



# MySQL Basics

SAE Wien

Alexander Hofbauer

hofbauer.alexander+sae@gmail.com

# SQL & MySQL

- + "Structured Query Language"
- + MySQL besteht aus
  - + Vorname der Tochter des Mitgründers Michael Widenius (My)
  - + Datenbanksprache SQL
- + seeeeeeehr hohe Verbreitung im Web Bereich

# SQL & MySQL

Datenbanksprache für

- + Definition
- + Abfrage
- + Datenmanipulation

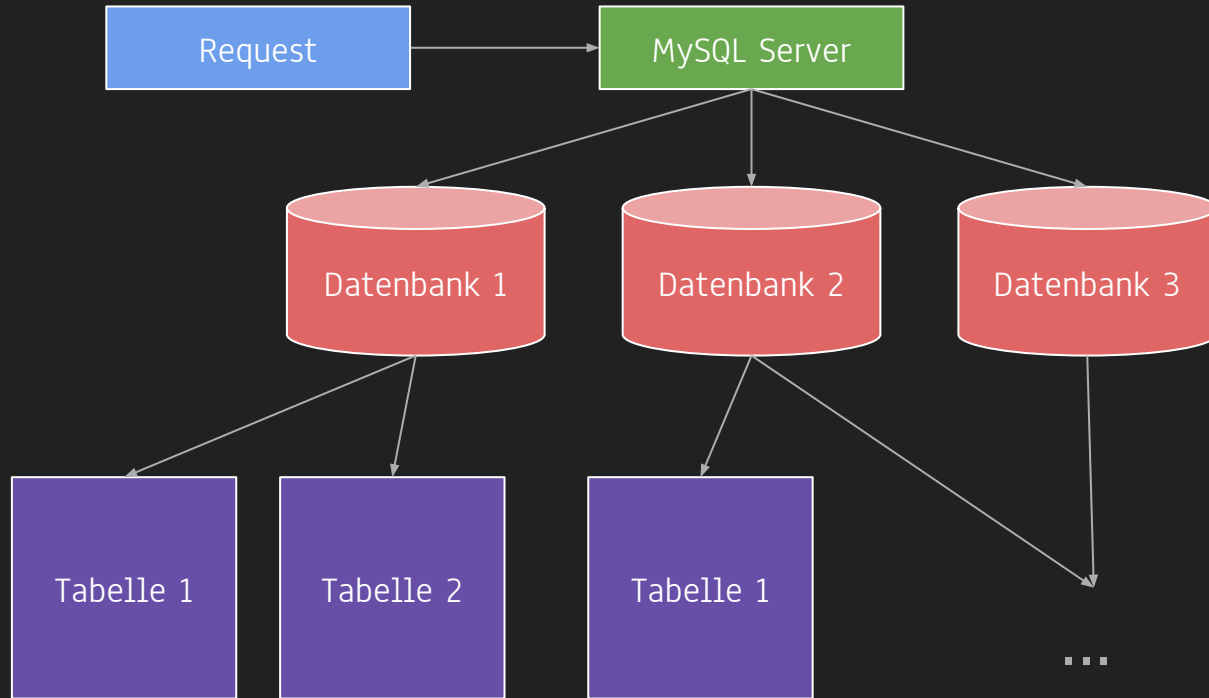
in relationalen Datenbanken

# SQL & MySQL

- + Syntax an natürliche englische Sprache angelehnt

```
USE importantdatabase;  
SELECT id,name FROM customers;
```

# Aufbau



# PhpMyAdmin

- + PHP Applikation zur Verwaltung von MySQL Datenbanken
- + Login über Browser
- + MAMP: [localhost:8888/phpmyadmin](http://localhost:8888/phpmyadmin)
- + XAMPP: [localhost/phpmyadmin](http://localhost/phpmyadmin)

# Syntax

# Datenbanken & Tabellen erstellen

```
-- Datenbank erzeugen
CREATE DATABASE movies;

-- Tabelle erzeugen
CREATE TABLE actors (
    ID integer,
    name varchar(20) not null
);
```



# Datensätze einfügen

```
-- Werte direkt setzen
INSERT INTO actors
    SET ID = '1', name = 'Johnny Depp';
```

```
-- mehrere Datensätze einfügen
INSERT INTO actors
    (ID, name)
VALUES
    ('2', 'Jennifer Aniston');
```

```
-- alle Werte pro Datensatz einfügen
insert into actors
values ('3', 'Johnny 5');
```

# Datensätze abfragen

```
-- alles aus actors abfragen
```

```
SELECT * FROM actors;
```

```
-- Spalten ID und name aus actors abfragen
```

```
SELECT ID, name from actors;
```

```
-- alles aus movies.actors abfragen
```

```
SELECT * FROM movies.actors;
```

```
-- alle Spalten aus actors abfragen, wo ID 1 ist
```

```
SELECT * FROM actors WHERE ID = '1';
```

# Datensätze abfragen

```
-- Verkettten: OR
SELECT * FROM actors WHERE
    ID = 2
OR
    name = 'Johnny 5';
```

```
-- Verkettten: AND
SELECT * FROM actors WHERE
    ID = '2'
AND
    name = 'Johnny 5';
```

# Datensätze abfragen - Limit

```
-- nur einen Datensatz abfragen
```

```
SELECT * FROM actors LIMIT 1;
```

```
-- ersten Treffer überspringen, 2 folgende zurückgeben
```

```
SELECT * FROM actors LIMIT 1, 2;
```

# Datensätze zählen

```
-- Anzahl  
SELECT count(*) FROM actors WHERE ID >= 2;  
  
-- Spalten in der Ausgabe umbenennen  
SELECT count(*) AS number FROM actors;
```

# Datensätze löschen

```
-- einen Eintrag löschen  
DELETE FROM actors WHERE ID = 3;  
  
-- alle Einträge einer Tabelle löschen  
DELETE FROM actors;
```

# Datensätze aktualisieren

```
UPDATE actors SET  
    name = 'WALL-E'  
WHERE  
    ID = 3;
```

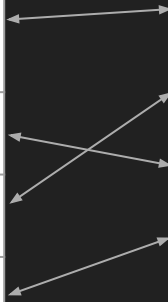
-- ACHTUNG: ohne WHERE-Clause werden alle Datensätze aktualisiert

# Relationen

## Beziehungen zwischen Tabellen

Movies	
id	name
1	A Nightmare On Elm Street
2	Wall-E
3	Friends
4	Short Circuit

Actors	
id	name
1	Johnny Depp
2	Jennifer Aniston
3	Wall-E
4	Johnny 5





# Relationen

```
SELECT actors.name FROM movies, movies_actors_mm, actors
WHERE actors.id = movies_actors_mm.actor_id
AND movies_actors_mm.movie_id = movies.id
AND movies.name = 'Friends';
```

movies	
id	name
1	A Nightmare On Elm Street
2	Wall-E
3	Friends
4	Short Circuit

movies_actors_mm	
movie_id	actor_id
1	1
2	3
3	2
4	4

actors	
id	name
1	Johnny Depp
2	Jennifer Aniston
3	Wall-E
4	Johnny 5

# Primärschlüssel

- + Eindeutige **Identifizierung** eines Datensatzes
  - + Pro Tabelle muss ein Primärschlüssel "unique" sein
- + automatisch vergebene IDs eignen sich sehr gut (**auto\_increment**)
  - + ID wird beim Insert nicht angegeben, da sie automatisch gesetzt wird
- + **auto\_increment** ist nur auf Primärschlüssel anwendbar
- + pro Tabelle kann nur eine Spalte Primary Key sein

# Fremdschlüssel

- + Datensätze aus anderen Tabellen identifizieren, um Zuordnungen modellieren zu können

Movies	
id	name
1	A Nightmare On Elm Street
2	Wall-E
3	Friends
4	Short Circuit

Movies_actors_mm	
movie_id	actor_id
1	1
2	3
3	2
4	4

Actors	
id	name
1	Johnny Depp
2	Jennifer Aniston
3	Wall-E
4	Johnny 5

# Gruppierung

- + Datensätze anhand bestimmter Kriterien gruppieren

```
SELECT name, sum(hours) AS summe FROM lectures  
GROUP BY (name);
```

lectures		
name	lecture	hours
Alex	PHP Basics	5
Alex	PHP OOP	5
Fabio	WS XY	8
Jeanne	Introduction	2.5

Result	
name	summe
Alex	10
Fabio	8
Jeanne	2.5

# Datensätze ordnen

```
-- Aufsteigend ordnen (kleinster Wert zuerst) <-- default
SELECT * FROM movies
ORDER BY movies.year ASC

-- Absteigend ordnen (größter Wert zuerst)
SELECT * FROM movies
ORDER BY movies.year DESC
```

# LIKE

```
-- % = beliebige Anzahl an Zeichen  
-- _ = genau ein Zeichen
```

```
SELECT * FROM actors WHERE name LIKE 'John%';
```

```
SELECT * FROM actors WHERE name LIKE 'Jen% An%';
```

```
SELECT * FROM albums WHERE name LIKE '%a1%'  
AND year like '199_';
```

# Copyright

Referenzen: <http://php.net>

Autor: [Alexander Hofbauer](mailto:hofbauer.alexander+sae@gmail.com) <hofbauer.alexander+sae@gmail.com>