COMP 430/533 Assignment 6 (Project 1)

Overview

Teams of 1-2 students will write database code and a Java or Python application that supports basic operations and queries on a database using the Swimmer schema from the previous two assignments.

Again, COMP 533 students need to support relays, while COMP 430 students do not.

Teams

Teams must consist of 1 or 2 students from the same course. A team cannot combine students from both COMP 430 and COMP 533. Both members of a team will receive the same score, unless there are exceptional circumstances

(Note: I had origially planned to allow teams of 3 students. However, I was also originally planning to have more difficult requirements for the project.)

Table definitions

The database tables should be defined following the schemas of Assignment 5 as provided in our sample solutions. However, you will slightly modify these for the "revised" solutions of Assignment 4, which corrects a couple minimum cardinalities and adds swimmer names.

Application language and library

You can use either Java and JDBC or Python 3 and psycopg2. JDBC is among the standard libraries for Java. Psycopg2 is among the packages that you have already installed in order to support Jupyter Notebooks. Both libraries have tutorials and examples available on the web. Both are relatively simple libraries that are based upon passing strings which are explicitly SQL code.

Organizing your code?

You will put the bulk of the code in one .sql file containing SQL and PL/PGSQL code. This includes code to initially create the schema's tables. Your business logic, i.e., CRUD operations, data integrity, and the required queries that the application supports, will be packaged in functions in this file. Your code should drop any prior definitions so that graders can simply run your code after having run another team's code.

You can load a .sql file from the command line: psql -f filename.sql, or from within psql: \i filename.sql.

Your application will support a basic user interface, supporting all the specified operations in a self-evident manner. Your user-interface will be evaluated for its ease of use, but not how fancy it is. For example, you should provide useful prompts and feedback to the user and check user input for errors. It should avoid any SQL injection attacks. However, the user-interface need not be graphical/window-based.

All your code should be in a ZIP file. Include a text file called README with instructions on how to run your code.

- (15 points) User-interface usability, feedback, input error-checking, avoiding SQL injection

Required operations

- (10 points) Read data from a user-provided filename in the client's file system, and add the data into the database. This does not delete any data from the database, unless new data conflicts with old. The file format is basically CSV, except that rows starting with an asterisk (*) have the name of a table (e.g., *Event), and the following lines contains data for that table. A sample data file will be provided.
- (5 points) Save all data to a user-provided filename. Use the same file format as for reading.
- (5 points) For each table, enter a row of data. The user should be reminded or prompted as to what data is expected in what order.
- (5 points) For the Participant, Event, and Org tables, update a row of data, given the primary key.

For queries, the basic data summary is a Heat Sheet. This displays a summary of an entire meet's information. It groups each Event, and for each Event, displays each Heat. Within each Heat, it displays each lane and the swimmer(s), school, time(s), and rank(s). For relays, times for both the individuals and groups are included. A Heat Sheet can be displayed before some or all heats happen, in which case those heats have no time or rank information.

We will only consider per-Event ranks. A swimmer or group can be in multiple heats for the same event. Their fastest time in the event is ranked, and their slower times are ignored for the ranking.

- (10 points) For a Meet, display a Heat Sheet.
- (10 points) For a Participant and Meet, display a Heat Sheet limited to just that swimmer, including any relays they are in.
- (10 points) For a School and Meet, display a Heat Sheet limited to just that School's swimmers.
- (10 points) For a School and Meet, display just the names of the competing swimmers.
- (10 points) For an Event and Meet, display all results sorted by time. Include the heat, lane, swimmer(s) name(s), and rank.
- (10 points) For a Meet, display the scores of each school, sorted by scores, and calculated as follows. Within each event, the top-ranked swimmers earn points for their school's score. In individual events, first through fifth place earn 6, 4, 3, 2, and 1 points, respectively. In relay events, first through third place earn 8, 4, and 2 points, respectively. (In reality, there are a variety of scoring systems for different levels of competition and different types of meets. This is one system in use.)