

# Introduction to Databases

## Data Definition and Datatypes

How Do RDBMS Work?



**SoftUni Team**  
Technical Trainers



**SoftUni**

Software University

<https://softuni.bg>

# Table of Contents

1. Data Management
2. Database Engine
3. Structured Query Language
4. MySQL
5. Table Relationships
6. Programmability



# Table of Contents

- 7. Data Types in MySQL Server
- 8. Database Modeling
- 9. Basic SQL Queries
- 10. Table Customization
- 11. Altering Tables
- 12. Deleting Data and Structures



[sli.do](https://sli.do)

**#Java-DB**



# **Data Management**

When Do We Need a Database?

# Storage vs. Management

## SALES RECEIPT

Date: 07/16/2016

Order#:[00315]

Customer: David Rivers

Product Oil Pump

S/N: OP147-0623

Unit Price: 69.90

Qty: 1

Total: 69.90

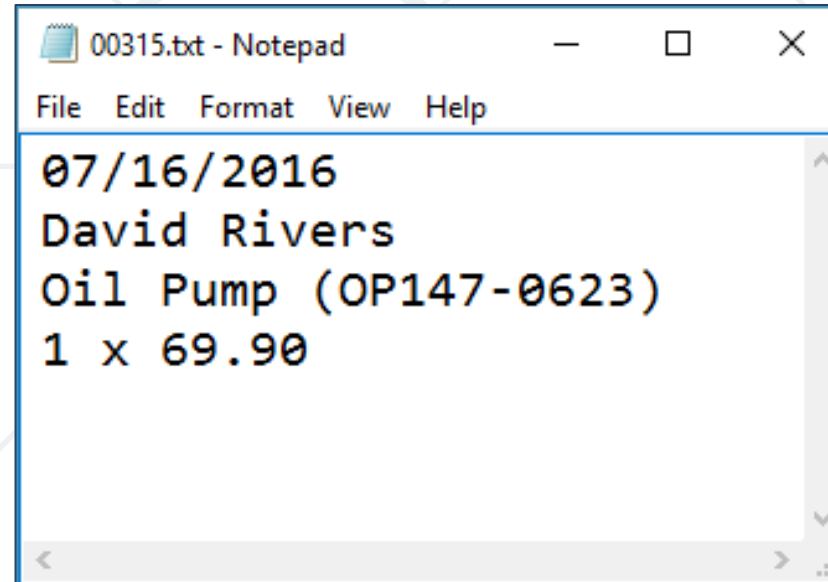
00315 – 07/16/2016

David Rivers

Oil Pump (OP147-0623)

1 x 69.90

# Storage vs. Management (2)



Order#	Date	Customer	Product	S/N	Qty
00315	07/16/2016	David Rivers	Oil Pump	OP147-063	1

# Storage vs. Management (3)

- Storing data is **not** the primary reason to use a database
- Flat storage **eventually** runs into **issues** with
  - Size
  - Ease of updating
  - Accuracy
  - Security
  - Redundancy
  - Importance





- A database is an **organized** collection of **related** information
  - It imposes **rules** on the contained data
  - Access to data is usually provided by a "**system**" (DBMS)  
**database management**
  - Relational storage first proposed by Edgar Codd in 1970

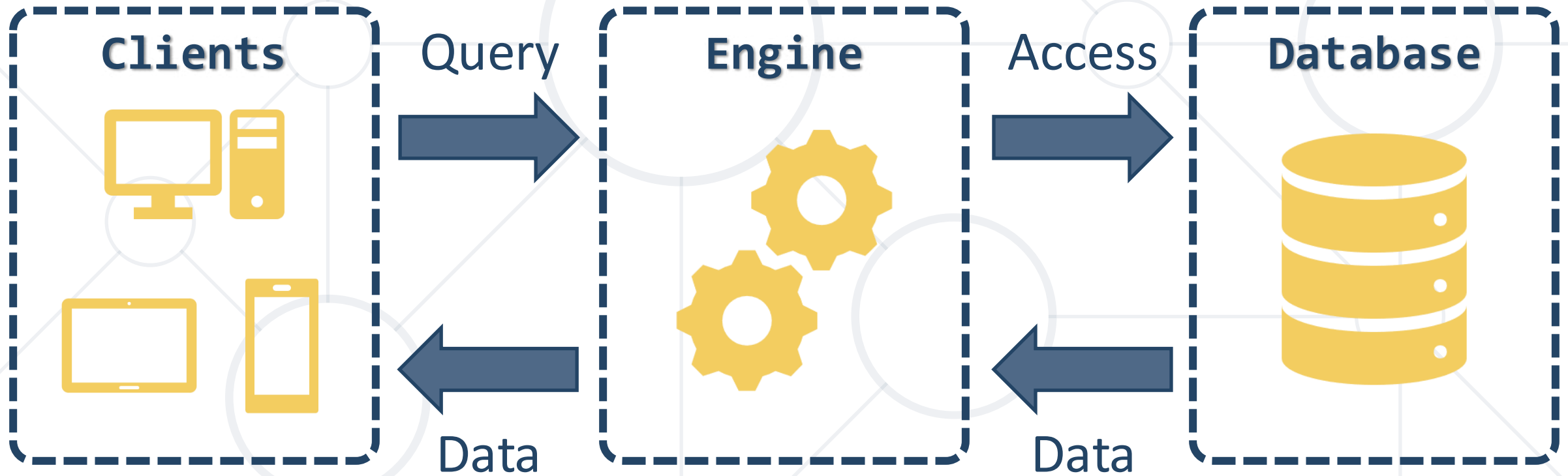
- Relational Data Base Management System
  - Database management
  - It parses requests from the user and takes the appropriate action
  - The user doesn't have direct access to the stored data
  - Data is presented by relations – collection of tables related by common fields
  - MS SQL Server, DB2, Oracle and MySQL



**Database Engine**

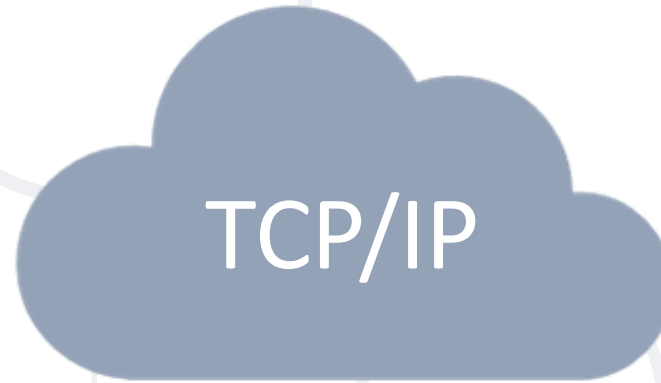
# Database Engine Flow

- SQL Server uses the Client-Server Model























# Client-Server Model

CLIENTS



DATABASE

# Top Database Engines

Rank			DBMS	Database Model	Score		
Dec 2019	Nov 2019	Dec 2018			Dec 2019	Nov 2019	Dec 2018
1.	1.	1.	Oracle 	Relational, Multi-model 	1346.39	+10.33	+63.17
2.	2.	2.	MySQL 	Relational, Multi-model 	1275.67	+9.38	+114.42
3.	3.	3.	Microsoft SQL Server 	Relational, Multi-model 	1096.20	+14.29	+55.86
4.	4.	4.	PostgreSQL 	Relational, Multi-model 	503.37	+12.30	+42.74
5.	5.	5.	MongoDB 	Document, Multi-model 	421.12	+7.94	+42.50
6.	6.	6.	IBM Db2 	Relational, Multi-model 	171.35	-1.25	-9.40
7.	7.	 8.	Elasticsearch 	Search engine, Multi-model 	150.25	+1.85	+5.55
8.	8.	 7.	Redis 	Key-value, Multi-model 	146.23	+1.00	-0.59
9.	9.	9.	Microsoft Access	Relational	129.47	-0.60	-10.04
10.	10.	 11.	Cassandra 	Wide column	120.71	-2.52	-1.10

Source: <http://db-engines.com/en/ranking>



**Structured Query Language**

# Structured Query Language

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



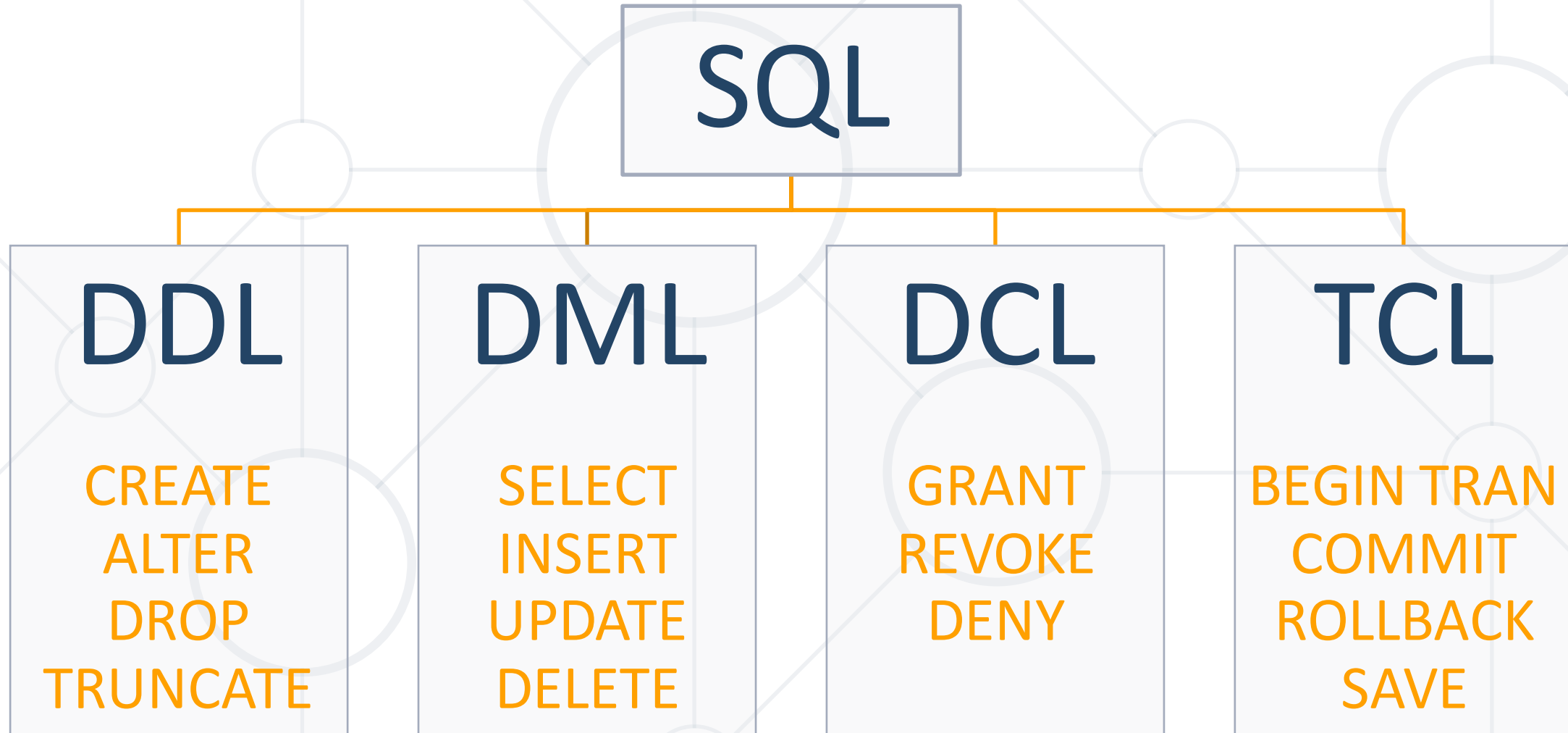
# Structured Query Language (2)

- 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





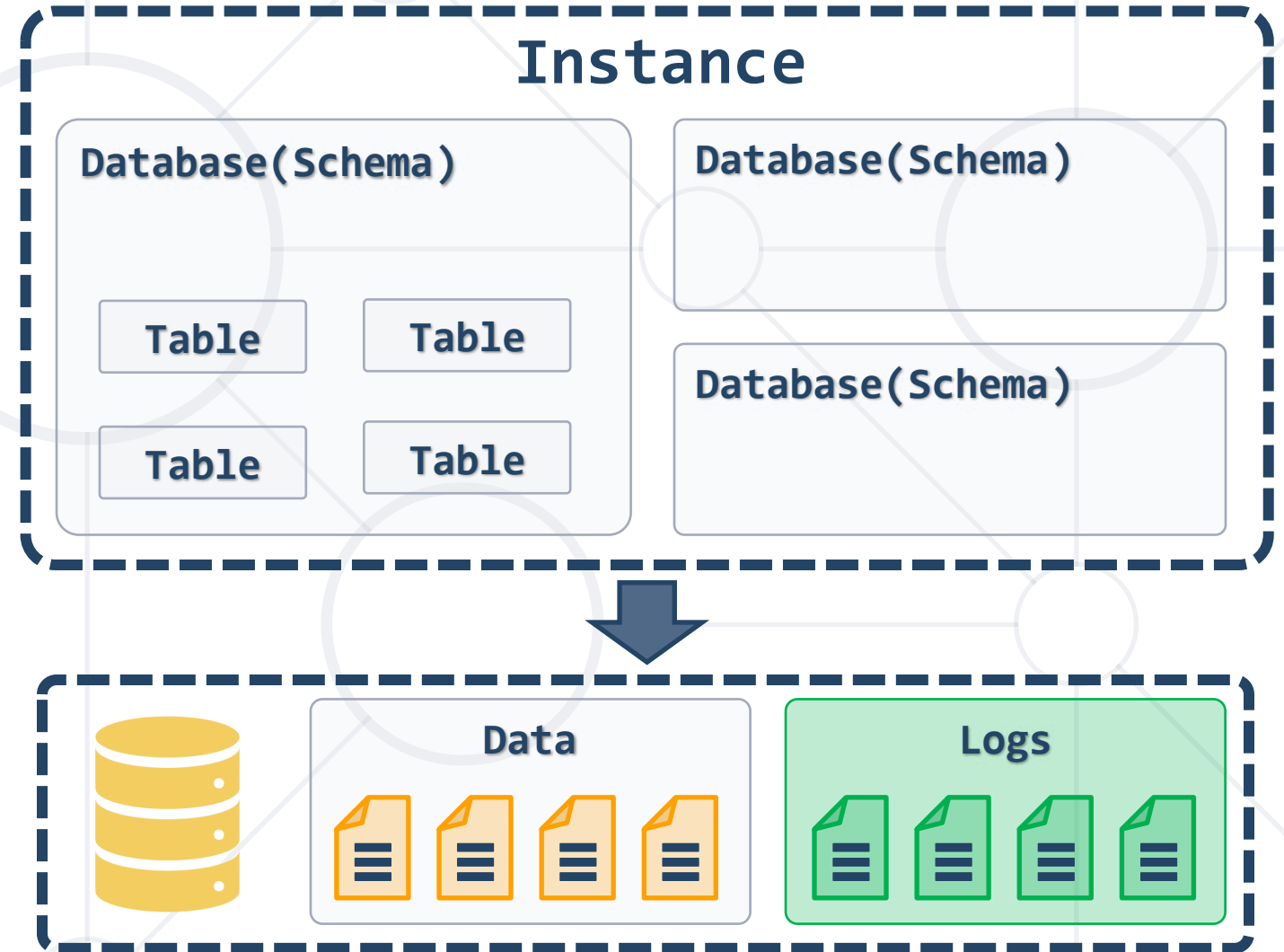
**MySQL**

- **Open-source** relational database management system
- Used in many **large-scale websites** like including Google, Facebook, YouTube etc.
- Works on **many** system platforms –  
MAC OS, Windows, Linux
- Download **MySQL Server**
  - **Windows:** [dev.mysql.com/downloads/windows/installer/](https://dev.mysql.com/downloads/windows/installer/)
  - **Ubuntu/Debian:** [dev.mysql.com/downloads/repo/apt/](https://dev.mysql.com/downloads/repo/apt/)



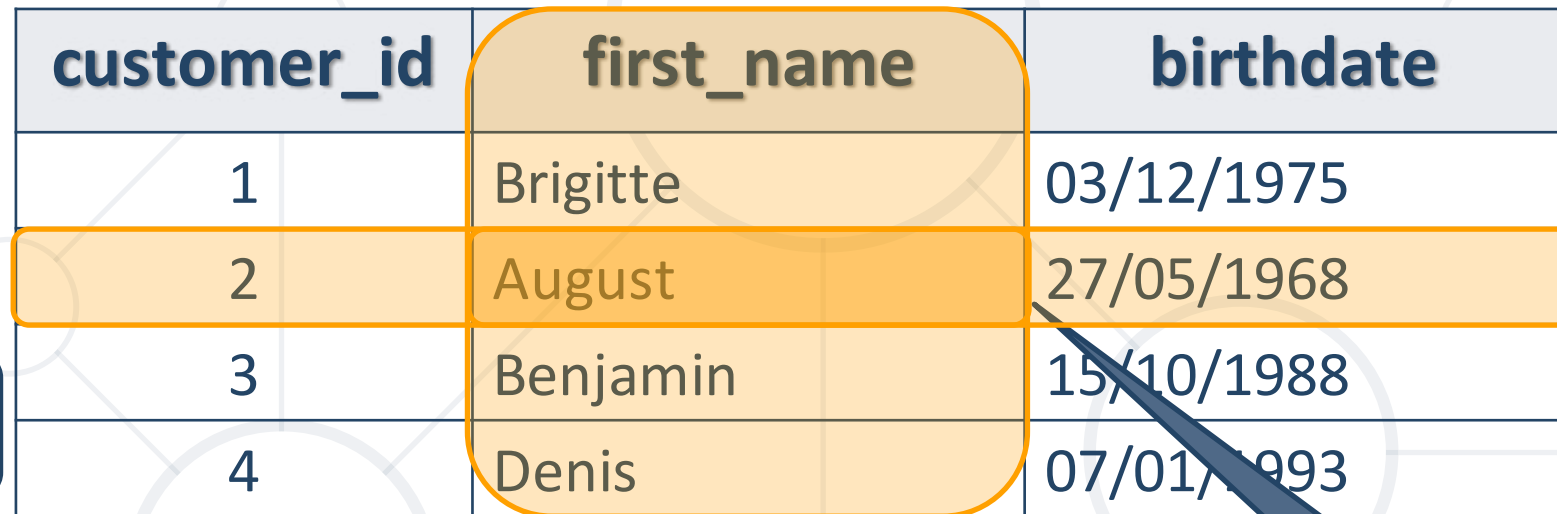
# MySQL Server Architecture

- Logical Storage
  - Instance
    - Database/Schema
      - Table
- Physical Storage
  - Data files and Log files
  - Data pages



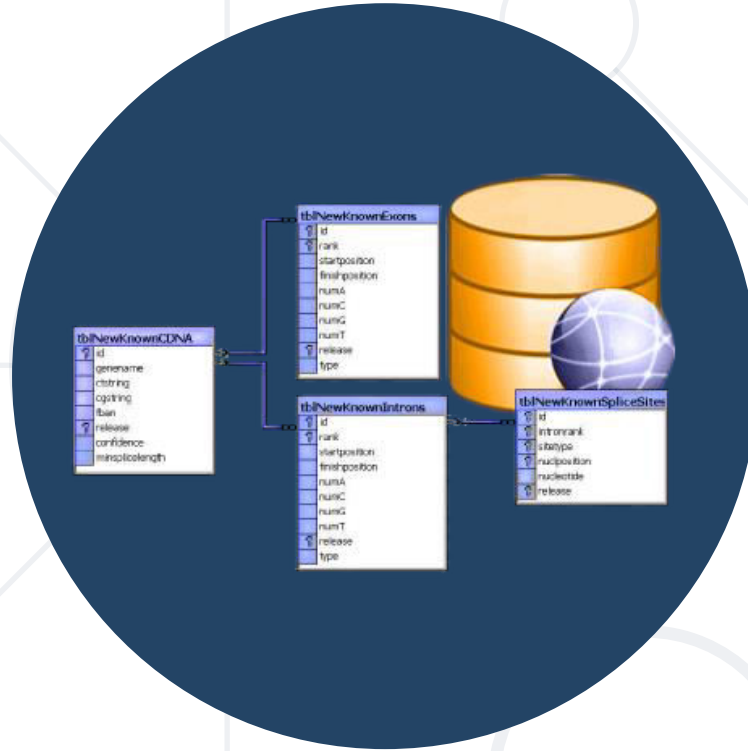
# Database Table Elements

- 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



# Table Relationships

Table Relationships



# Why Split Related Data?

first	last	registered	email	email2
David	Rivers	05/02/2016	drivers@mail.cx	david@homedomain.cx
Sarah	Thorne	07/17/2016	sarah@mail.cx	NULL
Michael	Walters	07/18/2016	walters_michael@mail.cx	NULL

Empty records

Redundant information

order_id	date	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

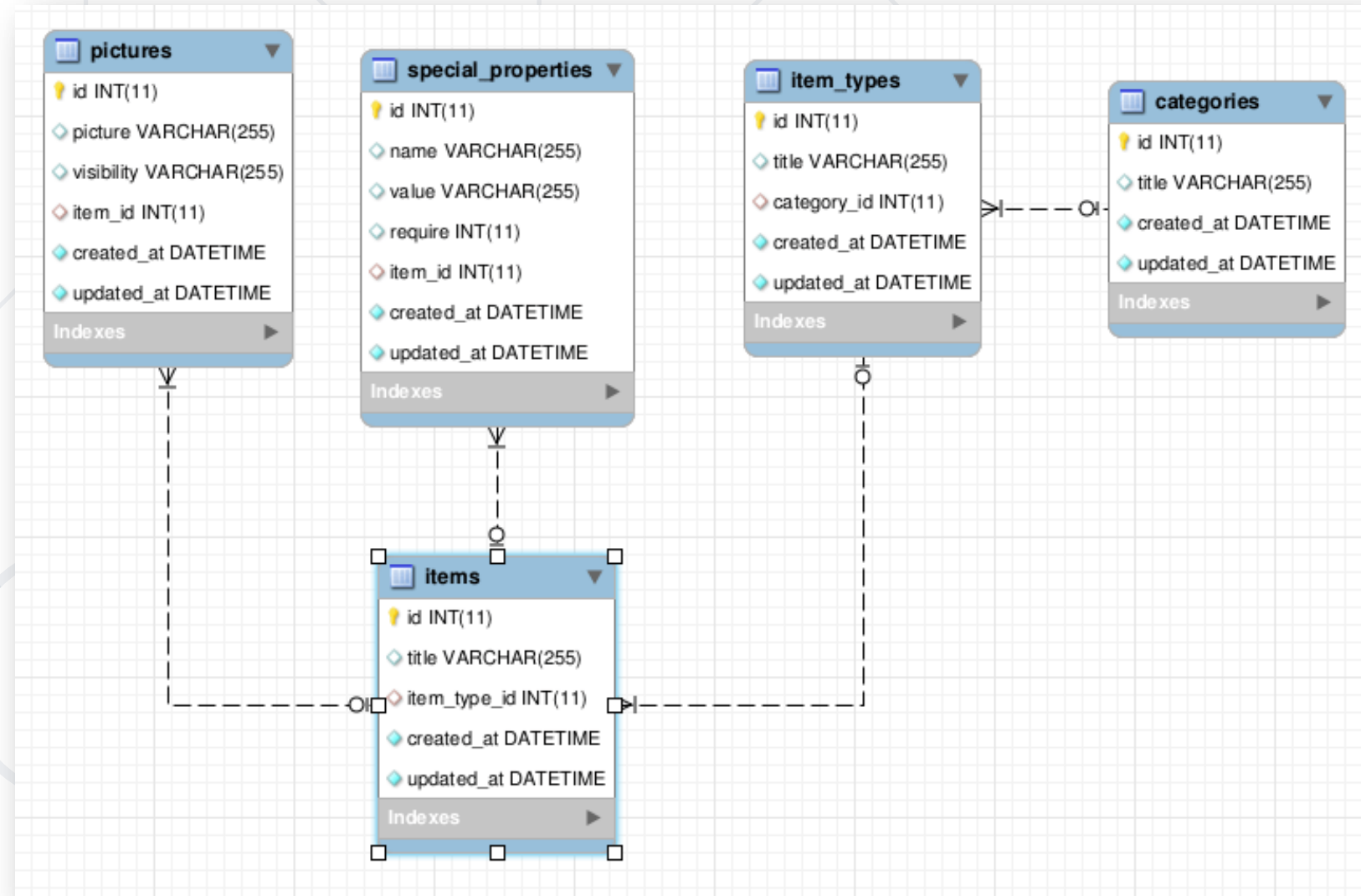
Primary Key

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

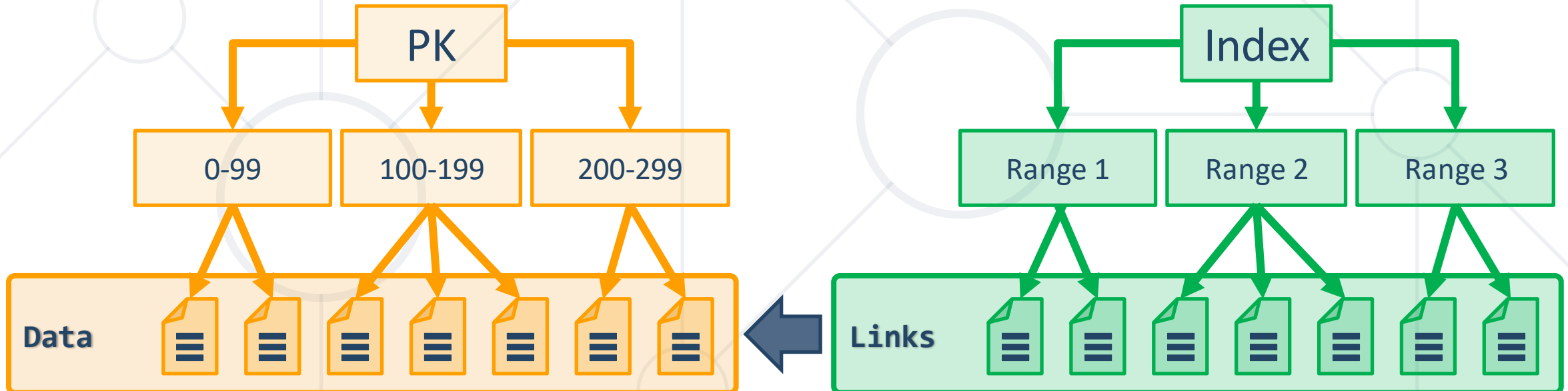




# **Programmability**

Customizing Database Behavior

- Indices make data lookup faster
  - Clustered – bound to the **primary key**, physically sorts data
  - Non-Clustered – can be **any field**, references the primary index
- Structured as an **ordered tree**



- Views are **prepared queries** for displaying **sections** of our data

```
CREATE VIEW v_employee_names AS
SELECT e.employee_id,
       e.first_name,
       e.last_name
FROM   soft_uni.employees AS e
```

```
SELECT * FROM v_employee_names
```

- Evaluated at **run time** – they do not increase performance

- A database can further be customized with reusable code
- **Procedures** – carry out a predetermined **action**
  - E.g. get all employees with salary above 35000
- **Functions** – receive **parameters** and return a **result**
  - E.g. get the age of a person using their birthdate and current date
- **Triggers** – **watch** for activity in the database and **react** to it
  - E.g. when a record is deleted, write it to an archive

```
CREATE PROCEDURE udp_get_employees_salary_above_35000()  
BEGIN  
    SELECT first_name, last_name FROM employees  
    WHERE salary > 35000;  
END
```

```
CALL udp_get_employees_salary_above_35000
```



```
CREATE FUNCTION udf_get_age (dateValue DATE)
RETURNS INT
BEGIN
  DECLARE result INT;
  SET result = TIMESTAMPDIFF(YEAR, dateValue, NOW());
  RETURN result;
END
```

```
SELECT udf_get_age( '1988-12-21' );
```



# **Data Types in MySQL Server**

Numeric, String and Data Types

- Numeric data types have certain range
- Their range can be changed if they are:
  - **Signed** - represent numbers both in the positive **and** negative ranges
  - **Unsigned** - represent numbers **only** in the positive range
- E.g. signed and unsigned INT:

Signed Range		Unsigned Range	
Min Value	Max Value	Min Value	Max Value
-2147483648	2147483648	0	4294967295

- **INT** [(M)] [UNSIGNED]
  - TINYINT, SMALLINT, MEDIUMINT, BIGINT

- **DOUBLE** [(M, D)] [UNSIGNED]

Digits stored for value

Decimals after  
floating point

- E.g. DOUBLE[5, 2] – 999.99
- **DECIMAL** [(M, D)] [UNSIGNED] [ZEROFILL]

- String column definitions include attributes that specify the **character set** or **collation**
  - **CHARACTER SET** (Encoding)
    - E.g. utf8, ucs2
  - **CHARACTER COLLATION** – rules for encoding comparison
    - E.g. latin1\_general\_cs, Traditional\_Spanish\_ci\_ai etc.
- Set and collation can be defined at the database, table or column level

Determines the storage of each character (single or multiple bytes)

Determines the sorting order and case-sensitivity

# CHARACTER COLLATION – Example

- **ORDER BY** with different collations

latin1_swedish_ci	latin1_german1_ci	latin1_german2_ci
Muffler	Muffler	Müller
MX Systems	Müller	Muffler
Müller	MX Systems	MX Systems
MySQL	MySQL	MySQL

# String Types (2)

- **CHAR** [(M)] - up to 30 characters
- **VARCHAR**(M) – up to 255 characters
- **TEXT** [(M)] – up to 65 535 characters
  - TINYTEXT, MEDIUMTEXT, LONGTEXT
- **BLOB** - **B**inary **L**arge **O**bject [(M)] - 65 535 ( $2^{16} - 1$ ) characters
  - TINYBLOB, MEDIUMBLOB, LONGBLOB

Column name	Column Type
title	VARCHAR(CHAR)
content	TEXT(LONGTEXT)
picture	BLOB(LONGBLOB)

- **DATE** - for values with a date part but **no time part**
- **TIME** - for values with time but **no date part**
- **DATETIME** - values that contain both date **and** time parts
- **TIMESTAMP** - both date **and** time parts

Column name	Column Type
birthdate	DATE
last_time_online	TIMESTAMP
start_at	TIME
deleted_on	DATETIME

**DATETIME and  
TIMESTAMP have  
different time ranges**



# Date Types (2)

- MySQL retrieves values for a given date type in a **standard output format**
  - E.g. as a string in either 'YYYY-MM-DD' or 'YY-MM-DD'

Data Type	Column Type
DATE	' 0000-00-00 '
TIME	' 00:00:00 '
DATETIME	' 0000-00-00 00:00:00 '
TIMESTAMP	' 0000-00-00 00:00:00 '
YEAR	0000



# Database Modeling

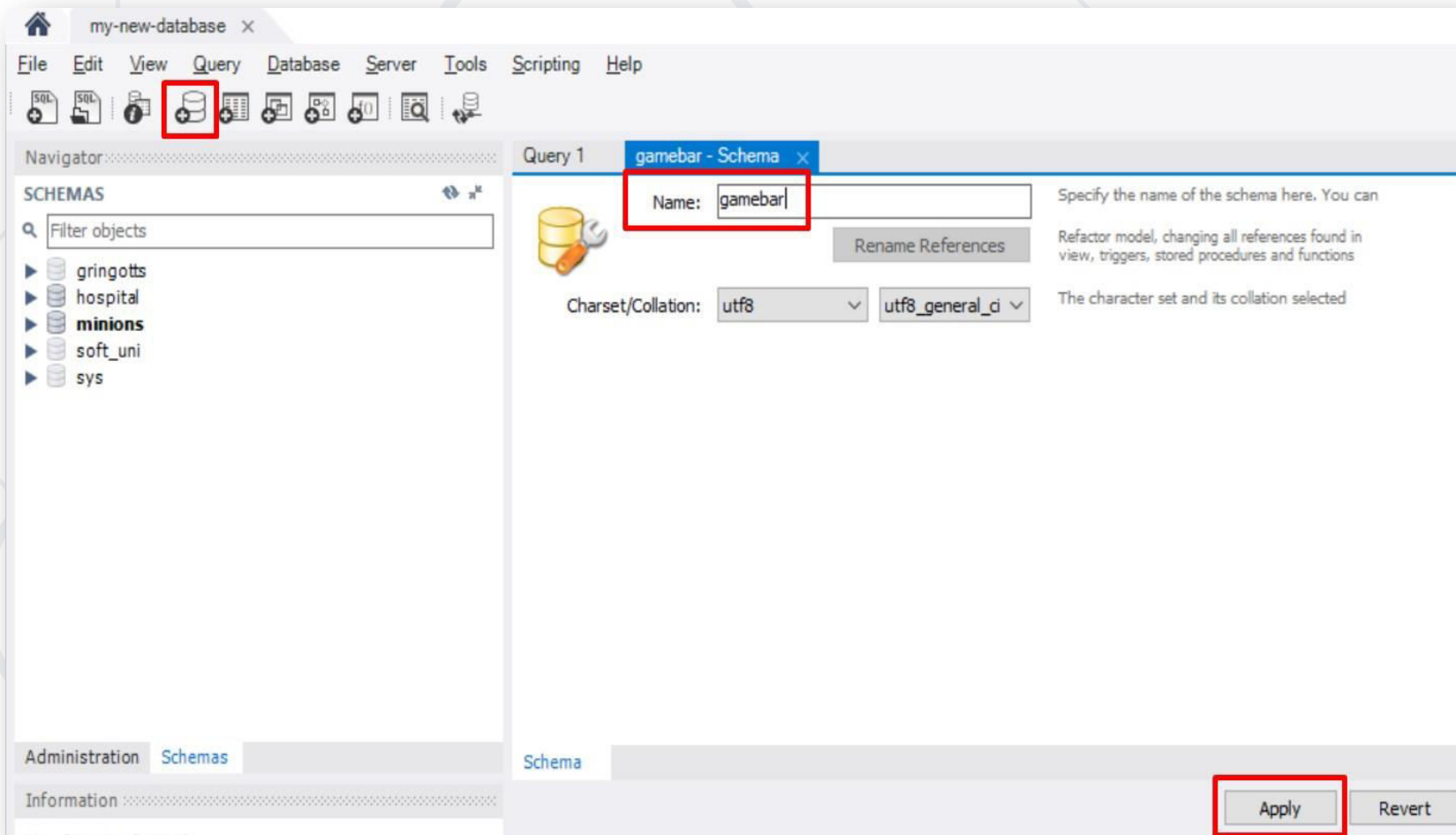
Data Definition Using GUI Clients

- We will **manage** databases with **MySQL Workbench**
- Enables us:
  - To **create** a new database
  - To create **objects in the database** (tables, stored procedures, relationships and others)
  - To **change** the properties of objects
  - To **enter records** into the tables



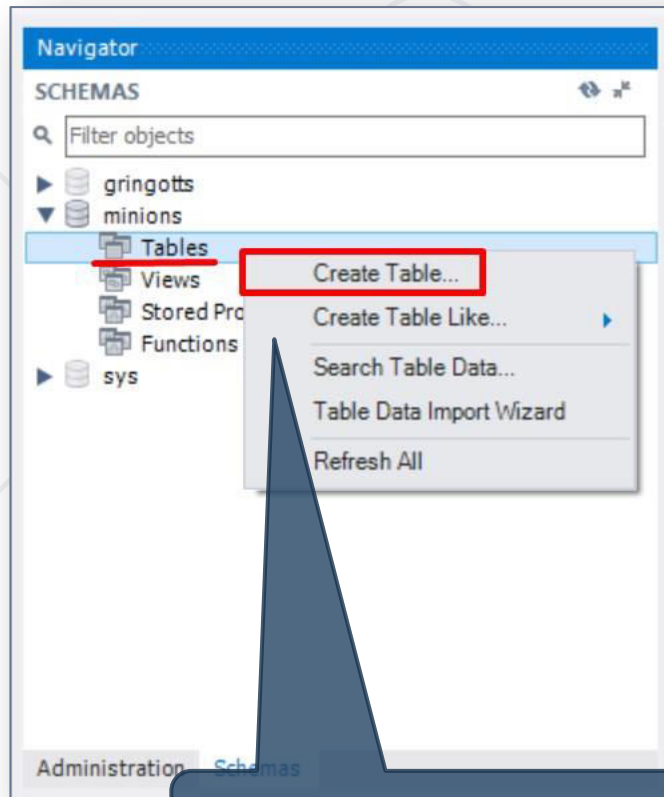
# Creating a New Database

- Select **Create new schema** from the **command menu**

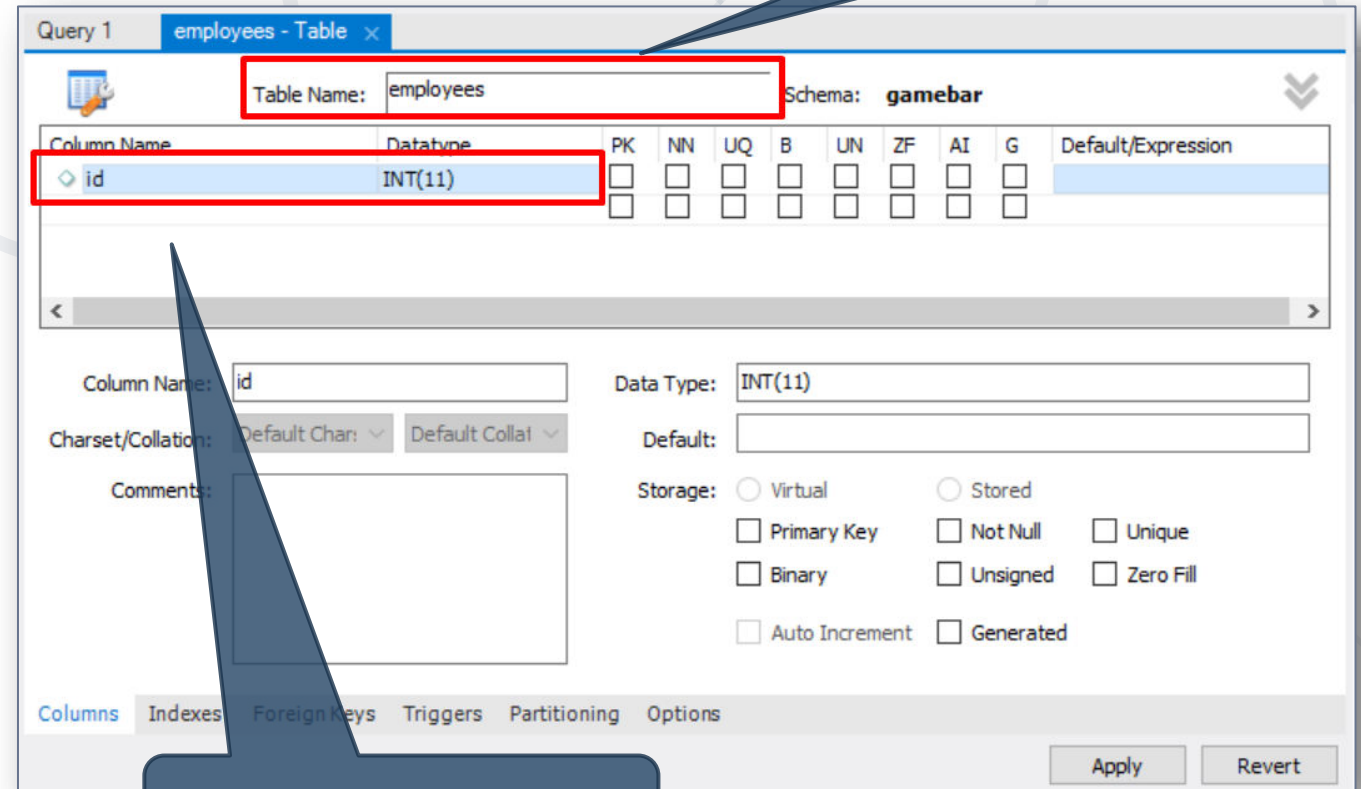


# Creating Tables

- Right click on “Tables” **Select Create Table**



Create new table



Set up table name

Add new record

# Creating Tables (2)

- A **Primary Key** is used to uniquely identify and index records

The screenshot shows a database management interface for creating a table named 'employees' in the 'gamebar' schema. The 'id' column is selected as the primary key, indicated by a red box around the 'PK' checkbox in the column list and another red box around the 'Primary Key' checkbox in the column properties section.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: id Data Type: INT(11)

Charset/Collation: Default Char: Default Collat

Comments:

Storage: ☒ Primary Key ☐ Not Null ☐ Unique ☐ Binary ☐ Unsigned ☐ Zero Fill ☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

# Creating Tables (3)

- Auto increment – on the "Default" field

Query 1 employees - Table x

Table Name: employees Schema: gamebar

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Column Name: id Data Type: INT(11)

Charset/Collation: Default Char: Default Collat

Comments:

Storage: ☐ Virtual ☐ Stored

☒ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☒ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

# Storing and Retrieving Data


- We can add, modify and read records with GUI Clients
- To insert or edit a record, click inside the cell

Query 1 employees - Table x

Table Name: employees Schema: gamebar

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	





```
CREATE TABLE people  
(  
  id INT NOT NULL,  
  email VARCHAR(50) NOT  
  NULL,  
  first_name VARCHAR(50),  
  last_name VARCHAR(50)  
);
```

# Basic SQL Queries

## Data Definition Using SQL

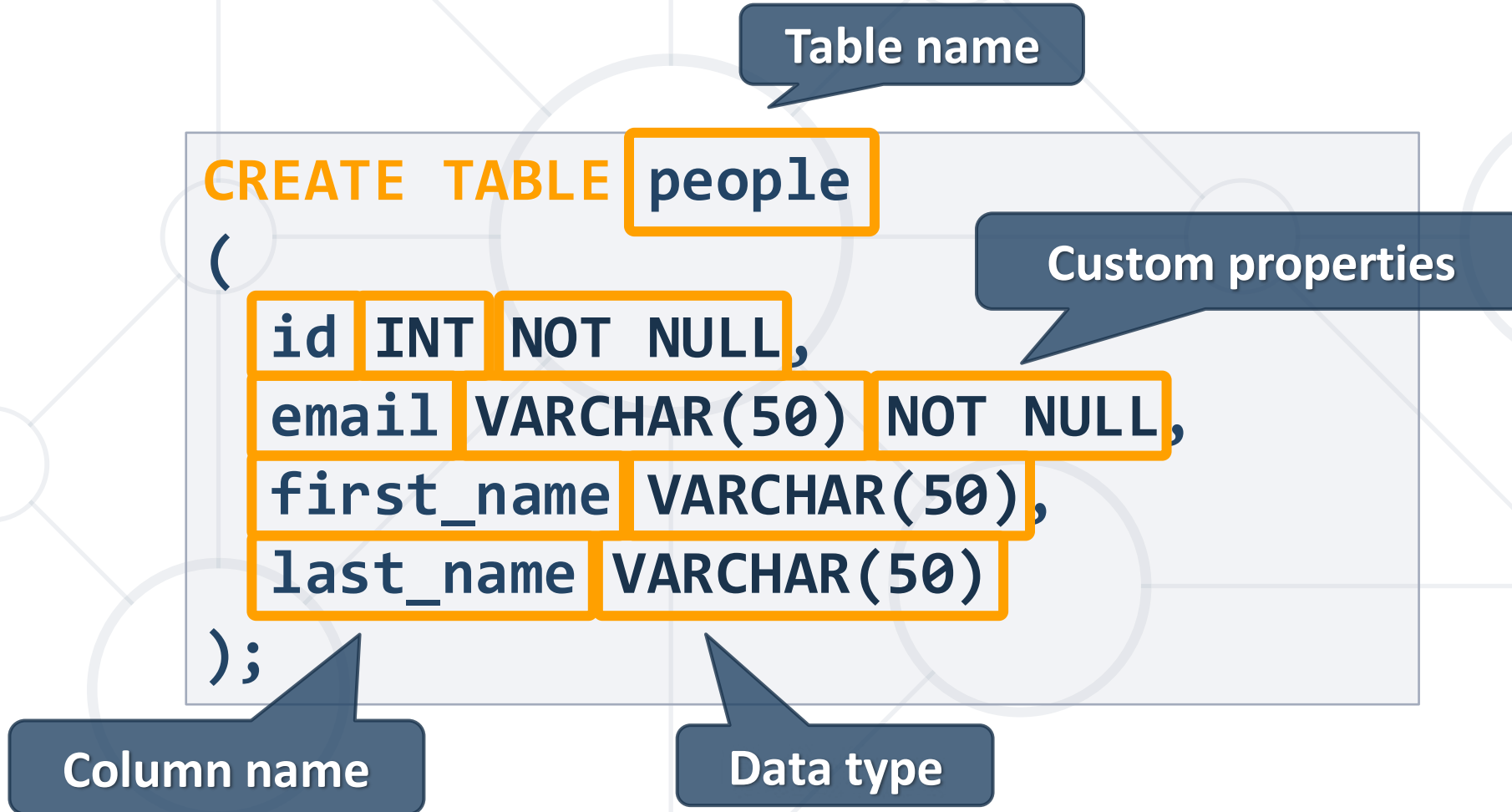
- We communicate with the database engine using SQL
- Queries provide greater **control** and **flexibility**
- To create a database using SQL:

Database name

```
CREATE DATABASE employees;
```

- SQL keywords are conventionally **capitalized**

# Table Creation in SQL



# Retrieve Records in SQL

- Get all information from a table

```
SELECT * FROM employees;
```

Table name

- You can limit the columns and number of records

```
SELECT first_name, last_name FROM employees  
LIMIT 5;
```

List of columns

Number of records



# Table Customization

Adding Rules, Constraints and Relationships

# Custom Column Properties

- Primary Key

```
id INT NOT NULL PRIMARY KEY
```

- Auto-Increment (Identity)

```
id INT AUTO_INCREMENT PRIMARY KEY
```

- Unique constraint – no repeating values in entire table

```
email VARCHAR(50) UNIQUE
```

- Default value – if not specified (otherwise set to **NULL**)

```
balance DECIMAL(10,2) DEFAULT 0
```

**Create** new Database "gamebar".

- **Create Tables:**

- "employees" – id, first\_name, last\_name
- "categories" – id, name
- "products" – id, name, category\_id

- **Insert** Data:

- Populate the "**employees**" table with 3 test values.



# Altering Tables

Changing Table Properties After Creation



- A table can be changed using the keywords **ALTER TABLE**

```
ALTER TABLE employees;
```

Table name

- Add new column

```
ALTER TABLE employees  
ADD salary DECIMAL;
```

Column name

Data type

# Altering Tables Using SQL (2)

- Delete existing column

```
ALTER TABLE people  
DROP COLUMN full_name;
```

Column name

- Modify data type of existing column

```
ALTER TABLE people  
MODIFY COLUMN email VARCHAR(100);
```

Column name

New data type

# Altering Tables Using SQL (3)

- Add primary key to existing column

```
ALTER TABLE people  
ADD CONSTRAINT pk_id  
PRIMARY KEY (id);
```

Constraint name

Column name  
(more than one for composite key)

- Add unique constraint

```
ALTER TABLE people  
ADD CONSTRAINT uq_email  
UNIQUE (email)
```

Constraint name

Columns name(s)

# Altering Tables Using SQL (4)

- Set default value

```
ALTER TABLE people  
ALTER COLUMN balance SET DEFAULT 0;
```

Default value

Column name

- **Alter table**
  - Add a new column – "middle\_name" to the "employees" table.
- **Adding Constraints**
  - Make "category\_id" **foreign key** linked to "id" in the "categories" table.
- **Modifying Columns**
  - Change the property "VARCHAR(50)" to "VARCHAR(100)" to the "middle\_name" column in "employees" table.



# Deleting Data and Structures

Dropping and Truncating

- Deleting structures is called **dropping**
  - You can drop **keys, constraints, tables** and entire **databases**
- Deleting all data in a table is called **truncating**
- Both of these actions **cannot be undone** – use with caution!

# Dropping and Truncating

- To delete all the entries in a table

```
TRUNCATE TABLE employees;
```

Table name

- To drop a table – delete data and structure

```
DROP TABLE employees;
```

Table name

- To drop entire database

```
DROP DATABASE soft_uni;
```

Database name



# Dropping and Truncating (2)

- To remove a constraining rule from a column
  - Primary keys, value constraints and unique fields

```
ALTER TABLE employess  
DROP CONSTRAINT pk_id;
```

Table name

Constraint name

- To remove **DEFAULT** value (if not specified, revert to **NULL**)

```
ALTER TABLE employess  
ALTER COLUMN clients  
DROP DEFAULT;
```

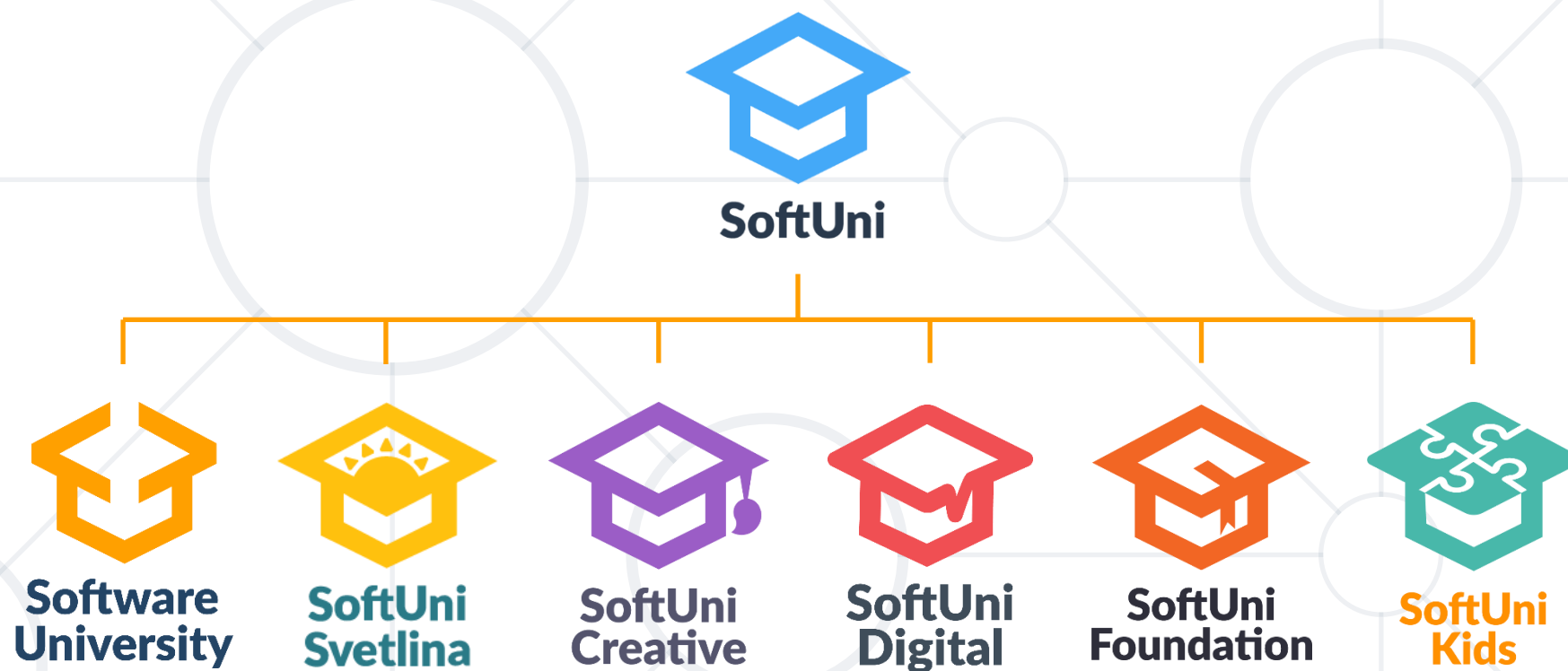
Table name

Columns name

- We communicate with the DB engine via **SQL**
- MySQL is a **multiplatform** RDBMS using SQL
- Table columns have a **fixed type**
- We can use GUI Clients to **create** and **customize** tables
- SQL provides **greater control**



# Questions?



# SoftUni Diamond Partners



**XS**software



**SBTech**  
*we know sports*



telenor



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето утре*

**SUPER  
HOSTING  
.BG**

**INDEAVR**

*Serving the high achievers*



**INFRAGISTICS®**



**STEMO®**  
*Computer Systems & Software*



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- 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)



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

