Assignment 2
JDBC

COMP3358 Distributed and Parallel Computing

Overview

- Setting up MySQL database
- Using MySQL JDBC driver
- Writing Java program
  - Connect to database
  - Create, read, update, delete

Installing MySQL server via terminal

- See: https://ubuntu.com/server/docs/databases-mysql

```
leo@ubuntu:~$ sudo apt install mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  mysql-server
0 upgraded, 1 newly installed, 0 to remove and 51 not upgraded.
Need to get 9,460 B of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://cn.ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 mysql-ser
ver all 8.0.36-0ubuntu0.22.04.1 [9,460 B]
Fetched 9,460 B in 1s (8,122 B/s)
Selecting previously unselected package mysql-server.
(Reading database ... 210105 files and directories currently installed.)
Preparing to unpack .../mysql-server_8.0.36-0ubuntu0.22.04.1_all.deb
Unpacking mysql-server (8.0.36-0ubuntu0.22.04.1) ...
Setting up mysql-server (8.0.36-0ubuntu0.22.04.1) ...
leo@ubuntu:~$
```

```
leo@ubuntu:~$ sudo service mysql status
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:>
     Active: active (running) since Mon 2024-02-05 11:27:23 CST; 1 day 6h ago
    Process: 73645 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=>
   Main PID: 73653 (mysqld)
     Status: "Server is operational"
      Tasks: 38 (limit: 4555)
     Memory: 355.2M
        CPU: 1min 43.698s
     CGroup: /system.slice/mysql.service
             └─73653 /usr/sbin/mysqld

2月 05 11:27:22 ubuntu systemd[1]: Starting MySQL Community Server...
2月 05 11:27:23 ubuntu systemd[1]: Started MySQL Community Server.
lines 1-14/14 (END)
```

Connect to MySQL server

- Install MySQL Client

```
leo@ubuntu:~$ sudo apt install  mysql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  mysql-client
0 upgraded, 1 newly installed, 0 to remove and 51 not upgraded.
Need to get 9,354 B of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://cn.ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 mysql-cli
ent all 8.0.36-0ubuntu0.22.04.1 [9,354 B]
Fetched 9,354 B in 1s (8,819 B/s)
Selecting previously unselected package mysql-client.
(Reading database ... 210105 files and directories currently installed.)
Preparing to unpack .../mysql-client_8.0.36-0ubuntu0.22.04.1_all.deb ...
Unpacking mysql-client (8.0.36-0ubuntu0.22.04.1) ...
Setting up mysql-client (8.0.36-0ubuntu0.22.04.1) ...
```

- Connect to the MySQL server

```
leo@ubuntu:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, 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>
```

Hints: JDBC URL for MySQL

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
conn = DriverManager.getConnection("jdbc:mysql://"+DB_HOST+
                                   "/"+DB_NAME+
                                   "?user="+DB_USER+
                                   "&password="
                                   +DB_PASS);
System.out.println("Database connection successful");
```

Preparing database

- Connect to the MySQL server using MySQL client
  - Enter root password
- Create a new database and table:

```
CREATE DATABASE c3358;

GRANT ALL ON c3358.* TO 'c3358@localhost' IDENTIFIED BY
'c3358PASS';

USE c3358;

CREATE TABLE c3358_2025 (
  name varchar(32) NOT NULL,
  birthday date NOT NULL,
  PRIMARY KEY name (name)
);
```

Create database named **c3358**

Create user named **c3358**, who have all access to database **c3358**

Create table named **c3358_2025**

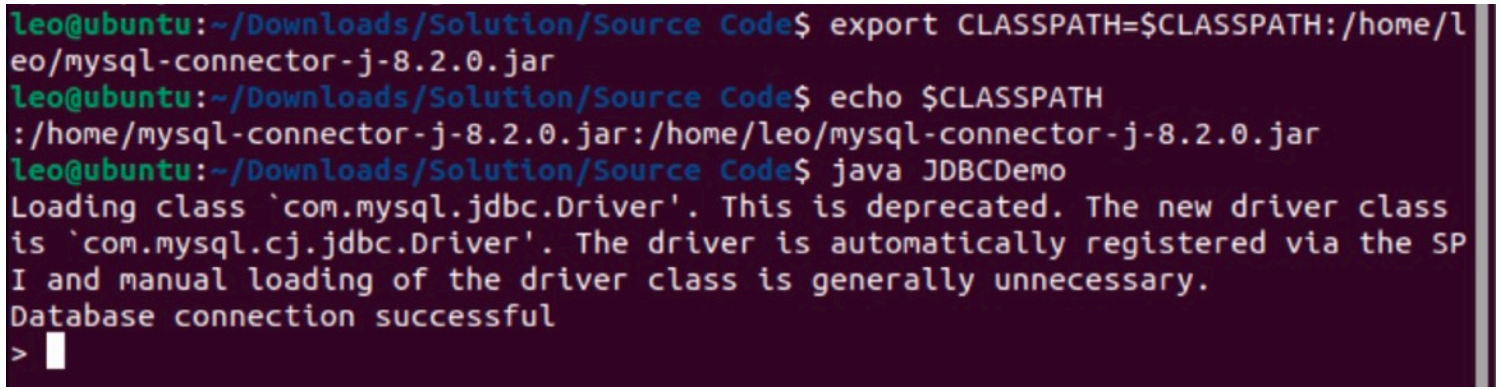
You can check it by the command DESCRIBE **c3358_2025**

JDBC driver

- Download MySQL JDBC driver:
  - http://dev.mysql.com/downloads/connector/j/
- Extract mysql-connecter-j-x.x.x.jar

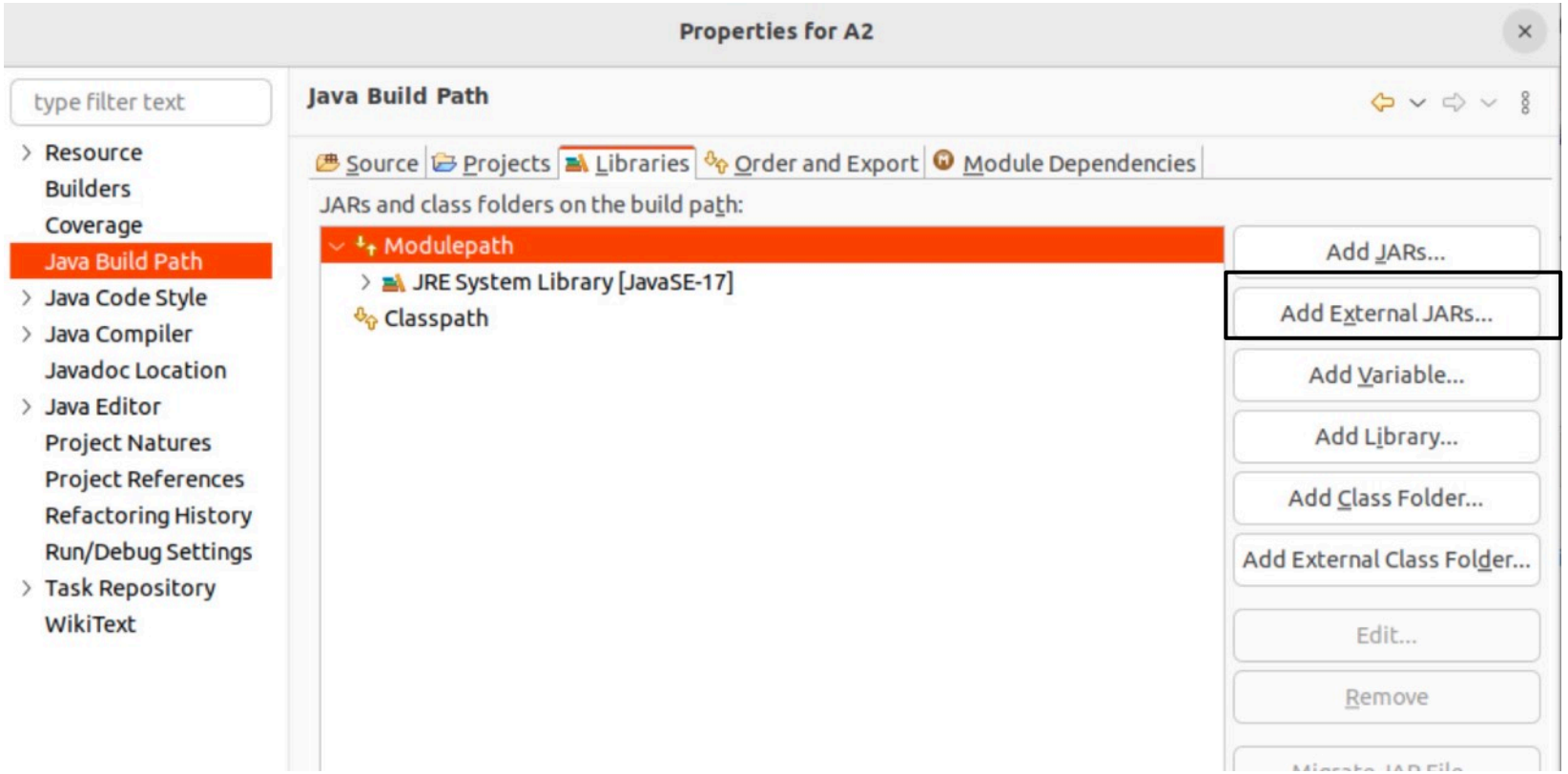
Add the path to CLASSPATH environment variable

export CLASSPATH=$CLASSPATH:/path/to/mysql-connector-java.jar



OR Setting up project in Eclipse

- Right click on your project → properties



Implementation: insert()

```
try {
    PreparedStatement stmt =
            conn.prepareStatement("INSERT INTO c3358_2025 (name, birthday) VALUES (?, ?)");

    stmt.setString(1, name);
    stmt.setDate(2, java.sql.Date.valueOf(birthday));
    stmt.execute();

    System.out.println("Record created");

} catch (SQLException | IllegalArgumentException e) {
    System.err.println("Error inserting record: "+e);
}
```

Use prepared statement to input parameters

Execute statement

Implementation: read()

```
try {
    PreparedStatement stmt = conn.prepareStatement("SELECT birthday FROM c3358_2025 WHERE name
= ?");
    stmt.setString(1, name);

    ResultSet rs = stmt.executeQuery();
    if(rs.next()) {
        System.out.println("Birthday of "+name+" is on "+rs.getDate(1).toString());
    } else {
        System.out.println(name+" not found!");
    }
} catch (SQLException e) {
    System.err.println("Error reading record: "+e);
}
```

Use result set object to retrieve results

Implementation: list()

Use statement object for queries without parameters

Specify list of columns

```
try {
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT name, birthday FROM c3358_2025");
    while(rs.next()) {
        System.out.println("Birthday of "+rs.getString(1)+" is on "+rs.getDate(2).toString());
    }
} catch (SQLException e) {
    System.err.println("Error listing records: "+e);
}
```

Use while loop to read all results

Implementation: update()

```
try {
    PreparedStatement stmt = conn.prepareStatement("UPDATE c3358_2025 SET birthday = ? WHERE name
= ?");
    stmt.setDate(1, java.sql.Date.valueOf(birthday));
    stmt.setString(2, name);

    int rows = stmt.executeUpdate();
    if(rows > 0) {
        System.out.println("Birthday of "+name+" updated");
    } else {
        System.out.println(name+" not found!");
    }
} catch (SQLException e) {
    System.err.println("Error reading record: "+e);
}
```

Use executeUpdate() for update

Return number of rows updated

Implementation: delete()

Always specify WHERE clause for delete!

Use executeUpdate() to check number of rows deleted

```
try {
    PreparedStatement stmt = conn.prepareStatement("DELETE FROM c3358_2025 WHERE name = ?");
    stmt.setString(1, name);
    int rows = stmt.executeUpdate();
    if(rows > 0) {
        System.out.println("Record of "+name+" removed");
    } else {
        System.out.println(name+" not found!");
    }
} catch (SQLException | IllegalArgumentException e) {
    System.err.println("Error inserting record: "+e);
}
```

Exercise
Part I: finish the implementation in the slides

(40%)

- Complete the implementation as suggested in the slides
- Add a new command "month" that takes one argument (the month) and displays the names of all records whose birthdays fall in that month.
- Try to do three operations: create, update and delete records in the database through your program
- Submit all your final *.java file(s) and a document (better in pdf format). The document should contain your screen shots (your operation outputs) on running each of the three operations.

Exercise
Part II: replace the storage in Assignment I

(60%)

- Design and create new Tables for UserInfo and OnlineUser in Assignment 1 and implement the method for insert(), read(), update(), and delete() on the new tables, so that all user data will be stored in the database instead of txt files.

- Submit all your final *.java file(s) and a document (better in pdf format).
- The document should contain the design of new tables and your screen shots on the content of the new tables after running several user commands in Assignment 1. Consider whether transactions should be implemented in this application, and provide your reasoning.