CSC343 Fall 2007
Assignment 2
SQL and Embedded SQL

Distribution date: Friday, October 26, 2007
Due date: Monday, November 12, 2007 1:00 p.m.

Instructions

1. Read this assignment thoroughly before you proceed. Failure to follow an instruction can affect your grade.
2. The database tables you are using are described in A2.DDL which must be downloaded from the assignment webpage.
3. You must submit your assignment both in paper format in the drop-box in Bahen room BA2220, and electronically via CDF by the due date.
4. You must submit the following 5 files:
    1. **cover.txt** the submission information for your assignment. All information must be complete. 5 marks will be deducted for missing or incomplete information. The format for this file can be downloaded from the web page.
    2. **a2drop** drop statements (see Interactive SQL Query section).
    3. **a2tables** creation statements for query result tables (see Interactive SQL Query section).
    4. **a2sql** your queries for this assignment (see Interactive SQL Query section).
    5. **Assignment2.java** your JDBC .java class (see JDBC and SQL Queries section). Be careful to submit the .java file, **not** the .class file. We will not mark .class files (and you could end up with a zero for this section)
5. To submit your assignment through CDF

    submit -c csc343h -a A2 cover.txt a2drop a2sql a2tables a2sql Assignment2.java

    If you are submitting late, you must submit to A2-late as follows

    submit -c csc343h -a A2-late cover.txt a2drop a2sql a2tables a2sql Assignment2.java

    For more information on submit, use man submit.
6. Your paper copy must contain all five submission files in the order shown above. Please do not use an envelope or any special cover – stick to plain paper. Staple all of your sheets and make sure they are securely fastened together. Any pages not attached will not be marked. Late assignments do not require a printed copy (as we will print late assignments).
7. You may work in groups of up to two people for this assignment. Please make only one paper and electronic submission per group.
8. There is a 10% penalty per late day. Assignments received at 1:01 p.m. on the due date are considered late. Assignments may be submitted up to three days late – after which they will not be accepted (at all). Please see the "Lateness Policy" on the "Assignments" webpage for exceptions to this rule.

SQL Queries [45 marks]

In this section, you must create views and queries **to be run in *psql* on the CDF machine.** In order to ensure that everything is run in the correct order (by the markers), you must create three files containing your statements, which can be read into *psql* and executed using the *psql* command:
`\i <FILENAME>`.

The files must be as follows:

1. **a2drop**: statements to drop any and all views and tables that you create.
2. **a2tables:** statements to create the tables which will hold the query results (Query1 through Query7), as well as any intermediate views you may require for this assignment.
3. **a2sql**: all queries that populate the Query? tables with results that satisfy the questions below.

Express the following queries in SQL. Follow these rules:

i. The output of each query must be stored in a separated table. You must create the table definition for the tables that are used to store the results of your queries (*Query1, Query2, etc.).* These definitions should be saved in a2tables.

ii. The statement to insert your query results might look something like:
"insert into Query1 select … <put your SQL query here> …" .

iii. Your tables **must** match the output tables specified for each query. The attribute names must be identical to those specified in *italics,* and they must be in the specified order.

iv. All of your statements must run on PostgreSQL on the CDF machine, so be sure to populate your tables with test data and run all your statements on CDF prior to submission.

NOTE: Failure to do this may cause your query to fail when (automatically) tested, and you will lose marks.

1. **[5 marks]** Find the number of female students in the 'Computer Science' department who are in their fourth year of study.
   Output Table : Query1
   attributes: *num* (the number of students)

2. **[5 marks]** Find the department with the highest number of instructors who do **not** have a PhD degree. If there is a tie, return all tied departments.
   Output Table: Query2
   attributes: *dname* (the department name)

3. **[5 marks]** For each department find the best student. The student rank is determined by the average grade of all of the courses they have completed. Courses for the current semester should not be included. (Hint: The current semester occurs in the largest year/term value.) If there is a tie for first place, include all students in the tie.
   Output Table: Query3
   attributes: *dept* (the name of the department)
   sid (the student id),
   sfirstname (student first name),
   slastname (student last name),
   avgGrade (average grade for the student)

4. **[5 marks]** Find the year between 2001 and 2006 in which the "Computer Science" department had the highest course enrollment, compared to other years. Course enrollment for a year is calculated as the sum of the total number of students enrolled in each course for all courses in all semesters for that year. (Hint: Use the calendar year; do not worry about starting a year in the Fall.)
   Output Table: **Query4**
   attributes: *year* (the year with highest enrollment)
   enrollment (the course enrollment for the year)

5. **[5 marks]** Find all the courses in 'Computer Science' department that are taught **only** in summer semester. Do not report duplicates.
   Output Table: **Query5**
   attributes: *cname* (the name of the course)

6. **[10 marks]** List all the students and courses for all situations where a student has taken a course without previously having taken the prerequisites. Report a student as many times as the number of courses with no prerequisites that they have taken (but not for each prerequisite).
   Output Table: **Query6**
   attributes: *fname* (student first name),
   lname (student last name),
   cname (course name),
   year (the year)
   semester (the semester),

7. **[10 marks]** Of all the courses offered in the 'Computer Science' department which had an enrollment of at least 3 students find the course with the highest average marks and the course with the lowest average marks. Include all tied results. (Hint: Do not need to consider the current semester which does not yet have any marks.)
   Output Table: **Query7**
   attributes: *cname* (name of the course)
   semester (the semester),
   year (the year),
   avgmark (the average mark for that course)

JDBC and SQL Queries [45 marks]

For this part of the assignment, you will create the class Assignment2.java which will allow you to process queries using JDBC. We will use the standard tables provided in the A2.DDL for this assignment. If you feel you need an intermediate view to execute a query in a method, you must create it in that method. You must also drop it before exiting that method.

Rules:

i. Standard input and output must not be used. This will halt the "automarker" and you will probably end up with a zero.
ii. The database, username, and password must be passed as parameters, never "hard-coded".
iii. Be sure to close all unused statements and result sets.
iv. All return values will be String, boolean or int values.
v. A successful action (Update, Delete) is when:
   1. It doesn't throw an SQL exception, and
   2. The number of rows to be updated or deleted is correct.
vi. When rows of data are returned as a String, they must be in the following contiguous format:
   - Columns are separated with a colon ":". There is no colon after the last column of a row.
   - Rows are separated with a pound-sign "#". There is no pound-sign after the last row of the result.
   - Leading and trailing spaces are eliminated
   - e.g., a result set that includes multiple rows of the columns firstname, lastname might look like "John:Mylopoulos# Faye:Baron#Solmaz:Kolahi#Dimitris:Tsirogiannis"

Class name	**Description**
Assignment2.java	Allows several interactions with a postgreSQL database. The second assignment for CSC343 Fall 2007.

Instance Variables (you may want to add more)

Type	**Description**
`Connection`	The database connection for this session.

Methods (you may want to add helper methods for redundant code)

Constructor	**Description**
`Assignment2()`	Identifies the postgreSQL driver using Class.forName method.

Method	**Description**
`boolean connectDB(String URL, String username, String password)`	Using the String input parameters which are the URL, username, and password respectively , establish the `Connection` to be used for this session. Returns **`true`** if the connection was successful.
`boolean disconnectDB()`	Closes the connection. Returns **`true`** if the closure was successful.
`boolean insertStudent(int sid, String lastName, String firstName, int age, String sex, String dname, int yearOfStudy)`	Inserts a row into the student table. *dname* is the name of the department. You have to check if the department exists, if the sex is one of the two values ('M' or 'F') and if the year of study is a valid number (>0 && < 6). Returns **`true`** if the insertion was successful, **false** otherwise.

Method	**Description**
`int getStudentsCount(String dname)`	Returns the number of students in department *dname*. Returns -1 if an error occurs.
`String getStudentInfo(int sid)`	Returns a string with student information of student with student id *sid*. The output is "firstName:lastName:sex:age:yearOfStudy:department". Returns an empty string "" if the student does not exist.
`boolean chgDept(String dcode, String newName)`	Changes the department name to the department name supplied (*newName*). Accepts the *dcode* and new department name as `String`s (in that order). Returns **`true`** if the change was successful, **false** otherwise.
`boolean deleteDept(String)`	Deletes the department identified by the input `String` dcode. Returns **`true`** if the deletion was successful, **false** otherwise.
`String listCourses(int sid)`	Returns a string with all the courses a student with student id *sid* has taken. Each course will be in a separate line, e.g. "courseName1:department:semester:year:grade# courseName2:department:semester:year:grade# ... " Returns an empty string "" if the student does not exist.
`boolean updateGrades(int csid)`	Increases the grades of all the students who took a course in the course section identified by *csid* by 10% :) Returns **true** if the update was successful, **false** otherwise. Do not not allow marks to go over 100%.
`String query7()`	Execute query 7 (highest and lowest average department marks) described in the Interactive SQL section above. Instead of inserting the results in a table, return them as a `String` in the same format as is specified for the output table for the query. Be sure to follow the prestated String rules involving colons and pound-signs for your return `String`. Do not use the views that you created in the SQL part. If you need views create them when you execute function query7().
boolean updateDB()	Create a table containing all the female students in "Computer Science" department who are in their fourth year of study. The name of the table is femaleStudents and the attibutes are: *sid* INTEGER (student id) *fname* CHAR (20) (first name) *lname* CHAR (20) (last name) Returns **true** if the database was successfully updated, **false** otherwise.