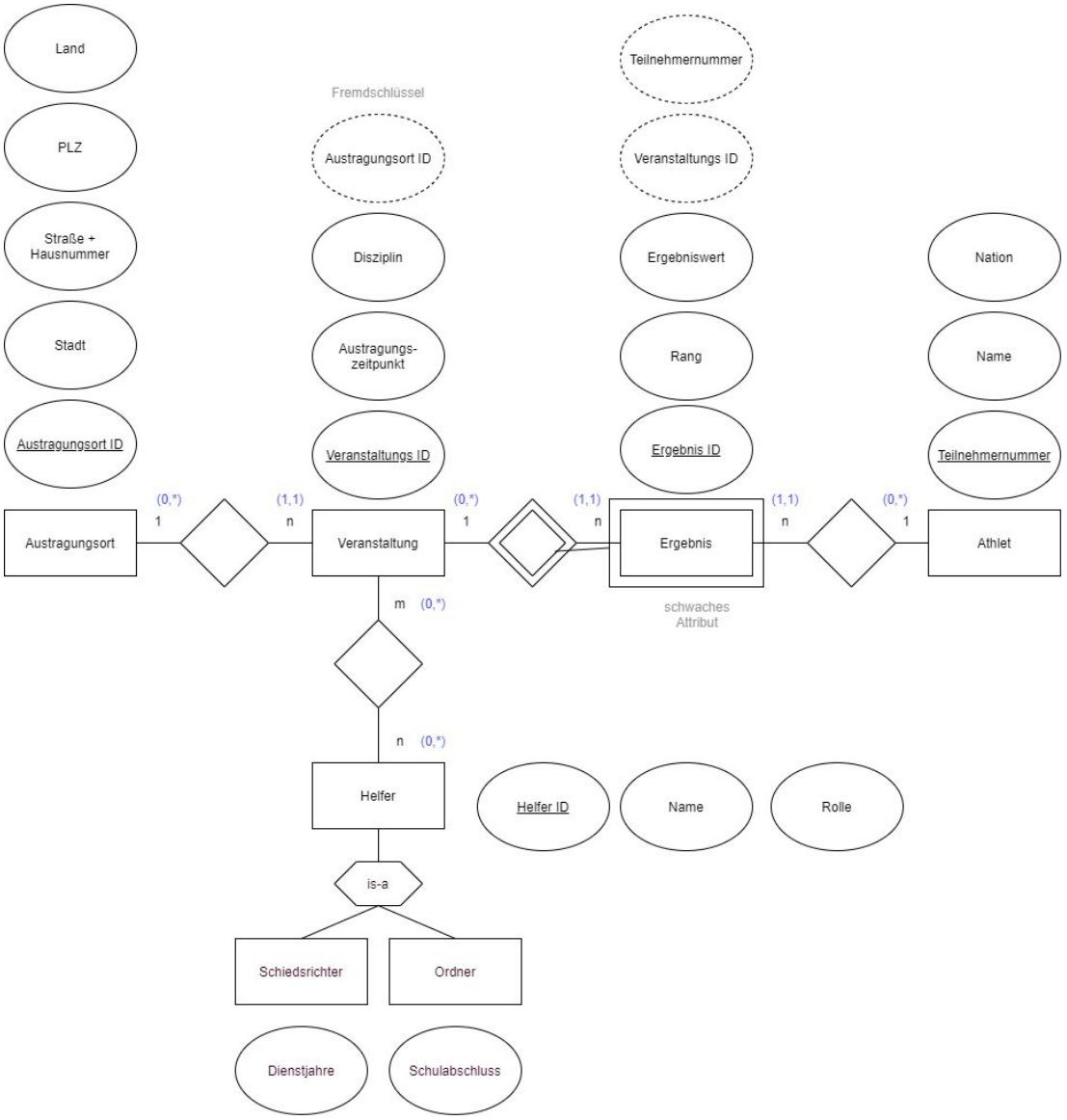
Kreuztabellen bei n:m Beziehungen mit zusammengesetzten Primärschlüssel

Artificial primary key that is a combination of the primary keys of two entities in a n:m relation.

Vorgehensweise:

1. ER-Diagramm
2. Relation in Tabellen darstellen
3. Textuelle Notation

Exercise:



Austragungsort: {[Austragungsort ID:integer (PK), Stadt:string, Straße+Hausnummer:string, PLZ:integer, Land:string]}

Veranstaltung: {[Veranstaltungs ID:integer (PK), Austragungszeitpunkt:datetime, Disziplin:string, Austragungsort ID:integer (FK)]}

Ergebnis: {[Ergebnis ID:integer (PK), Rank:integer, Ergebniswert:string, Veranstaltungs ID:integer (FK), Teilnehmernummer:integer (FK)]}

Athlet: {[Teilnehmernummer:integer (PK), Name:string, Nation:string]}

Helfer: {[Helfer ID: integer (PK), Name:string, Rolle:string]}

Helfer_Ergebnis_Liste: {[Helfer ID:integer (PK, FK), Veranstaltung ID:integer (FK, PK)]}

Normalformen

Ziel: Anomalien vermeiden
Änderungsanomalie, Einfügungsanomalie, Löschanomalie
Normalformen: rules to avoid redundancies and inconsistencies

1NF

- only atomic attributes
- NOTE: no more that one value per attribute, cell
- **Keinen Schas in den Feldern abspeichern, nichts mischen.**

Während der **Anforderungsanalyse in der Datenbankentwicklung ist folgende Rechnungsinformation** aufgenommen worden:

R.-Nr.	Datum	Name	Straße	Ort	Artikel	Anzahl	Preis
187	01.01.2012	Max Mustermann	Musterstr. 1	12345 Musterort	Bleistift	5	1,00 €

Nach der Anwendung der **Ersten Normalform (1NF)** sieht das Ergebnis folgendermaßen aus:

R.-Nr.	Datum	Name	Vorname	Straße	Hnr.	PLZ	Ort	Artikel	Anzahl	Preis	Währung
187	01.01.2012	Mustermann	Max	Musterstr.	1	12345	Musterort	Bleistift	5	1,00	Euro

Die **erste Normalform (1NF)** ist dann erfüllt, wenn die **Wertebereiche** der **Attribute** des Relationstypen **atomar** vorliegen.

2NF

- partial dependencies of an attribute ONLY on the one or all keys
- **It´s not ok when an attribute depends on only one of many keys.**

Die **Rechnungsinformationen** liegen nun in der ersten Normalform (1NF) vor:

R.-Nr.	Datum	Name	Vorname	Straße	Hnr.	PLZ	Ort	Artikel	Anzahl	Preis	Währung
187	01.01.2012	Mustermann	Max	Musterstr.	1	12345	Musterort	Bleistift	5	1,00	Euro

Nach der Anwendung der **Zweiten Normalform (2NF)** sieht das Ergebnis folgendermaßen aus:

Rechnung		
R.-Nr.	Datum	Knr.
187	01.01.2012	007

Kunde						
Knr.	Name	Vorname	Straße	Hnr.	PLZ	Ort
007	Mustermann	Max	Musterstr.	1	12345	Musterort

Rechnungsposition			
R.-P.-Nr.	R.-Nr.	Art.-Nr.	Anzahl
1	187	69	5

Artikel		
Art.-Nr.	Artikel	Preis
69	Bleistift	1,00

3NF

- eliminates additional transitive dependencies
- **It´s not ok, when an attribute depends on anything else than a key/ the keys.**

Die **Kundeninformationen** liegen nun in der zweiten Normalform (2NF) vor:

Kunde						
Knr.	Name	Vorname	Straße	Hnr.	PLZ	Ort
007	Mustermann	Max	Musterstr.	1	12345	Musterort

Nach der Anwendung der **Dritten Normalform (3NF)** sieht das Ergebnis folgendermaßen aus:

Kunde					
Knr.	Name	Vorname	Straße	Hnr.	PLZ
007	Mustermann	Max	Musterstr.	1	12345

Postleitzahl	
PLZ	Ort
12345	Musterort

Postleitzahl depends on Musterort which is NOT a key.

Setup a database

Software:

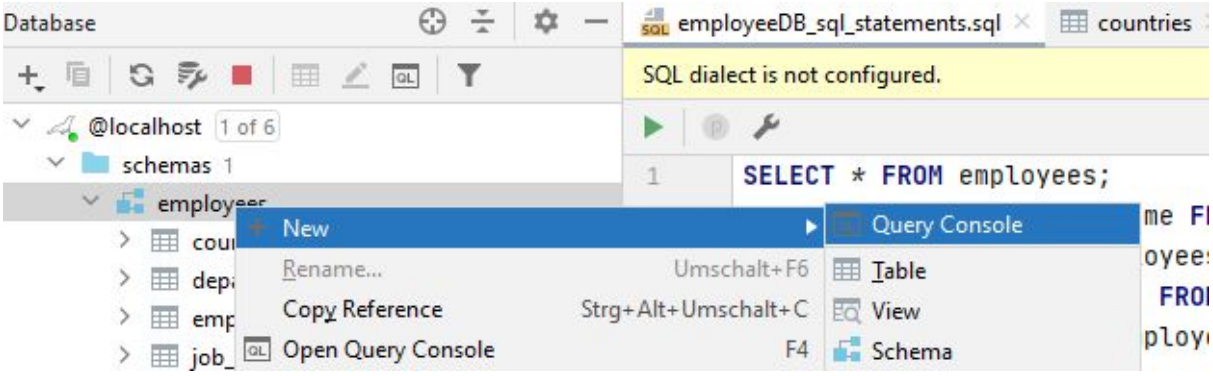
- XAMPP
    - **Apache** und **MySql** und immer aktivieren
    - http://localhost/dashboard/
- Datagrip von Jetbrains (development environment)

How is everything connected:

1. DB server
2. software or server type (e.g. MySql, Oracle, Maria DB,...) that runs on the server to manage the DB
3. graphic interface to work on the DB with software like PHP MyAdmin, Datagrip,...

→ **We used XAMPP to setup a localhost on our PC.** It automatically has Apache and MySql as server and server type included. The pre-installed graphic interface PHP MySql can be accessed by simply typing localhost in the browser window.
XAMPP Tutorial/Anleitung: Windows Installation & Konfiguration
https://en.wikipedia.org/wiki/XAMPP

Datagrip how-to:
Open previously saved sql file → click on database “employees” and select **new query console**



SQL queries:

SELECT	**attributes**, selected data in a relation
FROM	from which **relation**, which table
WHERE	**conditions**, selection criteria, combinations, filter
GROUP BY	aggregating data
HAVING	selections for groups (e.g. to be aggregated)
ORDER BY	sorting the results
AND	combines two conditions → &&
AS	renaming the column or table
DISTINCT	used with SELECT to return unique entries
COUNT ()	counts rows of selected column that are not null
COUNT(*)	count everthing
IS NULL	used with WHERE to test conditions
IS NOT NULL	used with WHERE to test conditions

First statements:

```
SELECT first_name, last_name FROM employees;

SELECT * FROM  employees;

SELECT * FROM employees WHERE employee_id = 100;

SELECT * FROM employees ORDER BY last_name asc;

SELECT * FROM employees ORDER BY hire_date asc;

SELECT * FROM employees ORDER BY manager_id asc, salary desc;
→ Kombi von Sortierungen

SELECT count(*) FROM employees;

SELECT count(*) AS 'Anzahl Mitarbeiter' FROM employees;
→ Spaltenausgabe benennen

SELECT count(first_name) AS "Anzahl Mitarbeiter", manager_id  FROM
employees GROUP BY manager_id;
→ wieviele Mitarbeiter ein Manager hat

SELECT DISTINCT manager_id FROM employees;
→ mit distinct Duplikate entfernen
```

```sql
SELECT DISTINCT manager_id FROM employees WHERE manager_id IS NOT NULL;
```
→ Nullwerte entfernen

```sql
SELECT DISTINCT manager_id FROM employees WHERE manager_id IS NOT NULL AND manager_id > 105;
```

List of SQL keywords
https://www.w3schools.com/sql/sql_ref_keywords.asp

Most commonly used sql commands:

https://www.codecademy.com/articles/sql-commands

```sql
ALTER TABLE countries ADD test_table int; -- adds a new column to a selected table

SELECT last_name FROM employees WHERE manager_id = 124 AND salary > 3000 ;-- combining 2 conditions: "All employees of manager 124 with salary > 3000."

SELECT country_id AS "Länder" FROM countries; -- remaning the column of a table

SELECT AVG(salary) FROM employees; -- average value of numeric column

SELECT last_name, first_name FROM employees WHERE salary BETWEEN 2000 AND 3000; -- filter the result within a certain range e.g a certain salary range

SELECT last_name,
   CASE
       WHEN salary > 10000 THEN 'extrem gut bezahlte Leute'
       WHEN salary > 5000 THEN 'gut bezahlte Leute'
       ELSE 'normales Gehalt'
   END
FROM employees; -- With CASE you can create different outputs.

SELECT COUNT(commission_pct) AS 'Anzahl Provisionen' FROM employees;
-- counts all rows that are not null

CREATE TABLE test_table (
```

```sql
    column1 varchar(10),
    column2 varchar(10),
    column3 varchar(10)
); -- creating a new table

INSERT INTO test_table (column1, column2, column3)
    VALUES
        ('Apfel', 'Gurke', 'Käse');

INSERT INTO test_table (column1, column2, column3)
    VALUES
        ('Birne', 'Karotte', 'Milch'); -- insert content per row

DELETE FROM test_table WHERE column1 = 'Apfel'; -- deletes the whole
row if there is a matching content

SELECT department_id, COUNT(*) FROM employees GROUP BY
department_id; -- only in combination with COUNT to aggregate data,
group results

SELECT department_id, COUNT(*) FROM employees GROUP BY department_id
HAVING COUNT(*) > 1; -- combining aggregating and conditions (WHERE
is not working) --> "All department with more that 1 employees."

SELECT commission_pct FROM employees WHERE commission_pct IS NULL;
-- IS NULL or IS NOT NULL

SELECT first_name FROM employees WHERE first_name LIKE 'E%'; --
search for a specific pattern

SELECT salary FROM employees ORDER BY salary desc LIMIT 3; -- limit
results e.g. top 3 salaries

SELECT MAX(salary) FROM employees;
SELECT MIN(salary) FROM employees; -- min and max values

SELECT manager_id FROM employees WHERE manager_id = 100 OR
department_id = 110; -- selection with OR condition

SELECT ROUND(salary, 1) FROM employees; -- round integers to defined
decimal numbers

SELECT DISTINCT commission_pct FROM employees; -- returns all unique
values (Duplikate entfernen)
```

```sql
SELECT SUM(salary) FROM employees; -- sum of values (does not count the number of lines!)

UPDATE test_table SET column1 = 'ApfelNEU' WHERE column1 = 'Apfel';
-- updates content of a specific row

WITH temporary_department_id (Vorname, Nachname) AS (
   SELECT first_name, last_name
   FROM employees )
SELECT *
FROM temporary_department_id;  -- store the result of a query in a temporary table
```

LIKE and the patterns that can be used:

Das ‚MUSTER' kann nach folgenden Strukturen aufgebaut werden:

- ‚L_S': Alle Zeichenketten die mit einem ‚L' beginnen, inklusive einem Folgezeichen und mit einem ‚S' enden.
- ‚BEST%': Alle Zeichenketten, die mit ‚BEST' beginnen.
- ‚%UNG': Alle Zeichenketten, die auf ‚UNG' enden.
- ‚%ST%': Alle Zeichenketten, die an irgendeiner Stelle das Muster ‚ST' enthalten.

To be clarified:

INNER JOIN
OUTER JOIN