Assignment 2
JDBC

COMP3358 Distributed and Parallel Computing

Overview

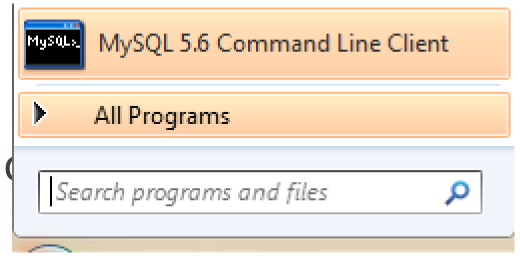
- Setting up MySQL database
- Using MySQL JDBC driver
- Writing Java program
  - Connect to database
  - Create, read, update, delete

Installing MySQL server

- You may use the department MySQL account instead
  - Get your account at: https://intranet.cs.hku.hk/csintranet/contents/technical/howto/database.jsp
- Or download and install MySQL Community Server
  - http://dev.mysql.com/downloads/mysql/

Preparing database (for own installation)

- Fire up MySQL command line client
  - Enter root password (configured during installati
- Enter these commands:



```
CREATE DATABASE c3358;

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

USE c3358;

CREATE TABLE c3358_2017_t4 (
  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**

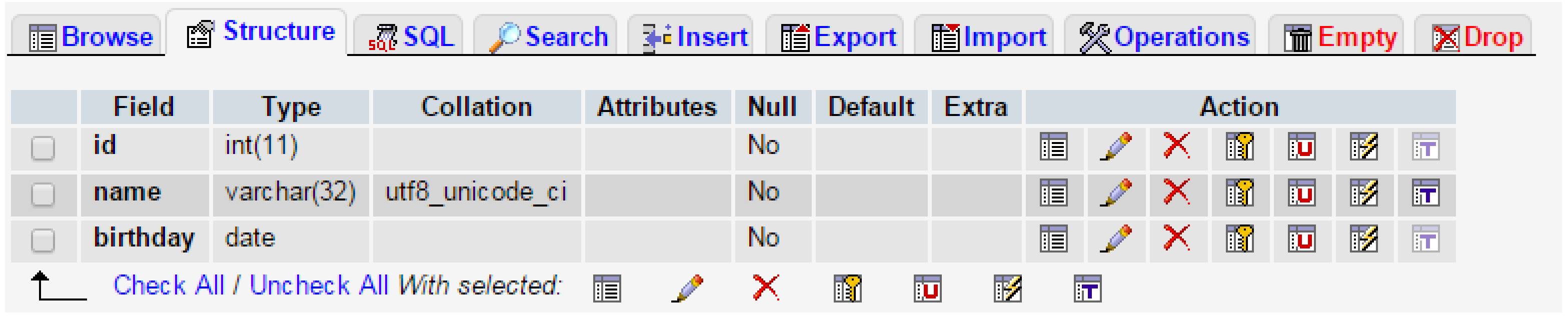
Create table named **c3358_2017_t4**

You can check it by the command DESCRIBE **c3358_2017_t4**

Preparing database (CS account)

- Login http://i.cs.hku.hk/tools/phpMyAdmin/
- Click on your database (your CS account name)
- Click “SQL”
- Execute this SQL

```
CREATE TABLE c3358_2017_t4 (
  name varchar(32) NOT NULL,
  birthday date NOT NULL,
  PRIMARY KEY name (name)
);
```



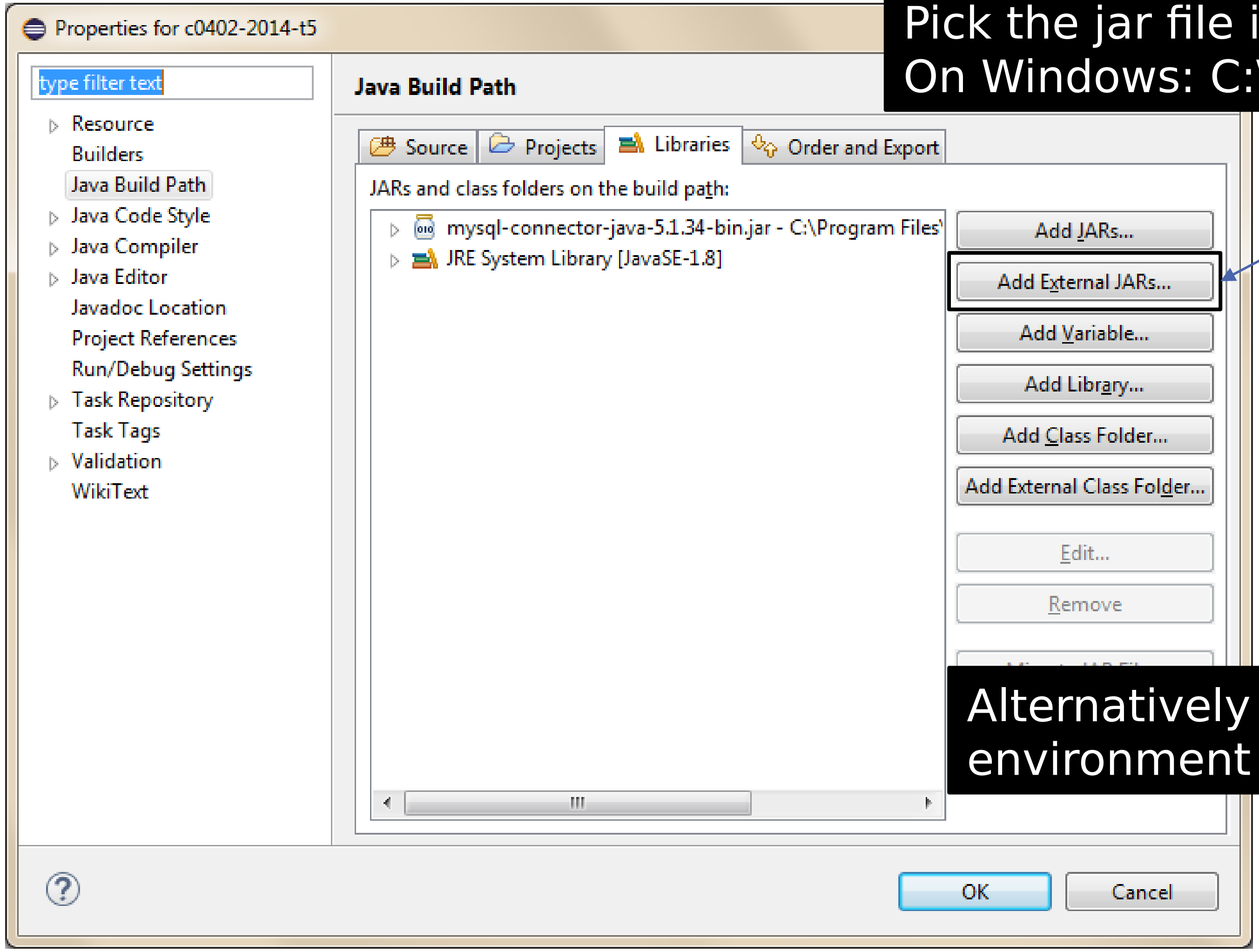
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	id	int(11)			No			
	name	varchar(32)	utf8_unicode_ci		No			
	birthday	date			No			

JDBC driver

- Download MySQL Connector/J (already installed in lab)
  - http://dev.mysql.com/downloads/connector/j/

Setting up Eclipse project

- Right click on your project properties



Pick the jar file in Connector/J installation
On Windows: C:\Program Files\MySQL\MySQL Connector J

Alternatively you can add the path to CLASSPATH environment variable

Connecting with Java

- Download JDBCDemo.java from Moodle
- Set up MySQL login

```
private static final String DB_HOST = "sophia";
private static final String DB_USER = "";
private static final String DB_PASS = "";
private static final String DB_NAME = "";
```

	CS MySQL	Your own server
DB_HOST	sophia	localhost
DB_USER	Your CS id	c3358
DB_PASS	Your MySQL password	c3358PASS
DB_NAME	Your CS id	c3358

- Set up JDBC connection

```
public JDBCDemo() throws … {
    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.");
}
```

Execute the program, type “exit” to end.

insert()

Use prepared statement to input parameters

```
try {
    PreparedStatement stmt =
            conn.prepareStatement("INSERT INTO c3358_2017_t4 (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);
}
```

Execute statement

read()

```
try {
    PreparedStatement stmt = conn.prepareStatement("SELECT birthday FROM c3358_2017_t4 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

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_2017_t4");
    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

update()

```
try {
    PreparedStatement stmt = conn.prepareStatement("UPDATE c3358_2017_t4 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

delete()

```
try {
    PreparedStatement stmt = conn.prepareStatement("DELETE FROM c3358_2017_t4 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);
}
```

Always specify WHERE clause for delete!

Use executeUpdate() to check number of rows deleted

Exercise

- Complete the implementation as suggested in this tutorial
- Add a command “birthday” which takes 1 argument (the birthday) and print out the names of all records with the specific birthday
- 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 word document. The document should contain your screen shots (your operation outputs) on running each of the three operations. The format of the doc is flexible.