

Database Basics

Database Management Systems and SQL



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

1. Databases: Introduction
2. SQL vs. NoSQL Databases
3. DBMS Systems (Database Engines)
4. Relational Databases, SQL and MySQL Database
5. NoSQL Databases and MongoDB



Have a Question?

sli.do

#fund-common



Databases: Introduction

Data Storage and Data Management

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



- Conventional data storage
 - Orders
 - Receipts



From Data Storage to Databases

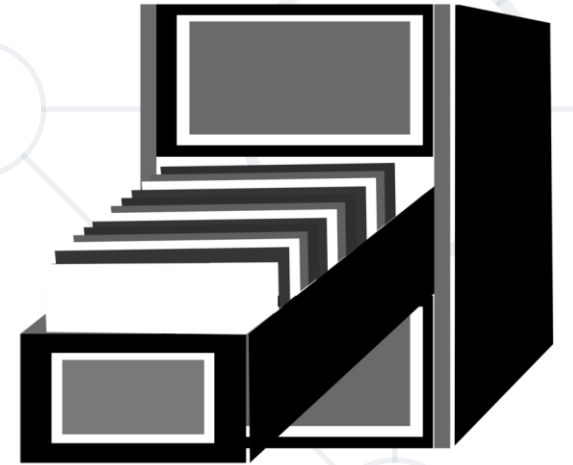
- We can **group related pieces of data** into separate columns:

Order#	Date	Customer	Product	S/N	Unit Price	Qty	Total
315	07/16/2016	David Rivers	Oil Pump	OP147-0623	69.90	1	69.90



Why Do We Need Databases?

- Storing data is **not** the primary reason to use a database
- Flat storage runs into **issues** with
 - Ease of searching
 - Ease of updating
 - Performance
 - Accuracy and consistency
 - Security and access control
 - Redundancy





SQL vs. NoSQL Databases

SQL Databases (Relational Databases)

- Relational (**SQL**) databases organize data in **tables**
 - Tables have strict structure (**columns** with 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



SQL Databases (Relational Databases) (2)

- Relational DB model organizes data into one or more **tables** of columns and rows 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	peter@gmail.com
6	Jayne	jayne@gmail.com

Orders

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



NoSQL Databases (Non-Relational Databases)

- A **NoSQL** databases have dynamic schema for **unstructured** data
- Data is stored in many ways
 - Document-oriented
 - Column-oriented
 - Graph-based
 - Key-value store



Scalability: Relational vs. NoSQL

- SQL are **vertically** scalable
 - You can increase the load on a single server by **increasing its resources** (CPU, RAM, SSD)
 - Or you can **replicate the data** to a cluster of several servers
- NoSQL are **horizontally** scalable
 - You handle more traffic by **sharding** and adding more servers in your NoSQL database cluster



Structure: Relational vs. NoSQL

- SQL databases are **table-based**
- Better option for:
 - Applications that require multi-row transactions, such as an accounting system
 - Complex transaction processing systems
- SQL databases hold **dynamic data**
- NoSQL databases implement **four main data models**
 - Document store
 - Wide-column store
 - Key-value data store
 - Graph store



- **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

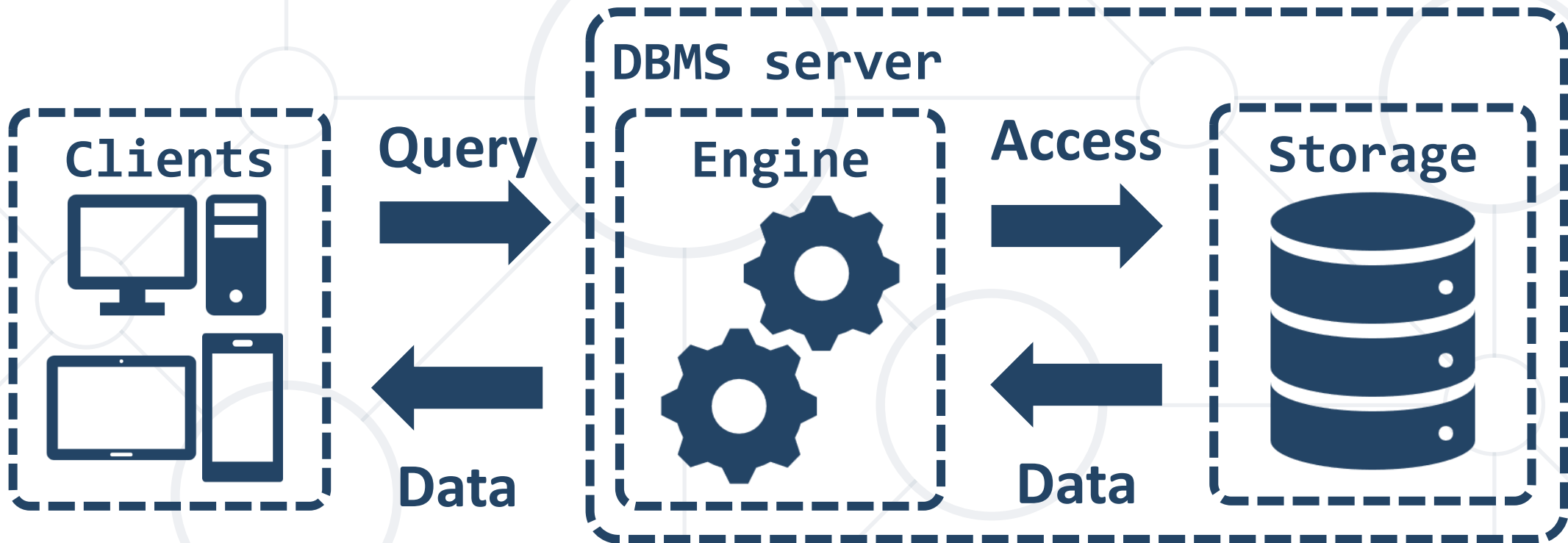




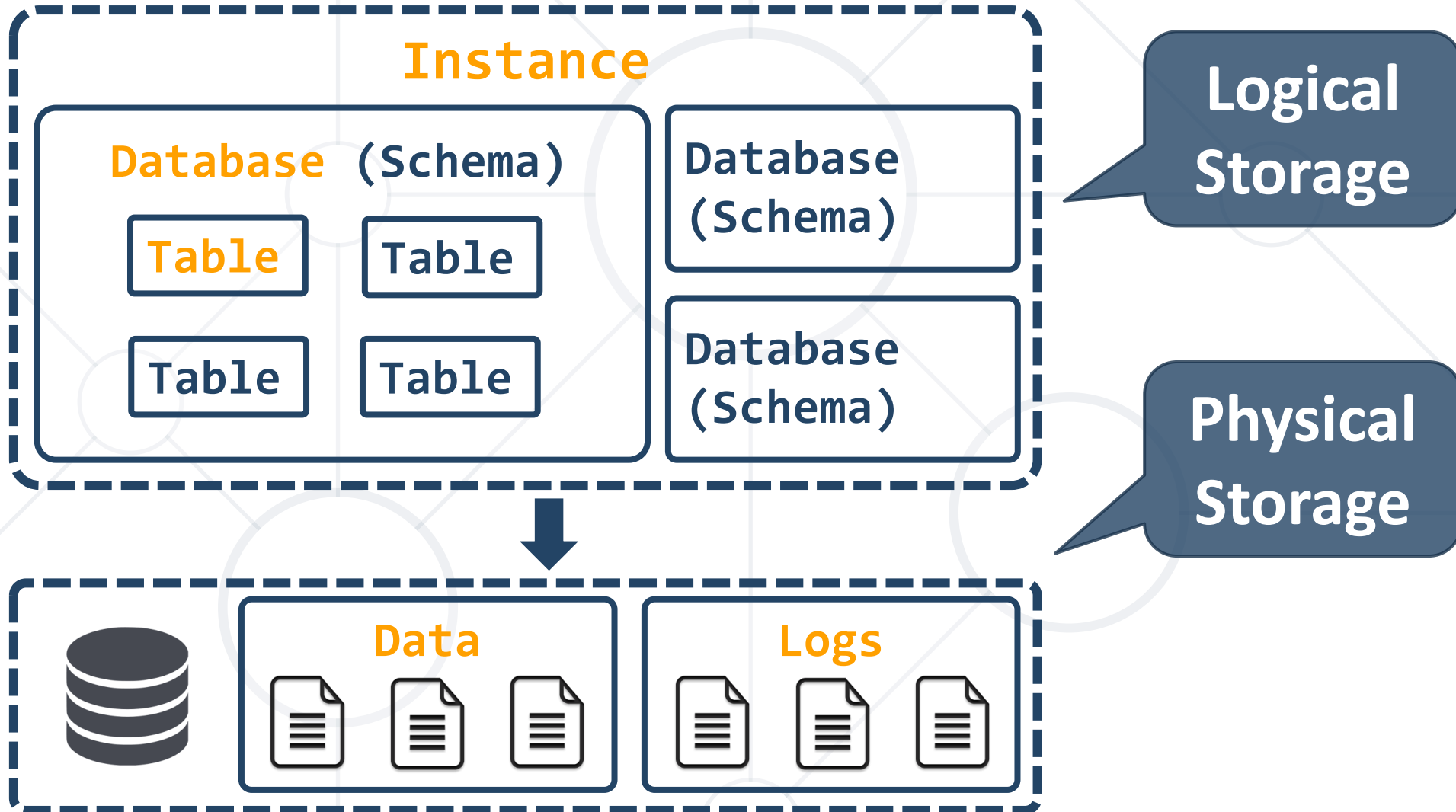
Database Management Systems (DBMS)

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

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



DBMS Server Architecture

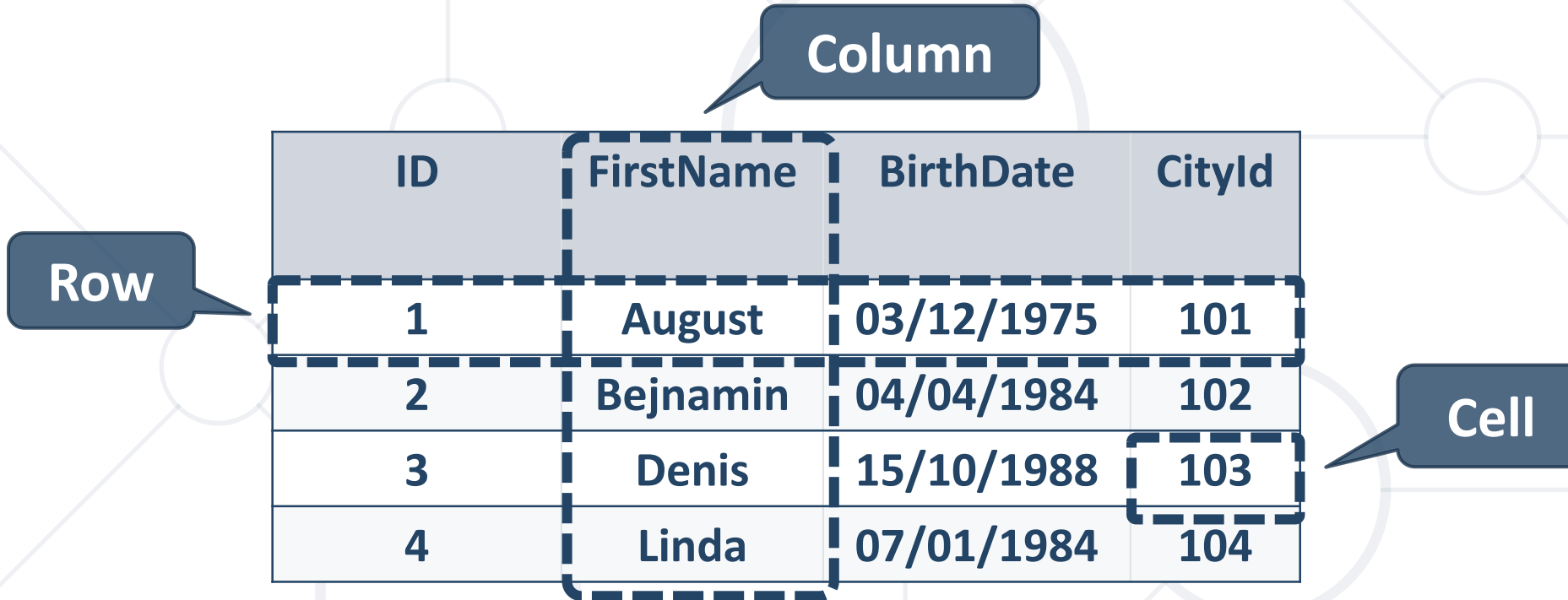




Relational Databases

RDBMS, the SQL Language and MySQL Database

- The **table** is the main **building block** in the relational databases



ID	FirstName	BirthDate	CityId
1	August	03/12/1975	101
2	Bejnamin	04/04/1984	102
3	Denis	15/10/1988	103
4	Linda	07/01/1984	104

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

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**

- 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	smith@yahoo.co.uk	John	Smith
2	pwh@gmail.com	Peter	White
3	anne@anne.com	Anne	Green
4	jason.jj@gmail.com	Jason	Anderson



- **MySQL** == **open-source** relational database management system (RDBMS), very popular, also known as MariaDB
 - Runs on **most** server platforms: Linux, Windows, macOS
- Used in many **large-scale** software projects
 - Amazon, Apple, Facebook, others
- In MySQL data is stored in **tables** with **relationships** between them
 - SQL is used to query / manipulate data



- phpMyAdmin (part of XAMPP)

- **phpMyAdmin** is Web-based MySQL admin tool
- **XAMPP** == Web server development stack
 - Apache + MariaDB + PHP + phpMyAdmin



- HeidiSQL

- GUI tool for managing MySQL, MSSQL and PostgreSQL
- Query / modify database
- Explore database objects



- 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
```

Creating Table and Inserting Values

■ Creating tables

```
CREATE TABLE people (  
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  email VARCHAR(40) NOT NULL,  
  first_name VARCHAR(40) NOT NULL,  
  last_name VARCHAR(40) NOT NULL  
)
```

Table name

Primary key
definition

Column name

Data type

■ Inserting values

```
INSERT INTO people(email, first_name, last_name)  
VALUES ('john@gmail.com', 'John', 'Smith')
```

- **Retrieve** all records from a table

```
SELECT * FROM people
```

* retrieves all columns

- You can **limit** (select) **the columns** to retrieve

```
SELECT first_name, last_name FROM people
```

List of columns

- You can **limit the number of rows**

```
SELECT first_name, last_name FROM people  
LIMIT 5
```

Number of
rows to return

- Retrieve all records, matching a **filter**

```
SELECT * FROM people  
WHERE email = 'peter@gmail.com'
```

Filter the returned rows by a condition

- Filter** and **sort** data

```
SELECT * FROM people  
WHERE id > 10 AND id < 20  
ORDER BY id
```

Filter by multiple conditions

Sort by given column / expression

- **Updating** rows

```
UPDATE people  
SET last_name = 'Adams'  
WHERE first_name = 'John'
```

Updates the last
name of person

```
UPDATE people  
SET first_name = 'Peter',  
    last_name = 'White',  
    email = 'pw@email.com'  
WHERE id = 42
```

Update
multiple
fields

Deleting Data and Objects

- **Deleting** table rows

```
DELETE FROM people WHERE id = 42
```

- Deleting (**dropping**) database objects

- Table

Delete all records in a table

```
TRUNCATE TABLE people
```

Delete the table itself

```
DROP TABLE people
```

- Entire database

```
DROP DATABASE employees
```

- These actions **cannot be undone**



Using MySQL and HeidiSQL

Live Demo



NoSQL Databases

Using MongoDB

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

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

Example of JSON
document in MongoDB

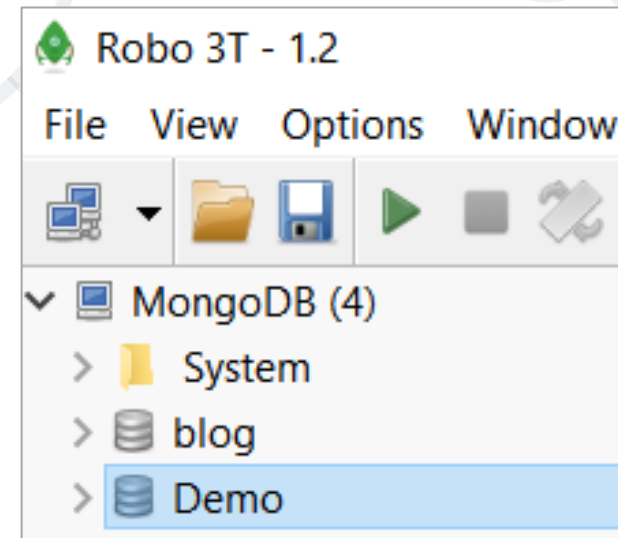
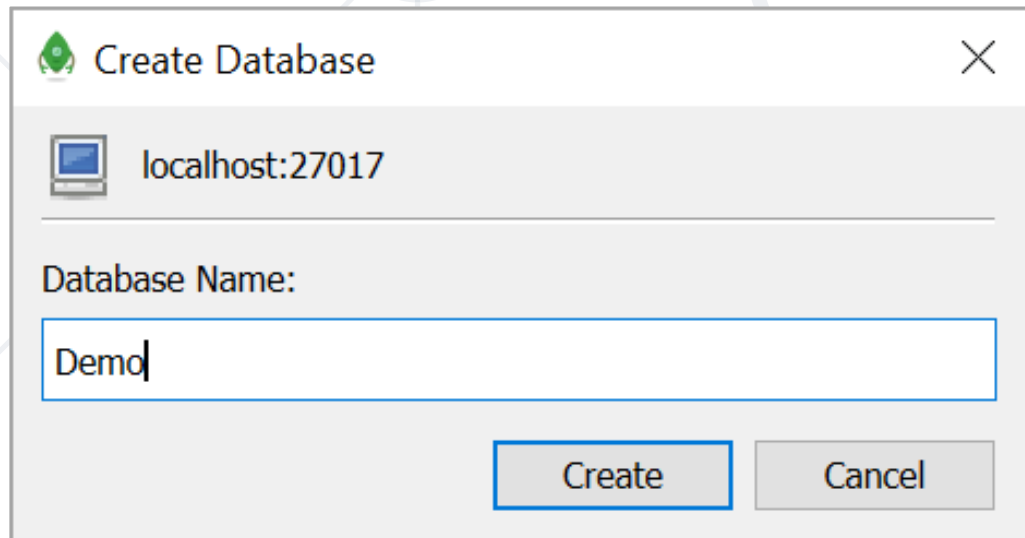
- **MongoDB** == free **open-source** cross-platform **document-oriented database**
 - Keeps collections of **JSON** documents (with or without schema)
- Sample usages: **mobile app** backend, product **catalog**, **poll** system, **blog** system, Web content management system (**CMS**)
- Supports evolving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

- MongoDB Compass
- Robo 3T
 - Powerful **GUI tool** for MongoDB
 - Fully-featured IDE with embedded **shell**
- NoSQLBooster (alternative)
 - Shell-centric cross-platform GUI tool
 - Object explorer and query builder



Creating a Database

- **Creating a MongoDB database** in Robo 3T is done using the GUI
- Right click on [**New Connection**] and select [**Create Database**]



Creating a Collection and Inserting Values

- **Creating a collection**

```
db.createCollection('people')
```

Collection name

- **Inserting a document** to existing collection

```
db.getCollection('people')  
  .insert({  
    firstName: 'Michael',  
    lastName: 'Smith',  
    email: 'michael@gmail.com'  
  })
```

Data is inserted
as **JSON object**

- Get **all entries** from a collection

```
db.getCollection('people').find({})
```

- Filter elements by **given criteria**

```
db.getCollection('people').find({ firstName: 'Michael' })
```

- Return **specified** fields

```
db.getCollection('people').find(  
  { firstName: 'Michael' },  
  { lastName: 1 }  
)
```

Retrieves an **entity** with
the desired **fields only**

- Update the **first** entry

```
db.getCollection('people').updateOne(  
  { firstName: 'Kate' },  
  { $set: { firstName: 'George', age: 25 } }  
)
```

Old values
(filter)

New object (replacement)

```
db.getCollection('people').updateOne(  
  { firstName: 'Kate' },  
  { $set: { firstName: 'George', lastName:  
    'Doe' } },  
  { multi: true }  
)
```

Update **all matching entries**

- Delete the **first entry** that matches given criteria

```
db.getCollection('people').deleteOne(  
  { firstName: 'George' }  
)
```

- Delete **all entries** that match given criteria

```
db.getCollection('people').deleteMany(  
  { firstName: 'George' }  
)
```



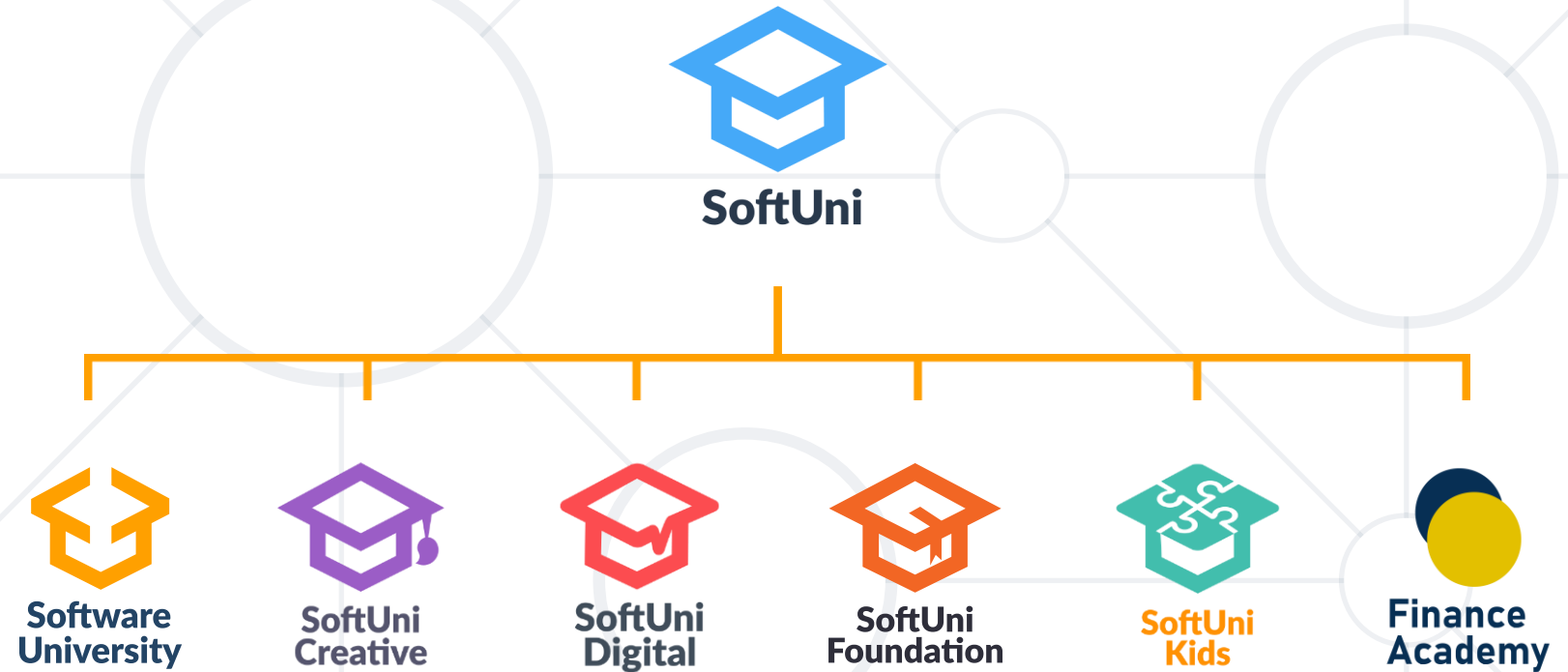
MongoDB and Robo 3T

Live Demo

- Database management systems (DBMS) **store** and **manage data**
 - Developers **communicate** with the DB engine via **SQL** commands or via **API**
- **MySQL** is open-source RDBMS: data is stored in **tables** and accessed via **SQL**
- **NoSQL** databases are more **flexible**
 - **MongoDB** stores entries in **JSON** format



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**



**Coca-Cola HBC
Bulgaria**

 **Flutter**TM
International

INDEAVR
Serving the high achievers



AMBITIONED

 **DRAFT
KINGS**

 **SOFTWARE
GROUP**



BOSCH

 **Postbank**
Решения за твоето утре

 **PHAR
VISION**



SmartIT

DXC
TECHNOLOGY

createX

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

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



Software
University



- 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>

