School of Computer Science, McGill University

# COMP-421 Database Systems, Winter 2016

Programming Project 3: Writing your Application

Due date: March, 17, 23:30pm

## 1 Assignment (Please turn in one solution per team)

This is the last assignment of your database project.

You have to work with XML, do some programming in the database (stored procedures) and build a "user-friendly," interactive application program front end using Java and the JDBC interface between Java code and SQL (see web-links for an example program).

Again, this deliverable is out of 100 pints, but only 90 are indicated below as specific tasks. The extra 10 points can be achieved, for instance, by building a graphical user-interface, by providing an extra stored procedure using a different language, or by looking up how triggers work and build one for your application.

1. (15 pts.) Extend your schema so that one of your relations (possibly a new one) contains at least one attribute of type XML. Fill this relation with at least 5 tuples containing all different values in the XML-typed attribute. Write two XQuery/XPath queries that select a subset of the tuples based on filtering conditions of the XML content. It should also provide an output that is differently formatted than the original XML data attributes (in case of XQuery).
   **Turn in the revised schema (only the relation added or the existing relation that was changed). Show the corresponding CREATE/ALTER TABLE statement. Provide a short description of the meaning of the new/changed relation / attribute(s). Show the insert statements (or the JDBC program + the file with the XML data used to fill the relation). Provide a short description of the content of the XML data that is inserted. Provide the two queries together with a description of what it is supposed to do, along with a script illustrating its execution. Your script should be sufficient to convince us that your commands run successfully.**
2. (15 Pts) Write one stored procedure (language of your choice) to perform operations on your project database. It should be nontrivial, illustrating a feature or features such as local variables, multiple SQL statements, loops etc. It should also involve a cursor. The stored procedure should use one or more parameters in a significant way. We encourage you to be imaginative. However, here are some sorts of things you might try if you can't think of something more interesting:
   - Compute some aggregate value from a relation and use that value to modify values in that or another relation.
   - Create a new relation and load it with values computed from one or more existing relations.
   - Enforce a constraint by searching your database for violations and fixing them in some way.

   **Hand in a listing of your programs and scripts showing them working. You should demonstrate that the programs had their intended effect by querying (before and after) some relation of your project database that was changed by the program. These queries may be included in the file that holds your programs for convenience.**

3. (50 Pts)
   Write a user-friendly application program for your database in Java. There is no need for a fancy interface. For example, a menu printed via simple I/O is ok.
   Your program should consist of a loop in which:
   - A list of at least five alternative options is offered to the user. An additional alternative should be quit.
   - The user selects an alternative.
   - The system prompts the user for appropriate input values.
   - The system accesses the database to perform the appropriate queries and/or modifications.
   - Data or an appropriate acknowledgment is returned to the user.

   Your program should follow the following guidelines.
   - Your options should include both queries and modifications.
   - Some of your options should contain more than one SQL statement.
   - Your program must handle errors appropriately. For Java, catch exceptions and print the error messages.

   For example, if your project were about skaters and competitions.
   - Look up whether a skater participates in a certain competition by skater name.
   - Enroll a skater S in a competition C. If the rating level is below 3, S cannot enroll in any competition. If it is between 3 and 6, S can enroll in regional competitions only, if its is between 7 and 9, he/she can enroll in regional and national levels, and only with a skating level of 10 can S enroll in all types of competitions. If S is not qualified for the competition C, return a list of alternative competitions for which the S has the minimum rating level and which are close to C in terms of the date.
   - A competition is cancelled: find all skaters participating and replace the participation with a competition close in time to the cancelled competition.
   - Add a new skater.
   - Increase the rating of skaters that were among the first 5 in at least 2 competitions of the highest level they can participate.
   - ...
   - Quit

   **Hand in your program and a script showing the program running. Each of the options should be exercised at least once in your script.**
4. (10 pts.) In class we discuss indexes that help to speed up queries. You can create and drop an index using: commands:

   ```
   CREATE INDEX <IndexName> ON <RelName>(<Attribute List>);
   e.g., CREATE INDEX skatersname ON Skaters(sname);
   DROP INDEX <IndexName>;
   ```

   Statements for more sophisticated indexes (unique, clustered etc.) can be found in the lecture notes and also in the DBS manuals.
   **Create at least two useful indexes for your project database. Do not build indexes on primary keys and unique keys. Database systems usually create indexes on these attributes automatically as they need them to check the uniqueness property. For each of the created indexes indicate why this index is useful by describing which application relevant queries would execute quicker**

Note: The project design was taken and adjusted from a project description of the CS145 Stanford database course.