5DV119 Introduction to Database Management Spring 2017

Obligatory Software Exercise

See deadlines on web site

1 Overall Purpose

The purpose of this exercise is to provide some experience in database programming, and with ODBC using either C or Python as the host language in particular. Regardless of the host language, the final product must work with both PostgreSQL and MySQL, and and will be tested on the Linux server `salt` of the Department of Computing Science. If C is used as the host language, the `gcc` compiler must also be used, with the library `-lodbc`, and all ODBC calls must be version 3.0 or later. There must be no compiler errors or warnings. If Python is used as the host language, the standard library `pyodbc` must be used as the bridge to ODBC. Either Python2 or Python3 may be used; `pyodbc` is installed for both. One language (C, Python2, or Python3) must be used for the entire solution; it is not permitted to use different languages in different parts of the solution.

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 overall task is to write two command-line programs to perform simple tasks using the Company database schema of the course textbook.

1. Display a summary report for all employees.
2. Change the number of hours which an employee works on a project.

For each program, the command-line call must take the following form:

```
<Program_Name> <DBNname> <UserID> <pwdarg> <Other_Parameters>
```

1. `<Program_Name>` must be `sumrpt` for the first program and `chghrs` for the second.
2. `<DBName>` is the ODBC database name.
3. `<UserID>` is the User ID associated with the database.
4. `<pwdarg>` is either the password for the database or else `-p`, in which case the program should prompt for the password on the command line and *not* echo the characters which are typed. For PostgreSQL, even though a password is not used, the program must take a password as input and then ignore it.
5. `<Other_Parameters>` are the other parameters for the program, as described below.

1.1 Program to Display a Summary Report

This program is to generate a summary report of the total working hours for each employee, including those which are not listed in the `Works_On` relation. The summary must be in the form of a single relation, with the following columns, in the order given: `SSN`, `LName`, `FName`, `MInit`, `Known_Hrs`, `Unknown_Hrs`, `Overtime`.

1. The attributes `SSN`, `LName`, `FName`, `MInit`, are exactly as in the `Employee` relation.
2. The attribute `Known_Hrs` gives the sum of the hours worked on all projects by the given employee. `NULL` is counted as zero for this summation.
3. The attribute `Unknown_Hrs` is `Yes` if the employee works on at least one project with `NULL` hours, and `No` otherwise.
4. The attribute `Overtime` may take one of three values: `Yes`, `No`, and `?`. If the sum of all known (*i.e.*, non-`NULL`) hours which the employee works on all projects is greater than 40, the value is `Yes`. If this sum plus 40 times the number of projects on which the employee works for `NULL` hours is still less than or equal to 40, then the answer is `No`. Otherwise, the answer is `?`.
5. The relation must be displayed using the vertical bar `|` as a column separator, and have a single header line followed by a separator line of dashes and vertical bars, as shown in the example below. Numerical values must use one place to the right of the decimal point. (The values were chosen randomly and are for illustration only; they do not reflect a correct computation on the initial database.)

```
|    SSN    |  LName  |   FName  | MInit | Known_Hrs | Unknown_Hrs | Overtime |
|-----------|---------|----------|-------|-----------|-------------|----------|
| 123456789 | Smith   | John     |   B   |    32.3   |     Yes     |    Yes   |
| 333445555 | Wong    | Franklin |   T   |     3.0   |      No     |      ?   |
```

6. Do not put any additional lines (such as an initial or final line of dashes) in the table. (This is necessary to allow automatic testing of the solution.)
7. There are no additional command-line arguments. Thus, the list `<Other_Parameters>` is empty and a call takes the following form.

```
sumrpt <DBNname> <UserID> <pwdarg>
```

1.2 Program to Change the Number of Hours which an Employee Works on a Given Project

This program changes the number of hours which a given employee works on a single project. There are three additional command-line parameters, so a call takes the following form:

```
chghrs <DBNname> <UserID> <pwdarg> <SSN> <PNo> <Hours>
```

1. The value of `<Hours>` must be a number no smaller than 0 and no larger than 40, or else `NULL`. All other values of `<Hours>` constitute an error.
2. If `<Hours>` is zero, then the command is interpreted as deleting entirely that the employee identified by `<SSN>` works on the project whose number is `<PNo>`.
3. If `<Hours>` is a non-zero number within the range identified above, then the command is interpreted as changing the number of hours which the employee identified by `<SSN>` works on the project `<PNo>`. If the employee does not currently work on that project, the command is interpreted as an insertion of an entry that the employee now works on the given project for the specified number of hours.
4. If `Hours` is `NULL`, then then the command is interpreted as changing to `NULL` the `Hours` which the employee identified by `<SSN>` works on the project `<PNo>`, or to insert that entry in case the employee does not currently work on the given project.
5. If the operation succeeds, a message of the form `Number of hours which employee <SSN> works on project <PNo> successfully changed to <Hours>` must be written to standard error.
6. If the operation does not succeed, a clear explanation of the reason must be written to standard error.

2 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. ODBC calls must be used when appropriate. In particular, do not build the query by concatenating strings containing the parameter values and the query template.

4. The program should not make modifications to the instance of any relation except `Works_On`.

5. It is permissible (even encouraged) to re-use the code provided in the example programs. However, if substantial parts are copied verbatim, a citation should be made, in the form of a comment in the program. This applies to re-using code from other sources as well (such as programs on the Web).

6. Remember that one language (C, Python2, or Python3) must be used throughout. It is not permissible to do some parts in one language and other parts in another.

3 Test data

A suite of test queries is available on the course Web site. These queries are provided to provide a way to test the system and it is strongly recommended that all submissions be tested on this suite, and the results checked for correctness. However, the system must work for all databases which satisfy the integrity constraints and all queries and updates which meet the conditions of Sections 1.1 and 1.2. Submissions may be tested on other databases and with other test queries than those provided.

4 Submission Rules

1. All solutions must work under both PostgreSQL and MySQL, using the database servers of the Department of Computing Science. This means the server `postgres` for PostgreSQL and the server `mysql` for MySQL.

2. The programming systems to be used are the `gcc` compiler on the server `salt` or the Python interpreters `python` or `python3` on that same machine. One of these three systems must be used for the entire project. It is not permitted to use different systems for different parts.

3. Java and JDBC may **not** be used instead of ODBC.

4. All submissions are to be electronic only.

5. The electronic soution must be uploaded using the submission system found at the following URL: `https://www8.cs.umu.se/kurser/5DV119/VT16/handin/`

6. Files to be included in the solution are selected one at a time, and then the entire set of files is uploaded with one click. If it is desired to submit a new solution, all files must be selected and uploaded again, even those which have not changed from the previous submission.

7. For solutions submitted in Python2 or Python3:

   (a) Each of the two main programs must be in its own file, named `sumrpt` and `chghrs` respectively. Do not append the extension `.py` to these files.
   (b) Each of these files must have a first line which makes the file executable without having to call the Python interpreter from the command line; *i.e.*; `#!/usr/bin/python` for Python2 and `#!/usr/bin/python3` for Python3. Remember that only one of Python2 and Python3 may be used for the entire project. If a solution works for both, so much the better, but it is still necessary to pick one interperter and use it in the first line of both main programs.
   (c) Any additional files (called as libraries) must have the extension `.py` as the final part of their names.

8. For solutions submitted in C:

   (a) The source for each of the two main programs must be in its own file, named `sumrpt.c` and `chghrs.c`, respectively.
   (b) All additional files must be source files in C (*e.g.*, `foo.c`) or header files (*e.g.*, `foo.h`). No other types of additional files are permitted.
   (c) It is assumed that all additional `.c` files will be included in the compilation of both programs. If it is necessary to use a `.c` file which is only to be used in the compilation of one of the two main programs, its name, without the `.c` extension, should end with an underscore followed by the name of the main program. For example, the file `foo1_sumrpt.c` will only be used in the compilation of `sumrpt.c`, the file `foo2_chghrs.c` will only be used in the compilation of `chghrs.c`, and the file `foo3.c` will be used in the compilation of both.
   (d) The compilations will be done using commands of the following forms:
```
gcc -lodbc sumrpt.c -o sumrpt [additional .c files]
gcc -lodbc chghrs.c -o chghrs [additional .c files]
```
   Usually, the order of the additional files does not matter, but to be safe, assume that they are listed in alphabetical order.

9. This exercise may be done either individually or in a group consisting of no more than four (4) individuals.

5 Further Notes

- Remember that there are point penalties for late submission. See the course syllabus.
- 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.

- So that solutions may be discussed in a class meeting, students and/or groups that are very late in preparing a solution may be required to solve an alternate problem to receive credit for this exercise.