#### University of Miskolc Faculty of Mechanical Engineering and Informatics

#### Java Web Application Development Technology N13020008

# MySQL

#### Tamás Tompa, PhD

assistant professor Department of Information Technology University of Miskolc



### What is MySQL?

• MySQL is the most popular and a free **Open Source** 

#### • Relational Database Management System (RDBMS)

- allows users to create, manage, and interact with relational databases
- in an RDBMS, data is stored in **tables** (rows and columns) where **relationships** between tables are **defined by keys** (like primary and foreign keys)
- An RDBMS system stores the data in the form of tables that might be related to each other



### What is MySQL?

- MySQL uses Structured Query Language (SQL) to store, manage and retrieve data, and control the accessibility to the data
- It is one of the best RDBMS being used for developing webbased software applications
- MySQL is written in C and C++
- Developed by Michael Widenius & David Axmark beginning in 1994



# Example

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00

SELECT \* FROM CUSTOMERS WHERE AGE = 25;

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
4	Chaitali	25	Mumbai	6500.00

#### What is database?



- Database is used to store a collection of data (which can either be structured or unstructured)
  - each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds
- Nowadays, we use **relational database management systems** (**RDBMS**) to store and manage huge volume of data

### What is RDBMS?



#### • A Relational DataBase Management System (RDBMS)

- enables you to implement a database with tables, columns and indexes
- guarantees the Referential Integrity between rows of various tables.
- updates the indexes automatically
- interprets an SQL query and combines information from various tables
- Elements:
  - 0 tables
  - o column
  - o row
  - o redundancy
  - o primary Key
  - o foreign Key
  - o etc.



# **RDBMS** Terminology

- Database A database is a collection of tables, with related data
- **Table** A table is a matrix with data. A table in a database looks like a simple spreadsheet
- **Column** One column (data element) contains data of one and the same kind, for example the column postcode
- **Row** A row (= tuple, entry or record) is a group of related data, for example the data of one subscription
- **Redundancy** Storing data twice, redundantly to make the system faster



# **RDBMS** Terminology

- **Primary Key** A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row
- Foreign Key A foreign key is the linking pin between two tables
  - "parent/child" relationship
  - can be make a specific column in the "child" table a foreign key that references a specific column in the "parent" table
- Compound Key A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique
- Index An index in a database resembles an index at the back of a book
- **Referential Integrity** Referential Integrity makes sure that a foreign key value always points to an existing row



# **RDBMS** Terminology





# Primary key and Foreign key

	users		orders			
user_id	_id email name		order_no	user_id	product_sku	
10	sadio@example.com	Sadio	93	11	123	
11	mo@example.com	Mohamed	94	11	789	
12	rinsola@example.com	Rinsola	95	13	789	
13	amalie@example.com	Amalie	96	10	101	

A row can only be added or updated in the **orders** table if the value in **orders.user\_id** matches an existing user ID in the **users** table.

This type of database rule is called a foreign key constraint.

### **RDBMS** visualization



- Entity Relationship (ER) Diagram can be used the graphical representation of relationships between tables
- Flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system
  - tables are represented as entities, and relationships between them are shown with connecting lines that indicate foreign keys and constraints, which help in understanding how tables relate to each other in the database

#### • Elements:

- entity
- attribute
- relationship
- primary Key (PK)
- foreign Key (FK)

### **RDBMS** visualization



#### • ER diagram elements

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse	$\bigcirc$	Attributes in ER Model
Diamond	$\diamond$	Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity



### ER diagram example



0 more information: <u>https://www.geeksforgeeks.org/introduction-of-er-model/</u>

### ER diagram example



Entity Relationship Diagram - Internet Sales Model





### Environment settings

#### • Download and install

• <u>https://dev.mysql.com/downloads/installer/</u>

Windows (x86, 32-bit), MSI Installer	8.0.40	306.4M	Download
(mysql-installer-community-8.0.40.0.msi)		MD5: 8c1bf3a205d5e191e36dc334a1	9f55d2   Signature

- Start MySQL server
  - mysqld command in the command prompt (run as an administrator)

- Services.msc, mysql service ...
- Setting Up a MySQL User Account





- **USE database\_name** This will be used to select a database in the MySQL
- SHOW DATABASES Lists out the databases that are accessible by the MySQL DBMS
- **SHOW TABLES** Displays the list of the tables in the current database
- **SHOW COLUMNS FROM** *table\_name:* Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table
- SHOW INDEX FROM table\_name Presents the details of all indexes on the table, including the PRIMARY KEY
- SHOW TABLE STATUS LIKE table\_name\G Reports details of the MySQL DBMS performance and statistics

# Most Important SQL Commands



- **SELECT** extract data from a database
- **UPDATE** update data in a database
- **DELETE** delete data from a database
- **INSERT INTO** insert new data into a database
- **CREATE DATABASE** create a new database
- ALTER DATABASE modify a database
- **CREATE TABLE** create a new table
- ALTER TABLE modify a table
- **DROP TABLE** delete a table
- **CREATE INDEX** create an index (search key)
- **DROP INDEX** delete an index



# Java connector to MySQL

- To communicate with databases Java provides a library known as JDBC (Java Database Connectivity)
- JDBC provides a set of classes and methods specifically designed for database connectivity, enabling Java developers to perform tasks such as establishing connections, executing queries, and managing data in MySQL databases
- Need to use a JDBC (Java Database Connectivity) driver to connect your Java application to a MySQL database
- Have to download MySQL Connector
  - mysql-connector-j-9.1.0.jar
  - https://dev.mysql.com/downloads/connector/j/
  - https://dev.mysql.com/downloads/file/?id=534782



# Java connector to MySQL





# JDBC methods

DriverManager.getConnection(String url, String user, String password)	Establishes a connection to the database using the specified URL, username, and password
createStatement()	Creates a Statement object for executing SQL queries
executeQuery(String sql)	Executes a SQL SELECT query and returns a ResultSet object containing the result set
executeUpdate(String sql)	Executes a SQL INSERT, UPDATE, DELETE, or other non-query statement
next()	Moves the cursor to the next row in the result set. Returns true if there is a next row, false otherwise
getInt(String columnLabel)	Retrieves the value of the specified column in the current row of the result set



# JDBC methods

prepareStatement(String sql)	Creates a PreparedStatement object for executing parameterized SQL queries
setXXX(int parameterIndex, XXX value)	Sets the value of a specified parameter in the prepared statement
<pre>executeQuery(), executeUpdate()</pre>	Execute the prepared statement as a query or update
setAutoCommit(boolean autoCommit)	Enables or disables auto-commit mode
commit()	Commits the current transaction
rollback()	Rolls back the current transaction



# JDBC steps

- 1. Load the JDBC driver specific to your database
- 2. Create a connection to the database using DriverManager.getConnection()
- 3. Create a "Statement" or "PreparedStatement" for executing SQL queries
- 4. **Use executeQuery()** for SELECT queries, or executeUpdate() for other statements
- 5. Iterate through the "ResultSet" to process the retrieved data
- 6. **Close "ResultSet"**, "Statement", and "Connection" to release resources
- 7. Wrap database code in try-catch blocks to handle exceptions
- 8. Use transactions if performing multiple operations as a single unit







## JDBC example class

import java.sql.Connection; import java.sql.DriverManager; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Statement;

```
public class DatabaseInteractionExample {
```

```
public static void main(String[] args) {
    try {
        // Load JDBC Driver
        Class.forName("com.mysql.cj.jdbc.Driver");
        // Load JDBC Driver");
        // Load JDBC Driver Driver");
        // Load JDBC Driver Driver Driver");
        // Load JDBC Driver Driver Driver Driver");
        // Load JDBC Driver Driv
```

```
// Connect to Database
Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database",
"your username", "your password");
```

```
// Execute Query
<u>Statement</u> statement = connection.createStatement();
<u>ResultSet</u> resultSet = statement.executeQuery("Your SQL Query");
```



# JDBC example class

}

}

```
// Process Results
    while (resultSet.next()) {
    // Process data
     }
    // Close Resources
    resultSet.close();
    statement.close();
    connection.close();
// Handle Exceptions
} catch (ClassNotFoundException | SQLException e) {
e.printStackTrace();
```

#### Variables



• Main purpose of a variable is to label a memory location(s) and store data in it so that it can be used throughout the program

- In MySQL, there are three types of variables:
  - User-Defined Variable
  - Local Variable
  - System Variables



#### Variables

#### • User-Defined Variable

- allows us to store a value in one statement and subsequently refer to it in another
- these variable names will have the symbol "@" as a prefix

**SELECT** @variable name = **value** 

SELECT @max\_salary := MAX(salary) FROM CUSTOMERS;

# MySQL

### Variables

#### • Local Variable

- local variable can be declared using the **DECLARE** keyword
- strongly typed variable, which means that we definitely need to declare a data type

```
DECLARE variable_name1, variabale_name2, ...
data type [DEFAULT default value];
```

DELIMITER // CREATE PROCEDURE salaries() BEGIN DECLARE Ramesh INT; DECLARE Ramesh INT; DECLARE Khilan INT DEFAULT 30000; DECLARE Kaushik INT; DECLARE Chaitali INT; DECLARE Total INT; SET Ramesh = 20000; SET Kaushik = 25000; SET Chaitali = 29000; SET Total = Ramesh+Khilan+Kaushik+Chaitali; SELECT Total,Ramesh,Khilan,Kaushik,Chaitali;

# MySQL

### Variables

#### • System Variables

- contains the data we need, to work with the database
- the SET command in MySQL can be used at the runtime to dynamically change the values of the system variables
- there are two variable scope modifiers
  - The GLOBAL variables are active throughout the lifecycle
  - The SESSION variables can be available only in the current session

SHOW [GLOBAL | SESSION] VARIABLES;

SHOW VARIABLES LIKE 'Stables';



### Database connection

- Must first establish a connection between client and the databse
- Connection parameters: consisting of a username and a password
- Set Password to MySQL Root mysql -u root password "new\_password";
- Reset Password

SET PASSWORD FOR 'root'@'localhost' = PASSWORD('password name'); FLUSH PRIVILEGES;

• Connect to the MySQL server from the command prompt mysql -u root -p



X

### MySQL server starting

#### • mysqld command (in the cmd)

Parancssor - mysqld

Microsoft Windows [Version 10.0.19045.5011] (c) Microsoft Corporation. Minden jog fenntartva.

C:\Users\Tompa\_Tamas>mysqld

Szolgáltatások

🖅 Futtatás 🛛 🗙						
	Adja meg a program, a mappa, a dokume internetes erőforrás nevét, és a Windows r	entum vagy az megnyitja azt.				
<u>M</u> egnyit	tás: services.msc	~				
	OK Mégse	T <u>a</u> llózás				

Szolgáltatások (he	Szolgáltatások (helyi)					
	MySQL80	Név	Leírás	Állapot	Indítási típus	Bejelent ^
	Alter and Antonio	🧟 Microsoft Office Click-to-Ru	Mana	Fut	Automatikus	Helyi rer
	A szolgáltatás indítása	Alicrosoft Passport	Folya	Fut	Manuális (ind	Helyi rer
		🍓 Microsoft Passport-tároló	A felh	Fut	Manuális (ind	Helyi szc
		Alicrosoft Tárolóhelyek tárfe	A Micr		Kézi	Hálózati
		Alicrosoft Update Health Ser	Maint		Letiltva	Helyi rer
		Microsoft Windows SMS-út	A meg		Manuális (ind	Helyi szc
		🤹 Microsoft-fiók bejelentkezés	A Micr	Fut	Manuális (ind	Helyi rer
		🍓 Minőségi audiovizuális Win	A qWa		Kézi	Helyi szc
		🍓 Mobilhálózati idő	Ez a sz		Manuális (ind	Helyi szc
		🧟 Mozilla Maintenance Service	A Moz		Kézi	Helyi rer
		🍓 Munkaállomás	Ügyfel	Fut	Automatikus	Hálózati
		🦓 Munkahelyi mappák	Ez a sz		Kézi	Helyi szc
		MySQL80			Kézi	Hálózati
	I	······································	1011010-001			101



 $\times$ 

### MySQL command line client

#### MySQL 8.0 Command Line Client

Enter password: \*\*\*\* Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 27 Server version: 5.6.20-log MySQL Community Server (GPL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>



# MySQL workbench

- The MySQL workbench is a graphical tool for working with MySQL servers and databases
- It is developed and maintained by Oracle
- This application includes various features such as data modelling, data migration, SQL development, server administration, database backup, database recovery and many more
- Features:
  - SQL Development
  - Data modelling
  - Server adminstration
  - Data migration



# MySQL workbench

MySQL Workbench					- 0
new connection ×					
File Edit View Query Database Serve	r Tools Scripting Help 00   1995				0
Navigator Query 1	SQL File 3 Administration - Server Status ×				SQLAdditions
MANAGEMENT					◄ ▶   🛐 👫 2   Jump to
<ul> <li>Server Status</li> <li>Client Connections</li> <li>Users and Privileges</li> <li>Status and System Variables</li> <li>Data Export</li> <li>Data Import/Restore</li> <li>INSTANCE S</li> <li>Startup / Shutdown</li> <li>Server Logs</li> <li>Options File</li> </ul>	Connection Name  New Connection  Host: Tompa-IIT-HP Socket: /tmp/mysql.so Port: 3306 Version: 5.6.20-log (Mys Compiled For: Win32 (x86) Configuration File: unknown Running Since: Tue Oct 29 18:0	ock SQL Community Server (GPL)) D0:08 2024 (0:01) Refresh	Server Status     CPU/Load       Running     0%       Traffic     3.01 KB/s	Connections 4 Key Efficiency 0.0%	Automatic context help disabled. Use the toolba manually get help for t current caret position of toggle automatic help
PERFORMANCE Dashboard Performance Reports Performance Schema Setup Memc. Semisi SSL Av	ilable Server Features mance Schema: On d Pool: N/a ached Plugin: N/a ync Replication Plugin: N/a vallability: Off	Windows Authentication:       Off         Password Validation:       n/z         Audit Log:       n/z         Firewall:       n/z         Firewall Trace:       n/z	Selects per Second	InnoDB Buffer Usage 5.1%	
Administration Schamas Ser	ver Directories				
Information Data [	Directory: C:\TT\Egyetem\CRM_Geothermal_Hou Directory: C:\TT\Egyetem\CRM_Geothermal_Hou	rizon_Europe2023.04-09\UwAmp\bin\database rizon_Europe2023.04-09\UwAmp\bin\database		0	
No object selected	pace in Data Dir: 211.69 GB of 475.81 GB available	Me	essage		Context Help Snippets Duration / Fetc



#### Create Database

• The **CREATE DATABASE** statement is a DDL (Data Definition Language) statement used to create a new database in MySQL RDBMS

**CREATE DATABASE** DatabaseName;

CREATE DATABASE hello world;

- Verification
  - once the database TUTORIALS is created, you can check it in the list of databases using the SHOW statement

SHOW DATABASES;

• If the database is existing then an error will be generated

CREATE DATABASE IF NOT EXISTS myDatabase

• In Java:

String sql = "CREATE DATABASE DatabaseName"; st.executeUpdate(sql);



Х



MySQL 8.0 Command Line Client

Enter password: \*\*\*\* Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 1 Server version: 5.6.20-log MySQL Community Server (GPL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.


#### Create Database



• The created database can be check in the MySQL Workbench

MySQL Workbench										
new connection ×										
File Edit View Query	Database	Server	Tools	Scripting	Help					
	<b>•</b>	<b>d</b> i i i i i i i i i i i i i i i i i i i	<b>₩</b> ₽							
Navigator			SQL F	ile 3						
SCHEMAS	0		1	F 🙇 🖸	9	🛛 🔀 I	Limit to 1000 rows	•   🏡   🕩	0, 1	4
Q Filter objects		1								
<ul> <li>crm_db</li> <li>hello_world</li> <li>Tables</li> <li>Views</li> <li>Stored Procedures</li> <li>Functions</li> </ul>										



#### Create Database (Java)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class createDatabase {
    public static void main(String[] args) {
         String url = "jdbc:mysql://localhost:3306/";
         String user = "root";
         String password = "password";
         ResultSet rs;
         try {
              Class.forName("com.mysql.cj.jdbc.Driver");
              Connection con = DriverManager.getConnection(url, user, password);
              Statement st = con.createStatement();
              //System.out.println("Connected successfully...!");
              String sql = "CREATE DATABASE TUTORIALS";
              st.execute(sql);
              System.out.println("Database created successfully...!");
         } catch(Exception e) {
             e.printStackTrace();
         }
    }
```



## Create Database (Python)

import mysql.connector

```
# creating the connection object
connection = mysql.connector.connect( host ="localhost", user ="root", password ="password" )
```

```
# creating cursor object
cursorObj = connection.cursor()
```

# creating the database cursorObj.execute("CREATE DATABASE MySqlPythonDB")
print("Database Created Successfully")

# disconnecting from server
connection.close()



### Drop Database

• The **DROP DATABASE** statement in MySQL is used to delete a database along with all the data such as tables, views, indexes, stored procedures, and constraints

DROP DATABASE DatabaseName;

DROP DATABASE TUTORIALS;

• Verification

SHOW DATABASES;

• Dropping a Database using mysqladmin

mysqladmin -u root -p drop DatabaseName

o Java

String sql = "DROP DATABASE DatabaseName;";
st.execute(sql);



## Drop Database

MySQL 8.0 Command Line Client	_	×
mysql> drop database hello_world; Query OK, 0 rows affected (0.01 sec)		^
mysql> show databases;		
Database		
<pre>information_schema   crm_db   mysql   performance_schema   </pre>		
mysql>		



## Drop Database (Java)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class DropDatabase {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/tutorials";
        String user = "root";
        String password = "password";
        ResultSet st;
        try {
           Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, user, password);
            Statement st1 = con.createStatement();
            //System.out.println("Connected successfully...!");
            String sql = "DROP DATABASE TUTORIALS";
            st1.execute(sql);
            System.out.println("Database dropped successfully...!");
         }catch(Exception e) {
            e.printStackTrace();
         }
     }
}
```

#### Select database



- Once you get connected with the MySQL server, it is required to select a database to work with
  - this is because there might be more than one database available with the MySQL Server
- To select a database in MySQL, can be use the SQL USE statement

USE DatabaseName;

USE hello world;

o Java

String sql = "USE Database\_name";
st.execute(sql);



 $\Box$ 

 $\times$ 



MySQL 8.0 Command Line Client

Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 3 Server version: 5.6.20-log MySQL Community Server (GPL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql≻ create database hello\_world; Query OK, 1 row affected (0.00 sec)

mysql> show databases;

```
+----+
| Database |
+----+
| information_schema |
| crm_db |
| hello_world |
| mysql |
| performance_schema |
+----+
5 rows in set (0.00 sec)
mysql> use hello world;
```

Database changed

mysql>

\_\_\_\_\_



#### Select database (Java)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class SelectDatabase {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/";
        String user = "root";
        String password = "password";
        System.out.println("Connecting to select database....!");
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, user, password);
            Statement st1 = con.createStatement();
            String sql = "USE TUTORIALS";
            st1.execute(sql);
            System.out.println("Database selected successfully...!");
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```



#### Show database (Java)

import java.sql.Connection; import java.sql.DriverManager; import java.sql.ResultSet; import java.sql.Statement;

```
public class ShowDatabase {
    public static void main(String[] args) {
         String url = "jdbc:mysql://localhost:3306/TUTORIALS";
         String user = "root";
         String password = "password";
         ResultSet rs;
         try {
           Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, user, password);
            Statement st1 = con.createStatement();
            //System.out.println("Database connected successfully...!");
           String sql = "SHOW DATABASES";
            rs = st1.executeQuery(sql);
           System.out.println("Show query executed successfully...!");
           System.out.println("Databases are: ");
           while(rs.next()) {
               String db = rs.getNString(1);
              System.out.println(db);
            }
         } catch(Exception e) {
            e.printStackTrace();
       }
```



#### Export database

- Exporting a database in MySQL is commonly used for backup purposes or transferring data between servers
- Can be export entire database or just a portion of it
- The simplest way of exporting a database is by using the **mysqldump** command-line tool

mysqldump -u username -p database\_name > output\_file\_path

- **username**: It is the MySQL username to use when connecting to the database
- **database\_name**: It is the name of the database to be exported
- **output\_file\_path**: It is the path of the backup file. This is where the backup data will be stored
- >: This symbol exports the output of the mysqldump command into a file named *output\_file\_path*

mysqldump -u root -p TUTORIALS > data-dump.sql



#### Import database

• In MySQL, to import an existing dump or backup file into a database, can use the **mysql** command-line tool

mysql -u username -p new database name < dumpfile path

- **username**: This is the MySQL username to use when connecting to the MySQL server
- **new\_database\_name**: The name of the database where you want to import the data
- **dumpfile\_path**: It is the path of the backup file. The data will be imported from this file
- <: This symbol **imports** the data from the file named *output\_file\_path*

mysql -u root -p testdb < **data**-dump.sql





Can create a new user account using the CREATE USER Statement in
To execute this statement, the current account must have the CREATE USER privilege or the INSERT privilege for the MySQL system schema

**CREATE USER** 'user name'@'host name' **IDENTIFIED BY** 'password';

- **user\_name** is the name of the user you need to create
- hostname specifies the host from which the user can connect
- **password** is the user's password

CREATE USER 'sample'@'localhost' IDENTIFIED BY '123456';

• Verification

SELECT USER FROM MySQL.USER;

SELECT \* FROM mysql.user;

```
String sql = "SELECT USER FROM MYSQL.USER";
statement.executeQuery(sql);
```



#### Create users

MySQL 8.0 Command Line Client	_	×
mysql> create user 'tompa' identified by 'passwd'; Query OK, 0 rows affected (0.00 sec)		^
<pre>mysql&gt; select user from mysql.user; ++   user   ++   tompa     root     uwamp   ++ 3 rows in set (0.00 sec)</pre>		
mysql>		



#### Create users (Java)

import java.sql.Connection; import java.sql.DriverManager; import java.sql.Statement;

#### public class CreateUsers {

```
public static void main(String[] args) {
    String url = "jdbc:mysql://localhost:3306/TUTORIALS";
```

```
String user = "root";
String password = "password";
```

```
try {
```

}

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection con = DriverManager.getConnection(url, user, password);
```

```
Statement st = con.createStatement();
```

```
//System.out.println("Database connected successfully...!");
```

```
String sql = "CREATE USER 'Vivek'@'localhost' IDENTIFIED WITH mysql_native_password
BY 'password'";
```

```
st.execute(sql);
System.out.println("User 'Vivek' created successfully...!");
Catch(Exception e) {
    e.printStackTrace();
}
```



#### Drop users

- Can be drop/delete one or more existing users in MySQL using the DROP USER Statement
- Once you delete an account, all privileges of it are deleted
- To execute this statement, you need to have CREATE USER privilege

DROP USER [IF EXISTS] 'username'@'hostname';

DROP **USER TestUser**@localhost;

## Drop users

mysq1)



MySQL 8.0 Command Line Client	_	×
mysql>		^
mysql>		
mysql> drop user 'tompa';		
Query OK, 0 rows attected (0.00 sec)		
mysql> select user from mysql.user;		
++		
user		
++		
root		
uwamp		
++		
2 rows in set (0.00 sec)		
mysal>		$\sim$



## Drop users (Java)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class DropUsers {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/TUTORIALS";
        String user = "root";
        String password = "password";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, user, password);
            Statement st = con.createStatement();
            //System.out.println("Database connected successfully...!");
            String sql = "DROP USER 'Vivek'@'localhost'";
            st.execute(sql);
            System.out.println("User 'Vivek' dropped successfully...!");
        } catch(Exception e) {
            e.printStackTrace();
    }
}
```



#### Create table

- SQL is used to store data in the form of structured tables
- These tables consist of fields and records
  - a field represents a column that defines the type of data to be stored in a table, and a record is a row containing the actual data
- The table creation command requires the following details:
  - name of the table
  - name of the columns
  - definitions for each column

```
CREATE TABLE table_name(
    column1 datatype,
    column2 datatype,
    ....
    columnN datatype,
    PRIMARY KEY( one or more columns )
);
```

```
CREATE TABLE CUSTOMERS (
    ID INT AUTO_INCREMENT,
    NAME VARCHAR(20) NOT NULL,
    AGE INT NOT NULL,
    ADDRESS CHAR (25),
    SALARY DECIMAL (18, 2),
    PRIMARY KEY (ID)
);
```

Verification: desc tablename; desc customers; or: show tables;





🔜 Kijelölés MySQL 8.0 Command Line Client	_	
<pre>mysql&gt; CREATE TABLE CUSTOMERS (     -&gt; ID INT AUTO_INCREMENT,     -&gt; NAME VARCHAR(20) NOT NULL,     -&gt; AGE INT NOT NULL,     -&gt; AGE INT NOT NULL,     -&gt; ADDRESS CHAR (25),     -&gt; SALARY DECIMAL (18, 2),     -&gt; PRIMARY KEY (ID)     -&gt; ); Ouery OK, 0 rows affected (0.01 sec)</pre>		
mysql> show tables; ++	MySQL Workbench	
Tables_in_hello_world   ++	File Edit View Query Database	Server To
customers   ++		
1 row in set (0.00 sec)	Navigator	ery 1 🗙 💈
mysql> _	SCHEMAS Filter objects Crm_db Filter objects Filter objects Crm_db Crm	1



#### Create table

#### • AUTO\_INCREMENT

• automatically increments the value in the ID column by one for each new record you add. It starts from the next available number

#### • NOT NULL

- the field to be NULL.
- if a user tries to create a record with a NULL value in that field, then MySQL will raise an error

#### • PRIMARY KEY

- used to define a column as a primary key
- ensures that every record in that column is unique
- can be also use it for multiple columns by separating them with commas



#### Queries

- The queries in MySQL are commands that are used to retrieve or manipulate the data from a database table
  - SELECT, UPDATE, DELETE, INSERT INTO, CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE DATABASE, ALTER DATABASE, CREATE INDEX, DROP INDEX, etc

CREATE DATABASE tutorials;

USE tutorials;

CREATE TABLE CUSTOMERS (
 ID int,
 NAME varchar(20),
 AGE int,
 PRIMARY KEY (ID)
);



## Insert Query

• The MySQL insert query can be used to insert records within a specified table

INSERT INTO table\_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

INTO	CUSTOMERS	(ID,	NAME,	AGE)	VALUES	(1,	"Nikhilesh",	28);
INTO	CUSTOMERS	(ID,	NAME,	AGE)	VALUES	(2,	"Tomy", 36);	
INTO	CUSTOMERS	(ID,	NAME,	AGE)	VALUES	(3,	"Joe", 22);	
INTO	CUSTOMERS	(ID,	NAME,	AGE)	VALUES	(4,	"Kate", 21);	
INTO	CUSTOMERS	(ID,	NAME,	AGE)	VALUES	(5,	"Jonh", 42);	

```
INSERT INTO CUSTOMERS VALUES
(1, 'Ramesh', '32', 'Ahmedabad', 2000),
(2, 'Khilan', '25', 'Delhi', 1500),
(3, 'Kaushik', '23', 'Kota', 2500),
(4, 'Chaitali', '26', 'Mumbai', 6500),
(5, 'Hardik','27', 'Bhopal', 8500),
(6, 'Komal', '22', 'MP', 9000),
(7, 'Muffy', '24', 'Indore', 5500);
```

#### Insert Query

#### MySQL 8.0 Command Line Client

#### mysql> INSERT INTO CUSTOMERS VALUES

-> (1, 'Ramesh', '32', 'Ahmedabad', 2000), -> (2, 'Khilan', '25', 'Delhi', 1500), -> (3, 'Kaushik', '23', 'Kota', 2500), -> (4, 'Chaitali', '26', 'Mumbai', 6500), -> (5, 'Hardik','27', 'Bhopal', 8500), -> (6, 'Komal', '22', 'MP', 9000), -> (7, 'Muffy', '24', 'Indore', 5500); Query OK, 7 rows affected (0.00 sec) Records: 7 Duplicates: 0 Warnings: 0

#### mysql> select \* from customers;

+	+	+	+	++
ID	NAME	AGE	ADDRESS	SALARY
+	+	++		++
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2500.00
4	Chaitali	26	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	9000.00
7	Muffy	24	Indore	5500.00
+	+	+	+	++
7 row	s in set (0	.00 sec	:)	
	`		,	







# • The MySQL update query can be used to modify the existing records in a specified table

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

**UPDATE** CUSTOMERS SET NAME = "Nikhil" WHERE ID = 1;

MySQL 8.0 Command Line Client

Update Query

```
mysql> UPDATE CUSTOMERS SET NAME = "Nikhil" WHERE ID = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql>
```



#### Alter Query

#### • The ALTER query in MySQL can be used to **add, delete, or modify columns in an existing table**

ALTER TABLE table\_name [ADD|DROP] column name datatype;

**ALTER TABLE** CUSTOMERS **ADD COLUMN** EMAIL varchar(50);

mysql> ALTER TABLE CUSTOMERS ADD COLUMN EMAIL varchar(50); Query OK, 7 rows affected (0.01 sec) Records: 7 Duplicates: 0 Warnings: 0

mysql> select \* from customers;

ID	NAME	AGE	ADDRESS	SALARY	EMAIL
+   1   2   3   4   5   6   7	+   Nikhil   Khilan   Kaushik   Chaitali   Hardik   Komal   Muffy	32 25 23 26 27 27 22	Ahmedabad Delhi Kota Mumbai Bhopal MP	+ 2000.00 1500.00 2500.00 6500.00 8500.00 9000.00 5500.00	NULL   NULL   NULL   NULL   NULL   NULL
+ 7 row:	s in set (0.	.00 sec	:)	+	++





5500.00 muffy@example.com

## Alter Query

mysql Query Rows

mysq

| ID +----

| 1 | 2 | 4 | 5 | 6 | 7

	) Use	e the up	padte	SÇ	)L con	nn	and to add ema	ail	data	. tc	eac]	h re	cord	
U	PDATE	CUSTON	HERS S	SET	EMAIL	=	'ramesh@example	e.c	om '	WHE	ERE I	D =	1;	
U	PDATE	CUSTON	MERS S	SET	EMAIL	=	'khilan@example	e.c	om '	WHI	ERE I	D =	2;	
U	PDATE	CUSTON	MERS S	SET	EMAIL	=	'kaushik@exampl	Le.	com'	WI	HERE	ID	= 3;	
U	PDATE	CUSTON	1ers S	SET	EMAIL	=	'chaitali@examp	ble	.com	• 7	VHERE	L ID	= 4;	
U	PDATE	CUSTON	4ers s	SET	EMAIL	=	'hardik@example	e.c	om '	WHE	ERE I	D =	5;	
U	PDATE	CUSTON	4ers s	SET	EMAIL	=	'komal@example.	CO	m'W	HEI	RE ID	) =	6;	
U	PDATE	CUSTON	1ers S	SET	EMAIL	=	'muffv@example.	CO	m'W	HEI	RE ID	) =	7;	
> UPDATH OK, 1 m matched > select	E CUSTOMERS row affecte : 1 Change t * from cu	SET EMAIL = ed (0.00 sec) ed: 1 Warnin ustomers;	= 'muffy@e ) ngs: 0 +	example	≥.com' WHERE	ID =	<ul> <li>new connection ×</li> <li>File Edit View Query Database</li> <li>7; Time Edit View Query Database</li> <li>Navigator</li> <li>SCHEMAS</li> <li>Filter objects</li> <li>-+ Control of the product o</li></ul>	Server	Tools Sc iile 4* × Selec	ripting 7 🙊 t * fr	Help	<b>) ()</b> 5 j	Limit to 1000 rows	~
NAME	AGE	ADDRESS	SALARY	EMAI	[L 		▼ 🗟 hello_world ▼ 🖶 Tables							
Nikhi   Khila   Chaita   Hardi   Komal   Muffy +	l   32 n   25 ali   26 k   27   22   24	Ahmedabad Delhi Mumbai Bhopal MP Indore	2000.00 1500.00 6500.00 8500.00 9000.00 5500.00	rame   khi]   chai   hard   koma   muff	esh@example. Lan@example. Lali@example. Hik@example.c H@example.c	com com com com com	vistomers     Customers     Golumns     Golumns     Indexes     Foreign Keys     Triggers     Vews     Views     Views     ID     NAME	< <tr>         Result           1           2</tr>	It Grid   III D NAME Nikhil Khilan Chaitali	Filte AGE 32 25 26	ADDRESS Ahmedabad Delhi	SALARY 2000.00 1500.00	Edit: 20 10 10 10 10 10 10 10 10 10 10 10 10 10	Ex

EMAIL

....

(Dec)

7

NULL

Muffy

NULL

24

NULL

Indore

NULL



#### Delete Query

• The Delete query in MySQL can be used to **delete existing** records in a specified table

DELETE FROM table name WHERE condition;

**DELETE FROM** CUSTOMERS WHERE ID = 3;

nysql> DELETE FR( Query OK, 1 row a nysql> select * f	OM CUST affecte from cu	OMERS WHERE d (0.00 sec) stomers;	ID = 3; )	
++   ID   NAME	+   AGE	ADDRESS	++   SALARY	EMAIL
1   Nikhil	32	Ahmedabad	2000.00	NULL
2 Khilan	25	Delhi	1500.00	NULL
4 Chaitali	26	Mumbai	6500.00	NULL
5 Hardik	27	Bhopal	8500.00	NULL
6 Komal	22	MP	9000.00	NULL
7   Muffy	24	Indore	5500.00	NULL
++	++		++	+
5 rows in set (0.	.00 sec	)		



## Truncate Query

• The MySQL truncate table query can be used to **remove all the records** but not the table itself

**TRUNCATE** [**TABLE**] table name;

TRUNCATE TABLE CUSTOMERS;



mysql> truncate table customers; Query OK, 0 rows affected (0.00 sec)

mysql≻ select \* from customers; Empty set (0.00 sec)



## Drop Query

mvsal>

#### • The MySQL drop query is used to **delete an existing table** in a database

DROP TABLE table name;

DROP TABLE CUSTOMERS;

MySQL Workbench





#### Constraints

- The MySQL constraints can be used to set certain rules to the column(s) in a table
- These constraints can **restrict the type of data that can be inserted** or updated in a particular column
- There are two types of MySQL constraints:
  - **column level constraints:** These type of constraints will only apply to a column in a table
  - **table level constraints:** These constraints will apply to the complete table

```
CREATE TABLE table_name (
    Column_name1 datatype constraint,
    Column_name2 datatype constraint,
    Column_name3 datatype constraint,
    .....);
```



#### Constraints

- NOT NULL: cannot insert or update a record without adding a value
- **UNIQUE**: every value in a column must be distinct
- **PRIMARY KEY**: uniquely identify each record in a table. Can be define primary key on a particular column in a table, it must contain UNIQUE values, and cannot contain NULL values
- FOREIGN KEY: used to link a field or collection of fields in one table to the primary key of another table. A table with the foreign key is called a child table and the table with the primary key is called the parent table or referenced table



#### Constraints

- **CHECK**: restricts the range of values that can be inserted into a column
- **DEFAULT:** used to assign a default value to a specific column in a table
- **CREATE INDEX**: used to create indexes for one more columns in a table
- AUTO\_INCREMENT: defined on a particular column of a table, it will automatically generate a unique number when a new record is inserted into that column



#### View

- MySQL views are a type of virtual tables
- They are stored in the database with an associated name
- Structure data in a way that users or classes of users find natural or intuitive
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more
- Summarize data from various tables which can be used to generate reports

**CREATE VIEW** view\_name **AS** select\_statements **FROM** table\_name;

CREATE VIEW first view AS SELECT \* FROM CUSTOMERS;



mysql>				
SELECT * F	ROM fi	irst_view;		
NAME	AGE	ADDRESS	SALARY	EMAIL
Nikhil	32	Ahmedabad	2000.00	ramesh@example.com
Khilan	25	Delhi	1500.00	khilan@example.com
Chaitali	26	Mumbai	6500.00	chaitali@example.com
Hardik	27	Bhopal	8500.00	hardik@example.com
Komal	22	MP	9000.00	komal@example.com
Muffy	24	Indore	5500.00	<pre>muffy@example.com</pre>
	nysql> SELECT * F NAME Nikhil Khilan Chaitali Hardik Komal Muffy	<pre>nysql&gt; _ SELECT * FROM f: NAME   AGE Nikhil   32 Khilan   25 Chaitali   26 Hardik   27 Komal   22 Muffy   24</pre>	<pre>&gt;</pre>	<pre>nysql&gt; SELECT * FROM first_view; NAME   AGE   ADDRESS   SALARY Nikhil   32   Ahmedabad   2000.00 Khilan   25   Delhi   1500.00 Chaitali   26   Mumbai   6500.00 Hardik   27   Bhopal   8500.00 Komal   22   MP   9000.00 Muffy   24   Indore   5500.00</pre>

View







#### View




#### View

#### • Create a 2. table:

```
CREATE TABLE ORDERS (
    ORDER_ID INT AUTO_INCREMENT,
    CUSTOMER_ID INT,
    AMOUNT DECIMAL(18, 2),
    ORDER_DATE DATE,
    PRIMARY KEY (ORDER_ID),
    FOREIGN KEY (CUSTOMER_ID)
    REFERENCES CUSTOMERS(ID)
);
```

• Add records to the ORDERS table:

```
INSERT INTO ORDERS (CUSTOMER_ID, AMOUNT, ORDER_DATE) VALUES
(1, 500, '2024-01-15'),
(3, 1500, '2024-02-10'),
(5, 2000, '2024-03-05'),
(2, 800, '2024-04-20'),
(4, 2200, '2024-05-17');
```



#### View

• Create a new view from the CUSTOMERS and the ORDERS tables:

CREATE VIEW customer orders view AS SELECT

- c.ID AS CustomerID,
- c.NAME AS CustomerName,
- c.AGE AS CustomerAge,
- c.ADDRESS AS CustomerAddress,
- c.SALARY AS CustomerSalary,
- o.ORDER ID AS OrderID,
- o.AMOUNT AS OrderAmount,
- o.ORDER DATE AS OrderDate

#### FROM

CUSTOMERS C

#### LEFT JOIN

ORDERS O **ON** C.ID = O.CUSTOMER ID;



#### View

#### • The select query for the view (in mysql cmd):

<pre>iysql&gt; select * from customer_orders_view;</pre>											
CustomerName	CustomerAge	CustomerAddress	CustomerSalary	OrderID	OrderAmount	OrderDate					
Nikhil	32	Ahmedabad	2000.00	1	500.00	2024-01-15					
Hardik	27	Bhopal	8500.00	3	2000.00	2024-03-05					
Khilan	25	Delhi	1500.00	4	800.00	2024-04-20					
Chaitali	26	Mumbai	6500.00	5	2200.00	2024-05-17					
Komal	22	MP	9000.00	NULL	NULL	NULL					
Muffy	24	Indore	5500.00	NULL	NULL	NULL					
+ (0.00 sec)	+	+	+	++	+	++					
	<pre>* from customer CustomerName Nikhil Hardik Khilan Chaitali Komal Muffy (0.00 sec)</pre>	<pre>* from customer_orders_view;   CustomerName   CustomerAge   Nikhil   32   Hardik   27   Khilan   25   Chaitali   26   Komal   22   Muffy   24 +</pre>	<pre>* from customer_orders_view;   CustomerName   CustomerAge   CustomerAddress   Nikhil   32   Ahmedabad   Hardik   27   Bhopal   Khilan   25   Delhi   Chaitali   26   Mumbai   Komal   22   MP   Muffy   24   Indore</pre>	<pre>* from customer_orders_view;   CustomerName   CustomerAge   CustomerAddress   CustomerSalary   Nikhil   32   Ahmedabad   2000.00   Hardik   27   Bhopal   8500.00   Khilan   25   Delhi   1500.00   Chaitali   26   Mumbai   6500.00   Komal   22   MP   9000.00   Muffy   24   Indore   5500.00</pre>	<pre>* from customer_orders_view; CustomerName   CustomerAge   CustomerAddress   CustomerSalary   OrderID Nikhil   32   Ahmedabad   2000.00   1 Hardik   27   Bhopal   8500.00   3   Khilan   25   Delhi   1500.00   4   Chaitali   26   Mumbai   6500.00   5   Komal   22   MP   9000.00   NULL   Muffy   24   Indore   5500.00   NULL (0.00 sec)</pre>	<pre>* from customer_orders_view; CustomerName   CustomerAge   CustomerAddress   CustomerSalary   OrderID   OrderAmount Nikhil   32   Ahmedabad   2000.00   1   500.00   Hardik   27   Bhopal   8500.00   3   2000.00   Khilan   25   Delhi   1500.00   4   800.00   Chaitali   26   Mumbai   6500.00   5   2200.00   Komal   22   MP   9000.00   NULL   NULL   Muffy   24   Indore   5500.00   NULL   NULL   Muffy   24   Indore   5500.00   NULL   NULL</pre>					



#### • The select query for the view (in MySQL Workbench):

View

INIYSQL WORDCHCH								
new connection ×								
File Edit View Query Database S	ierver Tools S	Scripting Help						
Navigator	SQL File 4" ×							
SCHEMAS 🚸		F Q O I	80 0	Eimit to 1000	rows • 🏂	1 a	1 7	
Q Filter objects	1 • sele	ct * from cus	stomer order	s view;				
<ul> <li>hello_world</li> <li>Tables</li> <li>customers</li> <li>Columns</li> <li>Indexes</li> <li>Foreign Keys</li> <li>Triggers</li> </ul>	<result grid="" th=""  ="" 🕕<=""><th>Filter Rows:</th><th></th><th>Export:</th><th>Wrap Cell Content:</th><th>ĪĀ</th><th></th><th></th></result>	Filter Rows:		Export:	Wrap Cell Content:	ĪĀ		
<ul> <li>hello_world</li> <li>Tables</li> <li>customers</li> <li>Columns</li> <li>Indexes</li> <li>Foreign Keys</li> <li>Triggers</li> <li>orders</li> </ul>	<      Result Grid      CustomerID	<ul> <li>Filter Rows:</li> <li>CustomerName</li> </ul>	CustomerAge	Export:	Wrap Cell Content: CustomerSalary	1A OrderID	OrderAmount	OrderDate
<ul> <li>hello_world</li> <li>Tables</li> <li>customers</li> <li>Columns</li> <li>Columns</li> <li>Indexes</li> <li>Foreign Keys</li> <li>Triggers</li> <li>orders</li> <li>Views</li> </ul>	<     Result Grid     CustomerID     1	<ul> <li>Filter Rows:</li> <li>CustomerName</li> <li>Nikhil</li> </ul>	CustomerAge 32	Export: Export: CustomerAddress	Wrap Cell Content: CustomerSalary 2000.00	I I	OrderAmount 500.00	OrderDate 2024-01-15
<ul> <li>hello_world</li> <li>Tables</li> <li>customers</li> <li>Columns</li> <li>Columns</li> <li>Indexes</li> <li>Foreign Keys</li> <li>Foreign Keys</li> <li>Triggers</li> <li>orders</li> <li>Views</li> <li>customer_orders_view</li> <li>first view</li> </ul>	< <tr> Result Grid   CustomerID   1   5</tr>	<ul> <li>Filter Rows:</li> <li>CustomerName</li> <li>Nikhil</li> <li>Hardik</li> </ul>	CustomerAge 32 27	Export: Export: CustomerAddress Ahmedabad Bhopal	Wrap Cell Content: CustomerSalary 2000.00 8500.00	I 3	OrderAmount 500.00 2000.00	OrderDate 2024-01-15 2024-03-05
<ul> <li>hello_world</li> <li>Tables</li> <li>customers</li> <li>Columns</li> <li>Indexes</li> <li>Foreign Keys</li> <li>Foreign Keys</li> <li>Triggers</li> <li>orders</li> <li>Views</li> <li>customer_orders_view</li> <li>first_view</li> <li>Stored Procedures</li> </ul>	< <tr> Result Grid   CustomerID   1   5   2</tr>	<ul> <li>Filter Rows:</li> <li>CustomerName</li> <li>Nikhil</li> <li>Hardik</li> <li>Khilan</li> </ul>	CustomerAge 32 27 25	Export: Export: CustomerAddress Ahmedabad Bhopal Delhi	Wrap Cell Content: CustomerSalary 2000.00 8500.00 1500.00	TA OrderID 1 3 4	OrderAmount 500.00 2000.00 800.00	OrderDate 2024-01-15 2024-03-05 2024-04-20
<ul> <li>hello_world</li> <li>Tables</li> <li>customers</li> <li>Columns</li> <li>Indexes</li> <li>Foreign Keys</li> <li>Triggers</li> <li>orders</li> <li>Views</li> <li>customer_orders_view</li> <li>first_view</li> <li>Stored Procedures</li> <li>Functions</li> </ul>	< <tr> Result Grid   CustomerID   1   5   2   4</tr>	Filter Rows: CustomerName Nikhil Hardik Khilan Chaitali	CustomerAge 32 27 25 26	Export: Export: CustomerAddress Ahmedabad Bhopal Delhi Mumbai	Wrap Cell Content: CustomerSalary 2000.00 8500.00 1500.00 6500.00		OrderAmount 500.00 2000.00 800.00 2200.00	OrderDate 2024-01-15 2024-03-05 2024-04-20 2024-05-17
<ul> <li>hello_world</li> <li>Tables</li> <li>customers</li> <li>Columns</li> <li>Indexes</li> <li>Foreign Keys</li> <li>Foreign Keys</li> <li>Triggers</li> <li>orders</li> <li>Views</li> <li>customer_orders_view</li> <li>first_view</li> <li>Stored Procedures</li> <li>Functions</li> </ul>	< <tr>         Result Grid         I           CustomerID         1           5         2           4         6</tr>	Filter Rows:       CustomerName       Nikhil       Hardik       Khilan       Chaitali       Komal	CustomerAge 32 27 25 26 22	Export: Export: Customer Address Ahmedabad Bhopal Delhi Mumbai MP	Wrap Cell Content: CustomerSalary 2000.00 8500.00 1500.00 6500.00 9000.00	I 3 4 5	OrderAmount 500.00 2000.00 800.00 2200.00	OrderDate 2024-01-15 2024-03-05 2024-04-20 2024-05-17

# MySQL

## Join

- A Join clause in MySQL is used to combine records from two or more tables in a database
- These tables are joined together based on a condition, specified in a **WHERE clause** 
  - Inner Join: An Inner Join retrieves the intersection of two tables. It compares each row of the first table with each row of the second table. If the pairs of these rows satisfy the join-predicate, they are joined together. This is a default join.
  - **Outer Join**: An Outer Join retrieves all the records in two tables even if there is no counterpart row of one table in another table, like Inner Join. Outer join is further divided into three subtypes: Left Join, Right Join and Full Join. We will learn about these Joins later in this tutorial.

```
SELECT a.ID, a.NAME, b.DATE, b.AMOUNT
FROM CUSTOMERS a, ORDERS b
WHERE a.ID = b.CUSTOMER ID;
```



## Join





## More MySQL example

- O <u>https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL\_Beg</u> <u>inner.html</u>
- 0 <u>https://www.tutorialspoint.com/mysql/index.htm</u>
- o <u>https://www.w3schools.com/MySQL/default.asp</u>



## JDBC - query

#### • Query for the (**previously created Customers**) table

```
package mysql query;
1
2
                                                                                       32
                                                                                       33
3@import java.sql.Connection;
                                                                                       34
4 import java.sql.DriverManager;
                                                                                       35
5 import java.sql.ResultSet;
                                                                                       36
6 import java.sql.Statement;
                                                                                       37
7
                                                                                       38
8 public class DatabaseQueryExample {
                                                                                       39
9
                                                                                       40
10
       // Database URL, username, and password
                                                                                       41
       static final String DB_URL = "jdbc:mysql://localhost:3306/hello_world";
11
                                                                                       42
12
       static final String USER = "root";
                                                                                       43
13
       static final String PASS = "root";
                                                                                       44
14
                                                                                       45
15⊝
       public static void main(String[] args) {
                                                                                       46
                                                                                       47
           Connection conn = null;
16
17
           Statement stmt = null;
                                                                                       49
18
                                                                                       50
19
           try {
                                                                                       51
               // Step 1: Register JDBC driver (optional for newer JDBC versions)
20
                                                                                       52
21
               Class.forName("com.mysql.cj.jdbc.Driver");
                                                                                       53
22
                                                                                       54
23
               // Step 2: Open a connection
                                                                                       55
24
               System.out.println("Connecting to the database...");
                                                                                       56
25
               conn = DriverManager.getConnection(DB_URL, USER, PASS);
                                                                                       57
26
                                                                                       58
27
               // Step 3: Execute a query
                                                                                       59
                                                                                       60
28
               System.out.println("Creating statement...");
                                                                                       61
29
               stmt = conn.createStatement();
                                                                                       62
               String sql = "SELECT ID, NAME, AGE, ADDRESS, SALARY FROM CUSTOMERS"
30
                                                                                       63
31
               ResultSet rs = stmt.executeOuerv(sal):
                                                                                       64
20
                                                                                       65
```

```
// Step 4: Process the ResultSet
    while (rs.next()) {
        // Retrieve data by column name
        int id = rs.getInt("ID");
        String name = rs.getString("NAME");
        int age = rs.getInt("AGE");
        String address = rs.getString("ADDRESS");
        double salary = rs.getDouble("SALARY");
        // Display values
        System.out.print("ID: " + id);
       System.out.print(", Name: " + name);
        System.out.print(", Age: " + age);
        System.out.print(", Address: " + address);
       System.out.println(", Salary: " + salary);
    }
    // Step 5: Clean up the environment
    rs.close();
    stmt.close();
    conn.close();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    // Finally block to close resources
    try {
        if (stmt != null) stmt.close();
        if (conn != null) conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

}



## JDBC - query

• Query for the (**previously created Customers**) Customers table

🖹 Problems 🛛 🖉 Javadoc 🖳 Declaration 📮 Console 🗡

<terminated > DatabaseQueryExample [Java Application] C:\eclipse\plugins\org.eclipse.justj Connecting to the database...

Creating statement...

ID: 1, Name: Nikhil, Age: 32, Address: Ahmedabad, Salary: 2000.0

- ID: 2, Name: Khilan, Age: 25, Address: Delhi, Salary: 1500.0
- ID: 4, Name: Chaitali, Age: 26, Address: Mumbai, Salary: 6500.0
- ID: 5, Name: Hardik, Age: 27, Address: Bhopal, Salary: 8500.0
- ID: 6, Name: Komal, Age: 22, Address: MP, Salary: 9000.0
- ID: 7, Name: Muffy, Age: 24, Address: Indore, Salary: 5500.0



#### JDBC – create table

import java.sql.Connection; import java.sql.DriverManager; import java.sql.SQLException; import java.sql.Statement;

#### public class CreateTableExample {

public static void main(String[] args) {
 // JDBC URL, felhasználónév és jelszó
 String url = "jdbc:mysql://localhost:3306/hello\_world";
 String user = "root";
 String password = "root";

```
// Kapcsolat és Statement objektum
Connection connection = null;
Statement statement = null;
```

#### try {

```
connection = DriverManager.getConnection(url, user, password);
statement = connection.createStatement();
```

```
statement.executeUpdate(sql);
System.out.println("The students table created successfully!");
```

```
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.close();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
```

Problems @ Javadoc Declaration Console ×
<terminated > CreateTableExample [Java Application] C:\ecl
The students table created successfully!

• Create of students table in the hello\_world database

### JDBC - insert record



```
🖹 Problems @ Javadoc 😣 Declaration 📮 Console 🛛
 30 import java.sql.Connection;
 4 import java.sql.DriverManager;
 5 import java.sql.PreparedStatement;
 6 import java.sql.SQLException;
 7
 8
   public class InsertStudentsExample {
       private static final String URL = "jdbc:mysql://localhost:3306/hello world";
 9
       private static final String USER = "root";
10
11
       private static final String PASSWORD = "root";
12
       public static void main(String[] args) {
13⊝
           InsertStudentsExample example = new InsertStudentsExample();
14
           example.insertStudent("John Doe", 20);
15
           example.insertStudent("Jane Smith", 22);
16
           example.insertStudent("Emily Johnson", 19);
17
       }
18
19
20⊝
       public void insertStudent(String name, int age) {
21
           String sql = "INSERT INTO students (name, age) VALUES (?, ?)";
22
23
           try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
24
                 PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
25
26
               preparedStatement.setString(1, name);
27
               preparedStatement.setInt(2, age);
28
29
               int rowsAffected = preparedStatement.executeUpdate();
               if (rowsAffected > 0) {
30
                   System.out.println("Record inserted successfully: " + name + ", age: " + age);
31
32
                }
33
34
           } catch (SQLException e) {
35
               e.printStackTrace();
36
                                                   Inserting records in the Students table
           }
                                              Ο
```

37

38 }

}

<terminated> InsertStudentsExample [Java Application] C:\eclipse\plugi Record inserted successfully: John Doe, age: 20 Record inserted successfully: Jane Smith, age: 22 Record inserted successfully: Emily Johnson, age: 19

mysql	> select * from	students;
+   id	+   name	++   age
1	John Doe	20
2	Jane Smith	22
3	Emily Johnson	19
+	+	++
3 row	s in set (0.00 s	sec)



## JDBC – update record

```
public class UpdateStudentExample {
   private static final String URL = "jdbc:mysql://localhost:3306/hello_world";
   private static final String USER = "root";
   private static final String PASSWORD = "root";
   public static void main(String[] args) {
                                                                                                          mysql> select * from students;
       UpdateStudentExample example = new UpdateStudentExample();
       example.updateStudentAge(1, 25);
       example.updateStudentName(2, "Michael Brown");
                                                                                                             id
                                                                                                                  name
                                                                                                                                       age
   }
   // age update
                                                                                                                  John Doe
                                                                                                                                        25
                                                                                                              1
   public void updateStudentAge(int id, int newAge) {
                                                                                                                  Michael Brown
                                                                                                              2
                                                                                                                                        22
       String sql = "UPDATE students SET age = ? WHERE id = ?";
                                                                                                              3
                                                                                                                  Emily Johnson
                                                                                                                                        19
       try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
                                                                                                             4
                                                                                                                  John Doe
                                                                                                                                        20
            PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
                                                                                                                  Jane Smith
                                                                                                              5
                                                                                                                                        22
                                                                                                                  Emily Johnson
           preparedStatement.setInt(1, newAge);
                                                                                                              6
                                                                                                                                        19
          preparedStatement.setInt(2, id);
                                                                                                            rows in set (0.00 sec)
           int rowsAffected = preparedStatement.executeUpdate();
           if (rowsAffected > 0) {
              System.out.println("Successfully updated the record with ID: " + id + " New age: " + newAge);
          } else {
              System.out.println("The record doens't exist with this ID.");
       } catch (SQLException e) {
           e.printStackTrace();
                                                   📳 Problems @ Javadoc 😣 Declaration 📮 Console 🗙
   }
                                                   <terminated> UpdateStudentExample [Java Application] C:\eclipse\plugins\org.eclipse.just
                                                   Successfully updated the record with ID: 1 New age: 25
                                                   Successfully updated the record with ID: 2 New name: Michael Brown
```

• Updating record (with specified ID) in the Students table



age

22

19

20

22

19

## JDBC – delete record

```
public class DeleteStudentExample {
   private static final String URL = "jdbc:mysql://localhost:3306/hello world";
   private static final String USER = "root";
   private static final String PASSWORD = "root";
                                                                                           mysql> select * from students;
    public static void main(String[] args) {
       DeleteStudentExample example = new DeleteStudentExample();
       example.deleteStudentById(1);
                                                                                              id
                                                                                                   name
   }
                                                                                                 | Michael Brown
   // delete record by ID
                                                                                                   Emily Johnson
                                                                                               3
    public void deleteStudentById(int id) {
                                                                                                   John Doe
                                                                                               4
       String sql = "DELETE FROM students WHERE id = ?";
                                                                                                   Jane Smith
                                                                                               5
       try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
                                                                                                   Emily Johnson
               PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
                                                                                             rows in set (0.00 sec)
           preparedStatement.setInt(1, id);
           int rowsAffected = preparedStatement.executeUpdate();
           if (rowsAffected > 0) {
               System.out.println("Successfully deleted the record with ID: " + id);
           } else {
               System.out.println("The record with this ID doens't exist.");
           }
                                                            🖹 Problems 🛛 🖉 Javadoc 😣 Declaration 🗐 Console 🗙
       } catch (SQLException e) {
           e.printStackTrace();
                                                            <terminated> DeleteStudentExample [Java Application] C:\
       }
                                                            Successfully deleted the record with ID: 1
   }
```

}

Deleting record (with specified ID) from the Students table



## Task: Create a Web Application Mus

- Create a Web Application with Form to save data to databse and display data from the database
- Objective:
  - design a web application using Java Servlets, JSP, and MySQL
  - the application should allow users to submit data via a form and view the stored data in a tabular format

#### • Requirements:

- Database Setup
  - create a MySQL database called user\_management.
  - create a table users with the following columns:
    - id (INT, Primary Key, Auto Increment)
    - name (VARCHAR(100))
    - email (VARCHAR(100), Unique)
    - o age (INT)



### Task: Create a Web Application My

#### • Java Servlet and JSP Setup

- develop a Java Servlet called UserServlet
- use JSP for the frontend

#### • Application Functionalities

- Form Submission:
  - Create a form in a JSP page (user\_form.jsp) to collect user information (name, email, and age)
  - When submitted, the form should send the data to UserServlet, which will insert the data into the users table in MySQL
- Display Data:
  - Add a button on the user\_form.jsp page to display all user records
  - When this button is clicked, UserServlet should retrieve all records from the users table and display them in a tabular format on a new JSP page (user\_list.jsp)



#### Task: Create a Web Application Mys

**CREATE DATABASE** user\_management; **USE** user management;

```
CREATE TABLE users (
```

id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(100) UNIQUE NOT NULL, age INT NOT NULL );

mysql> show tables;
Tables_in_user_management
users
1 row in set (0.00 sec)

mysql> CREATE DATABASE user\_management; Query OK, 1 row affected (0.00 sec) mysql> USE user\_management; Database changed mysql> CREATE TABLE users ( id INT AUTO INCREMENT PRIMARY KEY, -> name VARCHAR(100) NOT NULL, email VARCHAR(100) UNIQUE NOT NULL, -> age INT NOT NULL ); -> Query OK, 0 rows affected (0.01 sec) mysql> show databases; Database information schema crm db hello world mysql performance schema user management



## Task: Create a Web Application Mu

$\leftarrow$	$\rightarrow$ C			alhost:8080/MySQL_us	ser_manageme	nt_06/use	rform.html				
G goo	gle 🕒 YouTube	奈 időkép	💛 aliExpress	💶 Tompa Tamás Posta n	💁 fordító	ncore	🛃 neptun				
User	· Registrat	tion									
Name: Email: Age: 3	Tamas Tompa tamas.tompa1@ 36	Duni-miskol									
Submit	t		$\leftarrow$	$\rightarrow$ C	$\bigcirc$	🗅 local	host:8080/I	MySQL_user_	_manageme	nt_06/Users	Servlet
View A	II Users		G goo	gle 🕒 YouTube 🔶	időkép 🔽 alil	Express 🧧	Tompa Tar	nás <mark>P</mark> osta n	💁 fordító	ncore	🔝 nept
			User r	egistered successf	ully!						





## Thank you for your attention!

thank you