School of Computer Science, McGill University

COMP-421B Database Systems, Winter 2015

Programming Project 3: Writing your Application

Due date: March 26

Assignment (Please turn in one solution per team)

You have to do some programming in the database (triggers and 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).

Extra points can be achieved by building a graphical user-interface or by providing an extra stored procedure or trigger using a different language.

1. (10 Pts) Write a trigger. The trigger could check integrity constraints that span over several relations, i.e., reject a modification if it violates a constraint over several relations. You might also try to write a trigger for the other constraints that you described in your E/R model but that you could not guarantee so far (e.g., cost of the purchase is the sum of the prices of the individual items plus tax etc.). Another use for triggers is to write explicit logging information in a separate table. Hand in your code and a script showing the triggers declared. Also, the script should show the effect of two database modifications. One modification should fire the trigger, and the other not. Show in the script queries that demonstrate that the trigger has an effect in the first case and not in the second.

2. (10 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 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. (55 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 acknowledgement 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 you can have options such as
   - 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 it 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 five in at least two 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. (15 Pts) In class we discussed indexes that help to speed up queries. You can create and drop an index using commands:
   CREATE INDEX <IndexName> ON <RelName>(<Attrubute List>);
   e.g., CREATE INDEX skatername 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.

Run your most complex queries from your last project on your large database with the indexes and without the indexes, and measure the time in both cases. For that, you can include the queries in a Java application program. Measure the time before you start the query, and then again after the query executed. The difference of these two values is the query execution time (plus communication between program and DBS)
To measure the current time you can use in Java
// Returns the current time in milliseconds.
Public static long currentTimeMillis()
// (e.g. long currTime = System.currentTimeMillis())
Java only gives milliseconds so you might not be able to see a big difference.
Naturally these times may be affected by external factors such as system load, etc. Still, you will hopefully see a difference between the execution times with indexes and the times without. Turn in a script showing your commands to create indexes, and showing the relative times of query execution with and without indexes.