

# Databases

## Relational and Non-Relational Databases and MySQL



**SoftUni Team**  
**Technical Trainers**



**SoftUni**

**Software University**

<https://about.softuni.bg>

1. **Databases** – Introduction
2. **Relational** Databases: Tables and Relationships
3. **Non-Relational** (NoSQL) Databases
4. Database Management Systems (**DBMS**)
5. **SQL** Commands: SELECT, INSERT, UPDATE, DELETE
6. **JSON** Data Format
7. Working with **MySQL** + **Workbench**
8. Working with **MongoDB** + **Compass**



# Have a Question?



**sli.do**

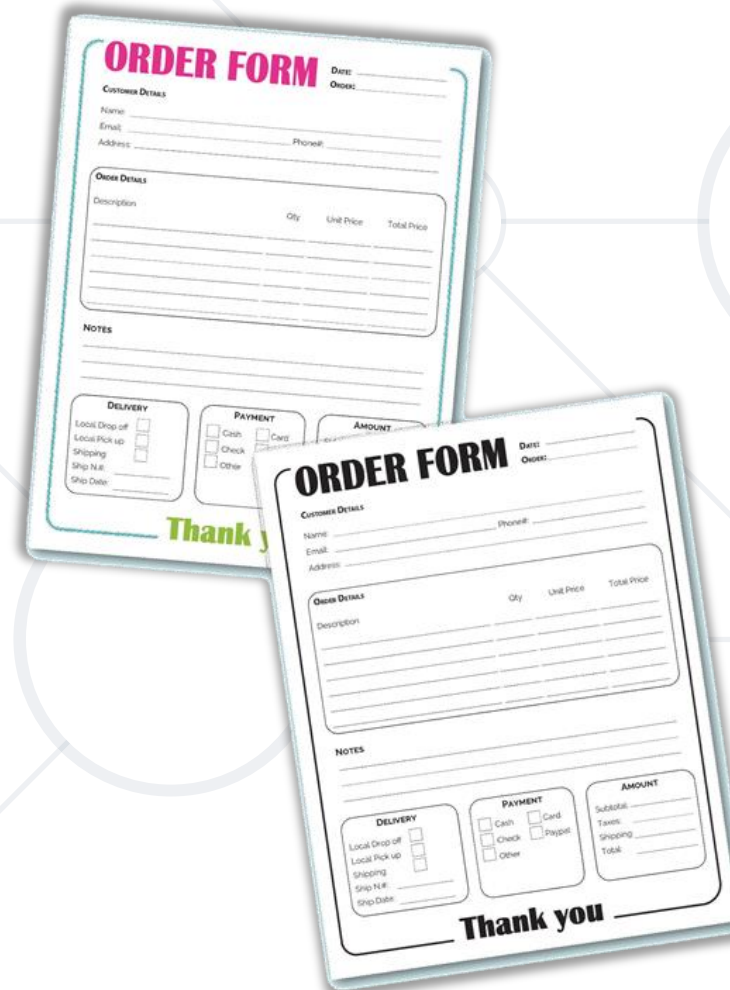
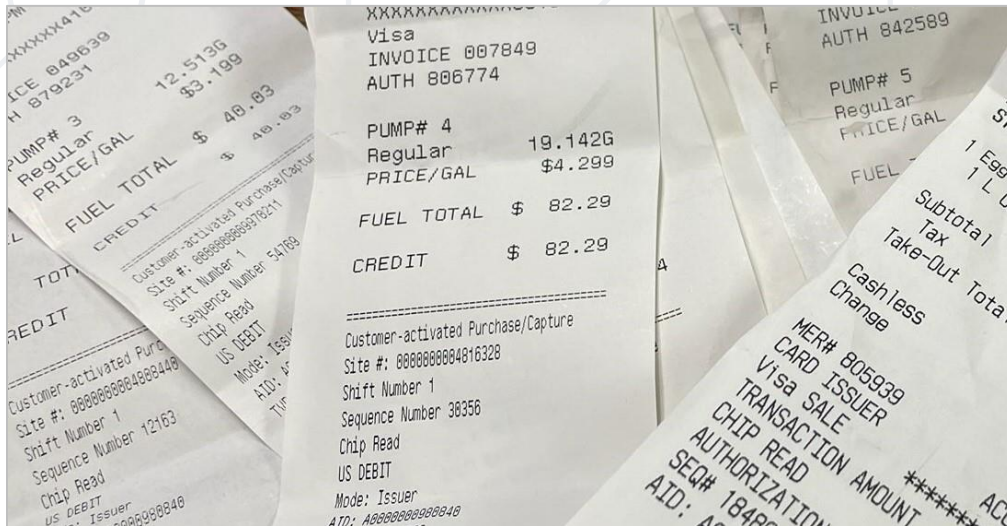
**#qa-fund**



# Databases

## Introduction

- Conventional (paper-based) data storage
  - Orders
  - Receipts



# From Data Storage to Databases

- We can group related pieces of data into separate columns:



ITEM NO.	DESCRIPTION	QUANTITY	UNIT PRICE	TOTAL
WM2536	Women's Tall - M	10	\$10.00	\$100.00
GH2403	Men's Tall - M	5	\$20.00	\$100.00
NC2267	Children's - S	10	\$5.00	\$50.00
YX6278	Men's - XL	5	\$10.00	\$50.00

- Thus, we keep data in **tables** (like in Excel)
- Tables may be **related** (e. g. Products and Orders)

# Why Do We Need Databases?

- **Data storage and processing** is a **common need** in the tech industry
- Data storage needs:
  - Ease of searching
  - Ease of updating
  - Performance
  - Accuracy and consistency
  - Security and access control
  - Redundancy





# What is a Database?

- A **database** is a collection of data, organized to be easily accessed, managed and updated
- Modern databases are managed by **Database Management Systems** (DBMS)
  - Define database **structure**, e. g. tables, collections, columns, relations, indexes
  - **Create / Read / Update / Delete** data (**CRUD** operations)
  - Execute **queries** (filter / search data)





- **Databases** hold and manage data in the back-end systems
- **Relational databases (RDBMS)**
  - Hold data in **tables + relationships**
  - Use the **SQL** language to query / modify data
  - Examples: MySQL, PostgreSQL, Web SQL in HTML5
- **NoSQL databases**
  - Hold **collections** of documents or key-value pairs
  - Examples: MongoDB, IndexedDB in HTML5





# Relational Databases

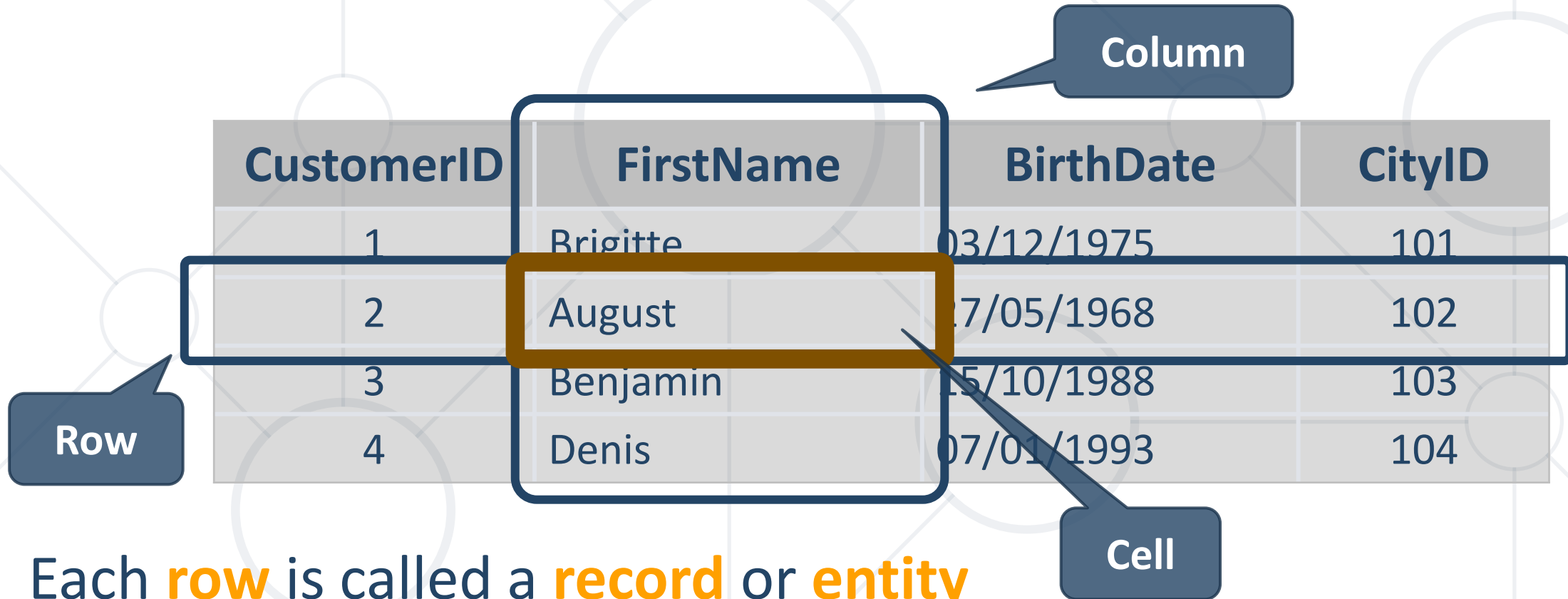
Tables, Relationships and SQL

# SQL Databases (Relational Databases)

- Relational (**SQL**) databases organize data in **tables**
  - Tables have strict structure (**columns** of certain **data types**)
  - Can have **relationships** to other tables
- Relational databases use the **structured query language (SQL)** for defining and manipulating data
  - Extremely powerful for complex queries
- **Relational databases** are the most widely used data management technology



- **Tables** are the main **building block** of relational databases



CustomerID	FirstName	BirthDate	CityID
1	Brigitte	03/12/1975	101
2	August	17/05/1968	102
3	Benjamin	15/10/1988	103
4	Denis	07/01/1993	104

- Each **row** is called a **record** or **entity**
- Columns (**fields**) define the **type** of data they contain

# The Relational DB Model

- Relational data is stored into one or more **tables** with a **unique key** identifying each row and **foreign keys** defining **relationships**

Items

ID	Order ID	Name	Quantity	Price
5	1	Table	1	200.00
6	1	Chair	1	123.12

Customers

ID	Name	Email
5	Peter	<a href="mailto:peter@gmail.com">peter@gmail.com</a>
6	Jayne	<a href="mailto:jayne@gmail.com">jayne@gmail.com</a>

Orders

ID	Customer ID	Date	Total Price
1	5	11/1/17	323.12
2	1	11/15/17	13.99





# **Non-Relational Databases**

NoSQL Databases and JSON Documents

# NoSQL (Non-Relational) Databases

- A **NoSQL** databases have dynamic schema for **unstructured** data
- Data may be stored in several ways:
  - **Document-oriented** (JSON store)
  - **Column-oriented** (table store)
  - **Graph-based**
  - **Key-value store**





- **NoSQL databases** don't use tables and SQL
  - Instead, use **document collections** or **key-value pairs**
- More **scalable** and **high performance**
- Examples: **MongoDB**, **Cassandra**, **Redis**, etc.

Example of  
JSON document  
in MongoDB

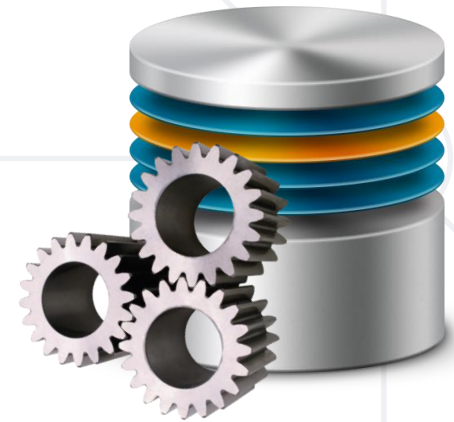
```
{  
  "_id": ObjectId("59d3fe7ed81452db0933a871"),  
  "email": "peter@gmail.com",  
  "age": 22  
}
```



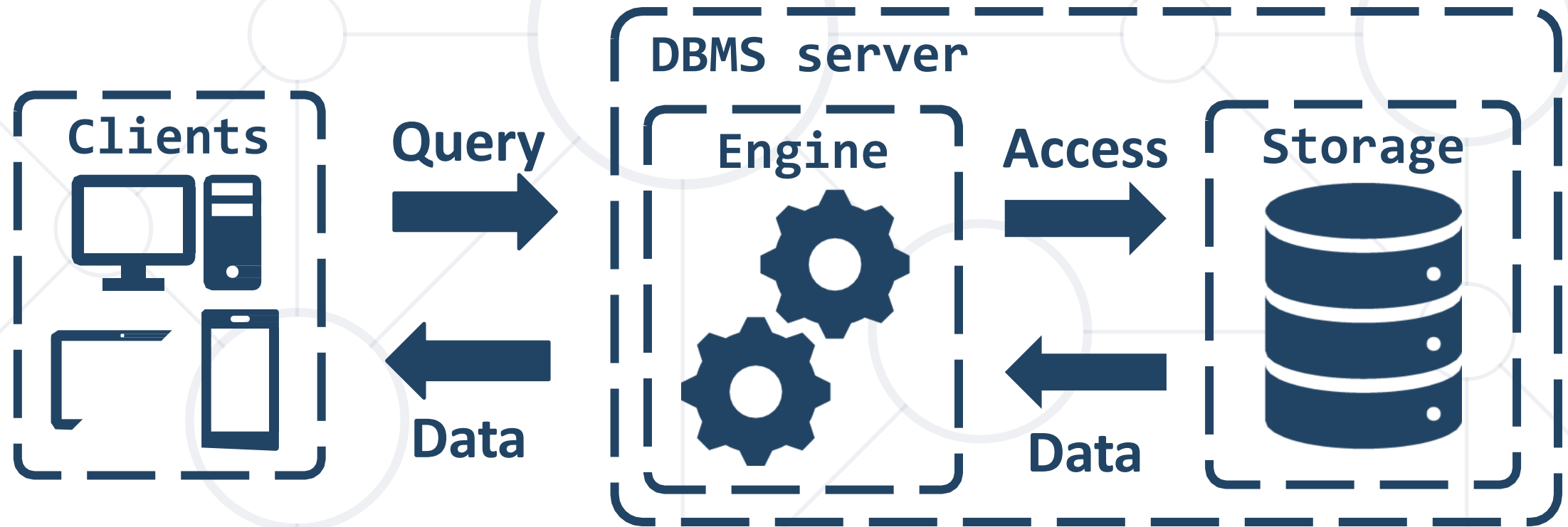
# Database Management Systems

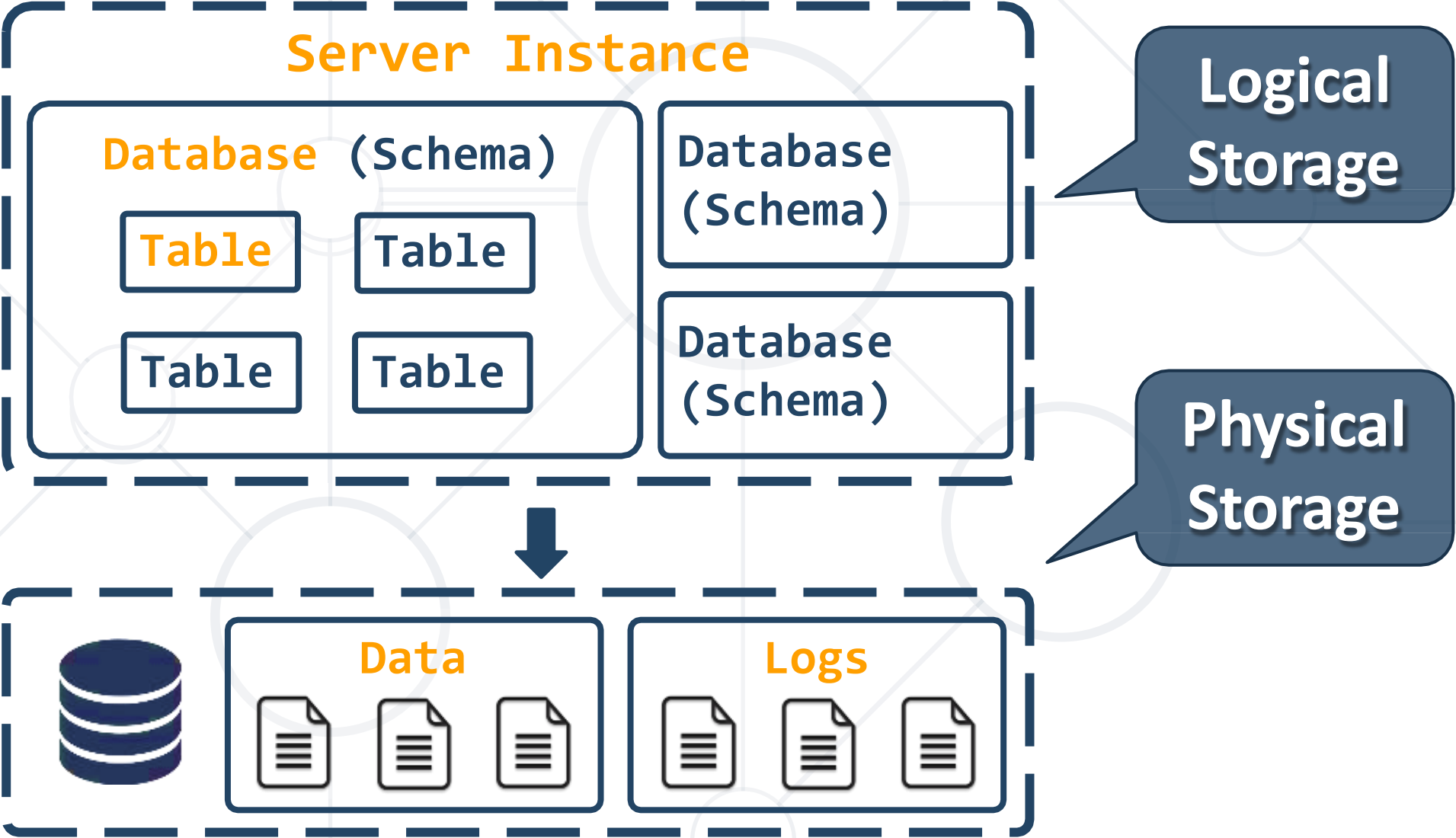
Database Servers

- A **Database Management System (DBMS)** is a software, used to **define, manipulate, retrieve** and **manage** data in a database
  - DBMS **stores and manages** the data itself, the data format, field names and data types, record structure and file structure
- **DBMS examples** (database servers):
  - MySQL, MS SQL Server, Oracle, PostgreSQL
  - MongoDB, Cassandra, Redis, HBase
  - Amazon DynamoDB, Azure Cosmos DB



- DBMS servers use the **client-server model**:





## ■ SQL databases examples:

- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- SQLite and Web SQL

## ■ NoSQL databases examples:

- MongoDB
- Redis
- Google BigTable
- Amazon DynamoDB
- Azure Cosmos DB
- Cassandra





# Structured Query Language

SQL Language



# Structured Query Language (SQL)

- **SQL** == query language designed for managing data in **relational** databases (RDBMS)
  - Used to communicate with the database engine
- Logically, SQL is divided into four sections:
  - **Data definition**: describe the **structure** of data
  - **Data manipulation**: **store** and **retrieve data**
  - **Data control**: define who can **access the data**
  - **Transaction control**: bundle **operations** together and perform **commit** / **rollback**



# Structured Query Language (1)

- Query language designed for managing data in a relational database
- Developed at **IBM** in the early 1970s
- To communicate with the DB engine we use **SQL**



# Structured Query Language (2)

- Subdivided into several language elements

- Queries
- Clauses
- Expressions
- Predicates
- Statements



- Logically divided in four sections
  - **Data Definition** – describe the structure of our data
  - **Data Manipulation** – store and retrieve data
  - **Data Control** – define who can access the data
  - **Transaction Control** – bundle operations and allow rollback

- We can **communicate** with the database engine via **SQL**
- **SQL commands** provide greater **control** and **flexibility**
- To create a database in MySQL:

```
CREATE DATABASE employees
```

Database  
name

- Display all databases in MySQL:

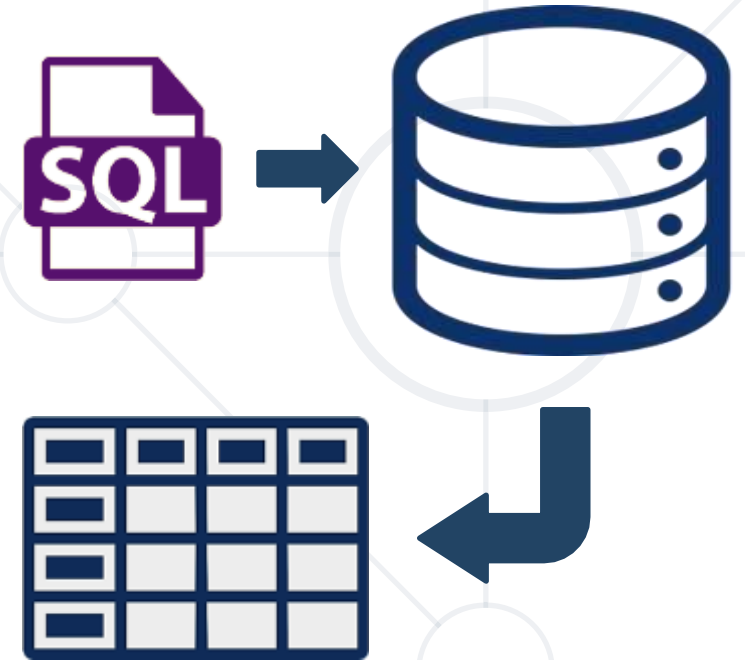
```
SHOW DATABASES
```

- Example of **SQL** query:

```
SELECT * FROM people
```

- The query is executed by the **DBMS** system
  - It returns a sequence of **data rows**, e.g.

id	email	first_name	last_name
1	<a href="mailto:smith@yahoo.co.uk">smith@yahoo.co.uk</a>	John	Smith
2	<a href="mailto:pwh@gmail.com">pwh@gmail.com</a>	Peter	White
3	<a href="mailto:anne@anne.com">anne@anne.com</a>	Anne	Green
4	<a href="mailto:jason.jj@gmail.com">jason.jj@gmail.com</a>	Jason	Anderson



# SELECT – Example

- Selecting **all** columns from the "Departments" table

```
SELECT * FROM Departments
```

DepartmentID	Name	ManagerID
1	Engineering	12
2	Tool design	4
3	Sales	273
...	...	...

- Selecting **specific** columns

```
SELECT DepartmentId, Name  
FROM Departments
```



DepartmentID	Name
1	Engineering
2	Tool design
3	Sales
...	...



```
SELECT FirstName, LastName, JobTitle FROM Employees
```

```
SELECT * FROM Projects WHERE StartDate = '1/1/2006'
```

```
INSERT INTO Projects(Name, StartDate)  
VALUES('Introduction to SQL Course', '1/1/2006')
```

```
UPDATE Projects  
    SET EndDate = '8/31/2006'  
    WHERE StartDate = '1/1/2006'
```

```
DELETE FROM Projects  
    WHERE StartDate = '1/1/2006'
```

# Filtering the Selected Rows

- Use **DISTINCT** to eliminate **duplicate** results

```
SELECT DISTINCT DepartmentID  
FROM Employees
```

- Filter rows by specific **conditions** using the **WHERE** clause

```
SELECT LastName, DepartmentID  
FROM Employees  
WHERE DepartmentID = 1
```

- Other **logical operators** can be used for greater control

```
SELECT LastName, Salary FROM Employees  
WHERE Salary <= 20000
```

# Sorting Result Sets

- Sort rows with the **ORDER BY** clause
  - ASC**: ascending order, default
  - DESC**: descending order

```
SELECT LastName, HireDate
FROM Employees
ORDER BY HireDate
```

```
SELECT LastName, HireDate
FROM Employees
ORDER BY HireDate DESC
```



LastName	HireDate
Gilbert	1998-07-31
Brown	1999-02-26
Tamburello	1999-12-12
...	...

LastName	HireDate
Valdez	2005-07-01
Tsoflias	2005-07-01
Abbas	2005-04-15
...	...



# MySQL

## Working with Relational Database

- **Open-source** relational database management system
- Used in many **large-scale websites** like including Google, Facebook, YouTube etc.
- Works on **many** system platforms – macOS, Windows, Linux
- Download **MySQL Community Server**
  - **Windows:** <https://dev.mysql.com/downloads/mysql/>
  - **Ubuntu/Debian:** <https://dev.mysql.com/downloads/repo/apt/>



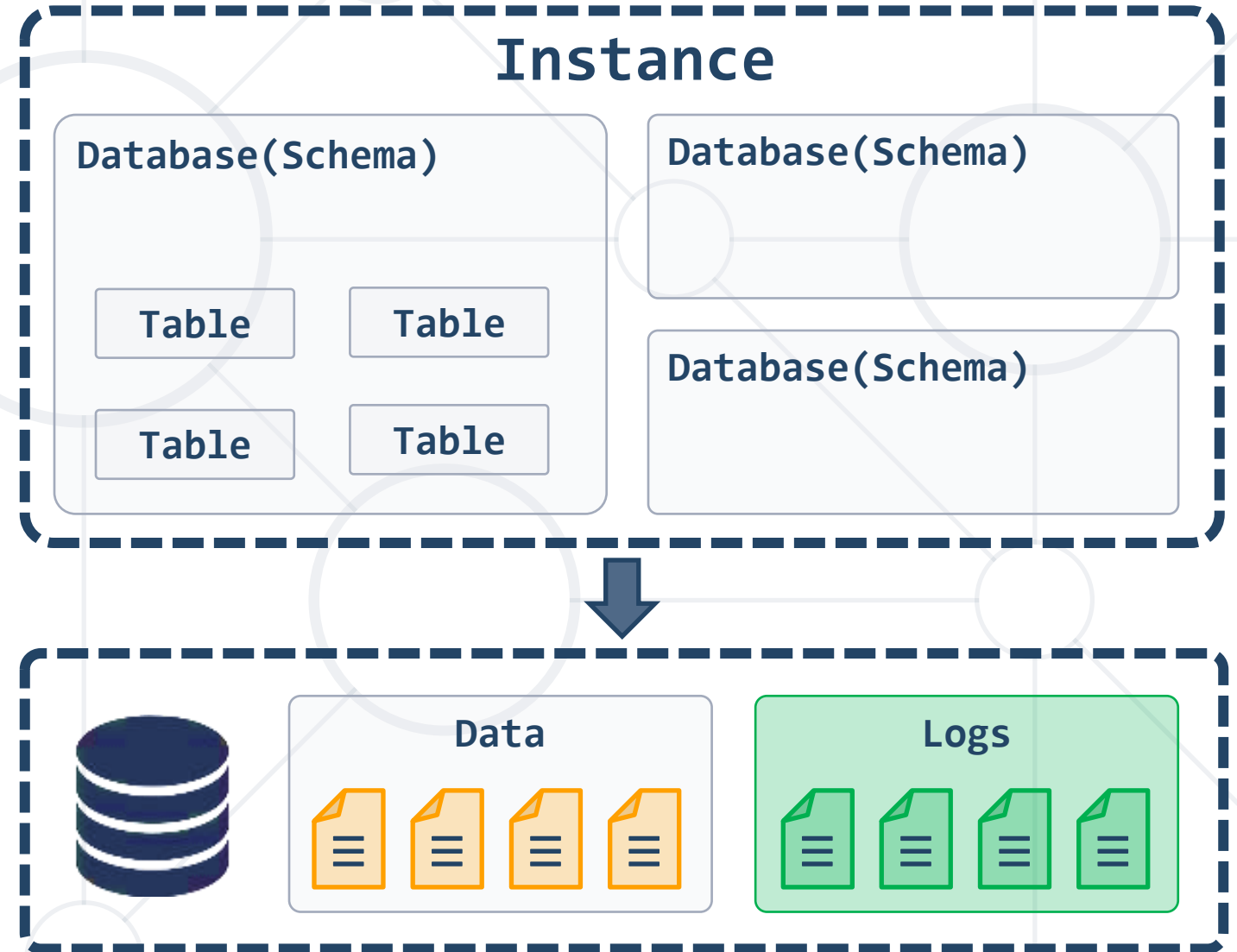
# MySQL Server Architecture

- **Logical Storage**

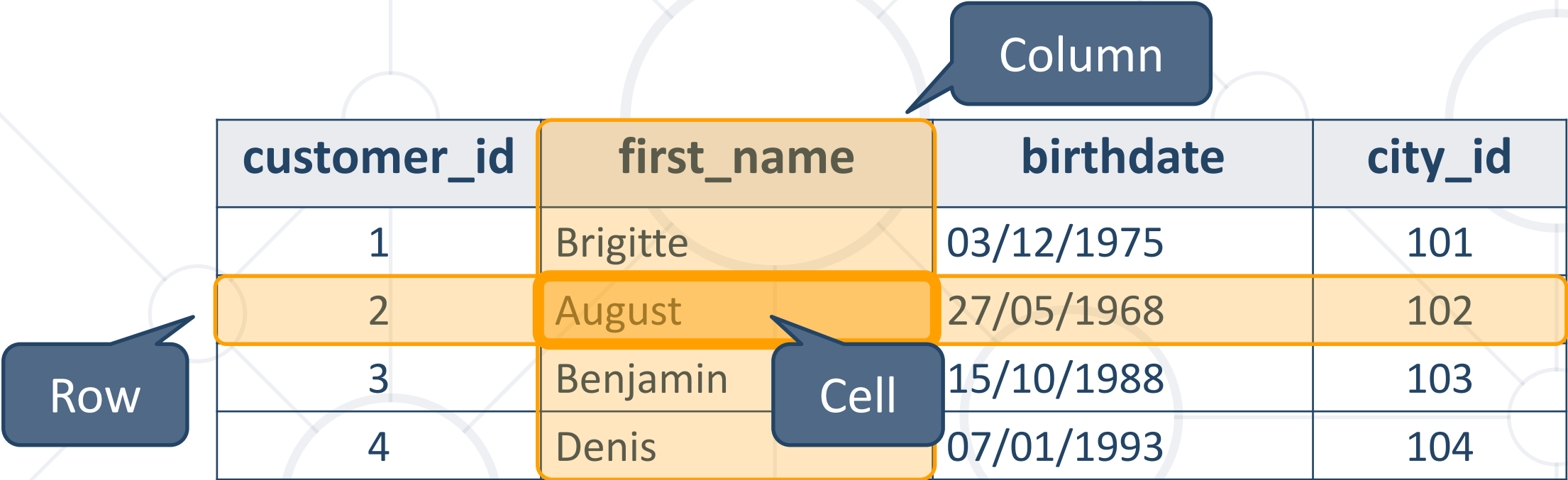
- Instance
- Database/Schema
- Table

- **Physical Storage**

- Data files and Log files
- Data pages



- The table is the main **building block** of any database



customer_id	first_name	birthdate	city_id
1	Brigitte	03/12/1975	101
2	August	27/05/1968	102
3	Benjamin	15/10/1988	103
4	Denis	07/01/1993	104

- Each **row** is called a **record** or **entity**
- Columns (**fields**) define the **type** of data they contain



# Why Split Related Data?

Empty records

first	last	registered	email	email2
David	Rivers	05/02/2016	drivers@mail.cx	NULL
Sarah	Thorne	07/17/2016	sarah@mail.cx	NULL
Michael	Walters	11/23/2015	walters_michael@mail.cx	walters_michael@abv.bg

Redundant information

		customer	product	s/n	price
00315	07/16/2016	David Rivers	Oil Pump	OP147-0623	69.90
00315	07/16/2016	David Rivers	Accessory Belt	AB544-1648	149.99
00316	07/17/2016	Sarah Thorne	Wiper Fluid	WF000-0001	99.90
00317	07/18/2016	Michael Walters	Oil Pump	OP147-0623	69.90

- We split the data and introduce **relationships** between the tables to **avoid** repeating information

user_id	first	last	registered
203	David	Rivers	05/02/2016
204	Sarah	Thorne	07/17/2016
205	Michael	Walters	11/23/2015

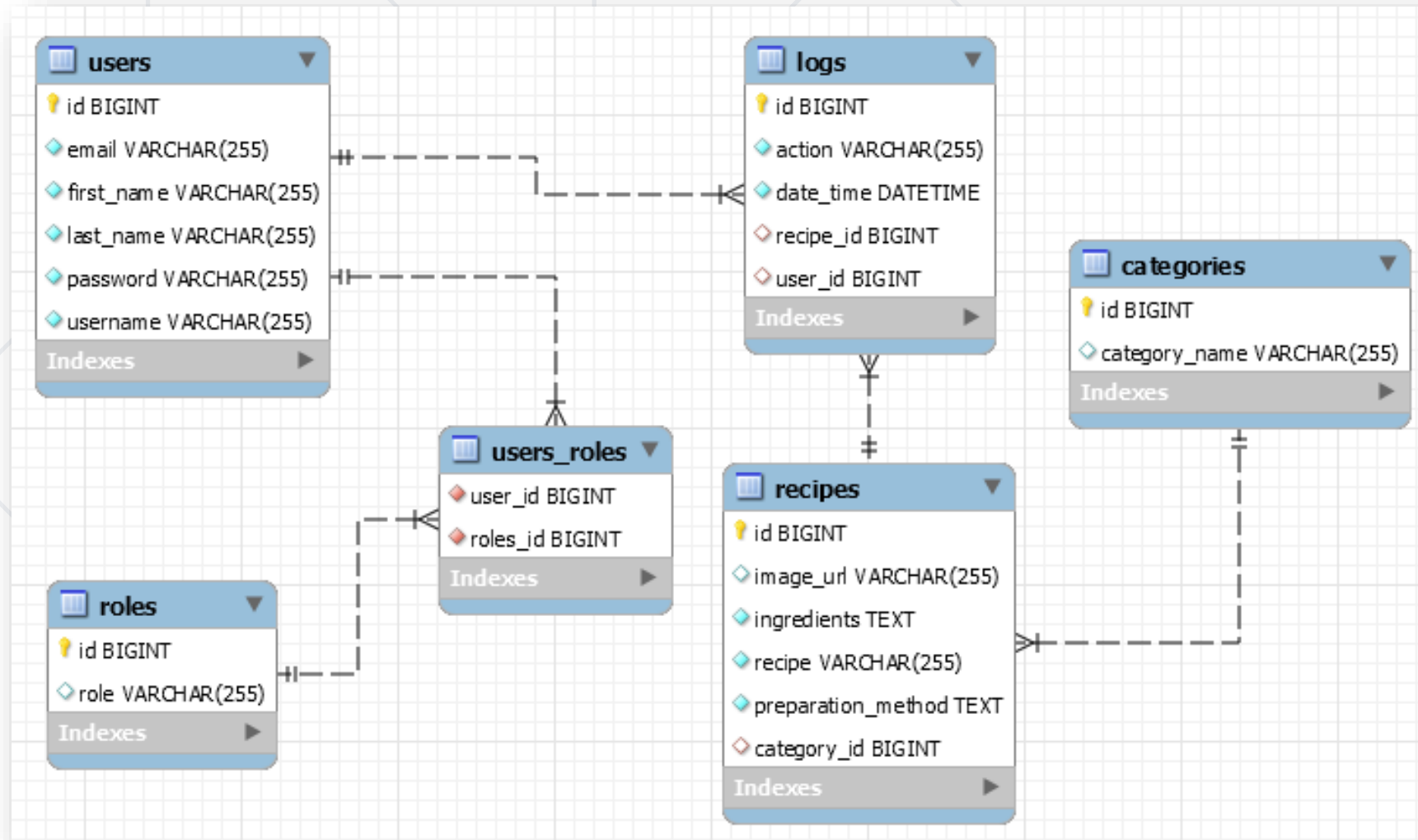
Primary Key

user_id	email
203	drivers@mail.cx
204	sarah@mail.cx
205	walters_michael@mail.cx
203	david@homedomain.cx

Foreign Key

- Connection via **Foreign Key** in one table pointing to the **Primary Key** in another

# E/R Diagrams





# JSON Data Format

Definition and Syntax

- **JSON** (**J**ava**S**cript **O**bject **N**otation) is a lightweight data format
  - Human and machine-readable plain text
  - Based on **JavaScript** objects
  - Independent of development platforms and languages
  - JSON data consists of:
    - Values (strings, numbers, etc.)
    - Key-value pairs: **{ key : value }**
    - Arrays: **[value1, value2, ...]**

```
{  
  "firstName": "Peter",  
  "courses": ["C#", "JS", "ASP.NET"]  
  "age": 23,  
  "hasDriverLicense": true,  
  "date": "2012-04-23T18:25:43.511Z",  
  // ...  
}
```

- The **JSON** data format follows the rules of object creation in JS

- **Strings, numbers** and **Booleans** :

```
"this is a string and is valid JSON"
```

```
3.14
```

```
true
```

- **Arrays** :

```
[5, "text", true]
```

- **Objects** (key-value pairs):

```
{  
  "firstName": "Svetlin", "lastName": "Nakov",  
  "jobTitle": "Technical Trainer", "age": 30  
}
```



# Mongo DB

## Working with Non-Relational Database

- **MongoDB** == free **open-source** cross-platform **document-oriented database**
  - Keeps collections of **JSON** documents (with or without schema)
- Sample usages: **mobile app** backend, product **catalog**, **poll** system, **blog** system, Web content management system (**CMS**)
- Supports evolving data requirements
  - The DB structure **may change** over the time
- Supports **indexing** for increased performance



# Install MongoDB

- Download from: [mongodb.com/try/download/community](https://mongodb.com/try/download/community)

## MongoDB Community Server

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with [MongoDB Atlas](#). Get access to advanced functionality such as auto-scaling, serverless instances (in preview), full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

## Available Downloads

Version	5.0.5 (current)	✓
Platform	Windows	✓
Package	msi	✓

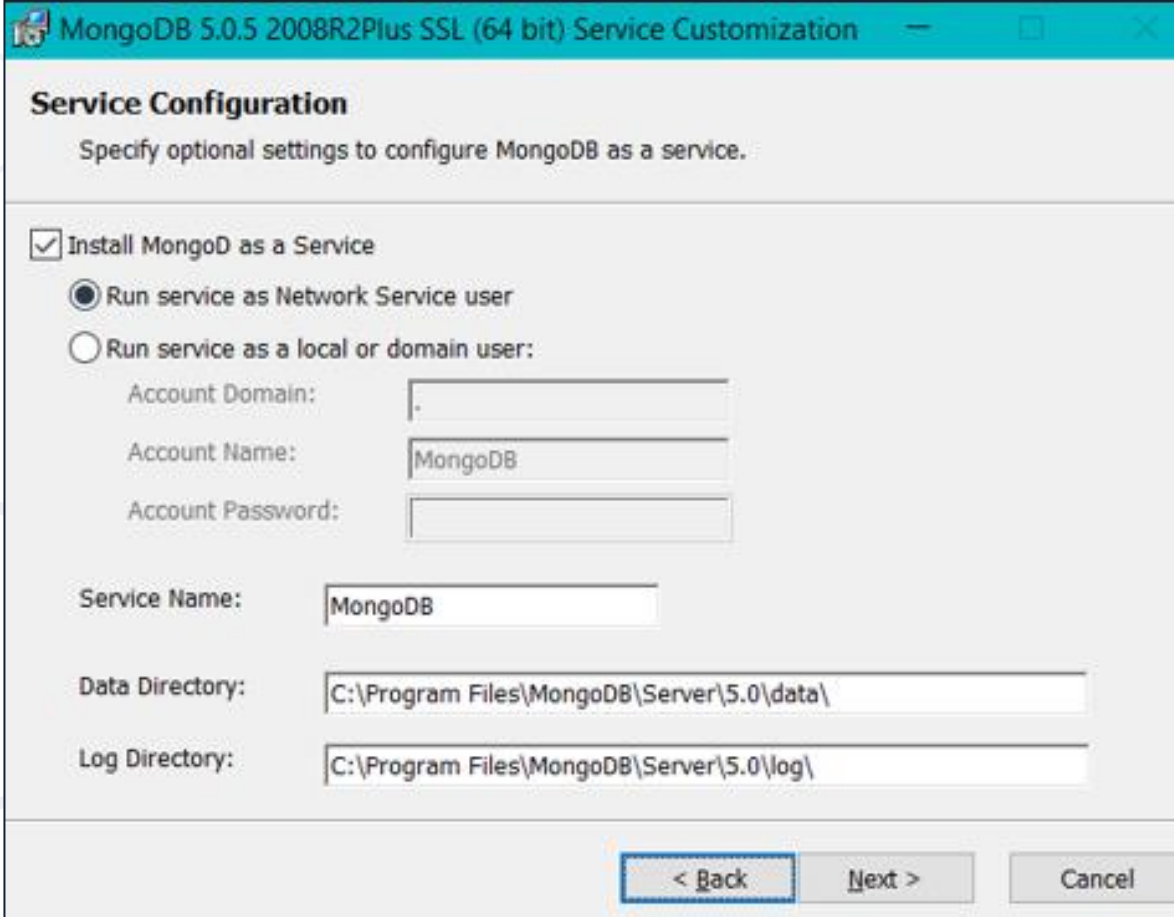


Download

Copy Link

- The package includes **MongoDB Compass**

- During installation, configure the **MongoDB service**:



**Service Configuration**  
Specify optional settings to configure MongoDB as a service.

☒ Install MongoDB as a Service

☒ Run service as Network Service user

☐ Run service as a local or domain user:

Account Domain:

Account Name:

Account Password:

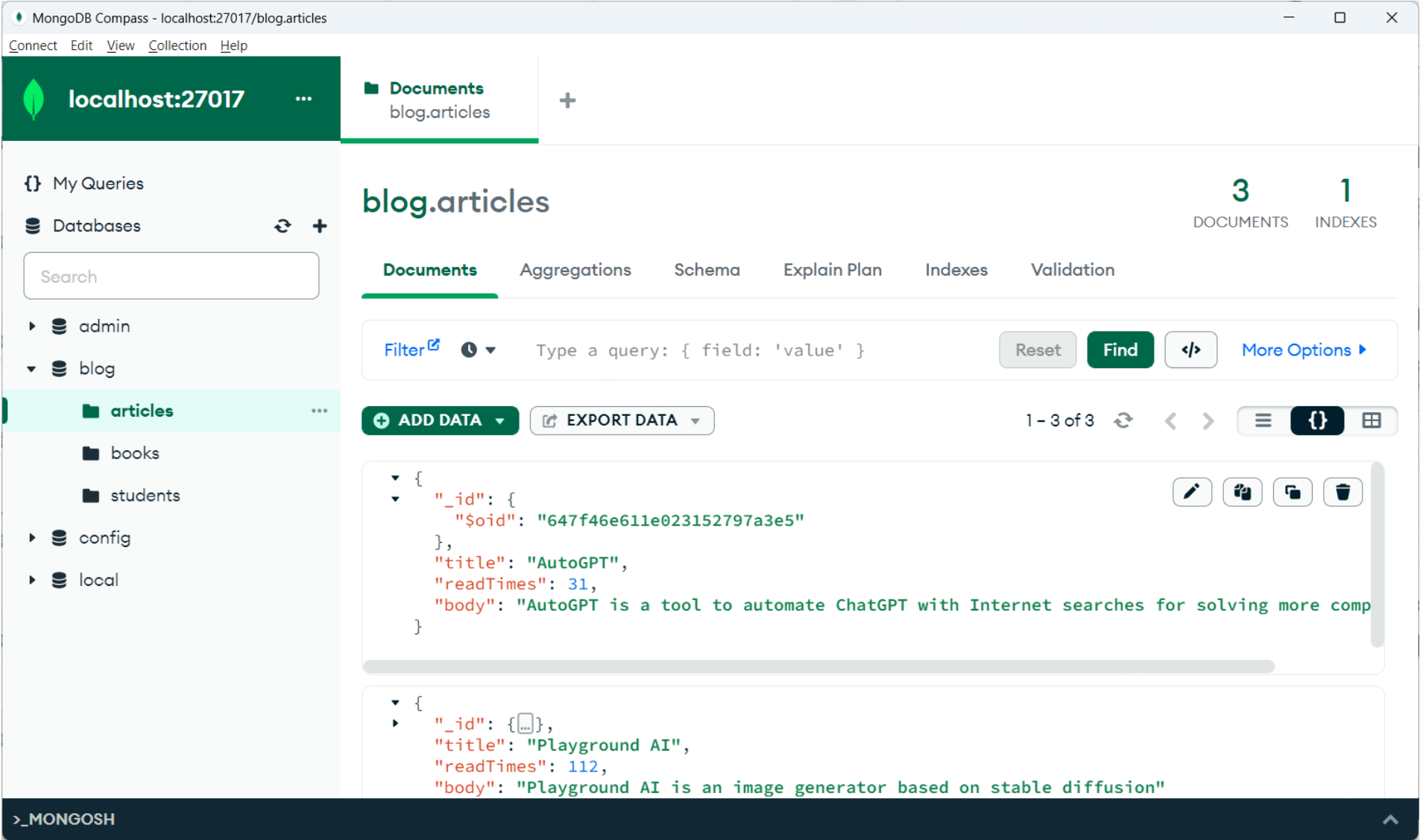
Service Name:

Data Directory:

Log Directory:

< Back   Next >   Cancel

- Choose one of the many
  - **MongoDB Compass** is included in the installer
- For example
  - Compass – <https://www.mongodb.com/products/compass>
  - Robo 3T – <https://robomongo.org/download>
  - NoSQLBooster – <https://nosqlbooster.com>



# MongoDB Queries Example

- Mongoose supports **many** queries
  - For equality/non-equality

```
Student.findOne({'lastName': 'Petrov'})
```

```
Student.find({}).where('age').gt(7).lt(14)
```

```
Student.find({}).where('facultyNumber').equals('12399')
```

- Selection of some properties

```
Student.findOne({'lastName': 'Kirilov'}).select('name age')
```

# Mongoose Queries Example 2

- Sorting

```
Student.find({}).sort({age:-1})
```

- Limit & skip

```
Student.find({}).sort({age:-1}).skip(30).limit(10)
```

- Different methods could be **stacked** one upon the other

```
Student.find({})  
  .where('firstName').equals('gosho')  
  .where('age').gt(18).lt(65)  
  .sort({age:-1})  
  .skip(10)  
  .limit(10)
```

- Install "**MongoDb Shell**" and run it from the command line:

- Type the command "**mongo**"

```
show dbs
```

```
use mytestdb
```

```
db.mycollection.insertOne({"name": "George"})
```

```
db.mycollection.find({"name": " George"})
```

```
db.mycollection.find({})
```

- Additional information at:

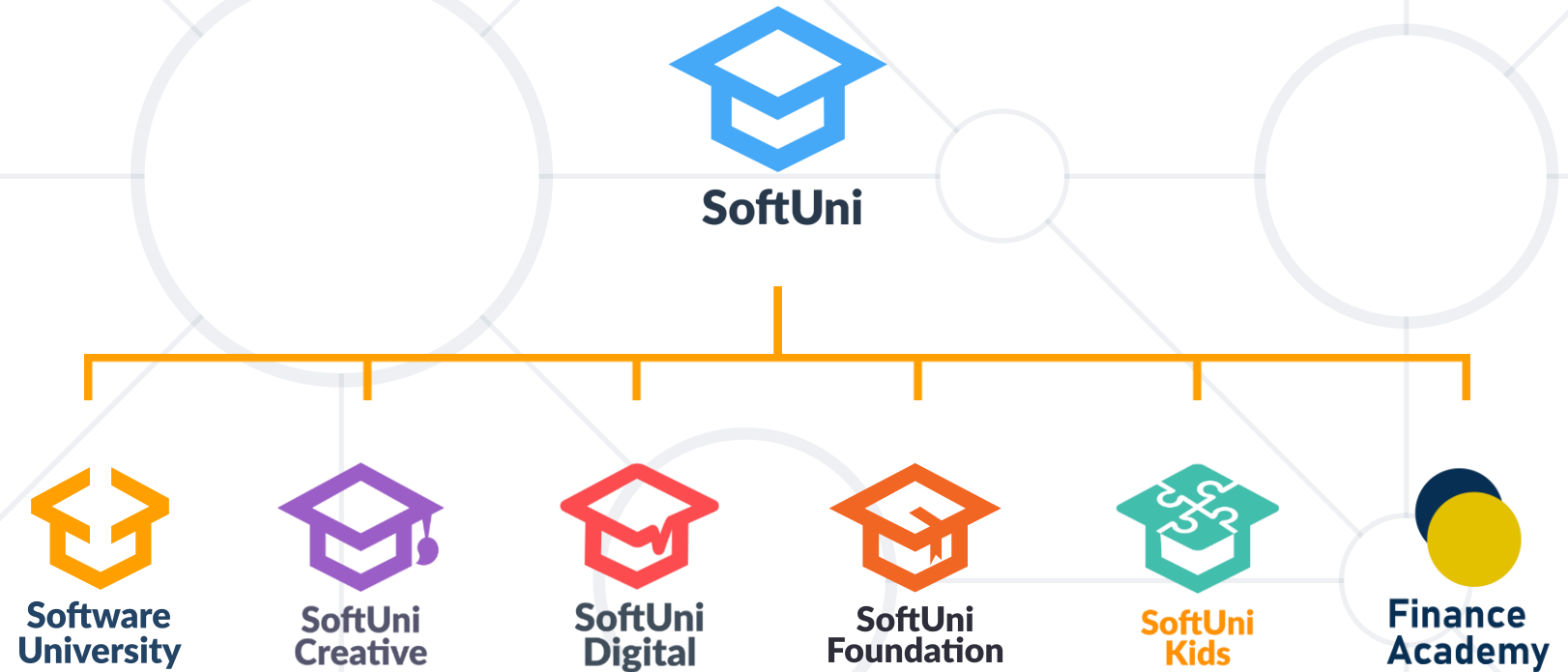
<https://www.mongodb.com/try/download/shell>

- **Databases**: store data tables and collections
- **Relational databases**: tables and relationships
- **Non-Relational**: document collections
- **DBMS** (database servers), e.g. MySQL, MongoDB
- **SQL** commands: SELECT, INSERT, UPDATE, DELETE, ...
- **JSON** document: {"name":"Joe", "age":25}
- Working with **MySQL + Workbench**
- Working with **Mongo DB + Compass**





# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**



**POKERSTARS**  
POKER | CASINO | SPORTS  
a Flutter International brand

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**



**SOFTWARE  
GROUP**

createX



**Postbank**  
Решения за твоето утре

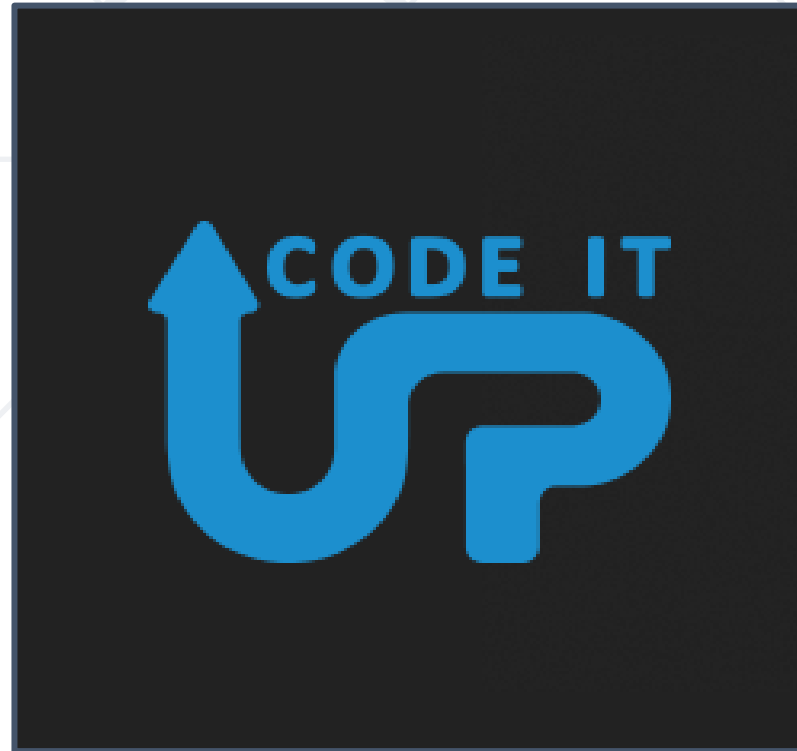


**BOSCH**

**DXC**  
TECHNOLOGY



**SmartIT**



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



Software University

