

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



MySQL®

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



MySQL®

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 web-based 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



MySQL®

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:
 - tables
 - column
 - row
 - redundancy
 - primary Key
 - foreign Key
 - etc.



MySQL®

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 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



MySQL®

RDBMS Terminology

CustomerID	FirstName	LastName	Birthdate
XY001	John	Doe	April 18, 1929
BR092	Mary	Green	March 4, 1980
PD500	Francesca	de la Gillebert	September 12, 1959
WI308	John	Green	March 4, 1980

Diagram illustrating RDBMS terminology:

- Column (attribute)**: Points to the **FirstName** column.
- Table (relation)**: Points to the entire table structure.
- Row (tuple)**: Points to the first row (tuple).
- Primary key**: Points to the **CustomerID** column, which is highlighted with a red box.
- Data value**: Points to the value "Green" in the **LastName** column of the fourth row.



MySQL®

Primary key and Foreign key

users			orders		
user_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)



MySQL®

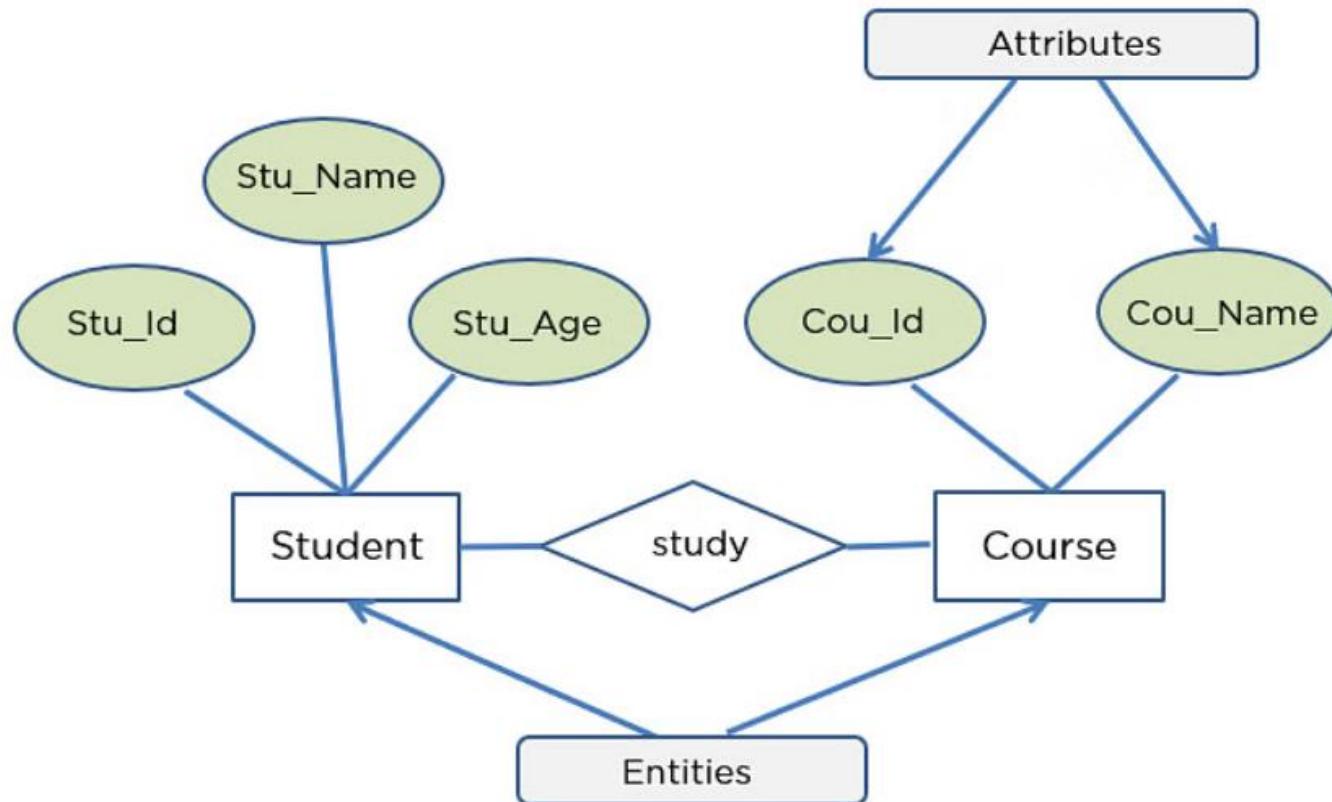
RDBMS visualization

○ ER diagram elements

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
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

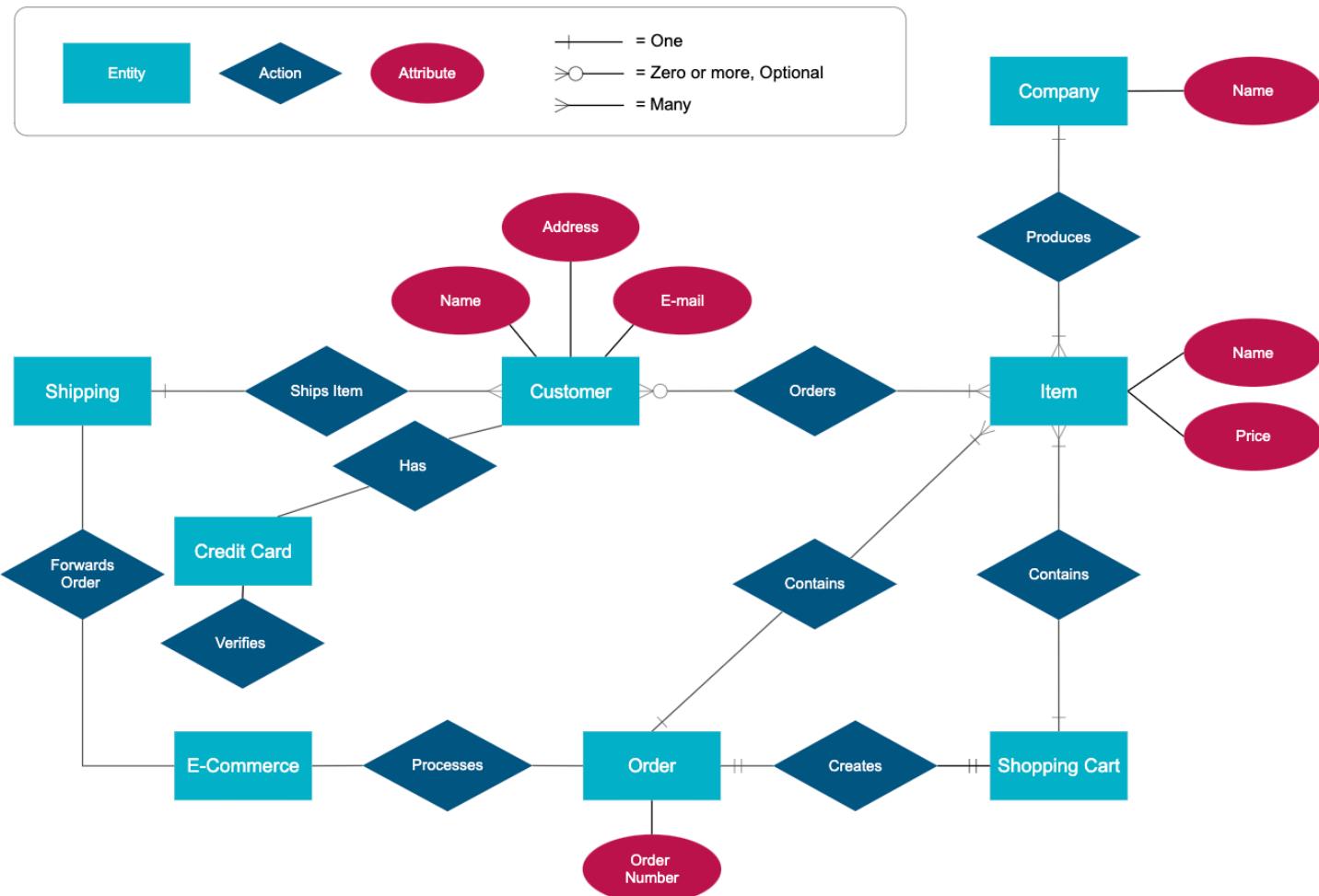


- more information: <https://www.geeksforgeeks.org/introduction-of-er-model/>



ER diagram example

Entity Relationship Diagram - Internet Sales Model





MySQL®

Environment settings

○ Download and install

- <https://dev.mysql.com/downloads/installer/>

Windows (x86, 32-bit), MSI Installer

8.0.40

306.4M

[Download](#)

(mysql-installer-community-8.0.40.0.msi)

MD5: 8c1bf3a205d5e191e36dc334a10f55d2 | [Signature](#)

○ Start MySQL server

- mysql command in the command prompt (run as an administrator)
- Services.msc, mysql service ...
- Setting Up a MySQL User Account

Administrative MySQL Commands



- **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

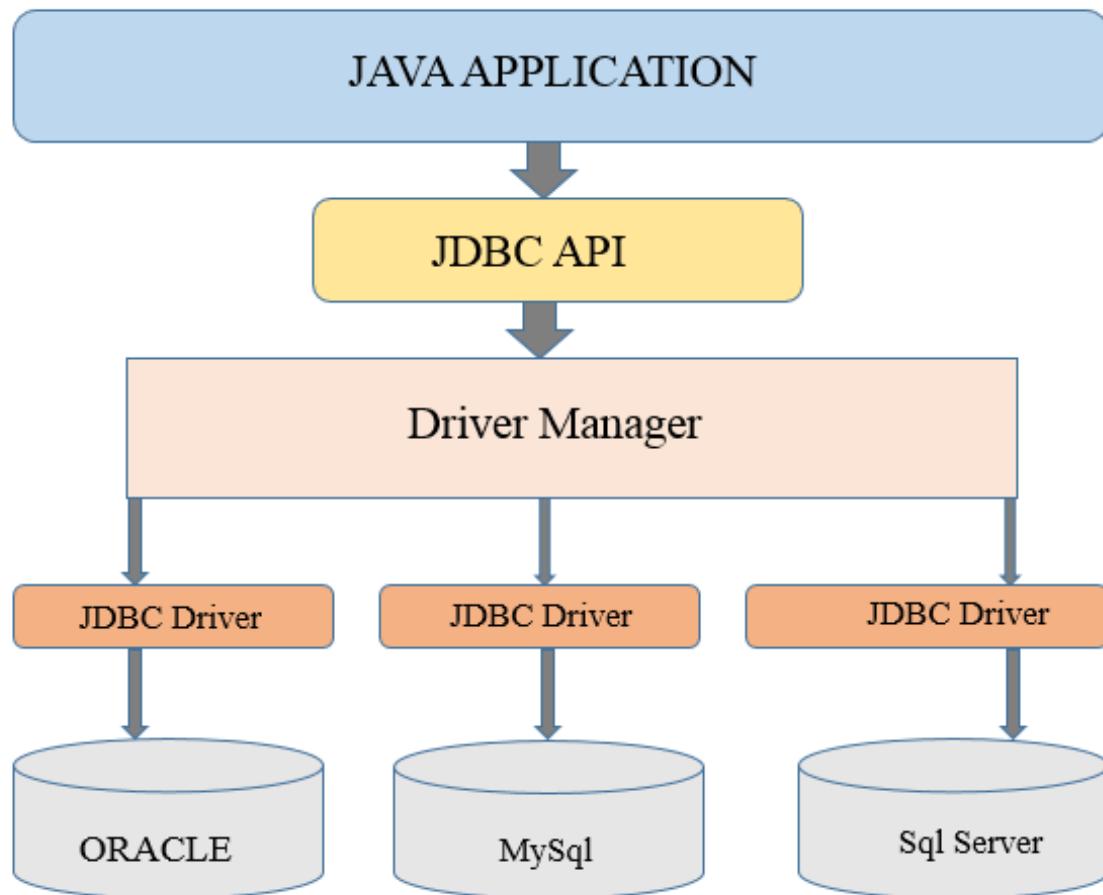


Java connector to MySQL

- 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



Activate



JDBC methods

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

`executeQuery() , executeUpdate()`

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



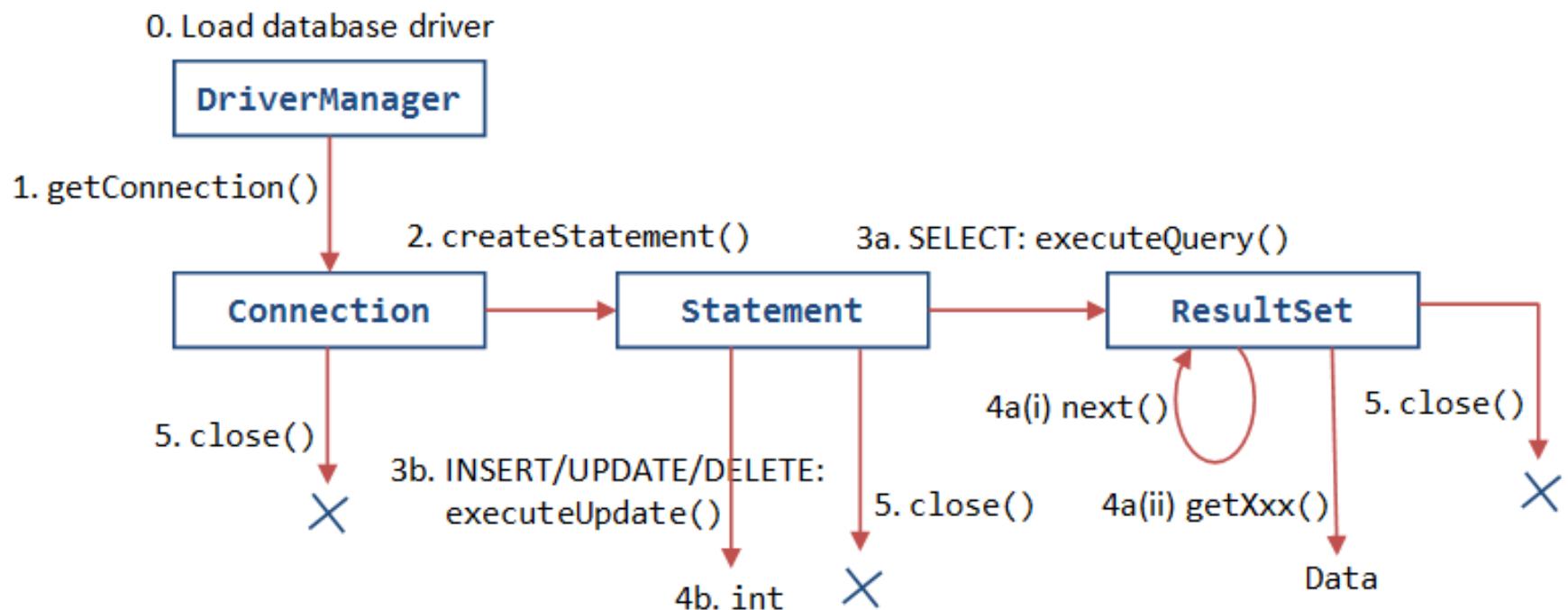
JDBC steps

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



MySQL®

JDBC steps





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");

            // Connect to Database
            Connection connection =
                DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database",
                "your_username", "your_password");

            // Execute Query
            Statement statement = connection.createStatement();
            ResultSet 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();
}
}
```



MySQL®

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;
```



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 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;
END //
```



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 '%table%';
```



Database connection

- Must first establish a connection between client and the database
- 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
```



MySQL®

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
```

Futtatás

Adja meg a program, a mappa, a dokumentum vagy az internetes erőforrás nevét, és a Windows megnyitja azt.

Megnyitás: services.msc

OK Mégse Tállozás...

Szolgáltatások

Fájl Művelet Nézet Súgó

Szolgáltatások (helyi)

A szolgáltatás [indítása](#)

Név	Leírás	Állapot	Indítási típus	Bejelent.
Microsoft Office Click-to-Ru...	Mana...	Fut	Automatikus	Helyi re...
Microsoft Passport	Folya...	Fut	Manuális (ind...	Helyi re...
Microsoft Passport-tároló	A felh...	Fut	Manuális (ind...	Helyi sz...
Microsoft Tárolóhelyek tárfe...	A Micr...	Kézi	Hálózati	
Microsoft Update Health Ser...	Maint...		Letiltva	Helyi re...
Microsoft Windows SMS-út...	A meg...		Manuális (ind...	Helyi sz...
Microsoft-fiók bejelentkezés...	A Micr...	Fut	Manuális (ind...	Helyi re...
Minőségi audiovizuális Win...	A qWa...	Kézi	Helyi sz...	
Mobilhálózati idő	Ez a sz...		Manuális (ind...	Helyi sz...
Mozilla Maintenance Service	A Moz...	Kézi	Helyi re...	
Munkaállomás	Ügyfel...	Fut	Automatikus	Hálózati
Munkahelyi mappák	Ez a sz...	Kézi	Helyi sz...	
MySQL80		Kézi	Hálózati	



MySQL®

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®

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 administration
 - Data migration



MySQL®

MySQL workbench

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas

Information

No object selected

Output

Action Output

Time Action

Message

Duration / Fetch

SQL Additions

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.

Connection Name: new connection

Host: Tompa-IIT-HP
Socket: /tmp/mysql.sock
Port: 3306
Version: 5.6.20-log (MySQL Community Server (GPL))
Compiled For: Win32 (x86)
Configuration File: unknown
Running Since: Tue Oct 29 18:00:08 2024 (0:01)

Refresh

Available Server Features

Performance Schema:	On	Windows Authentication:	Off
Thread Pool:	n/a	Password Validation:	n/a
Memcached Plugin:	n/a	Audit Log:	n/a
Semisync Replication Plugin:	n/a	Firewall:	n/a
SSL Availability:	Off	Firewall Trace:	n/a

Server Directories

Base Directory:	C:\TT\Egyetem\CRM_Geothermal_Horizon_Europe_._2023.04-09\UwAmp\bin\database
Data Directory:	C:\TT\Egyetem\CRM_Geothermal_Horizon_Europe_._2023.04-09\UwAmp\bin\database
Disk Space in Data Dir:	211.69 GB of 475.81 GB available

Context Help Snippets

MySQL Server 5.6

Server Status: Running CPU Load: 0% Connections: 4

Traffic: 3.01 KB/s Key Efficiency: 0.0%

Selects per Second: 0 InnoDB Buffer Usage: 5.1%

InnoDB Reads per Second: 0 InnoDB Writes per Second: 0



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);
```

Create Database



```
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.

mysql> create database hello_world
      ;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+----+-----+
| information_schema |
| crm_db            |
| hello_world        |
| mysql              |
| performance_schema |
+----+-----+
5 rows in set (0.01 sec)

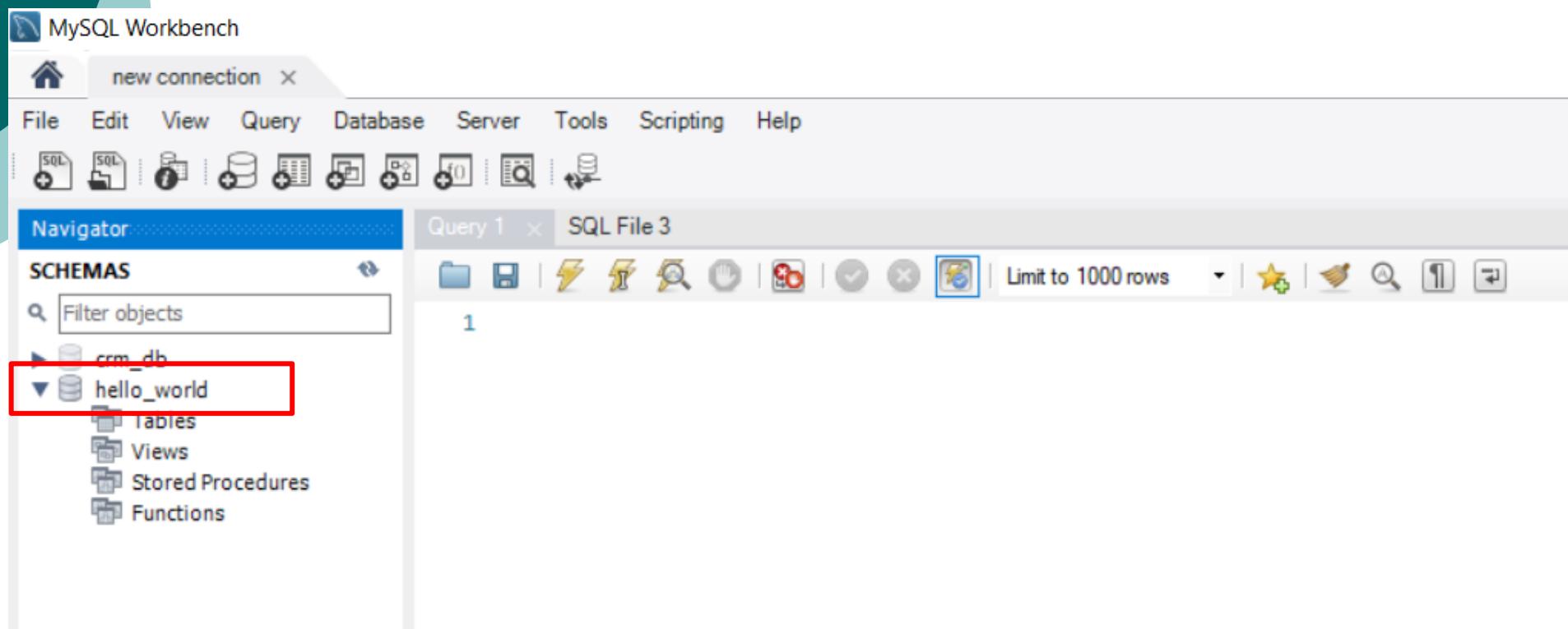
mysql>
```



MySQL®

Create Database

- The created database can be checked in the MySQL Workbench





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()
```



MySQL®

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
```

- Java

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



MySQL®

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 |  
+-----+  
| information_schema |  
| crm_db |  
| mysql |  
| performance_schema |  
+-----+  
4 rows in set (0.00 sec)
```

```
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();
        }
    }
}
```



MySQL®

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;
```

- Java

```
String sql = "USE Database name";
st.execute(sql);
```



MySQL®

Select database

MySQL 8.0 Command Line Client

- □ X

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.getString(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
```



Create users

- 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);
```



MySQL®

Create users

MySQL 8.0 Command Line Client

```
mysql> create user 'tompa' identified by 'passwd';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select user from mysql.user;
```

user
tompa
root
uwamp

```
3 rows in set (0.00 sec)
```

```
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;
```



MySQL®

Drop users

MySQL 8.0 Command Line Client

```
mysql>
mysql> drop user 'tompa';
Query OK, 0 rows affected (0.00 sec)

mysql> select user from mysql.user;
+-----+
| user |
+-----+
| root  |
| uwamp |
+-----+
2 rows in set (0.00 sec)

mysql>
```



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();
        }
    }
}
```



MySQL®

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;**



MySQL®

Create table

Kijelölés MySQL 8.0 Command Line Client

```
mysql> 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)
-> );
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> show tables;
+-----+
| Tables_in_hello_world |
+-----+
| customers             |
+-----+
1 row in set (0.00 sec)
```

```
mysql> _
```

MySQL Workbench

new connection ×

File Edit View Query Database Server Tools

SQL SQL Editor Tables Views Stored Procedures Functions

Navigator

SCHEMAS

Filter objects

crm_db

hello_world

Tables

customers

Views

Stored Procedures

Functions

Query 1 ×

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®

```
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 rows in set (0.00 sec)
```

mysql>

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for database management tasks. The left pane, titled 'Navigator', displays the 'SCHEMAS' section with a search bar for 'Filter objects'. Under 'hello_world', there is a 'Tables' section containing a selected table named 'customers'. This table has columns: ID, NAME, AGE, ADDRESS, and SALARY. Other sections for the 'customers' table include 'Indexes', 'Foreign Keys', and 'Triggers'. The right pane, titled 'SQL File 4*', contains a query editor with the following SQL statement:

```
1 • select * from customers;
```

Below the query editor is a 'Result Grid' table with the following data:

	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
*	NULL	NULL	NULL	NULL	NULL



MySQL®

Update Query

- 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  
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>
```



MySQL®

Alter Query

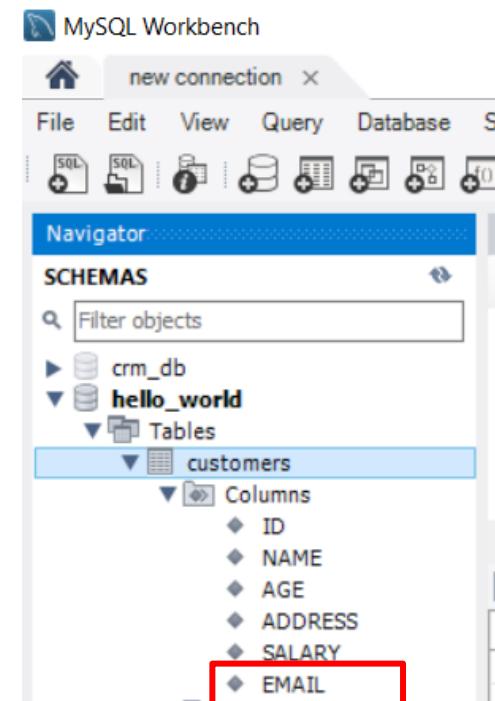
- The ALTER query in MySQL can be used to **add, delete, or modify columns in an existing table**

```
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 | Nikhil | 32 | Ahmedabad | 2000.00 | NULL |  
| 2 | Khilan | 25 | Delhi | 1500.00 | NULL |  
| 3 | Kaushik | 23 | Kota | 2500.00 | 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 |  
+----+-----+-----+-----+-----+  
7 rows in set (0.00 sec)
```





MySQL®

Alter Query

- Use the upadte SQL command to add email data to each record

```
UPDATE CUSTOMERS SET EMAIL = 'ramesh@example.com' WHERE ID = 1;
UPDATE CUSTOMERS SET EMAIL = 'khilan@example.com' WHERE ID = 2;
UPDATE CUSTOMERS SET EMAIL = 'kaushik@example.com' WHERE ID = 3;
UPDATE CUSTOMERS SET EMAIL = 'chaitali@example.com' WHERE ID = 4;
UPDATE CUSTOMERS SET EMAIL = 'hardik@example.com' WHERE ID = 5;
UPDATE CUSTOMERS SET EMAIL = 'komal@example.com' WHERE ID = 6;
UPDATE CUSTOMERS SET EMAIL = 'muffy@example.com' WHERE ID = 7;
```

The screenshot shows the MySQL Workbench interface. On the left, a terminal window displays the execution of an UPDATE query and a SELECT query. The UPDATE query changes the email address for customer ID 7 to 'muffy@example.com'. The SELECT query retrieves all data from the 'customers' table.

Terminal Output:

```
mysql> UPDATE CUSTOMERS SET EMAIL = 'muffy@example.com' WHERE ID = 7;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from customers;
+----+-----+-----+-----+-----+
| ID | NAME | AGE  | ADDRESS | SALARY | EMAIL
+----+-----+-----+-----+-----+
| 1  | Nikhil | 32   | Ahmedabad | 2000.00 | ramesh@example.com
| 2  | Khilan | 25   | Delhi      | 1500.00 | khilan@example.com
| 4  | Chaitali | 26   | Mumbai     | 6500.00 | chaitali@example.com
| 5  | Hardik | 27   | Bhopal    | 8500.00 | hardik@example.com
| 6  | Komal | 22   | MP        | 9000.00 | komal@example.com
| 7  | Muffy | 24   | Indore    | 5500.00 | muffy@example.com
+----+-----+-----+-----+-----+
```

MySQL Workbench Interface:

- File Bar:** new connection, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** SQL, Scripts, Results, Log, Schema, Structure, Data, Reports, Tasks, Utilities.
- Navigator:** Schemas (crm_db, hello_world), Tables (customers, first_view), Views.
- SQL Editor:** SQL File 4*, select * from customers; (Limit to 1000 rows).
- Result Grid:** Shows the data from the 'customers' table.

ID	NAME	AGE	ADDRESS	SALARY	EMAIL
1	Nikhil	32	Ahmedabad	2000.00	ramesh@example.com
2	Khilan	25	Delhi	1500.00	khilan@example.com
4	Chaitali	26	Mumbai	6500.00	chaitali@example.com
5	Hardik	27	Bhopal	8500.00	hardik@example.com
6	Komal	22	MP	9000.00	komal@example.com
7	Muffy	24	Indore	5500.00	muffy@example.com



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;
```

```
mysql> DELETE FROM CUSTOMERS WHERE ID = 3;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from customers;
+----+-----+---+-----+-----+-----+
| 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  |
+----+-----+---+-----+-----+-----+
6 rows in set (0.00 sec)
```



MySQL®

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)
```

The screenshot shows the MySQL Workbench interface. In the SQL tab, the command `TRUNCATE TABLE CUSTOMERS;` is entered. Below it, the result of the command is shown: `Query OK, 0 rows affected (0.00 sec)`. In the Navigator pane, under the `hello_world` schema, the `customers` table is selected. The Result Grid pane displays the table structure with columns: ID, NAME, AGE, ADDRESS, SALARY, and EMAIL. All rows in the grid are currently null.

ID	NAME	AGE	ADDRESS	SALARY	EMAIL
NULL	NULL	NULL	NULL	NULL	NULL



MySQL®

Drop Query

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

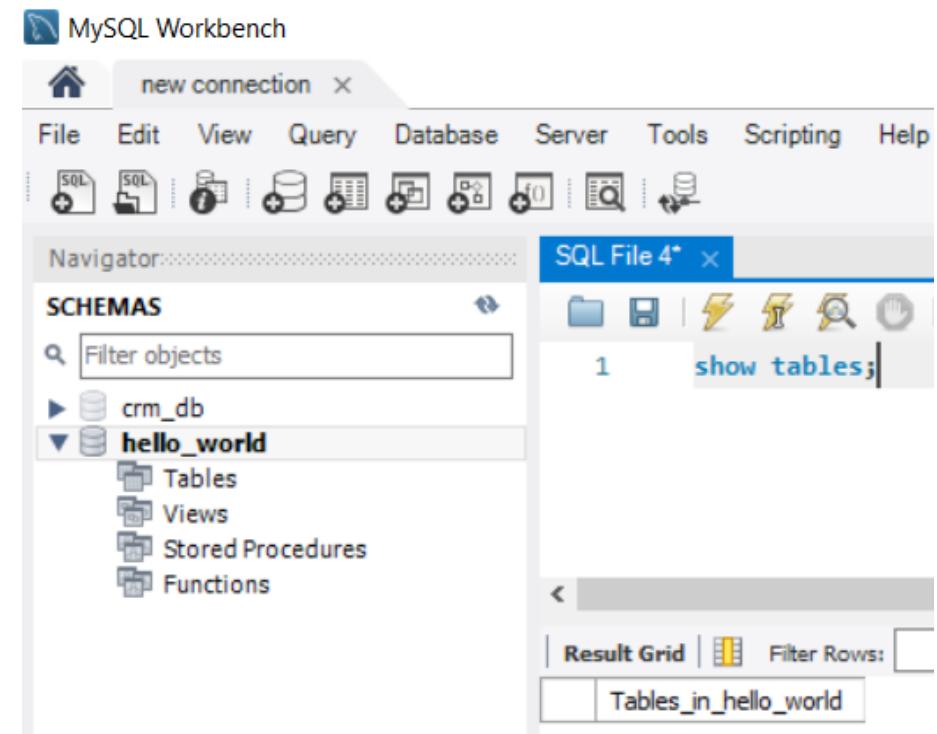
```
DROP TABLE table_name;
```

```
DROP TABLE CUSTOMERS;
```

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

mysql> show tables;
Empty set (0.00 sec)

mysql>
```





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 or 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®

View

```
mysql> CREATE VIEW first_view AS SELECT * FROM CUSTOMERS;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>
```

```
mysql> SELECT * FROM first_view;  
+----+-----+-----+-----+-----+  
| ID | NAME | AGE | ADDRESS | SALARY | EMAIL  
+----+-----+-----+-----+-----+  
| 1 | Nikhil | 32 | Ahmedabad | 2000.00 | ramesh@example.com  
| 2 | Khilan | 25 | Delhi | 1500.00 | khilan@example.com  
| 4 | Chaitali | 26 | Mumbai | 6500.00 | chaitali@example.com  
| 5 | Hardik | 27 | Bhopal | 8500.00 | hardik@example.com  
| 6 | Komal | 22 | MP | 9000.00 | komal@example.com  
| 7 | Muffy | 24 | Indore | 5500.00 | muffy@example.com  
+----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

The screenshot shows the MySQL Workbench interface. At the top, there's a menu bar with File, Edit, View, Query, and Database. Below the menu is a toolbar with various icons. The main area is titled "Navigator" and contains a "SCHEMAS" tree. Under the "hello_world" schema, there are "Tables" and "Views". The "customers" table is expanded, showing its columns: ID, NAME, AGE, ADDRESS, SALARY, and EMAIL. The "first_view" view is also expanded, showing the same six columns.

ID	NAME	AGE	ADDRESS	SALARY	EMAIL
1	Nikhil	32	Ahmedabad	2000.00	ramesh@example.com
2	Khilan	25	Delhi	1500.00	khilan@example.com
4	Chaitali	26	Mumbai	6500.00	chaitali@example.com
5	Hardik	27	Bhopal	8500.00	hardik@example.com
6	Komal	22	MP	9000.00	komal@example.com
7	Muffy	24	Indore	5500.00	muffy@example.com



View

Table A

A	B	C

View In SQL

Table B

D	E	F

SELECT
QUERY

View	A	B	C	D	E	F



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):

```
mysql> select * from customer_orders_view;
+-----+-----+-----+-----+-----+-----+-----+-----+
| CustomerID | CustomerName | CustomerAge | CustomerAddress | CustomerSalary | OrderID | OrderAmount | OrderDate |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      1 | Nikhil       |      32 | Ahmedabad     |    2000.00 |      1 |    500.00 | 2024-01-15 |
|      5 | Hardik        |      27 | Bhopal        |    8500.00 |      3 |   2000.00 | 2024-03-05 |
|      2 | Khilan         |      25 | Delhi          |    1500.00 |      4 |    800.00 | 2024-04-20 |
|      4 | Chaitali       |      26 | Mumbai         |    6500.00 |      5 |   2200.00 | 2024-05-17 |
|      6 | Komal          |      22 | MP             |    9000.00 |   NULL |   NULL     |   NULL      |
|      7 | Muffy          |      24 | Indore         |    5500.00 |   NULL |   NULL     |   NULL      |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```



MySQL®

View

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

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for database management tasks. The left sidebar is the Navigator, which displays the database schema. Under the 'hello_world' database, the 'Views' section is expanded, showing two views: 'customer_orders_view' (selected) and 'first_view'. The main workspace is titled 'SQL File 4*' and contains a single query:

```
1 •  select * from customer_orders_view;
```

The results are displayed in a 'Result Grid' table:

	CustomerID	CustomerName	CustomerAge	CustomerAddress	CustomerSalary	OrderID	OrderAmount	OrderDate
▶	1	Nikhil	32	Ahmedabad	2000.00	1	500.00	2024-01-15
▶	5	Hardik	27	Bhopal	8500.00	3	2000.00	2024-03-05
▶	2	Khilan	25	Delhi	1500.00	4	800.00	2024-04-20
▶	4	Chaitali	26	Mumbai	6500.00	5	2200.00	2024-05-17
▶	6	Komal	22	MP	9000.00	NULL	NULL	NULL
▶	7	Muffy	24	Indore	5500.00	NULL	NULL	NULL



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.

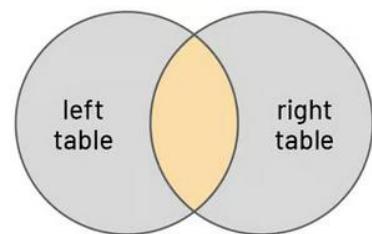
```
SELECT a.ID, a.NAME, b.DATE, b.AMOUNT  
FROM CUSTOMERS a, ORDERS b  
WHERE a.ID = b.CUSTOMER_ID;
```



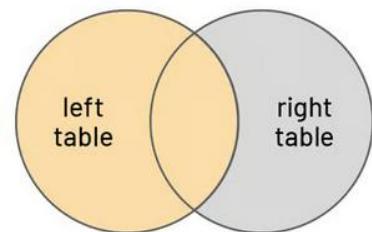
MySQL®

Join

INNER JOIN

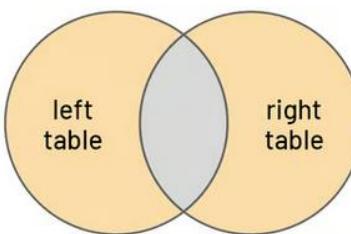


LEFT JOIN

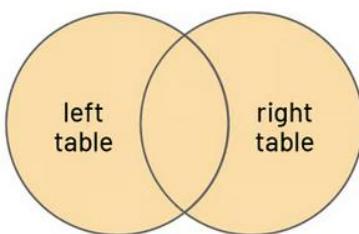


SQL JOIN Types

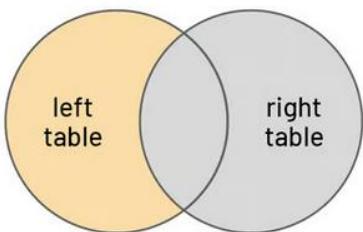
OUTER JOIN



UNION / FULL JOIN



RIGHT JOIN



CodingNomads

```
1 SELECT *  
2 FROM A  
3 INNER JOIN B ON A.key = B.key
```

```
1 SELECT *  
2 FROM A  
3 FULL JOIN B ON A.key = B.key
```

```
1 SELECT *  
2 FROM A  
3 FULL JOIN B ON A.key = B.key  
4 WHERE A.key IS NULL OR  
5 B.key IS NULL
```

```
1 SELECT *  
2 FROM A  
3 LEFT JOIN B ON A.key = B.key
```

```
1 SELECT *  
2 FROM A  
3 LEFT JOIN B ON A.key = B.key  
4 WHERE B.key IS NULL
```

```
1 SELECT *  
2 FROM A  
3 RIGHT JOIN B ON A.key = B.key
```

```
1 SELECT *  
2 FROM A  
3 RIGHT JOIN B ON A.key = B.key  
4 WHERE B.key IS NULL
```



MySQL®

More MySQL example

- https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_Beginner.html
- <https://www.tutorialspoint.com/mysql/index.htm>
- <https://www.w3schools.com/MySQL/default.asp>



JDBC - query

- Query for the (previously created Customers) table

```
1 package mysql_query;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.ResultSet;
6 import java.sql.Statement;
7
8 public class DatabaseQueryExample {
9
10    // Database URL, username, and password
11    static final String DB_URL = "jdbc:mysql://localhost:3306/hello_world";
12    static final String USER = "root";
13    static final String PASS = "root";
14
15    public static void main(String[] args) {
16        Connection conn = null;
17        Statement stmt = null;
18
19        try {
20            // Step 1: Register JDBC driver (optional for newer JDBC versions)
21            Class.forName("com.mysql.cj.jdbc.Driver");
22
23            // Step 2: Open a connection
24            System.out.println("Connecting to the database...");
25            conn = DriverManager.getConnection(DB_URL, USER, PASS);
26
27            // Step 3: Execute a query
28            System.out.println("Creating statement...");
29            stmt = conn.createStatement();
30            String sql = "SELECT ID, NAME, AGE, ADDRESS, SALARY FROM CUSTOMERS";
31            ResultSet rs = stmt.executeQuery(sql);
32
33            // Step 4: Process the ResultSet
34            while (rs.next()) {
35                // Retrieve data by column name
36                int id = rs.getInt("ID");
37                String name = rs.getString("NAME");
38                int age = rs.getInt("AGE");
39                String address = rs.getString("ADDRESS");
40                double salary = rs.getDouble("SALARY");
41
42                // Display values
43                System.out.print("ID: " + id);
44                System.out.print(", Name: " + name);
45                System.out.print(", Age: " + age);
46                System.out.print(", Address: " + address);
47                System.out.println(", Salary: " + salary);
48            }
49
50            // Step 5: Clean up the environment
51            rs.close();
52            stmt.close();
53            conn.close();
54        } catch (Exception e) {
55            e.printStackTrace();
56        } finally {
57            // Finally block to close resources
58            try {
59                if (stmt != null) stmt.close();
60                if (conn != null) conn.close();
61            } catch (Exception e) {
62                e.printStackTrace();
63            }
64        }
65    }
}
```



JDBC - query

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

A screenshot of the Eclipse IDE's Console view. The tab bar at the top shows 'Problems', 'Javadoc', 'Declaration', and 'Console' (which is selected). Below the tabs, the text output from a Java application is displayed. It starts with the application's name and path, followed by the process of connecting to a database and creating a statement. The final part of the output shows a list of customer records, each with an ID, name, age, address, and salary.

```
<terminated> DatabaseQueryExample [Java Application] C:\eclipse\plugins\org.eclipse.jst  
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
```



MySQL®

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();

            String sql = "CREATE TABLE students (" + "id INT AUTO_INCREMENT PRIMARY KEY,"
                + "name VARCHAR(100) NOT NULL," + "age INT NOT NULL" + ")";

            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();
            }
        }
    }
}
```

The screenshot shows an IDE interface with several tabs: Problems, @ Javadoc, Declaration, and Console. The Console tab displays the message: <terminated> CreateTableExample [Java Application] C:\ecl The students table created successfully!

- Create of students table in the hello_world database



MySQL®

JDBC - insert record

```
3④ import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.SQLException;
7
8 public class InsertStudentsExample {
9     private static final String URL = "jdbc:mysql://localhost:3306/hello_world";
10    private static final String USER = "root";
11    private static final String PASSWORD = "root";
12
13    public static void main(String[] args) {
14        InsertStudentsExample example = new InsertStudentsExample();
15        example.insertStudent("John Doe", 20);
16        example.insertStudent("Jane Smith", 22);
17        example.insertStudent("Emily Johnson", 19);
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();
30            if (rowsAffected > 0) {
31                System.out.println("Record inserted successfully: " + name + ", age: " + age);
32            }
33
34        } catch (SQLException e) {
35            e.printStackTrace();
36        }
37    }
38 }
```

Eclipse IDE screenshot showing the output of the Java application and the MySQL command-line interface.

Eclipse IDE Output:

```
Problems @ Javadoc Declaration Console ×
<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 Command Line:

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

- Inserting records in the Students table

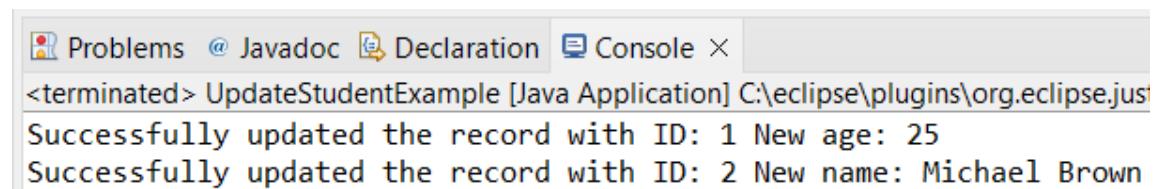


MySQL®

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) {  
        UpdateStudentExample example = new UpdateStudentExample();  
        example.updateStudentAge(1, 25);  
        example.updateStudentName(2, "Michael Brown");  
    }  
  
    // age update  
    public void updateStudentAge(int id, int newAge) {  
        String sql = "UPDATE students SET age = ? WHERE id = ?";  
  
        try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);  
             PreparedStatement preparedStatement = connection.prepareStatement(sql)) {  
  
            preparedStatement.setInt(1, newAge);  
            preparedStatement.setInt(2, id);  
  
            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 doesn't exist with this ID.");  
            }  
  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
mysql> select * from students;  
+----+-----+-----+  
| id | name | age |  
+----+-----+-----+  
| 1 | John Doe | 25 |  
| 2 | Michael Brown | 22 |  
| 3 | Emily Johnson | 19 |  
| 4 | John Doe | 20 |  
| 5 | Jane Smith | 22 |  
| 6 | Emily Johnson | 19 |  
+----+-----+-----+  
6 rows in set (0.00 sec)
```



The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the results of the JDBC update operations:

```
Problems @ Javadoc Declaration Console X  
<terminated> UpdateStudentExample [Java Application] C:\eclipse\plugins\org.eclipse.jus  
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

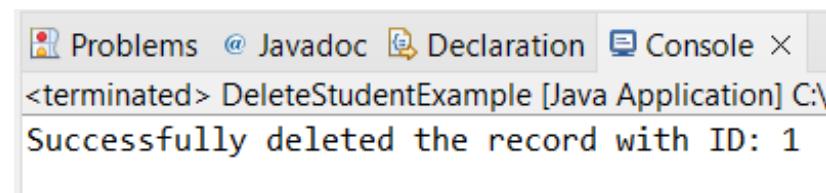


MySQL®

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";  
  
    public static void main(String[] args) {  
        DeleteStudentExample example = new DeleteStudentExample();  
        example.deleteStudentById(1);  
    }  
  
    // delete record by ID  
    public void deleteStudentById(int id) {  
        String sql = "DELETE FROM students WHERE id = ?";  
  
        try (Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);  
             PreparedStatement preparedStatement = connection.prepareStatement(sql)) {  
  
            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 doesn't exist.");  
            }  
  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
mysql> select * from students;  
+----+-----+---+  
| id | name      | age |  
+----+-----+---+  
| 2  | Michael Brown | 22 |  
| 3  | Emily Johnson | 19 |  
| 4  | John Doe   | 20 |  
| 5  | Jane Smith  | 22 |  
| 6  | Emily Johnson | 19 |  
+----+-----+---+  
5 rows in set (0.00 sec)
```



- Deleting record (with specified ID) from the Students table



Task: Create a Web Application

MySQL®

- **Create a Web Application with Form to save data to database 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)
 - age (INT)



Task: Create a Web Application

- **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

MySQL®

```
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

Screenshot of a web browser showing the User Registration page:

localhost:8080/MySQL_user_management_06/userform.html

User Registration

Name: Tamas Tompa

Email: tamas.tompa1@uni-miskolc.hu

Age: 36

Submit

View All Users

Screenshot of a web browser showing the confirmation message after registration:

localhost:8080/MySQL_user_management_06/UserServlet

User registered successfully!

Screenshot of a web browser showing the Registered Users page:

localhost:8080/MySQL_user_management_06/UserServlet?

Registered Users

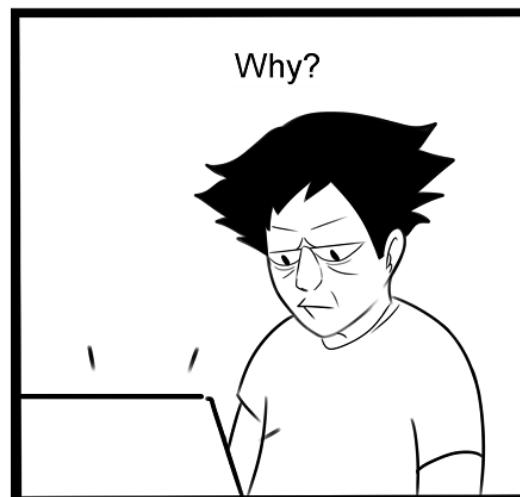
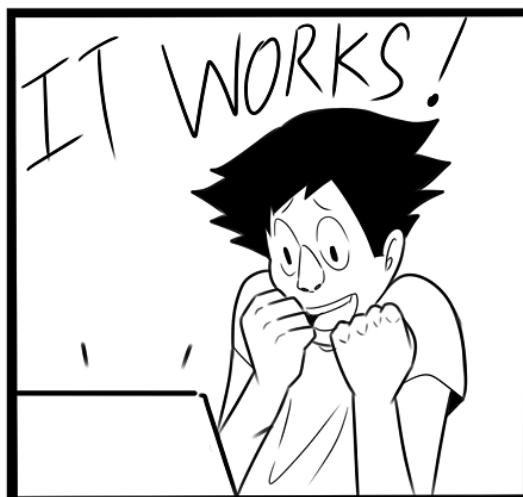
ID	Name	Email	Age
2	Tamas Tompa	tamas.tompa1@uni-miskolc.hu	36



MySQL®

:)

"IT WORKS. WHY?"



[HTTPS://TAPAS.IO/SERIES/GRUMPY-CODES](https://tapas.io/series/grumpy-codes)



CARDBOARDVOICE

Thank you for your attention!

thank you ☺