SQL

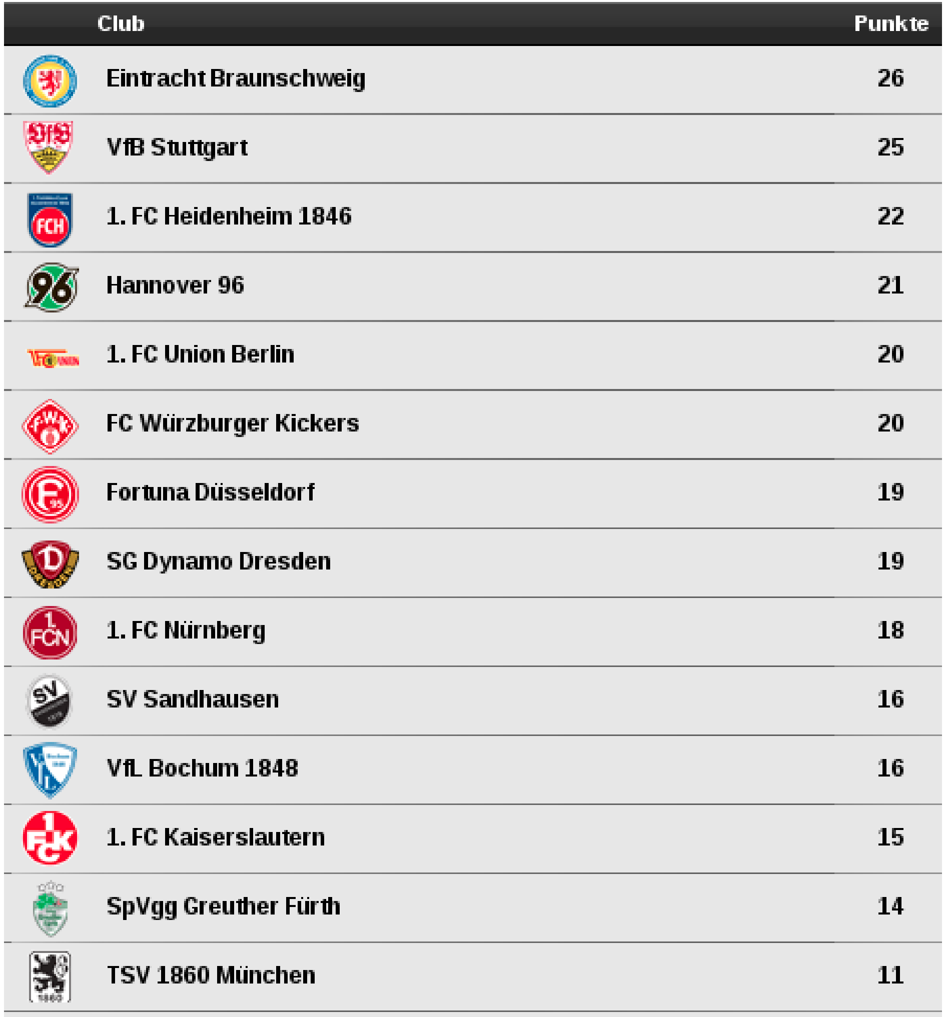
AW Academy

Structured Query Language

for Database Systems

AW Academy

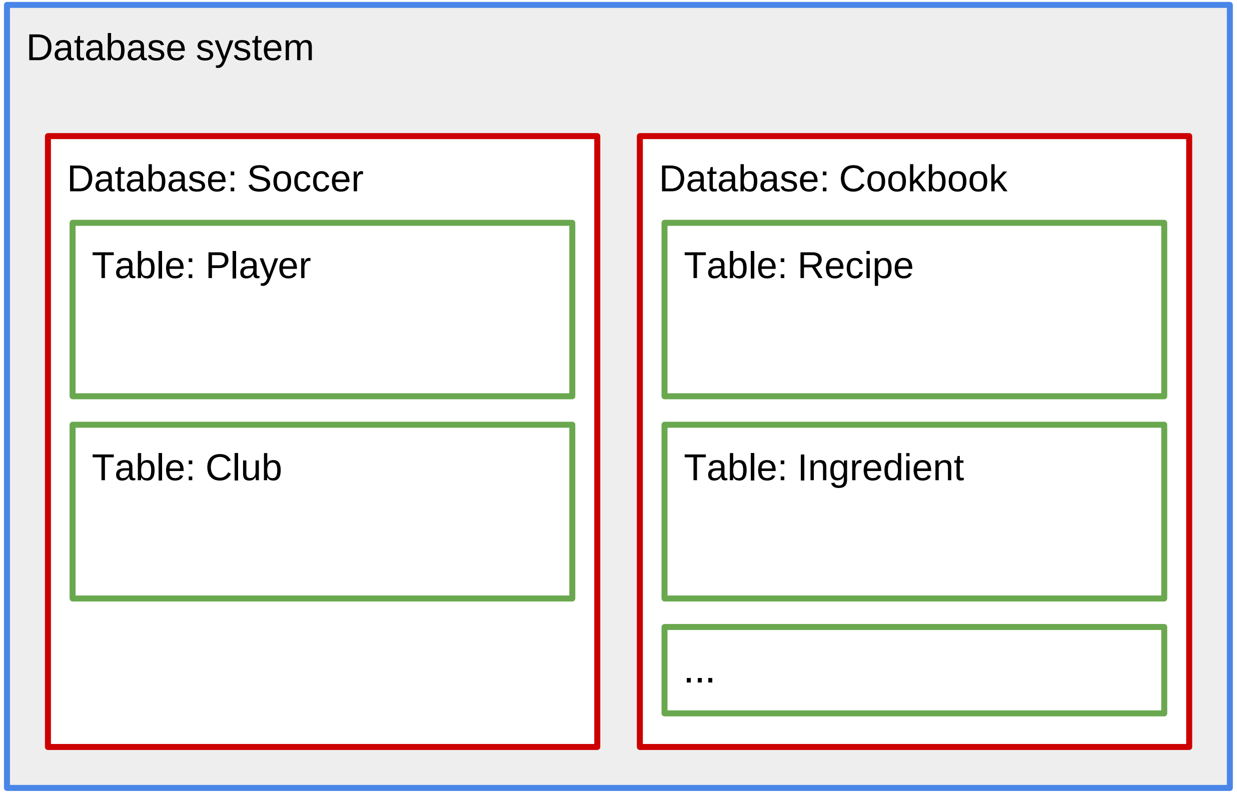
Behind every app, there is a database.
Most of them speak SQL.



Club	Punkte
Eintracht Braunschweig	26
VfB Stuttgart	25
1. FC Heidenheim 1846	22
Hannover 96	21
1. FC Union Berlin	20
FC Würzburger Kickers	20
Fortuna Düsseldorf	19
SG Dynamo Dresden	19
1. FC Nürnberg	18
SV Sandhausen	16
VfL Bochum 1848	16
1. FC Kaiserslautern	15
SpVgg Greuther Fürth	14
TSV 1860 München	11

Tables

- Columns
  - define structure of the data
- Rows
  - contain the data



Database system

Database: Soccer

Table: Player

Table: Club

Database: Cookbook

Table: Recipe

Table: Ingredient

...

Common SQL Database systems

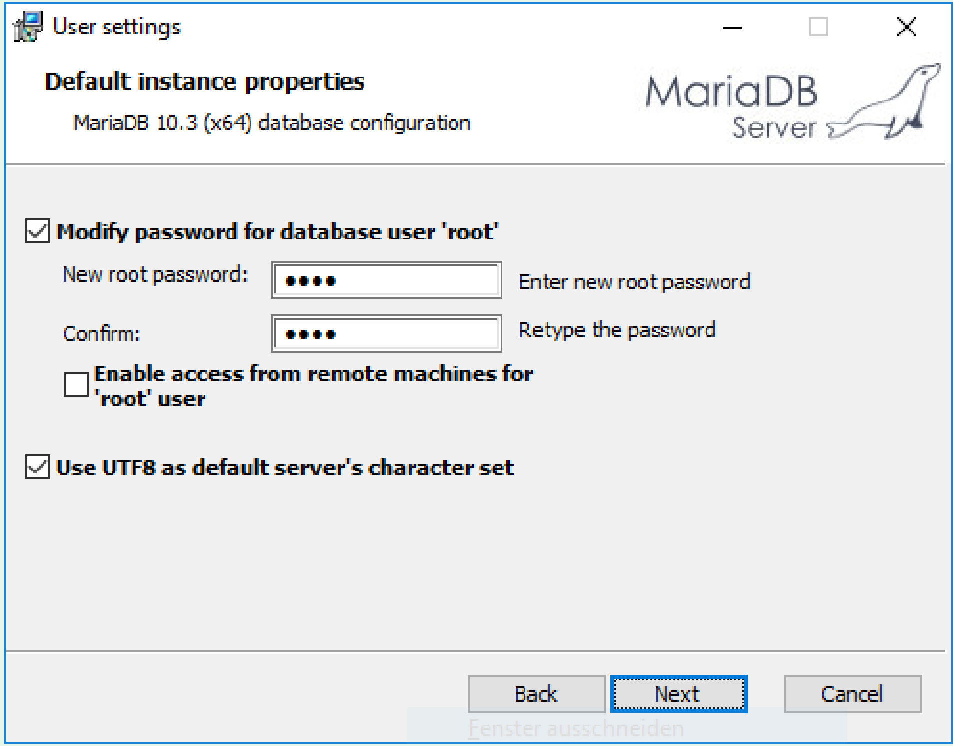
- MySQL and MariaDB
- Oracle RDBMS
- PostgreSQL
- SQLite

"Data Model"

Which tables are there and which columns do they have?

MariaDB Installation

- https://downloads.mariadb.org
- Stable Version
- Windows x86_64
- MSI Package
- Installation
  - Root password may be simple, access is only possible from your laptop (e.g. "root")
  - UTF8 as default character set
  - Leave other options unchanged



User settings

Default instance properties

MariaDB 10.3 (x64) database configuration

☑ **Modify password for database user 'root'**

New root password: •••• Enter new root password

Confirm: •••• Retype the password

□ **Enable access from remote machines for 'root' user**

☑ **Use UTF8 as default server's character set**

Connecting HeidSQL with server

!!! Trainer TODO: set up a MariaDB server where learners can connect to, import soccer.sql !!!

- Server:
  - Hostname: **`sql.xxxxxx.com`**
  - Username: **`academy`**
  - Password: **`java`**
- Select database: **`soccer`**
- Run query...

Reading Query

```sql
SELECT name, points FROM club
```

name	points
FC Philip	1
SpVgg Otto	3
SC Franzi	0
Gruber 1987	2
VfB Horsti	1
1. FC Monika	2

Type of queries

- Rows
    - Reading
    - Creating
    - Changing
    - Deleting
- Creating Tables
- ...

Creating Query

```
INSERT INTO club
(name, points)
VALUES
("SC Example", 3)
```

Create a row in the “club” table for your own club!

Relational Databaser

Database: Soccer

Table: Club

name	points
FC Philip	1
SpVgg Otto	3
SC Franzi	0
Gruber 1987	2
VfB Horsti	1
1. FC Monika	2

Table: Player

club	name	jersey_number
FC Philip	Max Muster	12
FC Philip	Sepp Huber	3
FC Philip	Georg Zufall	24
FC Philip	Fritz Beispiel	2
1. FC Monika	Paula Unsinn	5
1. FC Monika	Nina Nie	2

IDs: Unique and fixed

Database: Soccer

Table: Club

id	name	points
1	FC Philip	1
2	SpVgg Otto	3
3	SC Franzi	0
4	Gruber 1987	2
5	VfB Horsti	1
6	1. FC Monika	2

Table: Player

club_id	points	jersey_number
1	Max Muster	12
1	Sepp Huber	3
1	Georg Zufall	24
1	Fritz Beispiel	2
6	Paula Unsinn	5
6	Nina Nie	2

IDs: Eindeutig und Fest

Incrementing numbers vs. UUIDs

Your club's ID was automatically generated by the database system. What is it?

Creating Query

```
INSERT INTO player
(club_id, name, jersey_number)
VALUES
(1, "Klara Kick", 33)
```

Create rows for at least 3 players in your club in the table "player"

Reading Query: All columns

```
SELECT id, name, points FROM club
```

simplified:

```
SELECT * FROM club
```

Reading Query: Filter with **`WHERE`**

```sql
SELECT * FROM club
WHERE points > 1
```

Find all players with jersay number greater than 5!

Updating Query

```sql
UPDATE player SET club_id = 2
WHERE id = 3
```

```sql
UPDATE club SET name = "Soccer United 2022"
WHERE id = 1
```

```sql
UPDATE player SET jersey_number = jersey_number + 1
```

Deleting Query

```sql
DELETE FROM player WHERE id = 3
```

Reading Query: Sorting with **`ORDER BY`**

```
SELECT * FROM club
ORDER BY points DESC
```

Sort all players by ascending jersey number!
Hint: The opposite of “DESC” is “ASC”

Reading Query: Filter and Sort

Find all players of your team, sorted by jersey number!

Hint: “WHERE” goes before “ORDER BY”

Queries over multiple tables

Show all players and their clubs:

```
SELECT * FROM player
JOIN club ON club.id = player.club_id
```

Queries over multiple tables

Attention: For the column names to be unique, you might need to prefix the table name

```sql
SELECT * FROM player

JOIN club ON club.id = player.club_id
```

player

club_id	name	jersey_number
1	Max Muster	12
1	Sepp Huber	3
3	Paula Unsinn	5

club

id	name	points
1	FC Philip	1
2	SpVgg Otto	3
3	SC Franzi	0

```sql
SELECT * FROM player
JOIN club ON club.id = player.club_id
```

club_id	name	jersey_number	id	name	points
1	Max Muster	12	1	FC Philip	1
1	Sepp Huber	3	1	FC Philip	1
3	Paula Unsinn	5	3	SC Franzi	0

Queries over multiple tables

Find all players in clubs with more than 2 points!

```sql
SELECT player.* FROM player
JOIN club ON club.id = player.club_id
WHERE ...
```

Outlook

- Relations
  - 1:n or m:n
- Aggregation

```
SELECT COUNT(*) FROM player
```

Your MariaDB server

- Connect with HeidiSQL: **`localhost`**

Northwind Data

Import `northwind.sql` into your server

- In HeidiSQL: File → Run SQL File
- Refres 🔄
- Database `northwind` was created ✔

Purchse 10536

Ordered by: Lehmanns Marktstand, Frankfurt

- 15 × Queso Manchego La Pastora, á 38,00€
- 20 × Gorgonzola Telino, á 12,50€
- 30 × Geitost, á 2,50€
- 35 × Camembert Pierrot, á 34,00€

Where can we find this information in the database?

Purchase & Product

A product has multiple purchases

A purchase has multiple products

Shipping list for purchase 10536

```sql
SELECT Quantity, ProductName
FROM PurchaseProduct
JOIN Product ON Product.ProductId = PurchaseProduct.ProductId
WHERE PurchaseId = 10536
```

Virtual Columns

Price of a PurchaseProduct

```sql
SELECT *, UnitPrice*Quantity AS price
FROM PurchaseProduct
```

Aggregation

How many products are there?

```
SELECT COUNT(*) FROM Product
```

How many products are there?

```sql
SELECT COUNT(*) FROM Product
```

Other aggregation functions: **MIN, MAX, SUM, AVG**

even more: https://dev.mysql.com/doc/refman/8.0/en/group-by-functions.html

How often was product 22 bought?

```sql
SELECT SUM(Quantity)
FROM PurchaseProduct
WHERE ProductId = 22
```

Total price of all purchases

```
SELECT SUM(UnitPrice * Quantity)
FROM PurchaseProduct
```

Total price for purchase 10536

```
SELECT SUM(UnitPrice * Quantity)
FROM PurchaseProduct
WHERE PurchaseId = 10536
```

GROUP BY

How many products are there for every supplier?

```
SELECT SupplierId, COUNT(*) FROM Product
GROUP BY SupplierId
```

How much sale do we make for every catagory?

```sql
SELECT CategoryName, SUM(Quantity*PurchaseProduct.UnitPrice)
FROM purchaseproduct
JOIN product ON product.ProductId = purchaseProduct.ProductId
JOIN category ON category.CategoryId = product.CategoryId
GROUP BY category.CategoryId
```

How much sale do we make for every supplier?

Solution: How much sale do we make for every supplier?

```sql
SELECT CompanyName, SUM(Quantity*PurchaseProduct.UnitPrice)
FROM purchaseproduct
JOIN product ON product.ProductId = purchaseproduct.ProductId
JOIN supplier ON supplier.SupplierId = product.SupplierId
GROUP BY supplier.SupplierId
```

EXISTS

All suppliers with a product which costs less than 20

```sql
SELECT SupplierName
FROM supplier
WHERE EXISTS
(
   SELECT ProductName FROM product
   WHERE SupplierId = supplier.supplierId AND Price < 20
)
```

JOIN a table with itself

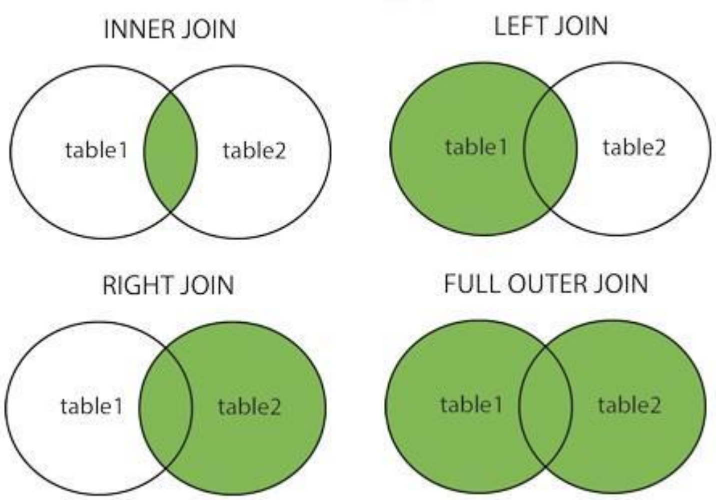
```sql
SELECT e1.FirstName, e1.LastName,
e2.FirstName AS BossFirstName, e2.LastName AS BossLastName
FROM employee AS e1
JOIN employee AS e2 ON e1.ReportsTo = e2.EmployeeID
```

INNER vs LEFT JOIN

NULL

- Values may be null, unless explicitly forbidden by the column

LEFT, RIGHT and OUTER Join



INNER JOIN is the default and in most cases what you want.

More Queries

- Sale volume by customer country?
- Sale volume by employee?
- Which customer buys most drinks?
- What has the biggest order in october 1996?
- Which region has most employees?
- Which product is best selling?

Which customer buys most drinks?

```
SELECT Customer.CompanyName, SUM(Quantity) AS drinks
 FROM PurchaseProduct AS pp
 JOIN Product ON Product.ProductId = pp.ProductId
 JOIN Purchase ON Purchase.PurchaseId = pp.PurchaseId
 JOIN Customer ON Customer.CustomerId = Purchase.CustomerId
 WHERE Product.CategoryId = 1
 GROUP BY Customer.CustomerId
 ORDER BY drinks DESC
```

Index

```
CREATE INDEX PurchaseDate ON Purchase (PurchaseDate);

CREATE INDEX ShippedDate ON Purchase (ShippedDate);

CREATE INDEX ShipPostalCode ON Purchase (ShipPostalCode);
```

Constraints

```
ALTER TABLE PurchaseProduct
ADD CONSTRAINT FK_Purchase_Details_Purchase
FOREIGN KEY (PurchaseID) REFERENCES Purchase (PurchaseID);

ALTER TABLE PurchaseProduct
ADD CONSTRAINT FK_Purchase_Details_Product
FOREIGN KEY (ProductID) REFERENCES Product (ProductID);
```

Database normalization

[Database normalization - Wikipedia]

Normal form

Columns can not be split further in a meaningful way

Every table models exactly one fact

SQL Injection

https://www.w3schools.com/sql/sql_injection.asp

xkcd: Exploits of a Mom

What can be done against that?

SQL Injection prevention

- Manual escaping of magic characters (dangerous)
- Prepared Statements:

```
PreparedStatement pstmt = con.prepareStatement(
    "UPDATE EMPLOYEES SET SALARY = ? WHERE ID = ?"
);
pstmt.setBigDecimal(1, 153833.00)
pstmt.setInt(2, 110592)
```

Money Transfer

Table: Bank Account

number	balance	dispo
55432	682.45	2000
12345	2985.30	0
90210	-20.84	1000
20018	722.98	2000

Steps for money transfer:

- Read balance and dispo for sender account
- Check, if amount is available (according to balance and dispo)
- Write new sender balance
- Write new receiver balance

Write SQL and pseudo code: Transfer of 500€ from 90210 to 55432

Money transfer - what could go wrong

Concurrency

- Read balance and dispo for sender account
- Check, if amount is available (according to balance and dispo)
- Write new sender balance
- Write new receiver balance

- Read balance and dispo for sender account
- Check, if amount is available (according to balance and dispo)
- Write new sender balance
- Write new receiver balance

Transactions

```
START TRANSACTION;

SELECT ... FOR SHARE;

UPDATE ...;

COMMIT;
```

Transactions

- Values read with **`FOR SHARE`** can not be changed outside of the transaction
- All **`UPDATE`**/**`INSERT`**/**`DELETE`**s are saved "at the same time"

ACID

What do the four letters mean in connection with MySQL or databases in general?

Bonus: There are also database systems that do not have these properties. Why?

What else can MySQL / MariaDB do?

- Replication
  - Leader/Follower (outdated terminology: Master/Slave)
  - Group
- Partitioning
- Access Control