**Lab 5 – Database Programming**

**This is a two-week lab**

The objective of this lab is to enable you to do database programming from within you programs specifically as it pertains to programs written in Java. It is simple to see how databases can be accessed from any programming language. For the sake of this lab, we will be focusing on the Java programming language.

In this lab, you will write a Java/python program that will allow the users to perform the list of operations that you had conceptualized in Lab 1 and operationalized through SQL commands in Lab 4 through a program command line.

**Step 1.**

Before you can write a program that accesses the database, you will need to have the driver for accessing your database in SQLite. This is called the JDBC driver (Java Database Connector). You can download the right driver from <https://bitbucket.org/xerial/sqlite-jdbc/downloads>

**Step 2.**

Now, you need to create a new Java project in your IDE of choice, NetBeans (*which has been installed in lab computers)*, Eclipse or any other environment. To be able to include JDBC related statements in your Java code, you need to include the following packages in your code:

import java.sql.\*;

You will have to first connect to the database and ensure that the connection is established without any problems. This can be achieved by:

Connection c = null;

try {

Class.*forName*("org.sqlite.JDBC");

c = DriverManager.*getConnection*("jdbc:sqlite:/home/Moviesion.db");

if(c == null)

throw new Exception("Connection to the database is null", null);

c.setAutoCommit(false);

}

catch (Exception e){

System.err.println("Problem Encountered!");

e.printStackTrace();

throw e;

}

System.out.println("Opened database successfully.");

You will need to include the full file path of your own database instead of /home/Moviesion.db. You can find more help here: <http://www.sqlitetutorial.net/sqlite-java/sqlite-jdbc-driver/>

**Step 3.**

Once you have created a connection to the database, you can access the tables through appropriate Java commands and send your SQL queries. Depending on the SQL query, you will receive appropriate data or response back from the database that you can use in your code. Let’s assume you want to create a new table, you can do this by:

Connection c = null;

Statement s = null;

try {

//Code to connect to database goes here (step 2)

s = c.createStatement();

String sql = "CREATE TABLE Movie (Id INTEGER PRIMARY ...";

s.executeUpdate(sql);

s.close();

c.close();

}catch(Exception e){

e.printStackTrace();

throw e;

}

System.out.println("Table created successfully.");

The sql variable will need to be populated with the appropriate CREATE TABLE query. Similarly, an INSERT statement in Java can be performed by using same code snippet but different sql query as follows:

Connection c = null;

Statement s = null;

try {

//Code to connect to database goes here (step 2)

s = c.createStatement();

String sql = "INSERT INTO Movie(Title, ...";

s.executeUpdate(sql);

s.close();

c.close();

}catch(Exception e){

e.printStackTrace();

throw e;

}

System.out.println("Records inserted successfully.");

An UPDATE or DELETE operation in SQL can be done quite similarly by replacing sql variable with proper sql query. Please make sure to include c.close() and s.close() after you have done inserting your data and you are done with using a database connection.

Now, a SELECT statement that retrieves some data has some more details that we need to explore:

First, we need to send the SELECT statement through JDBC to the database:

s = c.createStatement();

ResultSet rs = s.executeQuery("SELECT Id, Title, ... FROM Movie");

The database will respond with data that match the SELECT statement, all of which will be accessible through the rs variable. Depending on how many rows matched the select query, rs would contain zero or more rows. To access all the retrieved results, we need to loop through all the rows in rs:

while (rs.next()) {

//code to access rows goes here

}

Now depending on the type of the columns retrieved in the SELECT statement, you can use one of the following statements to retrieve the columns in each row; For instance,

String movieTitle = rs.getString("Title");

Retrieves the value of a column called Title whose type is String and places the value in a variable called movieTitle. Similarly, for other column types:

int movieId = rs.getInt("Id");

float movieRunningTime = rs.getFloat("RunningTime");

When such statements are placed inside the loop, every time that the loop iterates, one of the retrieved rows are accessed and the value for the columns of that row will be accessible through the getX() functions:

while (rs.next()) {

String movieTitle = rs.getString("Title");

int movieId = rs.getInt("Id");

float movieRunningTime = rs.getFloat("RunningTime");

System.out.println(String.*format*("The movie with Id = %d is %s and its

running time is %f", movieId, movieTitle, movieRunningTime));

}

For this lab, you will need to use the above JDBC functionality to fully implement all the queries that you had designed in Lab 4 through a Java program. When the Java program is run, it should show a prompt with all the possible actions that the user can execute. For instance,

>> welcome to the movie database application:

to see the list of movies, type 1

to see the list of directors, type 2

to add a new movie, type 3

Once the user chooses an option, the program should execute the appropriate SQL command and retrieve the results for the user. If needed, for example in the case of entering a new movie, the program should ask the user for additional information needed to create the new movie. For example

>>please enter the title of the movie:

Inception

>>please enter the name of the director:

Christopher Nolan

Once the operation is completed the program should provide the user with the retrieved appropriate data or message and return back to the original program menu and wait for a new command by the user.

**Deliverables**

You should complete the steps described above. Then, you will prepare and submit the results in one single zip file (*{YourFullName}*\_Lab4.zip) containing the following item:

1. You will need to provide a brief document as to how your code should be compiled and whether there are any known issues with the code or whether there are certain cautions that need to be taken when executing the code.
2. A dump of your tables that would include the structure and data of your tables. Make sure each of your tables have 10 rows in them. You can do this by performing the following command on terminal:

sqlite3 *{DatabaseName}* .dump

1. A zip file of all the source code of your program that can be compiled. Please include the clear java command needed to compile your code in your brief document (item #1). Failure to do so might lead to the inability of the TAs to mark your work.
2. A PDF document with the screenshot of your program when run and the data that is returned. Make sure you include one screenshot for each of the options in your program’s main menu. For instance, one screenshot when the user asked for the list of movies, one screenshot for when the user asked for all directors, one for when the user entered a new movie etc.