CS 2550 – Principles of Database Systems (Spring 2020)
Dept. of Computer Science, University of Pittsburgh

Assignment #3: Theory of Recovery & Serializability

Release: February 06, 2020 Due: 8PM, Thursday, February 13, 2020.

Goal

This is a *group* assignment. Please work on this assignment in the same group as that for HW2. The goal of this assignment is to practice the key concepts of recovery and serializability.

Description

Answer the following questions [for a total of 100 points]:

1. [15 points] Prove that if two histories are conflict equivalent then their serialization graphs are identical.
2. [40 points (10 points each subproblem)] Consider the following Histories:

$$H_1 = W_2(x)W_2(y)W_1(x)R_3(x)R_1(x)R_3(z)R_3(y)R_2(z)$$
$$H_2 = R_3(y)W_2(y)R_2(z)W_1(x)R_3(x)W_2(x)R_3(z)R_1(x)$$
$$H_3 = R_3(z)W_2(x)R_2(z)W_2(y)R_1(x)R_3(x)R_3(y)W_1(x)$$
$$H_4 = W_2(y)W_2(x)W_1(x)R_3(y)R_2(z)R_1(x)R_3(x)R_3(z)$$

   (a) Which of them are conflict equivalent?
   (b) Which of them are conflict serializable? Give the equivalent serial history.
   (c) For each history insert Commit operations to make them ACA, if possible.
   (d) For each history insert Commit operations to make them Strict, if possible.
3. [20 points (10 points each subproblem)] Consider the execution of three transactions T_1, T_2 and T_3.

```
T1:        T2 :       T3 :
read(z)    write(x)   write(x)
read(y)    read(z)    read(z)
write(x)   write(y)   read(y)
```

   (a) Give an example of histories of T_1, T_2 and T_3 that are not serial but serializable in the order of T_3, T_1, T_2.
   (b) Give another example of histories of T_1, T_2 and T_3 that are not serial but serializable in the order of T_1, T_2, T_3.
4. [15 points (3 points each subproblem)] Specify the compatibility table for the following set of database operations defined on a stack-queue (*Hint:* Identify the conditions for compatibility):
   - Pop(s): Removes and returns the top element on the stack s.
   - Push(s,X): Pushes X on the stack s and returns OK.
   - Top(s): Returns the top element on the stack s.
   - ExchangeB(s): Exchanges the two bottom elements on the stack s.
   - ExchangeT(s): Exchanges the two top elements on the stack s.

5. [10 points] *Snapshot isolation* ensures that all reads of transaction read the last committed values that existed at the time the transaction started and considers write-write conflicts. Can you prove or disprove whether snapshot isolation generates serializable executions?

What to submit

- Only one member of the team should submit the assignment. The team member whose pitt_user_name appears earlier in the alphabetical order should submit the assignment.
- You are required to submit **exactly one** PDF file under your **team_number** (e.g., team1.pdf). In addition to providing the answers, you are expected to: **include your team number, the names and pitt user names of the team members at the top of the PDF file**.
- Submit your assignments through the Web-based submission interface (at the class web page `http://db.cs.pitt.edu/courses/cs2550/current.term/`). **It is your responsibility to make sure the assignment was properly submitted.**
- Submit your assignment by the due date (**8PM, Thursday, February 13, 2020**). There is **no late submission**.

Academic Honesty

The work in this assignment is to be done *independently* by each group. Discussions with other students not in your group on the assignment should be limited to understanding the statement of the problem. Cheating in any way, including giving your work to someone else will result in an F for the course and a report to the appropriate University authority.