TDBC86 Database Concepts Fall 2001

Obligatory Software Exercise
Due date: December 21, 2001 at 1700

Note: This exercise may be submitted as late as January 10, 2002 at 1700 without loss of points for late submission. However, if it is not submitted on or before December 21, 2001 at 1700, then it may not be graded in time to receive a course grade within two weeks after the first examination.

After January 10, 2002, late penalties begin to accrue in any case.

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 an Access 97 database, using ODBC 3.0 calls, with the program written in C and built using Microsoft Visual C++ 6.0, as installed on the departmental machines which run Windows NT.
- 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. If you desire to use C++ instead of C, you must obtain the written permission of Jonas Westling in advance.
- 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 Jonas Westling in advance.

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 (See Figure 8.1(a)). 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.
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.

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.

Details of the Database Schema:

1. Due to limitations in the features of Microsoft Access, the Company schema cannot be implemented exactly as shown in Figure 8.1(a) the textbook. The modified schema to be used is similar to that which is embodied in the VBA and C generation programs provided on the course web page. Note that a null value for the `hours` field of the `Works_On` relation *is* allowed.

In addition, there is one major compromise in the schema declaration which must be made, due to a limitation in Microsoft Access:

2. Because Access does not support the decimal SQL type, the value for the `Hours` attribute of the `Works_On` relation is represented in the schema using tenths of an hour, or *deci-hours*. However, in your interface, in both input and output situations, the actual hours value must be used. Thus, your program must perform some simple translation of format.

Finally, 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:

3. The location of a project must be one of the locations of the department which controls the project..
4. The number of hours which an employee works on a project must be positive or null. Zero or negative values are not allowed.
5. 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.

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 (*e.g,* by using `_getch()`), and not by using `scanf()` or one of its relatives. Multiple levels of aborts would be a plus, but are not absolutely required.
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 du to incorrect or otherwise inappropriate input.

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. Values for 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.

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.

What to Turn In:

The following items must be submitted as hardcopy:

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. A suite of test queries, as well as test database, may be provided at a later date.

The following must be submitted in electronic form, to the e-mail address `proj-tdbc86`.

1. A Zip file containing the user manual, source code, and the executable object for your program. The file must be compatible with the `WinZip` program, which will be used to unpack it.

Further Notes:

- As stipulated in the course syllabus, this exercise may be done either individually, in a group of two, or in a group of three.
- Remember that there are point penalties for late submission. See the course syllabus.
- For this project, quality points will also be awarded. Thus, the number of points received for this work will depend both upon when it is submitted, and upon the quality of the work.
- 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.
- Remember that a correct solution must work for all instances of the database, and not just the Access database provided.
- Grades for solutions submitted after the due date may not be recorded in time to be included with results for the first examination.
- If you have solved this problem for a previous offering of the course, you may re-use your old solution, subject to the following conditions: (a) You may not work with any partners, except possibly those with whom you worked to prepare the solution in the previous course. (b) You must explicitly note any partners from the previous course with whom you submitted a joint solution for that course. Note that grading criteria may not be identical between years, so that a solution which was found to be satisfactory last year may not be evaluated similarly this year.

- **Finally, remember that all work for this course, including this project, 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.**