Programming Assignment #4
Due Wednesday May 27

For this assignment you will want to refer to the section on *Transactions* in Handout #25: *Oracle 7.3.2 SQL*, and to all of Handout # 37: *Using Oracle PL/SQL.*

For all problems of this assignment you may use your small database if it is sufficient to demonstrate that you have done the required work.

1. Write two PL/SQL programs to perform operations on your PDA database. Each program should be complicated enough to involve at least one local variable, more than one SQL statement, and some data modification. In addition, at least one should involve a cursor. We encourage you to be imaginative. However, here are some things you might try if you can't think of something more interesting:

   - Compute some aggregate value from a relation and use that value to modify values in that or another relation.
   - Create a new relation and load it with values computed from one or more existing relations.
   - Enforce a constraint by searching your database for violations and fixing them in some way.

   Turn in a listing of your programs and scripts showing them working. You should demonstrate that the programs had their intended effect by querying (before and after) some relation of your PDA that was changed by the program. These queries may be included in the file that holds your PL/SQL programs for convenience.

2. Write two PL/SQL procedures or functions. They should perform some data modifications, and at least one should involve more than one SQL statement, but otherwise the procedure bodies can be simple. However, each one should use parameters in a significant way. Turn in listings of your code and scripts showing the procedures called at least once each. Also, show in your script that the procedures had their intended effect by querying (before and after) some relation of your PDA that was changed by the procedure.

3. Create at least five "interesting" triggers for your PDA. For each one, show your `create trigger` statement, its successful execution, and trigger firings in response to database modifications. You should include:

   - At least one trigger that is "`for each row`" and at least one that is not.
   - At least one scenario in which a trigger `when` clause is satisfied and at least one in which it is not.
   - At least one pair of triggers where the two triggers are identical except one is specified as `before`, the other is `after`, and they behave differently under the same scenario.

   (over)

4. This problem is similar in spirit to Problem #2 in Programming Assignment #3: You are to do some sleuth work to determine what restrictions Oracle imposes, except this time you'll be considering triggers instead of views. We are interested in having you discover restrictions on *interactions among multiple triggers* (including between triggers and themselves) and *interactions between triggers and other constraints*. We are not interested in syntactic restrictions on individual triggers, since they are presented clearly already in Handout #37: *Using Oracle PL/SQL*. As last time, your sleuth work could involve sifting through `help` pages or Oracle books, but we prefer that you do it experimentally. Regarding trigger interaction restrictions, Handout #37: *Using Oracle PL/SQL* says:

   > The restrictions on `<trigger_body>` include:
   >
   > - You cannot modify the same relation whose modification is the event triggering the trigger.
   > - You cannot modify a relation connected to the triggering relation by another constraint such as a foreign-key constraint.

   We believe the actual restrictions may be more complex. You are to create or attempt to create a series of triggers on your PDA to determine precisely what Oracle allows. You should then provide a concise characterization of those trigger sets allowed by Oracle. You should support your claims by demonstrating:

   (a) trigger sets meeting the criteria that can be created without errors;

   (b) trigger sets not meeting the criteria that generate errors when you attempt to create them.

   As above, note that your restrictions will involve *sets* of triggers rather than individual triggers, and the restrictions may involve the potential interaction of triggers with constraints such as keys and referential integrity.

5. This is a trivial problem to exercise transaction support in Oracle. Since it would be difficult for you to simulate multiple users operating on your database, you will simply experiment with the properties of transaction `commit` versus `rollback`.

   (a) Show a session in which you perform one or more data modification commands, then commit the transaction using "`commit;`". Issue queries before and after executing the transaction to demonstrate that the modification has been made on the database.

   (b) Now show a session in which you perform one or more similar data modification commands, but then abort the transaction using "`rollback;`". Issue queries before and after executing the transaction to demonstrate that the modification has *not* been made on the database.