Written Assignment #6
Due Wednesday May 27

1. (**In this problem you will be graded on simplicity as well as correctness.**)
Consider one of our favorite relational schemas— `Product`, `PC`, `Laptop`, and `Printer`—given for example in Exercise 6.6.2 of the textbook on the top of page 358.

   (a) Give an example of two client transactions T_1 and T_2 operating on this database. Assume that transaction T_1 has isolation level `serializable`. (But note that true serializability is guaranteed only when *all* transactions have this isolation level.) For transaction T_2, consider isolation levels `read uncommitted` and `read committed`. Design your transactions so that there is some result allowed when T_2 uses `read uncommitted` that is not allowed when T_2 uses `read committed`. Assume that transactions T_1 and T_2 both commit. More specifically, give:

      i. The two transactions T_1 and T_2, each specified as a sequence of one or more SQL statements followed by `commit`. Your transactions should be as simple as possible; e.g., they certainly do not need to involve all of the relations in the schema.

      ii. A simple initial set of data for the relations used by your transactions (the initial state).

      iii. The state of the data after T_1 and T_2 have both executed such that this final state could result when T_2 uses `read uncommitted`, but could not result when T_2 uses `read committed`. Also specify the interleaving (ordering) of the statements between the two transactions that produced this final state.

   (b) Do part (a) again, except this time for transaction T_2 consider isolation levels `read committed` and `repeatable read`, where the final state could result when T_2 uses `read committed`, but could not result when T_2 uses `repeatable read`.

   (c) Do part (a) again, except this time for transaction T_2 consider isolation levels `repeatable read` and `serializable`, where the final state could result when T_2 uses `repeatable read`, but could not result when T_2 uses `serializable`.

2. (**In this problem you will be graded on simplicity as well as correctness.**)
Continue with the schema used in Problem #1.

   (a) Suppose multiple clients are operating on the database, and all of their transactions use isolation level **serializable**. Give a realistic example where the final state of the database depends on the order in which concurrent transactions are serialized. Please specify:

      i. An initial state for one or more relations.

      ii. Transactions issued by two clients concurrently (one transaction per client, one or more SQL statements per transaction).

      iii. Two different valid final states of the relations, depending on which transaction executed first.

   (b) Specify a simple scenario in which an application program operating on this database might decide to abort a transaction based on user input. You may specify the scenario using prose or simple pseudocode, whichever you prefer.

(over)

3. Recall from Problem #4 in Written Assignment #4 the relation `Flights(from,to,cost, airline)`, containing nonstop flights from one city to another. We were interested in part (c) in writing a SQL query that would return the cheapest cost of flying between each pair of cities, regardless of the number of stops. While it is not possible to write such a query in SQL2 (which doesn't support recursion), it is possible to compute the answer using SQL in conjunction with a programming language, and that's your task in this problem. You may use SQL embedded within a C-like language as presented in Section 7.1 of the textbook, or you may use a PL/SQL-like language as presented in Handout # 37: *Using Oracle PL/SQL.* In neither case are we particularly concerned with correct syntax, and you can certainly "wing it" on syntax for things like checking for end conditions. We're mostly interested in your overall algorithm, and the interaction between programming constructs and SQL.

4. Do Exercise 7.4.1 parts (b), (d), and (e) on page 407 of the textbook. The hardest part of this problem is finding the queries and updates referred to in the problem.

5. Do Exercise 7.4.3 on page 407 of the textbook.

6. Do Exercise 7.4.4 on page 408 of the textbook.