CSCI 585 – Database Systems – Fall 2013
Professor Dennis McLeod
Assignment 3
Due: November 14, 2013 @ 02:00 PM PST
[Electronic submission via https://www.uscden.net/]

In this assignment you will realize a practical database implementation based on a given relational schema within the MySQL database management system, write and run queries on it, and access it programmatically using Java Database Connectivity (JDBC). The tasks that you will accomplish in this assignment area as follows:

- create a database using DDL statements in the `mysql` command line tool
- populate the database row-by-row from provided raw data using JDBC
- write SQL queries to the database to be executed from the `mysql` command line tool
- query and manipulate the database programmatically using JDBC
- completely remove all traces of your database from the DBMS

Preliminaries: Installing and Using MySQL Community Server

The current Generally Available release of MySQL is version **5.6.14**. Download link is given below. Detailed installation instructions can be found in the reference manual, which also provides a tutorial, extensive documentation of client and administrative programs, data types, SQL statement syntax, etc.

Download: http://www.mysql.com/downloads/mysql/5.6.html
Reference Manual: http://dev.mysql.com/doc/refman/5.6/en/

We will be using the `mysql` command line client to access the database; you can read more about it at http://dev.mysql.com/doc/refman/5.6/en/mysql.html. Other database management tools like MySQL Workbench, GUI Tools, phpMyAdmin, etc. are **not** required for this assignment; use of these tools will neither be accepted nor supported by the TAs and graders – *use at your own risk*.

Preliminaries: Java

For this assignment, we will use the **Java Standard Edition (SE) 6** or higher. Note that we will not be able to accommodate any compatibility issues if you use an older version.

Download & instructions: http://www.oracle.com/technetwork/java/javase/downloads/index.html
Documentation: [Java 6] [Java 7]
API Specification: [Java 6] [Java 7]

Resources for those who are new to Java:

The Java Tutorials: http://download.oracle.com/javase/tutorial/
Thinking in Java (free e-book): http://www.mindview.net/Books/TIJ/

In case you don't have access to a machine with requested version of the JDK, the USC Student Compute Facility (SCF) has Java SE 6 installed on its servers; you can use this to verify that your programs will compile correctly in Java 6. Documentation on how to access SCF resources is available at http://www.usc.edu/its/unix/servers/scf.html. To setup your SCF server environment to use Java SE 6, run the following command from the terminal. Note this will only work for the current login session. To make this permanent, you need to append this command to your `~/.login` file:

```
source /usr/usc/jdk/1.6.0_23/setup.csh
```

You are free to use any Integrated Development Environment (IDE) such as Eclipse, NetBeans, etc. if you wish. However, you **must be able to compile and execute your code from the Windows command line (or Linux/Mac shell) environment**, outside of the IDE.

Preliminaries: JDBC

To connect to your MySQL database from within your Java program, you will need to use JDBC. The current Generally Available release of the official MySQL JDBC driver is **MySQL Connector/J 5.1.26**. It can be obtained from http://www.mysql.com/downloads/connector/j/

JDBC Documentation: [Java 6] [Java 7]
API Specification: [Java 6] [Java 7]
Tutorial: http://download.oracle.com/javase/tutorial/jdbc/basics/index.html
MySQL Connector Documentation: http://dev.mysql.com/doc/refman/5.6/en/connector-j.html

You may NOT use any other third-party libraries apart from MySQL Connector/J.

Connecting to the Database

To connect to your MySQL database from within your Java programs, you will need to supply it with the necessary connection parameters. For this assignment, these parameters **must** be stored in a separate configuration file to be passed as an argument to your Java program during execution. Your program should read the configuration file and use the parameters contained therein. **Do not hardcode any of the connection parameters in your Java source code.**

A sample `dbparams.txt` file has been provided for your reference. The five lines of this plain text file correspond to the host, port, database name, username and password for the database. Default values for host and port values have been used; you may need to adjust them accordingly if you had changed your settings when installing MySQL. Obviously, you will need to change the username and password according to your individual system setup in order to connect to your own MySQL instance.

For grading, we will of course use a different `dbparams.txt` configuration file based on the setup of the machine used for grading. Make sure you **follow the configuration file format. We will not be able to grade your assignment if your program fails to parse the connection parameters from our database configuration file.**

Application Domain Description

StarCraft II is a real-time strategy video game that is widely featured in professional video game competitions, with sponsored players and teams, and multiple ongoing leagues and tournaments. You will implement the database backend for a tournament results tracker using the open source MySQL Community Server pseudo-relational database management system.

Obviously, such a system will need to keep track of various players; each player has a *tag*, or nickname that they use in the game, along with their real name. As neither is guaranteed to be unique, an artificial player ID is introduced here. Also recorded are the date of birth of the player, their nationality (using a two-letter country code), and the "race" that they typically play as within the game, indicated using a single letter representing *P*rotoss, *T*erran or *Z*erg.

The system also stores information about professional gaming teams, both past and present: the team name and date of founding; for former teams, the date of disbandment is also stored. Players' membership in teams are recorded with the start date, and where applicable, the end date (if the player is currently still on the team, there is no end date).

The application keeps track of the name of each tournament and the region that it was played in (represented by a two-letter code). Some tournaments are flagged as a "major" event (generally characterized by large amount of prize money and participation of strong players).

The most important part of the system, of course, records the results of each match: the date and tournament at which it was played and the two players in the match. A match consists of one or more games; the score of each player represents the number of games that they each won in that match, and the player who wins more games is the match winner. A flag indicates if the match was "offline", i.e. it was played in-person at a physical event (as opposed to "online" over the Internet).

Finally, we keep track of the amount of prize money (in USD) that players win at tournaments.

Part 1: Creating the database (15 points)

The schema for our database is given by Fig. 5. Primary keys have been underlined. Foreign keys have been omitted, but you are expected to identify and implement them accordingly.

You are expected to prepare and submit a `createdb.sql` file containing SQL statements that will create a database `ResultTracker` containing the tables listed in Fig. 5. Primary keys and foreign keys *must* be implemented. Choose the most appropriate data type for each field[1].

[1] Hint: refer to the domain description, and consult the sample data first.

Players

player_id	tag	real_name	nationality	birthday	game_race

Teams

team_id	name	founded	disbanded

Members

player	team	start_date	end_date

Tournaments

tournament_id	name	region	major

Matches

match_id	date	tournament	playerA	playerB	scoreA	scoreB	offline

Earnings

tournament	player	prize_money	position

Fig. 5: Schema for `ResultTracker`

Part 2: Populating the database (10 points)

You are given samples of the following files, which contain sample data for the respective tables.

- `players.csv`
- `teams.csv`
- `members.csv`
- `tournaments.csv`
- `matches.csv`
- `earnings.csv`

These are standard comma-separated value files, which you may open and examine using any standard text editor. Observe that each row represents one record, and fields are comma-separated.

Note: The data files provided to you are samples demonstrating file format and attribute domains only. When grading, we will test your programs using a different set of unseen data. As such, you may wish to create additional data for your own testing purposes.

You are required to implement a Java program `Populate.java` that will accept filenames of the input data files as command line parameters, open and parse each file, and populate the data contained within them into your database by executing ***individual `INSERT` statements for each row*** using JDBC. Note: when executing the same statement repeatedly, you should use the JDBC `PreparedStatement` construct for greater efficiency as compared to the more generic `Statement` class.

Your program should be compiled and executed from the Windows command line or Linux/Mac shell. It should be compiled like this[2]:

```
> javac -classpath .;mysql-connector-java-5.1.26-bin.jar Populate.java
```

And executed like this (as one single long command):

```
> java -classpath .;mysql-connector-java-5.1.26-bin.jar ↩
     Populate <dbparams filename> <players filename> <teams filename> ↩
      <members filename> <tournaments filename> <matches filename> ↩
      <earnings filename>
```

Do not hardcode the locations and/or filenames of the configuration file and input data files into your program. We will use different filenames at different locations for grading.

Sample command line batch files / shell scripts for both compilation and execution are provided. Note that every time you run this program, it should start by first removing the previous data in your tables; otherwise the tables will have redundant or incorrect data.

Part 3: SQL queries on the database (55 points)

Write the following queries in SQL and run them on your MySQL database via the `mysql` command line tool. Prepare and submit them as separate files `q1.sql` to `q8.sql`. If two or more SQL statements are needed for a single question, they should be written after each other in one file.

Do not create or use views. Storing intermediate query results is not an appropriate use of views.

Q1. List the real name and birthday of each non-Korean ("KR") player who was born in 1985.

Q2. Give the tag, real name, and nationality of all Zerg (Z) players that are currently on a team, along with the name of their current team.

Q3. List the tournaments that give out USD10,000 or more of total prize money. For each tournament, give the name, region, and the total prize purse. Sort your results with the tournament with the most prize money first.

Q4. List all offline matches where one player defeated the other with a 4-0 score. For each such match, give the name of the tournament that it was in, the date that the match was played, and show the match result in a single field, according to the following format:

WinnerTag (WinnerRace) 4-0 LoserTag (LoserRace)

for example:

Bomber (T) 4-0 Jaedong (Z)

[2] Commands shown are for Windows. Linux/Mac users please consult sample shell scripts provided.

Q5. List the *former* members of the team "ROOT Gaming". Give the tag and real name of each player, and the date of their *most recent* departure from the team. *Caution:* some of the *current* members of the team rejoined the team after leaving it in the past – they should be *excluded* from the results of this query.

Q6. A "triple crown" is the accomplishment of having won a major championship (i.e. came in first position in a major tournament) in each of the three main regions, namely: Europe (EU), America (AM) and Korea (KR). List the tag and game race of each player who has managed to attain a triple crown at least once.

Q7. Extra credit (Optional)

From among the Protoss (P) players who have played at least 10 games against Terran (T) opponents, find those who were able to win more than 65% of their "P vs. T" matches. Give the tag, nationality, and P vs. T win rate (in percent) of these players. Sort the players according to their P vs. T win rate, with the most successful player first.
(Hint: the `IF()` function can help to tremendously simplify things here.)

Q8. Extra credit (Optional)

List all teams founded *before* 2011 that are still active (not yet disbanded). For each such team, give the team name, date founded, and the number of current team members who play Protoss, Terran and Zerg, respectively. Sort the teams alphabetically by name.

Part 4: Querying the database using JDBC (20 points)

You are required to implement a Java program `Hw3.java` that provides the capability to run queries on the system from the Windows command line or Linux/Mac shell environment. It should be compiled by running the provided `compile.bat` (Windows) or `compile.sh` (Linux/Mac):

```
$ javac -classpath .;mysql-connector-java-5.1.26-bin.jar Hw3.java
```

and executed like this:

```
$ java -classpath .;mysql-connector-java-5.1.26-bin.jar ↩
      Hw3 <dbparams filename> <query_number> <query_arguments>
```

Depending on the query number, your program should take the arguments provided and process the corresponding query, then terminate. (Query details are given below.)

Sample command line batch files / shell scripts are also provided:

- `p3q1.bat` and `p3q2.bat` (Windows)
- `p3q1.sh` and `p3q2.sh` (Linux/Mac)

You should test your program to ensure that they can be executed using the provided `*.bat` or `*.sh` files, depending on your OS. **Do not modify the contents of any of the `*.bat` or `*.sh` files; in particular, the classpath must be used as-is in order to ensure compliance with the grading environment.** You are also encouraged to come up with your own test cases using different query arguments. Remember that you can "reset" your database by running `dropdb.sql` (see Part 5), and then `createdb.sql` and `Populate.java` again.

Q1.

Given a year and month, provide the real name, tag and nationality of players who were born in that month. For instance, given:

```
$ java -classpath .;mysql-connector-java-5.1.26-bin.jar ↩
     Hw3 dbparams.txt q1 1990 10
```

Your program should find players who were born in October 1990 and print the results to the screen in tabular form.

Q2.

Given a player id and a team id, add that player as a member of the specified team, with the start date set according to the current system time. If the player is currently a member of another team, the database should also be updated to reflect their departure from the "old" team, with the end date set as above. If the player was already a current member of the given "new" team, no changes are necessary. For instance, given:

```
$ java -classpath .;mysql-connector-java-5.1.26-bin.jar ↩
     Hw3 dbparams.txt q2 1660 35
```

Your program should add player `1660` as a member of team `35`, if they were not already a member of that team. If player `1660` is presently a member of different team, that membership record must be updated as well.

Your program should display informative messages about the changes being made to the database (or lack thereof) based on the logic described above. It should also provide confirmation on whether each operation succeeded or not, and reasons for failure where applicable.

Important note on JDBC (Part 2 and Part 4)

You are expected to **incorporate sensible logic** into your programs. For example, if your program is asked to run a query involving an inexistent player or team, you should return an informative error message. Likewise, you should print an informative message if a query returns no results, to distinguish it from lack of output due to other errors.

Your programs are also expected to **handle database errors gracefully**, and return informative error messages before exiting rather than terminating abruptly. For instance, the connection to the database may fail, or the database user (as given in the configuration file) may not have the necessary permissions to perform the required operations.

You may assume, however, that your program will always be run according to the given command format – i.e., query arguments will be given in the specified order, and there will never be missing arguments. Dates given as argument for Q1 are also guaranteed to be valid dates.

Part 5: Cleaning up

Prepare and submit a `dropdb.sql` file that drops your entire database. It should remove your entire database, leaving no trace behind – this includes all types and tables that you created.

No marks are allocated for this section; however, failure to clean up fully will incur a penalty.

FAQ

Q: Can I use a different version of MySQL or a different RDBMS?

A: For consistency, all submissions will be graded using MySQL 5.6.14 as the common platform. This means graders will take your submitted SQL statements and execute them in 5.6. We do not guarantee compatibility with other versions; your submission must work in 5.6.14 during grading.

Q: Can I use MySQL Workbench / MySQL GUI Tools / phpMyAdmin / other tools?

A: This assignment does not require MySQL Workbench, GUI Tools, phpMyAdmin or other similar tools; use of such tools is unsupported. Note that we will not be able to provide assistance if you encounter issues when using such tools.

Q: What are these *.CSV files, and how do I open them?

A: Please see: http://en.wikipedia.org/wiki/Comma-separated_values

Q: What is an `*.sql` file? How do I make one? How do I run it?

A: An `*.sql` file is a text file containing SQL commands. You can use any plain text editor of your choosing to create and edit `*.sql` files. To run the files using the `mysql` command line client, see http://dev.mysql.com/doc/refman/5.6/en/batch-commands.html. Note that this is the same procedure that will be used to run your submitted files according during grading.

Of course, you can also use the `mysql` command line client interactively and keep track of your SQL statements and queries separately in a text file, then reorganize your solutions into the `*.sql` files for final submission.

Q: I am having trouble creating foreign keys! / I cannot insert data because of foreign key errors!

A: You must create the referenced table before you can create the foreign key that references it. There are no circular references in the schema so you should be able to create all the tables in some order that satisfies this. Also note that the column types must be exactly the same.

Likewise, if you are trying to insert some data that includes a reference to another table, you must have populated the other table first, otherwise you would be violating a referential integrity constraint by trying to reference something that does not exist.

Q: Can I use Eclipse / NetBeans / (insert IDE here) for Java programming?

A: Yes, but, you **must ensure that you are able to compile and execute your code from the Windows command line (or Linux/Mac shell) environment**, outside of the IDE. You should check using the provided `*.bat` or `*.sh` files before submission to ensure conformance.

Q: Can I use other java libraries or packages?

A: No. You may **NOT** use any other third-party libraries apart from MySQL Connector/J.

Q: What package should I use for my Java program?

A: The default package will suffice – no package declaration is needed.

Q: Where will the `dbparams` file / input files be located?

A: The location of all files will be specified using relative path with respect to the working folder, as demonstrated in the given `*.bat` and `*.sh` files.

Q: How should I set up my classpath variable?

A: Do **not** use the CLASSPATH system environment variable. Your environment variables only work on your system, and it is not guaranteed that another system (the one used for grading!) will have its environment variables defined exactly the same as yours. Furthermore, the program source files (`*.java`) and compiled class files (`*.class`) may end up in different filesystem locations than you have on your computer. For these reasons, you should not rely on environment variables for the compilation/execution of your program.

The recommended way to ensure that your programs will compile and run correctly anywhere is by using the -classpath switch in the compiler (`javac`) and interpreter (`java`) to specify the classpath on compilation or execution, as demonstrated in the given `*.bat` and `*.sh` files.

During grading:

- your source (`*.java`) files will be placed in the working directory
- after compilation, the class files (`*.class`) will end up in the same directory
- we will place the needed MySQL connector `.jar` file there as well
- and then execute from there.

Q: Can I use file names that are different than the ones specified for submission?

A: No! You **must not** alter the name, case, spacing or extension of any of the specified `*.sql` and `*.java` files. The grading processes uses these exact names and altering them will cause some files not to be graded.

Submission Instructions:

- This assignment is to be submitted electronically via the Assignments section of the USC DEN Blackboard at https://www.uscden.net. A guide on how to submit your is available at: http://ondemand.blackboard.com/r91/movies/bb91_student_submit_assignment.htm
  - You should have up to ten SQL files: `createdb.sql`, up to eight query files `q1.sql` to `q8.sql`, and `dropdb.sql`. You should also have two Java source files, `Populate.java` and `Hw3.java`. Failure to adhere to filenames will incur a 5 point penalty. Make sure the file extensions are correct! (e.g. *not* `*.sql.txt`)
  - Also prepare a `readme.txt` containing your name, USC id and email address. You may also (optionally) include comments in this file.
  - Do not submit `*.class` files. We will compile your program from submitted source.
  - Compress your `readme.txt`, SQL files and Java source files into a single zip archive with the filename `lastname_firstname_hw3.zip`. Use only standard zip format. Do **not** use other formats such as zipx, rar, ace, etc. Improper filename or format will incur a 5 point penalty. There is no need to include MySQL Connector/J in your submission.
  - Make sure that you have attached the file the when submitting. Failure to do so will be treated as non-submission. Late policy applies until correctly submitted.
  - Successful submission will be indicated in the assignment's submission history. We advise that you verify the timestamp, download and double check your zip file for good measure.
  - It is ok to modify and re-submit your **entire file**, but **only until the deadline.** Only the most recent on-time submission will be graded. If you need to make any changes, you must re-package and re-submit **the whole zip archive, containing all the files.** Also, **late amendments and/or additions will not be accepted.**
  - **Submit early** so that if anything goes wrong, you will have enough time left to rectify it.
- DEN students: Please follow the above electronic submission instructions outlined above. Do not submit to the DEN Homework Center.
- This assignment is due ***before*** the class at the specified due date and time.
- Late submissions: Follow the standard submission instructions above. Late submissions will be automatically detected according to submission timestamp. ***Do not submit by email.*** There is a 20% deduction per day or part thereof, starting from the submission deadline.
- Note that you must complete ***all*** the assignments and take ***both*** exams in order to pass the course.

Discussion Board and Student Collaboration Policy

- You should work on this assignment ***individually*** and within the realm of the USC Academic Integrity Guidelines.
- We encourage you to discuss general issues related to this assignment with other students ***without*** revealing and/or hinting at any answers.
- Warning: all submissions will be inspected using an automatic plagiarism detection system.

- A discussion board for this assignment is available on DEN's Blackboard system.
- Use the discussion board as your main resource to post questions related to the assignment.
- The TAs will participate in the discussions and answer questions on the board.
- Do not ask TAs homework-specific questions by email.
- The discussion boards are moderated, so your posts will not show up until after they have been approved. Please do not re-post the same message.

- ***Start your homework early***. Although the discussion board will remain open until the assignment deadline, the TAs cannot guarantee that they will be able to answer any/all last minute questions posted less than 24 hours before the deadline.

---

[3] Caution to students: referring to Aligulac will ***not*** help you – compared to the original, both schema and data have been heavily modified and simplified for the purposes of this assignment. ***Nothing*** at Aligulac should be interpreted as a suggestion of what a "correct" solution might look like.