CS145 Programming Assignment #5

Due Sunday May 30[*]

NOTE: *For this assignment you will want to refer to* Introduction to Pro*C — Embedded SQL.[†]

1. **Final Step of Your PDA.**

   The final step of your PDA is to build a user-friendly interactive application front end to your personal database using C or C++. Your program should consist of a continuous loop in which:

   (a) A list of at least five alternative options is offered to the user. (Obviously, an additional alternative should be *quit*.)
   (b) The user selects an alternative.
   (c) The system prompts the user for appropriate input values.
   (d) The system accesses the database to perform the appropriate queries and/or modifications.
   (e) Data or an appropriate acknowledgment is returned to the user.

   You should include both queries and modifications. For example, the interface for a registrar's database might include in its menu:

   - A number of useful queries on the database, with both input and output in a format more convenient and pleasing than raw interactive SQL.
   - Option to register a new student.
   - Options to add/drop classes from a student's study list.
   - Options to update a student's name, address, or major.
   - Option to assign grades to students.

   You should code your interface using embedded SQL commands in a C or C++ program, as described in *Introduction to Pro*C — Embedded SQL*. We are not expecting anything particularly fancy in terms of the interface itself. For example, a menu printed via `printf` in C is fine. Also, handling of SQL errors can be simple. You can write a `sqlerror` routine that just prints the error message from Oracle, or you can copy the error handler from one of our sample programs.

   Please turn in your C/C++ code along with a script showing an interaction with your program in which all of its features are exhibited. You may use your small database if it is sufficient to demonstrate that you have done the required work.

   Additional Notes: At the beginning of the quarter several students asked if they could write a Web-based front end to their database, use other application programming languages (such as Java), or use

[*]Please refer to CS145 Course Information Page (`http://www.stanford.class/cs145/info.html`) for submission instructions and late policy. Note that Sunday May 30 marks the beginning of the "dead week." Normally, no assignments are due during the dead week. However, as you might recall, the original deadline for all our programming assignments was Friday. Therefore, please regard this Sunday deadline as an "extension": you should be able to complete this assignment by Friday, before the dead week begins.

[†]`http://www.stanford.edu/class/cs145/or-proc.html`

other database access protocols (such as JDBC). It is fine with us if you deviate from this assignment; however, we cannot necessarily provide programming support when you do, and the same due date and late policy hold for all students.

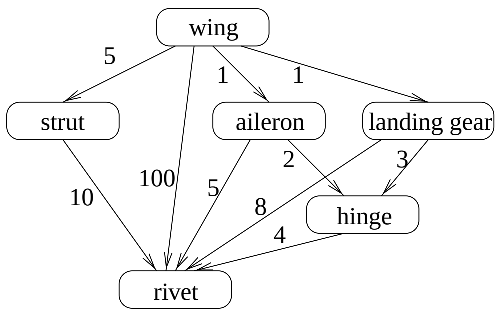
For all you Java addicts, there is a directory `/usr/pubsw/apps/oracle/8.0.4/jdbc/` which contains a `README` file, a `lib` directory with necessary Java classes, and a `samples` directory with some sample Java code. Hopefully this will be enough to get you started.

2. **Implementing Recursive Queries in Oracle.** (This part is completely optional; however, you can get a 10% bonus for this programming assignment if your solution beats ours!)

Suppose that an aircraft manufacturer maintains a table of all the parts used in a certain kind of airplane and the subparts from which each part is assembled. The table has the following schema:

`Parts(part, subpart, qty)`

Directory `/usr/class/cs145/src/PA5/` contains the files you need for this problem. File `setup.sql` creates `Parts` and populates it with sample data. A good way of looking at this data is to draw a directed acyclic graph. The nodes representing parts are connected by edges labeled with numbers, which represent how many of each kind of subpart are needed to assemble each part.



Given two parts P_1 and P_2, if P_2 is used to assemble P_1 (either directly or indirectly, or both), we are interested in the total number of P_2's required to assemble one P_1. As a concrete example, suppose we are interested in the total number of rivets used in a landing gear. This question requires us to recursively explore all components used in a landing gear, to discover how many rivets are used at each level. Note that we cannot simply add up the number of rivets used at each level—we must also take into account how many times each subassembly is used. For instance, a landing gear needs 8 rivets and 3 hinges; each hinge needs 4 rivets; so, a landing gear needs a total of $8 + 3 \times 4 = 20$ rivets.

Complete the code in `do.sql` so that it correctly computes the contents of the following table:

`PartsExplosion(part, subpart, qty)`

This table should record the total number of `subpart`'s used in a `part`, for each pair of `part` and `subpart` where the `subpart` is used directly and/or indirectly in the `part`. For the sample database, the correct answer should contain 11 rows, one of which should record the fact that a wing uses 183 rivets. An empty `PartsExplosion` table has been created for you in `setup.sql`. You may use `cleanup.sql` to clean up all tables created by `setup.sql`.

Please turn in your `do.sql` file. We will test it over our sample database.