**COMP 373 Databases**

Spring 2022

Bernie Boscoe

This document outlines your midterm/final project

**Overview**

You have the option of doing either a “standard” or an “open” final project.

The “standard” project is to design and build a database with a given dataset from scratch, then write a series of complex SQL statements to retrieve data. This is an **individual** project.

The “open” project is to build a backend using Spring boot that connects to a database with datasets of your choice from scratch. You may have **up to 2 people** per project. If there is any problem working with your team members that you cannot resolve by yourself, bring it to me as soon as possible. Last-minute complaints of the form “my partner did nothing” will not be entertained. Feel free to talk to me if you need any suggestions. You do not have to know anything about Spring boot if you want to do this project, just a willingness to learn something new.

If you would just like to learn the important basics and do the standard project, that is completely acceptable. The open project is for those who would like a challenging task, there is no penalty for not choosing this route.

**Grading**

Here are the important dates:

2/4: Your choice is due, if you decide on the Open project, you must complete step 1 parts a, and b of Open, where you will describe the dataset you will develop/use, ideas for what you might ask of it, and list your team members (BOTH people must ALWAYS submit to moodle!). If you choose the standard project, you may indicate that to me by just naming your choice.

3/6 Midterm portion of project is due. For standard, this is Parts 1 and 2 described below. For open, Parts 1,2 and 3 are due. I will be doing informal checkins as well to ascertain your progress up until the midterm point.

Each project will be graded based on the following criteria:

For the standard project, the following rubric applies: 1) timeliness- is your project on time? 2) Steps completion- have you done each of the steps outlined in the instructions? 3) Documentation- have you documented clearly each step you have done in a document, including screen shots if necessary to show results? 4) Accuracy-- did you implement the appropriate queries or other instructions as they were described?

For the “Open” final project, you will be graded based on the completeness and quality of your final product against your proposed work, as well as turning it in on time, and clear documentation of the steps of your work, including screenshots if necessary.

**“Standard” Project**

In the standard project, you will realize a practical database design and implementation based on a given dataset. Within the MySQL database management system, you will write and run queries on it, and access it programmatically using Java Database Connectivity (JDBC).

The tasks that you will accomplish in this project are as follows:

* design a database based on a given dataset
* convert design into schemas
* create a database using DDL statements in the mysql command line tool
* populate the database row-by-row (in loops) 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

In short, StarCraft II is a popular video game that allows top players to make tons of money through tournaments. Your job is to implement a database backend for a tournament results tracker using MySQL.

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, unique 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 amounts 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: Design database**

We provided you with a small sample of the dataset. Design the database schema. Start with an E/R diagram and convert it to a relational schema. Identify any constraints that hold in your application domain, and code them as database constraints. It is important to go over the samples of real data to validate your design. Do not forget to apply database design theory and check for redundancies.

**Part 2: Creating the database tables in MySQL**

Once your schema has been validated, 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 your schema. **Primary keys and foreign keys *must* be implemented**. Choose the most appropriate data type for each field by referring to the domain description and consulting the sample data first.

**Part 3: Inserting data to the database through JDBC**

You will be given the following files, which contain data for the respective tables you created.

* 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. You need to examine these data to make sure that you define the constraints of certain attributes - such as **not null, on default**.

You are required to implement a set of Java programs that accept table name and csv file names as command line arguments, open and parse each file and insert the data contained within them into your database.

You have the following options to do this task:

* (Not the best way to do this) For each table and each csv file, create a separate .java file to parse each file and insert into its corresponding table. In this case, you will need to create 6 java files and name them as follows:
  + PlayersInsert.java
  + TeamsInsert.java
  + TournamentsInsert.java
  + MatchesInsert.java
  + EarningsInsert.java
  + MembersInsert.java
* OR you can write one java file (**Insert.java**) that takes all of the table names and csv filenames as command line arguments and parse each set of data. If you choose to do this approach, your command line arguments will look something like this:

java Insert Players players.csv Teams teams.csv Members members.csv Tournaments tournaments.csv Matches matches.csv Earnings earnings.csv

You should use the **JDBC PreparedStatement** for greater efficiency as compared to the more generic Statement class and to reduce the risk of SQL injection.

**Part 4: SQL queries on the database**

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 q6.sql**. If two or more SQL statements are needed for a single question, they should be written after each other in one file.

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 USD 10,000 or more of total prize money. For each tournament, give the name, region, and the total prize. Sort your results with the tournament with the most prize money first.

Q4. Give the real name, game race and tournament name of all players that have over scored their opponent by at least 3 points in any matches. To clarify, you should return the player who won the match, not the one that lost by at least 3 points.

Q5. 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.

Q6. 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 5: Querying the database using JDBC**

You are required to implement a Java program **FinalProjectQuery.java** that provides the capability to run queries on the system and display results in the console.

Q1. Given a year and month, provide the real name, tag, nationality and the number of wins of players who were born in that month and that year.

For example, the following run command should find players who were born in May 1990 and print the results to the screen in the given format.

java FinalProjectQuery q1 1990 05

Output:

Young Jin Kim, SuperNova, KR, 7

...

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 example, the following run statement 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 a different team, that membership record must be updated as well.

java FinalProjectQuery q2 1660 35

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.

**“Open” Project**

The “open” project is to build a backend using Spring boot that connects to a database with datasets of your choice from scratch.

**Basic Requirements**

* You need to design and develop relational databases in mySQL. Any noSQL databases will not be considered as valid options;
* You need to use Spring boot framework to implement your backend logic;
* You need to write REST API endpoints to CRUD from your database;
* You may deploy your database on AWS RDS OR have it running on localhost;
* A frontend is optional.

Specifically, you will need to complete the following tasks over the course of this semester.

1. Pick a topic/an area that you are interested in and start looking for a real dataset. Your dataset should be relatively substantial. When choosing, keep the following questions in mind:

a. How do you plan to acquire the data to populate your database? Use of real datasets is highly recommended. You may use program-generated “fake” datasets if real ones are too difficult to obtain.

b. How are you going to use the data? What kind of queries do you want to ask? How is the data updated? Your application should support both queries and updates.

c. Since you are asked to create a relational database, the dataset should contain interesting and complex relationships.

2. Design the database schema. Start with an E/R diagram and convert it to a relational schema. Identify any constraints that hold in your application domain, and code them as database constraints. If you plan to work with real datasets, it is important to go over some samples of real data to validate your design (in fact, you should start Task 7 below as early as possible, in parallel to Tasks 3-6). Do not forget to apply database design theory and check for redundancies.

3. Create a sample database using a small dataset. You may generate this small dataset by hand. You will find this sample database very useful in testing, because large datasets make debugging difficult. It is a good idea to write some scripts to create/load/destroy the sample database automatically; they will save you lots of typing when debugging.

4. Think about how a typical user would use your site or app. Optionally, it might be useful to build a “canned” demo version of the site or app first (i.e., with hard-coded rather than dynamically generated responses). Do not spend too much time on refining the look of your interface; you just need to understand the basic “flow” in order to figure out what database operations are needed in each step of the user interaction.

5. Write SQL queries that will supply dynamic contents for the web pages or app you designed for Task 4. Also write SQL code that modifies the database on behalf of the user. You may hard-code the query and update parameters. Test these SQL statements in the sample database.

6. Setup Spring boot project to create models, controllers and repositories for your database. For each of the queries you need, create an API endpoint.

7. Acquire the large “production” dataset, either by downloading it from a real data source or by generating it using a program. Make sure the dataset fits your schema. For real datasets, you might need to write programs/scripts to transform them into a form that is appropriate for loading into a database. For program-generated datasets, make sure they contain enough interesting “links” across rows of different tables, or else all your join queries may return empty results.

8. Test the endpoints you developed in Postman for Task 6 in the large database. Do you run into any performance problems? Try creating some additional indexes to improve performance. Can you retrieve correct data from the database?

9. [Optional] Once you have tested all the endpoints, you may build the frontend portion of the project.