Data Structures and Representation (ZEIT2103) – 2022

Assignment 2 – Files, Databases, and SQL – Olympic Medals

100 Marks = 13%

Due Date: 11:55 pm, Friday 6 May 2022

Assignment Summary

In this assignment, you will create a multi-table database and comment on the advantages and disadvantages of this design. I will specify the relational database schema you must follow. You will write code to implement my design by creating tables in a MySQL database. You will then populate your tables with data from comma-separated text files and run some queries on your new database.

Assignment Background

This assignment tests your understanding and ability to use Java, JDBC, and SQL to access a MySQL database.

The problem you will solve involves a database[1] of Olympic medal winners from 1896 to 2016. The database has four tables with information about Olympic cities, events, athletes, and medals.

In this assignment, you will create a multi-table database and comment on the advantages and disadvantages of this design.

Getting Started – Resources

I have given you a completed class called `OlympicDBUserInterface.java`. This class will be the entry point to the program and will provide a simple console-based menu from which you can launch the implementations for each of the tasks in this assignment. This class should not be modified.

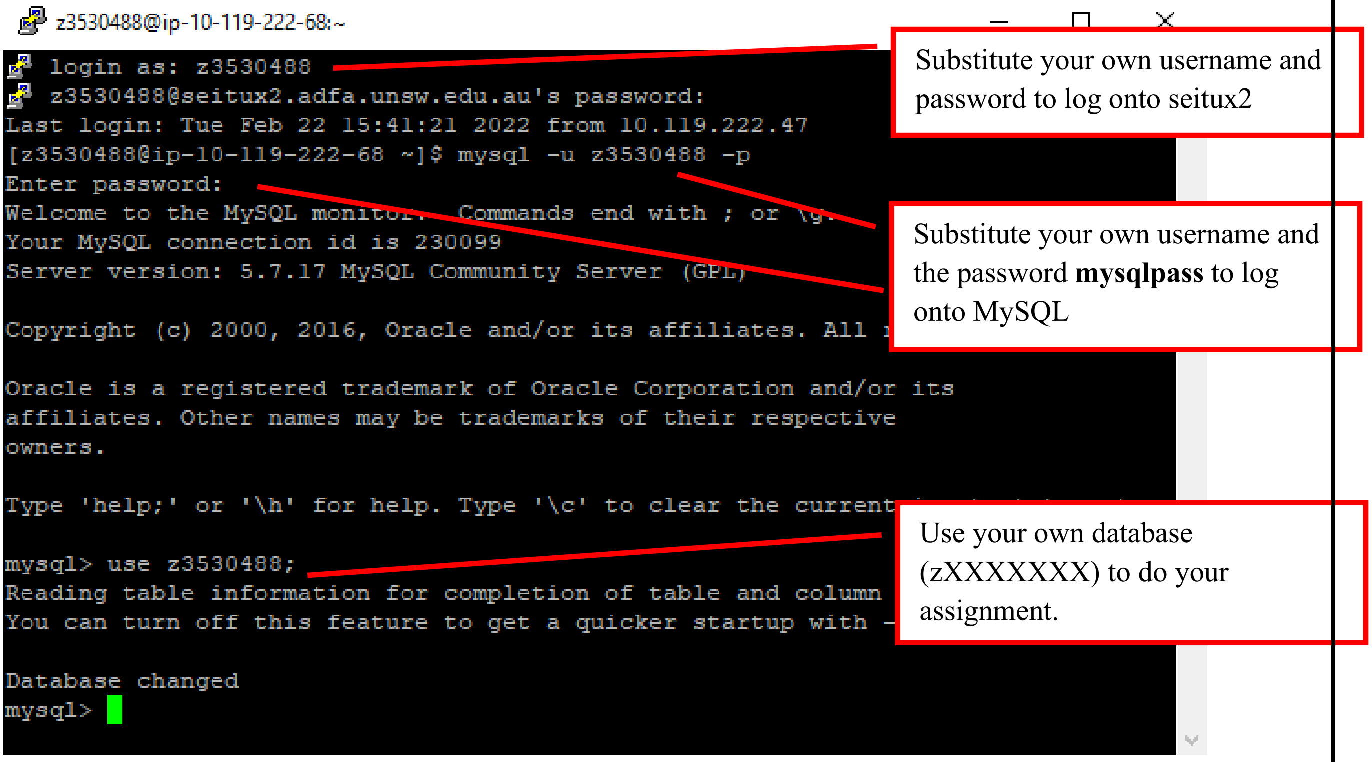
You also `have` been provided with a skeleton class called `OlympicDBAccess.java` that will contain method stubs such that the two classes can compile. Your tasks, as follows, will be to iteratively complete the methods within this class. You should first ensure that you are able to compile the classes and execute the menu, although it won't do much yet!

Accessing MySQL

1. Log onto `seitux2.adfa.unsw.edu.au` using putty or a similar tool. Use your zID and zPass to login.
    a. Further information about how to access putty and move files to/from seitux2 can be found here: how to access putty.pdf (Note: you must be logged into Moodle to access this document. Alternatively, it is available on the course Teams channel).
2. Login to MySQL using the following command with your zID:
```
mysql -u zXXXXXXX -p
```
3. You will be prompted for a password. Enter the student password as `mysqlpass`

[1] Data from https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results/version/2
Image from https://7news.com.au/sport/winter-olympics/jakara-anthony-finally-receives-her-gold-medal-after-stunning-womens-moguls-win-c-5607923, Credit: Getty Images

   a. **<u>THIS IS NOT YOUR zPASS!</u>**
4. Once logged in, you will see a MySQL prompt like the image below.
5. To change into your own database, use the 'use' command with your own zID:
   ```
   use zXXXXXXX;
   ```
6. To have a look at the tables in the database type:
   ```
   show tables;
   ```
   If you use show tables, you should see your database is currently empty.



```
z3530488@ip-10-119-222-68:~
login as: z3530488
z3530488@seitux2.adfa.unsw.edu.au's password:
Last login: Tue Feb 22 15:41:21 2022 from 10.119.222.47
[z3530488@ip-10-119-222-68 ~]$ mysql -u z3530488 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 230099
Server version: 5.7.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All r

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

mysql> use z3530488;
Reading table information for completion of table and column
You can turn off this feature to get a quicker startup with -

Database changed
mysql>
```

Substitute your own username and password to log onto seitux2

Substitute your own username and the password **mysqlpass** to log onto MySQL

Use your own database (zXXXXXXX) to do your assignment.

Assignment Tasks

Task 1 – Creating the Database [25 marks]

In this task, you will write Java code that builds a multi-table database and an accompanying method to drop the tables in the database. You will use the schema below for your database. There will be 4 tables with names as shown. You will use the auto-increment feature in SQL to create unique IDs for records in each of the four tables, which will also serve as the primary keys.

1. Implement the constructor in `OlympicDBAccess.java`, which should initialise the JDBC database connection such that it is available for use in the remaining tasks. Your constructor should gracefully handle any errors that may arise.
2. Implement the method `createTables()` that creates the tables according to the schema below, including the primary and foreign key constraints. Note, you may write additional helper methods as you see fit (e.g., `createOlympicsTable(…)` etc.), but do not change the signatures or names of the existing methods.

Olympics

ID	INT NOT NULL AUTO INCREMENT (primary key)
year	INT
season	VARCHAR(7)
city	VARCHAR(23)

Events

ID	INT NOT NULL AUTO INCREMENT (primary key)
sport	VARCHAR(26)
event	VARCHAR(86)

Athletes

ID	INT NOT NULL AUTO INCREMENT (primary key)
name	VARCHAR(94)
noc	CHAR(3)
gender	CHAR(1)

Medals

ID	INT NOT NULL AUTO INCREMENT (primary key)
olympicID	INT, foreign key references Olympics(ID)
eventID	INT, foreign key references Events(ID)
athleteID	INT, foreign key references Athletes(ID)
medalColour	VARCHAR(7)

3. Implement the method `dropTables()` to drop all tables from the database. This should work even if there are no tables present. This method may be useful when testing your data population method(s) in Task 2.
4. In your assignment report, answer the following:
   a) For the Athletes table, would using the name as the primary key be acceptable?
   b) For the Medals table, is there an alternative (possibly composite) primary key that could be used rather than having an explicit row ID? If so, what benefits would using this composite key have?
   c) Why is 3NF desirable in a database?

Notes: (1) You will likely want to test your SQL directly, via putty, before putting it in your Java program. (2) Don’t forget to include the library **mysql-connector-java-5.1.29-bin.jar** in your Java project’s build path. This is what allows the Java program to communicate with MySQL. (3) You may find it necessary to transfer and run your code directly from seitux2 (see the *How to access putty.pdf* document for further help on how to do so). Otherwise, you may be unable to access the database, depending on where you are running your code from.

Task 2 – Populating the Database [45 marks]

In this task, you will implement the `populateTables()` method to populate each table in your database. Again, you may wish to write helper methods to handle each of the tables separately, with the primary method calling each of the helper methods as appropriate. Remember that each of the tables has an auto-incremented ID, so be sure you are inserting the data into the correct columns. A simplistic timing mechanism is already present. Report the time taken to populate all the tables in your assignment report.

For the first three tables (Olympics, Events, Athletes), the data in the files matches the columns in the corresponding tables (except for the auto-incremented ID column). Each line in the file corresponds to a record that should be inserted into the appropriate table. These files should be relatively straightforward to read and insert.

1. The file `olympics.csv` contains information about each Olympic session. This data file contains 3 columns: the year, the season (Summer or Winter), and the name of the city. Data should be inserted into the Olympics table.
2. The file `events.csv` contains information about each sport/event. This data file contains 2 columns: the sport, and the name of the specific event. Data should be inserted into the Events table.
3. The file `athletes.csv` contains information about each (unique) Athlete that has received a medal. This data file contains 3 columns: the name, the National Olympic Committee (NOC; a three-letter code representing the country), the gender (M or F). Data should be inserted into the Athletes table.

The fourth table (Medals) will take a bit more processing to populate as the data in the file does not directly match the schema for the Medals table. Rather, you will need to use the data in the file to lookup the corresponding olympicID, athleteID, and eventID from the appropriate tables, then use these IDs when populating the Medals table.

4. The file `medals.csv` contains information about each medal that has been won for 120 years of Olympics. This data file contains 9 columns: the athlete's name, the athlete's gender, the athlete's NOC, the Olympic year, the Olympic season, the Olympic city, the event sport, the event name, and the medal colour. Populating this table can be done in a few steps, which need to be done for every row in the data file.
   a. Use the athlete's name, gender, and NOC to find the corresponding ID from the Athletes table.
   b. Use the Olympic year, season, and city to find the corresponding ID from the Olympics table.
   c. Use the event's sport and name to find the corresponding ID from the Events table.
   d. Insert the olympicID, athleteID, eventID, and medal colour into the Medals table.

Notes: (1) Reading the files and populating the database will likely take somewhere between 10 and 15 minutes using a reasonable implementation. Particularly inefficient solutions may take significantly longer. (2) You will need to ensure that the first three tables are populated before attempting to populate the Medals table, as otherwise, you will not be able to look up the IDs. (3) In each of the datafiles, String data is encapsulated in double quotes ("…") – these quotations should not be removed and are there to make inserting the data easier!

Task 3 – Querying the Database [30 marks – no assistance given]

In this task, you will implement the `runQueries()` method in Java that queries your database and returns information. The result of each query should be appropriately formatted and displayed to the console. Again, you may wish to write helper methods to handle each of the queries separately, with the main `runQueries()` method calling, and displaying the results of each of the helper methods in turn. For each query, include the output from your program in your assignment report.

1. The number of distinct events that have the sport 'Athletics'.
2. The year, season, and city for each Olympics, ordered with the earliest entries first.
3. The total number of each medal colour awarded to athletes from Australia (NOC: AUS) over all Olympics in the database.
4. The name of all athletes from Ireland (NOC: IRL) who won silver medals, and the year / season in which they won them.

Marking and Submission

You are required to refer to and follow the document '**Requirements and Expectations for Programming Assignments'**. If you do not, you are likely to be seriously disappointed by your final mark!

If you are aiming for a basic pass/credit result, you need to show your mastery of the basics by completing the following tasks:

- provide and discuss the required designs and specifications, with appropriate justifications;
- ensure that your Java and SQL code is well designed, with design decisions documented;
- correctly and efficiently use your Java code to implement the classes and methods specified in the relevant tasks; use SQL appropriately within this code
- add suitable Javadoc comments to your Java code for each class, attribute and method, and include inline comments on methods where necessary to clarify the intent of the code;
- design your Java code in a modular way using a sensible set of methods;
- ensure that the layout and indentation of your Java code and SQL is consistent and readable;
- include appropriate test strategies for the code you developed; and
- correctly use proper Java and SQL name conventions

To aim for a higher mark, you need to complete all the specified tasks to the basic standard (above) and also discuss the extensions you tackled, the research you conducted, your designs, evidence of your rigorous testing, such as an error-handling code.

When you have completed your work, you need to submit your `OlympicDBAccess.java` and `ass2.pdf` through Moodle. Note: you do not need to submit `OlympicDBUserInterface.java` as it should not have been modified.