INTRODUCTION TO DATABASE

Instructor : Dr.Vshidi Asl

Presenter : Parsa Hamzei

Fall 1403

What is DataBase ?

A database is an organized collection of structured information, usually stored electronically in a computer system. It allows efficient data storage, retrieval, and management. Databases are fundamental in storing large amounts of data in a format that can easily be queried, updated, and managed.

Why should we use DataBase in our project ?

Using a **database** in your project helps you manage data in a structured, scalable, and secure manner. It ensures that your data remains organized, accessible, and protected, making it an essential component for nearly every software project.

HOW TO :

1_ Install MYSQL

2_ Write SQL Commands

3_ Connect Java Project to MYSQL

WHAT IS SQL DATABASE ?

SQL is a domain-specific language used for managing data, especially in Relational Database Management Systems (RDBMS). A relational database organizes data into tables (or "relations") composed of rows and columns, establishing relationships between different tables.

Example: In an RDBMS, you can store customer information in one table and their purchases in another table, then link the two via a relationship.

MySQL is an open-source RDBMS, named after the daughter of its co-founder ("My") and the acronym SQL (Structured Query Language).

Installing MYSQL :

To install MySQL , go to www.mysql.com , navigate to the MySQL Community Downloads section , and choose the package that suits your needs.

The recommended method is to download the MySQL Installer. You can easily open this page or install it directly from this link

When you're installing Make sure to include the Server, Workbench, and Router components during installation .

Creating a DataBase :

First, design your database schema, defining tables, columns, and data types. You can create your database with this SQL command:

SQL
```sql
CREATE DATABASE 'database name ' ;
```

To delete a data base , you can use this command :

SQL
```sql
 DROP SHEMA 'database name' ;
```

Some essential data types used in SQL include:

1. INT , for numberic data
2. VARCHAR(M) , For string data (M represents the maximum number of characters)
3. DECIMAL(M,N) , For decimal numbers, where M is the total length, and N is the number of decimals.
4. BLOB , For binary data like images or files

5.DATE , for date , formatted as ‘YYYY-MM-DD’

6.TIMESTAMP , For date and time, formatted as ‘YYYY-MM-DD HH-MM-SS’

Common SQL Constraints :

1. PRIMARY KEY

The PRIMARY KEY is a field (or a combination of fields) that uniquely identifies each record in a table , Each table can have only one PRIMARY KEY, and the column defined as PRIMARY KEY cannot contain NULL values.

2.AUTO_INCREMENT

AUTO_INCREMENT is used in SQL to automatically generate a unique number whenever a new record is inserted into a table , It is typically used for **primary keys** where you need a unique identifier for each row.

3.UNIQUE

The UNIQUE constraint ensures that all values in a column are distinct (no duplicates) , You can apply the UNIQUE constraint to multiple columns, but unlike the PRIMARY KEY, a table can have more than one UNIQUE constraint.

4.NOT NULL

The NOT NULL constraint ensures that a column cannot have NULL values.

5.DEFAULT

The DEFAULT constraint allows you to specify a default value for a column if no value is provided during record insertion.

6.FOREIGN KEY

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table and It creates a relationship between two tables.

To make a foreign key , we use this syntax :

SQL

```
FOREIGN KEY (column name) REFERENCES table2(column name)
ON DELETE SET NULL / CASCADE
```

Cascade key , will delete the data if the object from table delete !

Creating and Managing Tables:

After creating database you should creat the tables

For example if you want to creat table for students , you can use following command :

SQL

```
USE ‘database name ’ ;

CREATE TABLE students (
 id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  age INT,
);
```

To modify your table columns , you can use this command :

SQL

```sql
ALTER TABLE Students
MODIFY COLUMN birthdate 'new type'
-- ADD , DROP
```

You can modify , add or delete a column !

For inserting data to your table , you can use this syntax :

SQL

```sql
INSERT INTO students (name, age) VALUES ('parsa’, 20);
```

To edit this student , you can use following :

SQL
```sql
UPDATE Students

SET name = "hamzei"
WHERE student_id = 1
```

This command will change the name of student with student id =1 to hamzei .

To delete this data , just write DELETE instead of UPDATE

Then define condition (for example : WHERE student_id = 1)

SQL QUERIES :

To show everything from table , we use :

```
SQL
SELECT * FROM students
```

To retrieve specific columns :

```
SQL
SELECT name FROM students
```

We can also use condition , order and limit the result :

SQL

```sql
SELECT * FROM students
ORDER BY student_id DESC
where student_id<100

LIMIT 5
```

The result will be 5 students with less than 100 id , descending ordered by name , we can also write LIMIT 2,5 and its means skip 2 and then show 5 students !

JOINING TABLES :

To join tables and write better queries we have many ways but the most simple way is like these :

```sql
SELECT * FROM students , units
WHERE students.id = units.student_id
```

In this example , we have a foreign key in units table , named student_id that references to column named id in student table

Another way to join tables is this query :

SQL

```
SELECT * FROM units

JOIN students
ON students.student_id = units.student_id

JOIN teachers
ON teachers.teacher_id = units.teacher_id

JOIN courses
ON courses.course_id = units.course_id
```

This query will join four tables(units , teachers , students , courses)

CONNECTIING JAVA TO MY SQL :

To connect a Java application to MySQL, you need a JConnector . To download the connector go to the MySQL website , navigate to the "Downloads" section , and find "JConnector" to download it . Alternatively , you can download it directly from this link .

After downloading , unzip file to add the jar file to dependencies

There is many ways to add this file to the dependencies

But the best way that I recommend to you is using intellij idea !

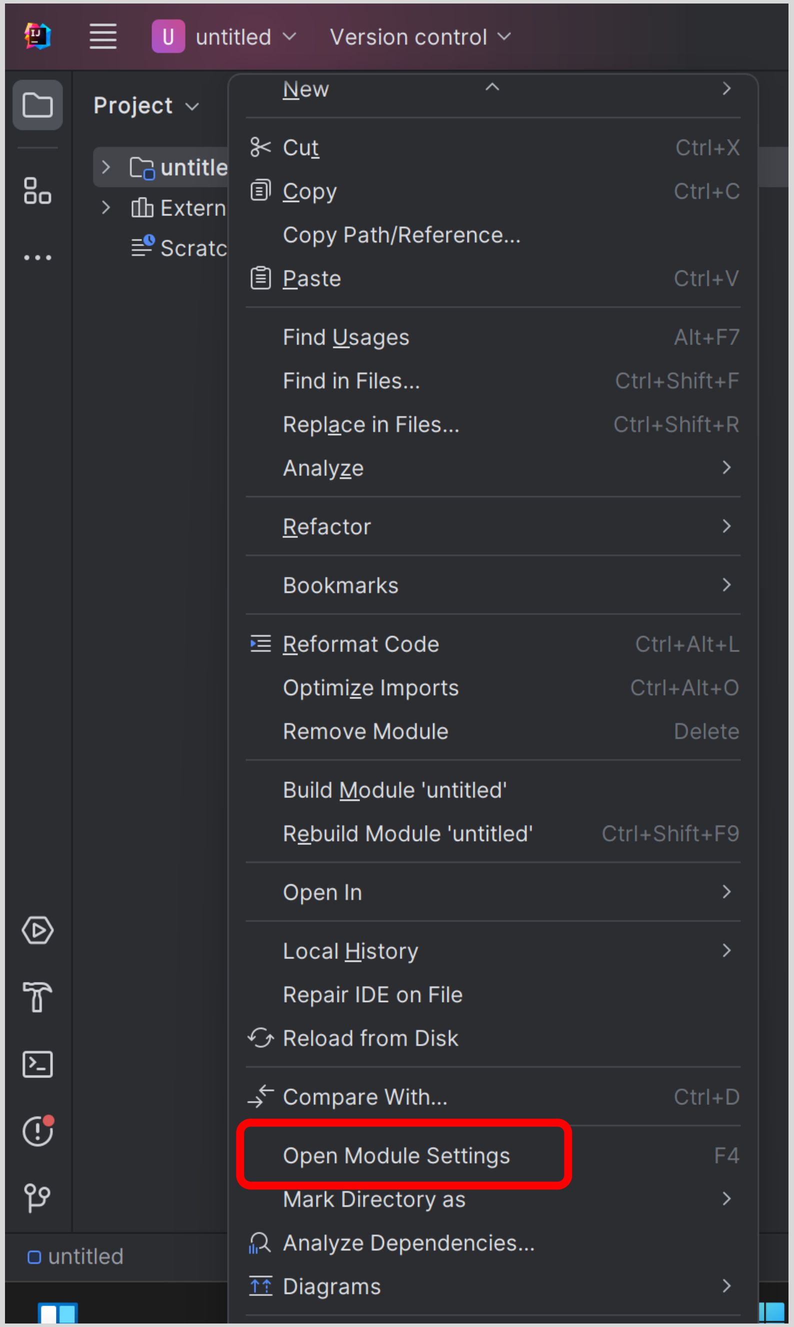
To add connector to dependency follow these steps :

1. Open intellij idea

2. Create new project

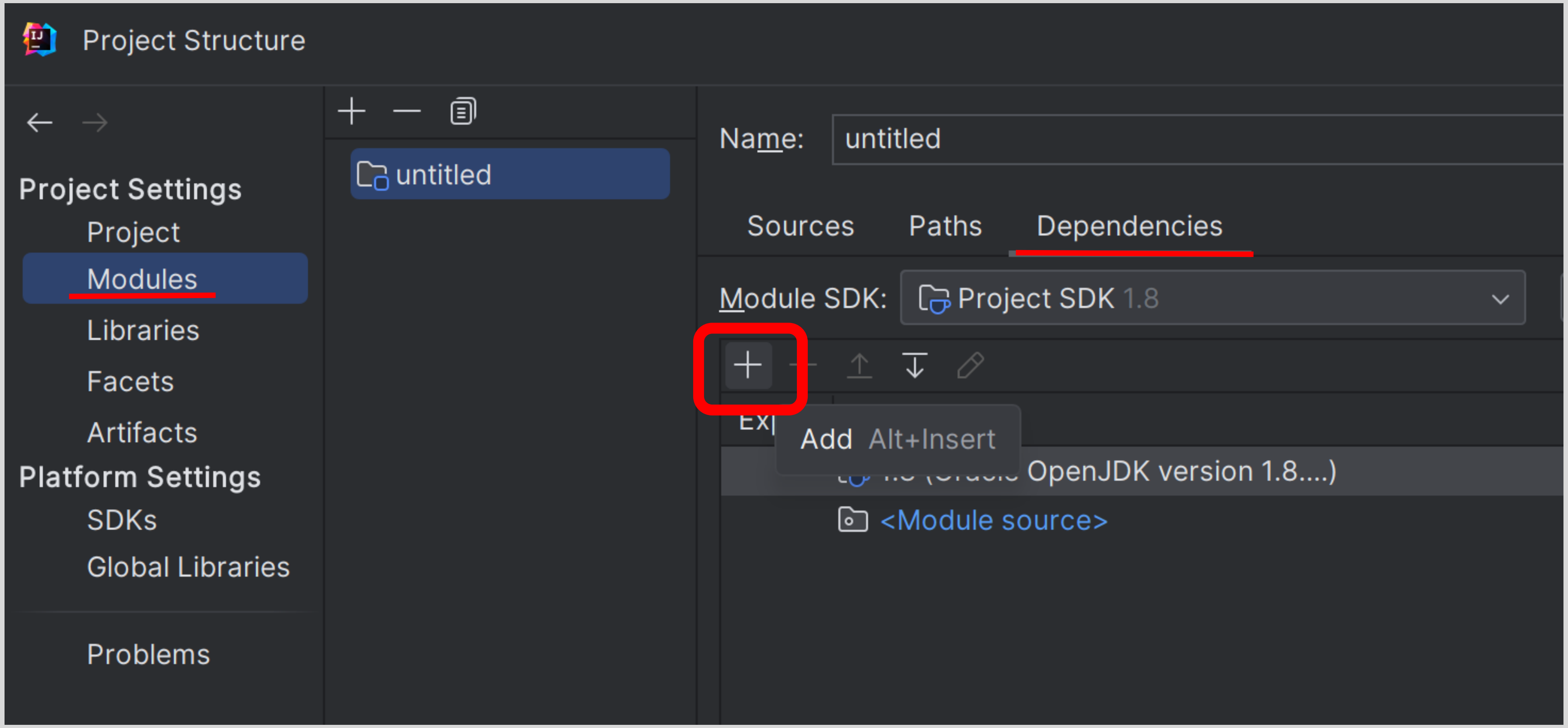
3.Right click on your project folder

4.Click on ‘open module setting’



5. In module section , dependencies tab , click on add

6.Use the jar file you unziped and click ok !



Now we should make connection to MYSQL

To make connection , import these :

JAVA

```java
import java.sql.Connection;

import java.sql.DriverManager;
import java.sql.SQLException;
```

Then use the library to make connection and remember to use try catch for making connection :

JAVA

```
try {
    String URL = "jdbc:mysql://localhost:your port /database name ";
    String USER = user name;
    String PASSWORD = your user password;
    Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);
    if (connection != null) {
        System.out.println("Connected to the database successfully!");
    } else {
        System.out.println("Failed to make connection!");
    }
} catch (SQLException e) {
    System.out.println("MySQL connection failed!");
}
```

Now it must connect to database .

To send queries to database and get result use following structure :

JAVA

```java
Statement statement = connection.createStatement();

String sql = "your query";
ResultSet resultSet = statement.executeQuery(sql);
while (resultSet.next()) {
    String first_column = resultSet.getString("first_name");
     String secound_column = resultSet.getString("last_name");
     System.out.println(first_column+" "+secound_column);
}
```

Good job !

Now you can send queries and pull datas into resultSet then use a loop to get all of the data from table !

In my project , the server had a method called ‘updateDatabase’ which was used to update the data in the database. It worked like this :

JAVA

```java
public static void UpdateDataBase(){

    try (Connection connection = DriverManager.getConnection("database", "user", "password")) {
        String query = "DELETE FROM assignment";
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        preparedStatement.executeUpdate();
        for(int i = 0 ; i < Assignments.size() ; i++){
            query = "INSERT INTO assignment (name, deadline, available) VALUES (?, ?, ?)";
            preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, Assignments.get(i).getName());
            preparedStatement.setString(2, Assignments.get(i).getDeadline());
            preparedStatement.setString(3, Assignments.get(i).getAvailable());
            preparedStatement.executeUpdate();
        }
        System.out.println("assignments updated !");
```

JAVA

```
query = "DELETE FROM courses";
preparedStatement = connection.prepareStatement(query);
preparedStatement.executeUpdate();
for(int i = 0 ; i < courses.size() ; i++){
    query = "INSERT INTO courses (name, units, date_of_exam) VALUES (?, ?, ?)";
    preparedStatement = connection.prepareStatement(query);
    preparedStatement.setString(1, courses.get(i).getCourseName());
    preparedStatement.setInt(2, courses.get(i).getUnitsPerCourse());
    preparedStatement.setString(3, courses.get(i).getDate_Of_Exam());
    preparedStatement.executeUpdate();
}
System.out.println("courses updated !");
query = "DELETE FROM teachers";
preparedStatement = connection.prepareStatement(query);
preparedStatement.executeUpdate();
for(int i = 0 ; i < teachers.size() ; i++){

    query = "INSERT INTO courses (teacher_id, first_name, last_name,courses,projects)
             VALUES (?, ?, ?,?,?)";
    preparedStatement = connection.prepareStatement(query);
```

JAVA

```java
preparedStatement.setString(1, String.valueOf(teachers.get(i).getTeacherid()));
preparedStatement.setString(2, teachers.get(i).getFirst_Name());
preparedStatement.setString(3, teachers.get(i).getLast_name());
String add = "";
List<Course> c = teachers.get(i).getCourses();
for(int j = 0 ; j < c.size() ; j++){
    if(j==c.size()-1){
        add+=c.get(j).getCourseName();
    }
    else{
        add+=c.get(j).getCourseName()+"  ";
    }
}
preparedStatement.setString(4, add);
add = "";
List<Assignment> a = teachers.get(i).getProjects();
for(int j = 0 ; j < a.size() ; j++){
    if(j==a.size()-1){
        add+=a.get(j).getName();
    }
```

JAVA

```
                else{
                    add+=a.get(j).getName()+" ";
                }
            }
            preparedStatement.setString(5,add );
            preparedStatement.executeUpdate();
        }
        System.out.println("teachers updated !");
        query = "DELETE FROM terms";
        preparedStatement = connection.prepareStatement(query);
        preparedStatement.executeUpdate();
        query = "DELETE FROM students";
        preparedStatement = connection.prepareStatement(query);
        preparedStatement.executeUpdate();
        for(int i = 0 ; i<students.size() ; i++){
            query = "INSERT INTO students (firs_name, last_name,
                     email,stdent_id,birthdate,terms) VALUES (?, ?, ?,?,?,?)";
           preparedStatement = connection.prepareStatement(query);
```

JAVA

```
preparedStatement.setString(1, students.get(i).getName());

preparedStatement.setString(2, students.get(i).getLast_Name());
preparedStatement.setString(3, students.get(i).getEmail());
preparedStatement.setString(4, String.valueOf(students.get(i).getSID()));
preparedStatement.setString(5,students.get(i).getBirthdate());
preparedStatement.setInt(6, students.get(i).getNumber_Of_Terms());
preparedStatement.executeUpdate();
List<Term> T = students.get(i).getTerms();
for(int j = 0 ; j < T.size() ; j++){
    List<SCORE> scores = new ArrayList<SCORE>();
    List<Course> Courses_Assigned = new ArrayList<Course>();
    List<Teacher> listOfTeachers = new ArrayList<Teacher>();
    for(int k = 0 ; k < scores.size() ; k++){
        query = "INSERT INTO terms (student_id, term_number, teacher,course,score)
                 VALUES (?, ?, ?,?,?)";
        preparedStatement = connection.prepareStatement(query);
        preparedStatement.setString(1, String.valueOf(students.get(i).getSID()));
```

JAVA

```
                    preparedStatement.setInt(2, j);

                    preparedStatement.setString(3, listOfTeachers.get(k).getLast_name());
                    preparedStatement.setString(4, Courses_Assigned.get(k).getCourseName());
                    preparedStatement.setDouble(5,scores.get(k).getScore());
                    preparedStatement.executeUpdate();
                }
            }
        }
        System.out.println("students updated !");
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

This method performs the following operations:

1. Establishes a Database Connection:

The code starts by establishing a connection to a MySQL

2. Updating the assignments Table:

The code begins by deleting all records in the assignment table .

After clearing the table, the code iterates over list . For each entry in the list

It inserts a new row into the assignment table .

This ensures that the assignment table is refreshed with the latest data from the Assignments list.

3. Updating the courses Table:

Similarly, the code deletes all records in the courses table

4. Updating the teachers Table:

The code deletes all records from the teachers table .

It then iterates over the teachers list and for each teacher Inserts the teacher's teacher id , first name , and last name into the teachers table.

The code concatenates the names of the courses and projects (using spaces between course or project names) the teacher is assigned to, and inserts them into the courses and projects columns of the teachers table

5. Updating the students and terms Tables:

First, it deletes all records from both the students and terms tables:

The code then iterates over a list of students. For each student It inserts the student's first name, last name, email, student id, birthdate, and the number of terms they've completed into the students table .

After inserting the student data, the code iterates over the student's academic terms (each represented by a list of Term objects). For each term, it inserts the term's student id, term number, and other related data into the terms table.

Each term may contain multiple SCORE objects (representing the student's scores) and related information about the course and teacher associated with that term .

6. Error Handling:

If an exception occurs at any point during these database operations, it is caught by the catch block, and the stack trace is printed , helping you debug the problem.