Programming Assignment #3
Due Wednesday May 20

For this assignment you will want to refer to the sections on *Constraints* and *Views* in Handout #25: *Oracle 7.3.2 SQL.*

1. 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 `referencing` 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 Handout #25: *Oracle 7.3.2 SQL.*

   (b) Reload your small hand-created PDA database. Did you get any key, referential integrity, or `check` constraint violations?

   (c) Reload your large computer-generated PDA database. Did you get any key, referential integrity, or `check` constraint violations?

   (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
   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

(over)

2. For this problem you are to do some sleuth work: Your task is to determine what criteria exactly Oracle uses in deciding whether a view is updatable, i.e., whether it is possible to perform `insert`, `delete`, and/or `update` statements on the view. While your sleuth work could involve sifting through `help` pages or Oracle books, we prefer that you do it experimentally. Write a series of views along with modification commands on the views to determine when Oracle allows views to be updated and when it does not. As discussed in class and in the textbook, some SQL views are obviously updatable, some are obviously not updatable (due to ambiguities), and some are theoretically updatable but it is difficult for a system to determine the correct update translations. In your solution to this problem you should attempt to provide a concise characterization of those views that Oracle allows to be updated, and you should support your claim by demonstrating:

   (a) views meeting the criteria that can be updated, and

   (b) views not meeting the criteria that cannot be updated.

   If separate criteria apply for `insert`, `delete`, and `update` commands then these should be included in your solution. You may use your PDA schema and data for this problem if you like, or you may use a separate, simpler database.

   The command for defining a view in Oracle is:

   `create view` *view-name* `as` (*query*)

   If you want your view to use different attribute names than those from the base relations you can write:

   `create view` *view-name* (*attr-name*$_1$, *attr-name*$_2$, ..., *attr-name*$_n$) `as` (*query*)

   You can get rid of a view using the command:

   `drop view` *view-name*

3. Create and test at least three views in your PDA that might be useful for an application. For each one, define the view and write at least two queries that use the view.