Introduction to Databases
Part 1

jan.schulz@devugees.org

Agenda

1. Database Definition
2. Database Types
3. Relational Databases
4. Our Online-Shop
5. SELECT
6. SELECT with aggregation
7. SELECT with JOINs

1. Database Definition

Database:

A **place** to store data.

What is a place?

1. Database Definition

Database:

A **place** 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 **files**
  - 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
  - **C**reates, **R**eads, **U**pdates and **D**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

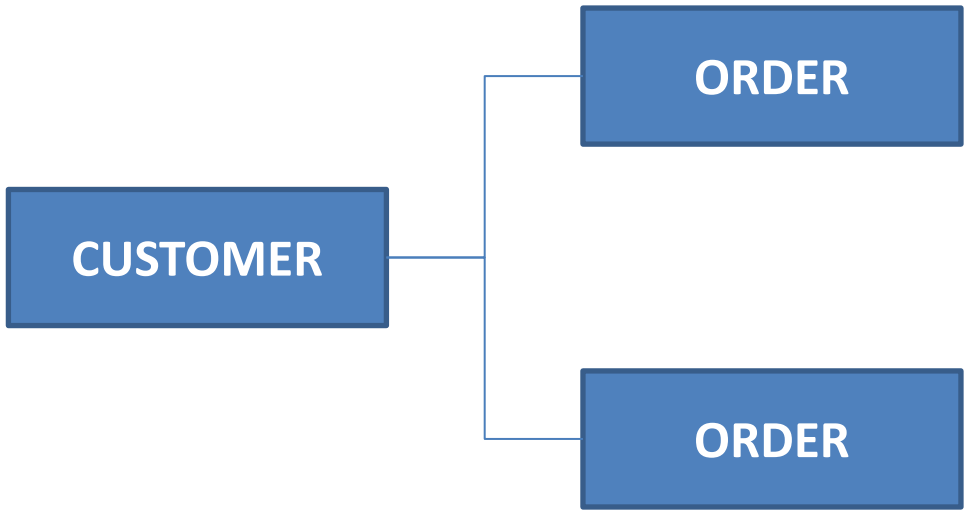
- When a customer registers, the shop creates a new entry in the table
  - **customers**
- When a customer buys something, the shop creates a new entry in the tables
  - **orders**
  - **order_details**

6. SELECT with JOINS

- One customer ...

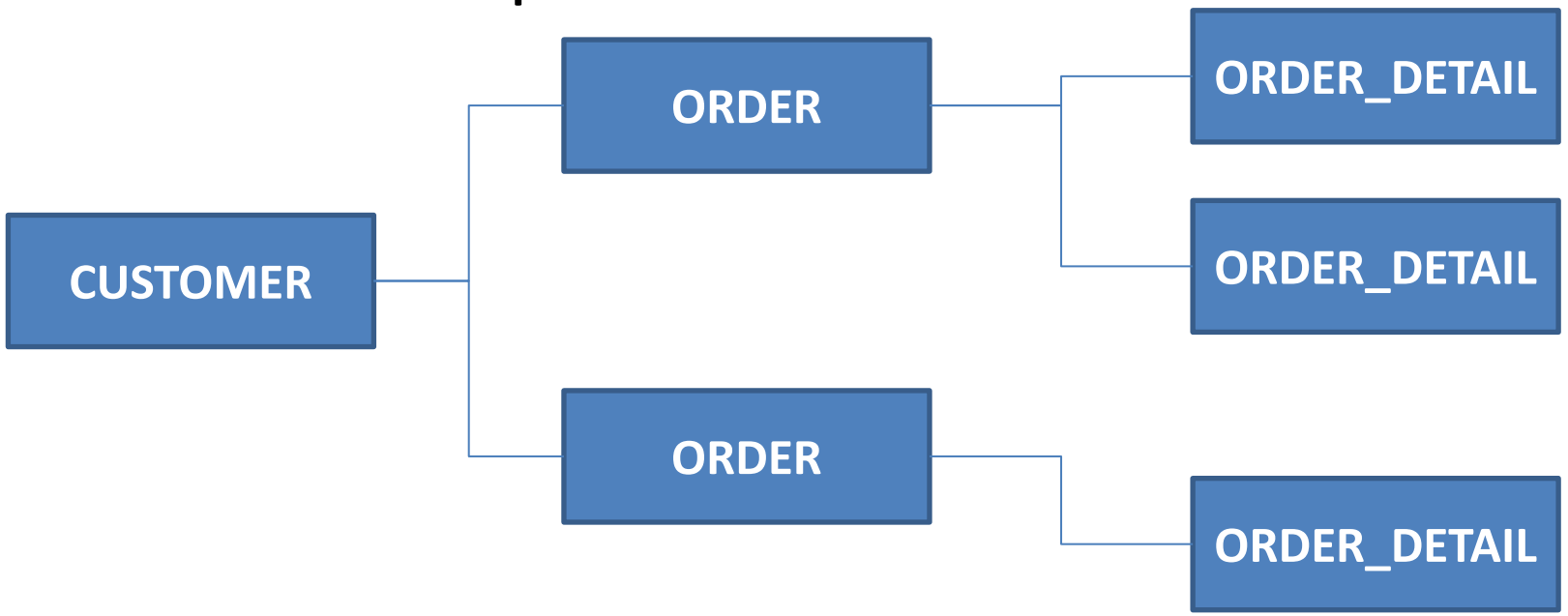
6. SELECT with JOINS

- One customer ... Can have multiple orders



6. SELECT with JOINS

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

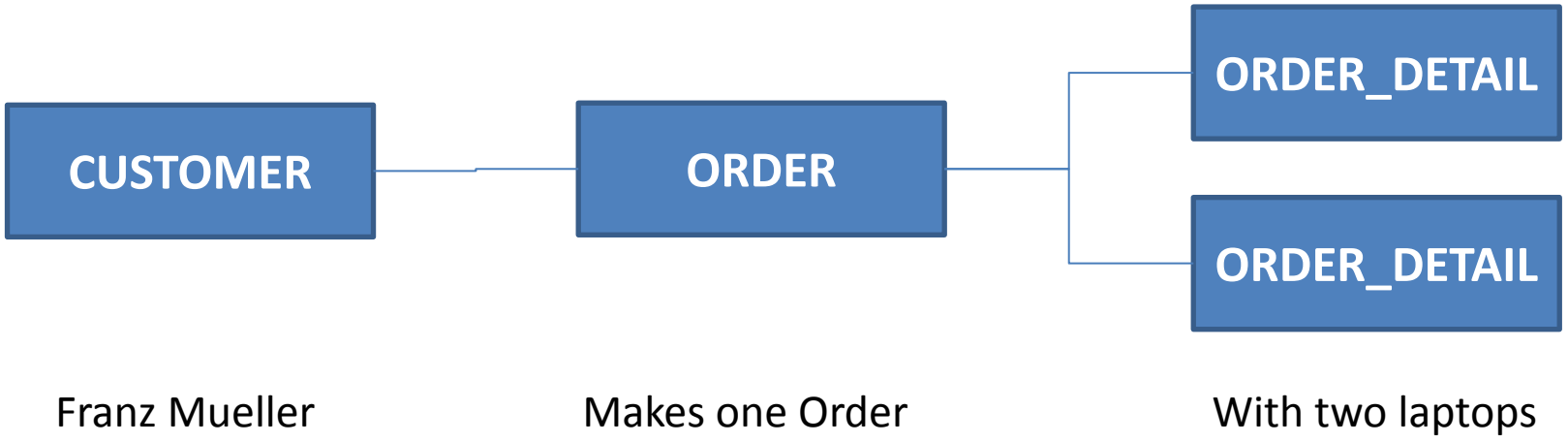


6. SELECT with JOINS

- Examples?

6. SELECT with JOINS

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



6. SELECT with JOINS

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