5DV052 Advanced Data Models and Systems Spring 2011
Obligatory Exercise 1
Due date: May 03, 2011 at 0800 (8am)

For this assignment, you may work in groups of up to three persons. There is no bonus for working alone or in smaller groups.

1 Overall Purpose

The purpose of this exercise is to provide some experience in database programming, and with ODBC using C as the host language in particular. Key points include the following:

The final product **must** work with PostgreSQL, using ODBC 3.0 calls, with the program written in C and built using gcc as installed on the departmental Linux and Solaris machines.

You are of course free to develop the software on any platform, and in any environment, you wish, as long as the final product is compatible with the specifications identified above.

The user-program interface must be a console (text) interface. If you wish to deviate from this norm and implement a GUI, you must obtain the written permission of the instructor in advance.

2 General Description of Supported Tasks

The overall task is to write an interactive, menu-based interface to a database built upon the Company database schema of the textbook of Elmasri and Navathe. It is not necessary to have access to a copy of that book. An ODBC program which generates that database is available on the course Web site. It is easy to see a description of both the schema and the database by examining the content of the files `company_tables.h` and `company_tuples.h`, or by running the program and then examining the database schema which is generated using the `\d` command of PostgreSQL and the data using simple SQL queries.

The interface must be capable of managing the following tasks.

1. Change the name of a project.
2. Change the location of a project.
3. Change the department in which a project is located.

4. Add an employee to a project with null hours. If the employee already works on the project, report an error. Do not update to null hours.
5. Delete an employee from a project.
6. Change the number of hours which an employee works on a given project. The old and/or new value may be null.
7. Display a summary report for a given project, which lists, the project name, number, location, and department, as well as a list of all employees working on that project, showing SSN, all names, and number of hours. A separate binary field (yes/no), indicating whether or not an employee has a null hours field for that project, should also be part of the summary. Null hours should not be interpreted as zero, but rather as unknown.
8. Display a list of all *overtime employees*; that is, all employees who are working more than 40 hours per week (on all projects combined). The list must give the full employee name and SSN, as well as the number of hours worked. Nulls do not affect the hours total.

Identify projects and departments by name, not by number. Identify employees by SSN.

3 Details of the Database Schema

There are some “implicit” constraints in the schema which are not characterized in Figure 8.1(a) of the text, but which your program should enforce:

1. The location of a project must be one of the locations of the department which controls the project.
2. The number of hours which an employee works on a project must be positive or null. Zero or negative values are not allowed.
3. The total number of hours which an employee works on all projects combined must not exceed 80. Null values should be ignored in making this computation.

4 Nature of the User Interface

The primary goal of this project is to obtain experience in database programming, not the design of user interfaces. Therefore, the interface is to be a simple text-based menu-driven application, which prompts the user to select one of a number of choices, and then prompts for input values and/or provides output values. Part of the top level might appear as follows:

```
Enter the number associated with your choice:

0: Exit the program
1: Change the name of a project.
2: Change the location of a project.
...
```

Some selected items may have further sub-menus, of course, and most will require further interaction with the user.

The interface must have the following features:

1. It must prompt repeatedly for new queries and then execute them, until a special menu item which terminates the session is selected.
2. There should be an "abort" key, which allows one to abort the current query and back up "one level" in the menu interface. The keystroke used in Emacs for this purpose, Ctrl-G, is a good choice.) It must be possible to type this key at any point in an input sequence, and realize the action immediately. This implies that input must be processed character-at-a-time, and not by using scanf() or one of its relatives. Multiple levels of aborts would be a plus, but are not absolutely required. (The ODBC example program which shows to to handle passwords also illustrates character-at-a-time input under Linux.)
3. Within entry of data at a user prompt, it should always be possible to erase the last character which was typed using the backspace key.
4. The program must absolutely not require the user to input SQL statements. The interface is to be usable by a "naive" user who can only select menu items and type simple text values at prompts.
5. It must present tabular results in a neat and organized fashion, with headings, and it must provide reasonable clarification for other information displayed or actions taken, including those taken due to incorrect or otherwise inappropriate input.

5 Further Notes on the Implementation

1. Although concurrency directives are not part of this project, the software must nonetheless be written with an eye towards the idea that concurrency is an issue. To this end, the program must not "cache" values which it has computed on one query so that they may be used in the next. Rather, it must fetch all required values directly from the database for each new query.
2. Input values for hours should be rounded to the nearest tenth.

3. When an update cannot be executed, the reason must be stated.
4. The ODBC name for the database to connect to must be Company.
5. ODBC calls must be used when appropriate. For example, to bind variable parameters in an SQL query use the API call SQLBindParameter; do not build the query by concatenating strings containing the parameter values and the query template in C.
6. The program should not make modifications to the instances of the Employee, Department, or Dependent relations.

6 User Manual

It must be possible to learn to use the software without (a) having to read the source code, and (b) without having to experiment excessively. To this end, the project submission must include a concise user manual which explains how to use the software.

The user manual is an essential part of this project, and a substantial part of the grade will be based upon it. This documentation must be clear, complete, and well written.

7 Test data

A suite of test queries will be supplied well before the due date.

8 Items to Submit

The following items must be submitted as hardcopy, by placing them in the appropriate laboratory mailbox on the fourth floor of MIT-huset, as well as in electronic form vie e-mail to `odbc-5dv052@cs.umu.se`.

1. The user manual for your software.
2. The source code.
3. A transcript of a session which illustrates the principal features of your software package. This transcript must include, but is not limited to, the suite of test queries to be supplied.

A submission is considered to be complete only after both hardcopy and electronic versions have been submitted.

9 Final Points

- An ODBC program which generates a test company database for this exercise is available on the course Web site. Use this database for the test queries.
- A correct solution must work for all legal instances of the database, and not just for those queries in the test suite. The grader may run additional queries against your submission.
- This exercise may be done either individually, in a group of two, or in a group of three.
- This project is worth 200 points total. For each working day, or fraction of a working day, that it is submitted late, 20 points will be deducted from the grade.
- It is not allowed to copy the work of others. The submission must be the original work of the individual(s) in the working group.
- The grader reserves the right to interview members of the working group about the solution.
- All submitted materials must be in English. You will not be graded on the finer points of English grammar. However, if you are not reasonably professional in the preparation of your work (for example, if you do not bother to run your user manual through a spelling checker), then you may lose some points.