CS145 Programming Assignment #3

Due Sunday May 16*

NOTE: *For this assignment you will want to refer to* Oracle Version 8 SQL† *and* Using Oracle PL/SQL.‡ *For Problems 2 and 3 of this assignment you may use your small database if it is sufficient to demonstrate that you have done the required work.*

1. **Constraints.** So far in your PDA the system has not enforced any constraints that hold over your relations—not even keys. For this problem you will recreate your PDA schema, adding specifications for keys, referential integrity, and other constraints.

   (a) Modify your PDA `CREATE TABLE` statements as follows.

   - For each relation in your schema, if the relation has one or more keys then modify the `CREATE TABLE` statement to declare one `PRIMARY KEY` and to declare all other keys as `UNIQUE`.
   - For each referential integrity constraint that should hold in your schema, specify the constraint using a `REFERENCES` clause within the appropriate `CREATE TABLE` statement. You may use the default option for handling referential integrity violations (violations will generate an error). We expect that everyone's PDA should include at least one referential integrity constraint. If your PDA has no natural referential integrity constraints, then it probably is either far simpler than we asked for, or a poor design—please contact one of the course staff.
   - Add at least two attribute-based and two tuple-based `CHECK` constraints to relations of your database schema. Remember that these constraints are more limited in Oracle than in the SQL2 definition; see *Oracle Version 8 SQL* for details.

   Please turn in a `.sql` file containing all your `CREATE TABLE` statements, and a script showing their successful execution in Oracle.

   (b) Reload your small hand-created PDA database. Did you get any key, referential integrity, or `CHECK` constraint violations? Please turn in the `.log` files generated by Oracle bulk loader.

   (c) Reload your large computer-generated PDA database. Did you get any key, referential integrity, or `CHECK` constraint violations? Please turn in the `.log` files generated by Oracle bulk loader.

   (d) You don't necessarily need to modify your program for generating data if it creates violations. However, for this part of the problem you should start with a database (small or large) that does not create violations. Write data modification commands to illustrate the following seven scenarios:

      1. An `INSERT` command creating a key violation
      2. An `UPDATE` command creating a key violation
      3. An `INSERT` command creating a referential integrity violation

*Please refer to CS145 Course Information Page (`http://www.stanford.class/cs145/info.html`) for submission instructions and late policy.

†`http://www.stanford.edu/class/cs145/or-nonsql2.html`

‡`http://www.stanford.edu/class/cs145/or-plsql.html`

4. A `DELETE` command creating a referential integrity violation
5. An `UPDATE` command creating a referential integrity violation
6. An `INSERT` command creating a `CHECK` constraint violation
7. An `UPDATE` command creating a `CHECK` constraint violation

Please turn in a `.sql` file containing all seven commands, and script showing their *unsuccessful* execution in Oracle.

2. **Oracle PL/SQL.** Before starting this problem, please read *Using Oracle PL/SQL*.

   (a) Write a PL/SQL program to perform operations on your PDA database. Your program should be complicated enough to involve at least a cursor, more than one SQL statement, and some data modification. 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.
   - Populate a new relation 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 `.sql` file containing your program and a script showing it working. You should demonstrate that your program had its intended effect by querying (before and after) some relation of your PDA that was changed by the program. For convenience, these queries may be included in the same `.sql` file.

   (b) Write a PL/SQL stored procedure or function for your PDA. It should involve more than one SQL statement and use parameters in a significant way, but otherwise the procedure/function body can be simple. Turn in a `.sql` file containing your `CREATE PROCEDURE/FUNCTION` statement, and a script showing its successful creation and execution. Also, show in the script the results of queries that demonstrate the procedure/function had its intended effect.

3. **Triggers.** Create at least two "interesting" triggers for your PDA. For each one, show your `CREATE TRIGGER` statement, its successful execution, and the effect of two database modifications. One modification should cause the trigger to fire, and the other not. Show in the script the results of queries which demonstrate that the trigger had an effect in the first case but not in the second.

4. **Implementing Bag Difference in Oracle.** Oracle does not support `EXCEPT ALL`, the SQL bag difference operator. For this problem, your job is to write your own PL/SQL procedure for bag difference. Copy the four `.sql` files in directory `/usr/class/cs145/src/PA3/` to your own working directory.

   - `setup.sql` will create and initialize two source tables `cs145_R` and `cs145_S`. Both tables contain duplicate tuples. `setup.sql` will also create an empty table `cs145_T`, which you should use to store the result of the bag difference between `cs145_R` and `cs145_S`.
   - `exceptall.sql` contains the skeleton of the `CREATE PROCEDURE` command for a procedure named `cs145_except_all`. You need to modify this file and fill in the PL/SQL code for `cs145_except_all` so that it will correctly compute the bag difference between `cs145_R`

and `cs145_S` and store the result in `cs145_T`. Your code should not modify any table except `cs145_T`.

- `run.sql` will run `cs145_except_all` and compute a "checksum" of the result table. If you had implemented `cs145_except_all` correctly, `cs145_T` should contain 4 tuples with a sum of 13.
- `cleanup.sql` is provided for your convenience: it will clean up everything created by the other `.sql` files.

Please turn in your modified `exceptall.sql` file. We will run it on our secret test cases. `^_^`

Additional notes and hints:

- There are many ways to implement bag difference in Oracle. Be creative. If you come up with something smarter than our sample solution, your reward will be a 10% bonus for this programming assignment!
- If you are really stuck, here are some hints: One way is to use nested cursor loops. Another method uses only one cursor, but it relies on the `ROWNUM` feature of Oracle. `ROWNUM` lets you control how many tuples to process in a SQL statement. For example, "`DELETE FROM` *R* `WHERE` *cond* `AND ROWNUM <=` *n*`;`" only deletes the first *n* tuples in *R* that satisfy *cond*.
- Please feel free to modify `setup.sql` to test your code on other instances of `cs145_R` and `cs145_S`. Also, feel free to post tricky test cases on the class newsgroup and share them with fellow CS145 hackers!