Introduction to Databases
Part 1

jan.schulz@devugees.org

Agenda

1. Database Definition

Database:

A <u>**place**</u> to store data.

What is a place?

1. Database Definition

Database:

A <u>**place**</u> to store data.

What is a place?

1. a file
2. or a block in your RAM

1. Database Definition

Database:

A **place** to store data.

What is a place?

1. a file
2. or a block in your RAM
3. or **both**

2. Types of Databases

- Save data Primarily as **<u>files</u>**
  - MySQL
  - Oracle
  - Postgres
  - MongoDB
  - CouchDB
- Save data primarily as blocks in **RAM**
  - Redis

2. Types of Databases

- Save data Primarily as **files**

= PERSISTENT DATABASES

- Save data primarily as blocks in **RAM**

= IN-MEMORY DATABASES

2. Database Types

- What types of data?

2. Database Types

- What types of data?
- Primitives
  - Strings
  - Numbers
  - Booleans
  - Nulls
- Non-Primitives
  - Objects

2. Database Types

- Databases can be categorized in how they save **Objects**

2. Database Types

- Databases can be categorized in how they save **Objects**
- **Document-Oriented Databases** save them as **JSON**:

```
var persons = [
    {firstname: ‘Andreas‘, lastname: ‘Schmidt‘, age: 32},
    {firstname: ‘Manfred‘, lastname: ‘Mustermann‘, age: 30},
    {firstname: ‘Julia‘, lastname: ‘Müller‘, age: 25},
];
```

2. Database Types

- Databases can be categorized in how they save **Objects**
- **Relational Databases** save them as **Tables**

Firstname	**Lastname**	**Age**
Andras	Schmidt	32
Manfred	Mustermann	30
Julia	Müller	25

2. Database Types

- Relational Databases
  - MySQL
  - Oracle
  - Postgres
- Document Oriented Databases
  - MongoDB
  - CouchDB

2. Database Types

- Relational Databases
  - MySQL
  - Oracle
  - Postgres
- Document Oriented Databases
  - MongoDB
  - CouchDB
- **(** Key-Value Databases **)**
  - Redis

3. Relational Databases

- Store information in tables
- **Relational** = two or more tables can **relate** to each other
- SQL = Structured Query Language
  - **<u>C</u>**reates, **<u>R</u>**eads, **<u>U</u>**pdates and **<u>D</u>**eletes data = **CRUD**
  - Query = A command to the database

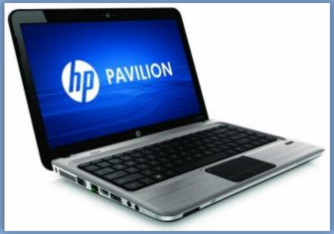
4. Our Online-Shop

ONLINE-SHOP.COM

Register | Login

PCs | /Laptops | Macs

450


600

320

350

550

650

450

700

4. Our Online-Shop

- Install MySQL-Server
  - **$ sudo apt-get install mysql-server**
- Install MySQL-Workbench
  - **$ sudo apt-get install mysql-workbench**
- Create new database “online-shop“
- Import Online-Shop database
  - **$ mysql –uroot –ppassword < online-shop.sql**

4. Our Online-Shop

- When a customer registers, the shop creates a new entry in the table
  - **customers**

4. Our Online-Shop

- When a customer registers, the shop creates a new entry in the table
  - **customers**
- When a customer buys something, the shop creates a new entry in the tables
  - **orders**
  - **order_details**

5. SELECT

```
SELECT
    firstname, lastname, city
FROM
    customers
```

5. SELECT

```
SELECT
    firstname, lastname, city
FROM
    customers
ORDER BY
    city ASC
```

5. SELECT

```
SELECT
    firstname, lastname, city
FROM
   customers
ORDER BY
   city ASC
LIMIT
   0, 5
```

5. SELECT aggregation

```
SELECT
    count(*)
FROM
   customers
```

5. SELECT aggregation

```
SELECT
    count(*)
FROM
    customers
GROUP BY
    city
```

5. SELECT with JOINS

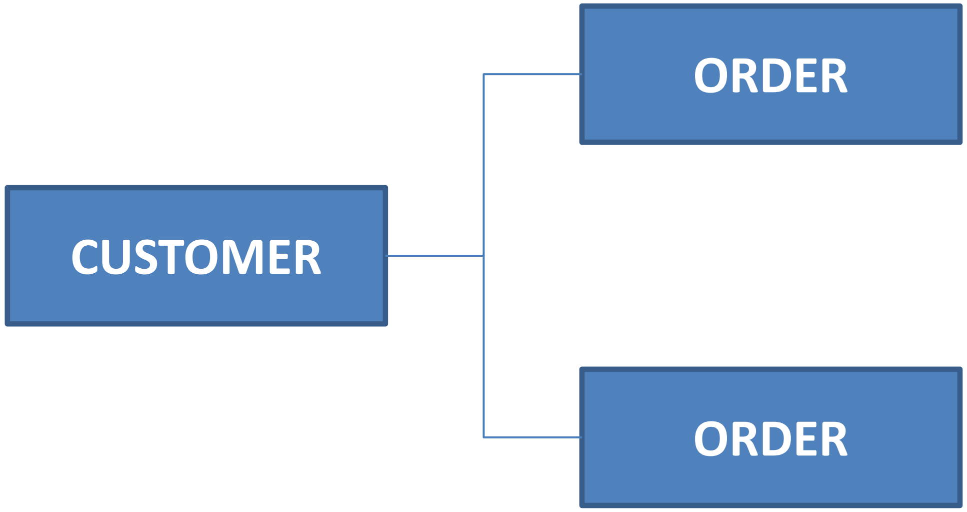
- When a customer registers, the shop creates a new entry in the table
  - **customers**
- When a customer buys something, the shop creates a new entry in the tables
  - **orders**
  - **order_details**

6. SELECT with JOINS

- One customer …

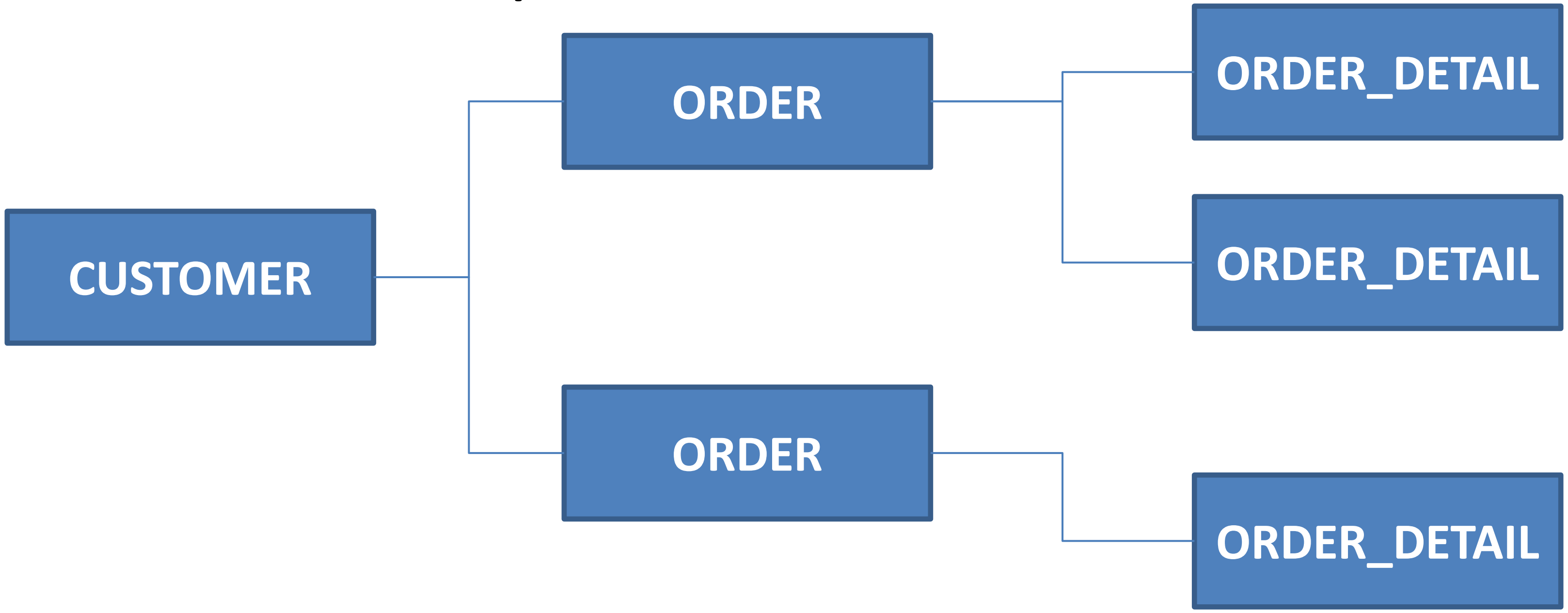
6. SELECT with JOINS

- One customer ... Can have multiple orders



6. SELECT with JOINS

- One customer … Can have multiple orders
  … With multiple order details

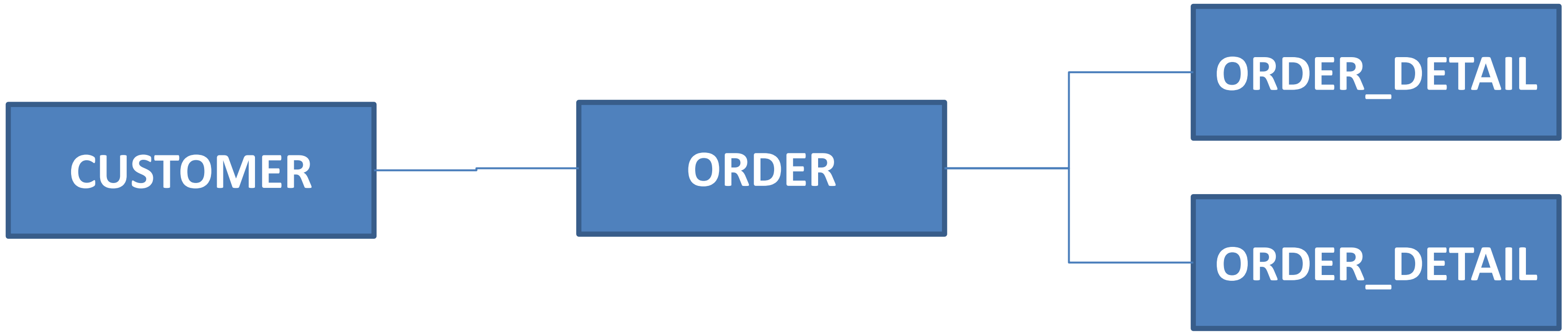


6. SELECT with JOINS

- Examples?

6. SELECT with JOINS

- Examples?
  - Franz Mueller buys two Laptops. One HP and one Sony Laptop.



Franz Mueller   Makes one Order   With two laptops

6. SELECT with JOINS

```
SELECT
   c.*, o.*
FROM
   customers c
JOIN
   orders o
ON
   c.id = o.customer_id
```

7. Types of JOINs

- Join (aka inner join)
  - If a match exists, show both tables
- Left Join
  - If a match exists, show both tables
  - Otherwise, show left table anyway
- Right Join (= mirrored Left Join)
  - If a match exists, show both tables
  - Otherwise, show right table anyway

8. UNION

- UNION concatenates two tables, if they have the same fields

```
SELECT ‘hallo‘ as x UNION ‘world‘ as x;
```

9. Virtual Tables

- A virtual table is a table that does not exist on the persistent memory of the SQL-database

```
Select x from
   (SELECT ‘hallo‘ as x UNION ‘world‘ as x) as vt;
```

-> Virtual Tables need aliases e.g. a name

10. SELECT CASE

- SELECT (CASE WHEN 1+2=3 THEN ‘it is 3‘ ELSE 0 END) AS SOMETHING
- SELECT (CASE WHEN CITY=‘Berlin‘ THEN ‘Ein Berliner‘ ELSE ‘Kein Berliner!‘ END) AS Berliner

11. Backup Tables

- Backup Tables are made before new changes to production tables will occur

```
Create table myOrders as select * from orders;
```

11. Backup Tables

- Backup Tables are made before new changes to production tables will occur

Create table backupOrders as select * from orders;

Rollback:

Insert into orders(select * from backupOrders);

12. Update

- Update orders set paid = now() where customer_id = ‘19‘;

-> What would that do?

12. Update

- Update orders set paid = now() where customer_id = ‘19‘;

-> What would that do?

User 19 has paid all its orders.

13. Creating tables

- What is the table‘s purpose?
- What fields do we need?
- What primary/foreign keys do we need?

13. Creating tables

Task:

We want to introduce certain payment methods for our shop.

13. Design Databases

<u>**Task:**</u>

We want to introduce certain payment methods for our shop.

-> Please draw a ER-diagram of the existing shop and find a way to introduce payment methods.

13. Design Databases

Task:

How would you design your own blog – database wise? Please draw an ER-diagram and implement it by creating a new database ‘myblog‘ on your local MySQL-database. Fill in some example data.

Requirements:
- Users can login
- Users can be administrators or regular users
- Regular users can be activated/deactivated/banned by Administrators
- All users can post
- Posts can be categorized
- Posts and Categorys can be disabled/enabled
- Guests can leave a guestbook entry
- Administrators can delete guestbook entries

Rule 1: Everything is tracked by the database.