CS5530 Written Assignment #5

Due: April 18, Wed, 11:30am.

Problem 1. [40 pts]

- Database
  - Employee-Table(ssn, name, salary)
    - E1(132, Smith, 20K)
    - E2(456, Kelley, 40K)
    - E3(678, Johnson, 400K)
    - E4(792, Preeston, 40K)
    - E5(865, Johnson, 60K) ...
  - DPT-table(dnumber, dname, budget)
    - D1(1, Marketing, 1M)
    - D2(2, Engineering, 2M)
    - D3(3, R&D , 4M)
    - D4(4, HR , 1M) ...

Part 1. Consider the following schedules

1) T1.R(E1), T1.W(E1), T2.R(E2), T2.W(E2), T1.R(D1), T1.commit, T2.commit
2) T1.R(E1), T1.W(E1), T2.R(E2), T2.W(E2), T2.W(E1), T1.R(E1), T1.commit, T2.commit
3) T1.R(E where salary>40K and salary<100k), T2.Insert(into E, (999, Bob, 50k)), T2.commit, T1.R(E where salary>40k and salary<100k), T1.commit
4) T1.R(E where salary>40K and salary<100k), T2.Insert(into E, (999, Bob, 50k)), T1.R(E where salary>40k and salary<100k), T2.commit, T1.commit.

For each schedule, please explain if it is valid in 1) the serializable isolation level; 2) the repeatable read isolation level.

Part 2. Consider the following scenario:

T_1	T_2
Begin Transaction	
	Begin Transaction
T1.Read(E1)	
T1.Update(set E1.salary=E1.salary*1.10)	
T1.Read(Select * from Employee)	
	T2.Update(set D4.budget=2M)
	T2.Delete(E5)
T1.insert(Into E, (999, Bob, 50k))	
T1.Read(Select * from Employee)	
T1.commit	
System Crash	

Figure 1: Sequence of events

Show the content of the table Employee after the system has recovered from the system crash.

Problem 2. [60 pts]

1. In the following schedules, $R_i(A)$ stands for a Read(A) operation by transaction i and $W_i(A)$ stands for a Write(A) operation by transaction i. For each of the following schedules show if it is conflict-serializable and give a conflict-equivalent serial schedule if it is one. Hint: use its dependency graph.

   (a) $R_1(A), R_2(B), W_3(A), R_2(A), R_1(B), T1.Commit, T2.Commit, T3.Commit$

   (b) $R_1(A), R_2(B), W_1(A), R_3(C), W_2(B), W_3(C), R_4(D),$
   $R_4(A), W_4(D), T1.Commit, T2.Commit, T3.Commit, T4.Commit$

   (c) $R_3(E), R_1(D), W_2(C), W_3(A), R_1(E), W_4(B), R_1(B), W_3(E),$
   $R_4(A), W_4(C), T1.Commit, T2.Commit, T3.Commit, T4.Commit$

2. For the following 2 schedules, show if each is allowed in strict 2PL, and if not, what happens.

   An example is given: (the schedule will not be allowed in strict 2PL as it is).

   $S_1(A), R_1(A), X_2(A), W_2(A), X_1(B), R_1(B), W_1(B), T1.Commit, X_2(B), W_2(B), T2.Commit$

T_1	T_2
S(A)	
R(A)	
	X(A), Blocked
X(B)	
R(B)	
W(B)	
Commit	
Release S(A)	
	W(A)
	X(B), Blocked
Release X(B)	
	W(B)
	Commit+Release its locks

OR

T_1	T_2
S(A)	
R(A)	
	X(A), Blocked
X(B)	
R(B)	
W(B)	
Commit	
Release S(A)	
Release X(B)	
	W(A)
	X(B)
	W(B)
	Commit+Release its locks

   (a) $S_1(A), R_1(A), S_2(B), R_2(B), S_3(C), R_3(C), X_3(D), W_3(D), T_3.Commit,$
   $X_2(C), W_2(C), T_2.Commit, X_1(B), W_1(B), T_1.Commit$

   (b) $X_1(A), R_1(A), X_2(B), R_2(B), X_3(C), R_3(C), S_1(B), R_1(B), S_2(C), R_2(C),$
   $S_3(A), R_3(A), W_1(A), T_1.Commit, W_2(B), T_2.Commit, W_3(C), T_3.Commit$

3. For the following schedules, show what happens in each case.

   An example of non-strict 2PL is given (either is fine; in fact, there could be even more. But showing one is enough):

   $S_1(A), R_1(A), X_1(B), X_2(A), W_2(A), R_1(B), W_1(B), T1.Commit, X_2(B), W_2(B), T2.Commit$

T_1	T_2
S(A)	
R(A)	
X(B)	
	X(A), Blocked
Release S(A)	
	W(A)
R(B)	
W(B)	
Release X(B)	
Commit	
	X(B)
	W(B)
	Release X(A), Release X(B)
	Commit

OR

T_1	T_2
S(A)	
R(A)	
X(B)	
Release S(A)	
	X(A)
	W(A)
R(B)	
W(B)	
Release X(B)	
Commit	
	X(B)
	Release X(A)
	W(B)
	Release X(B)
	Commit

   (a) $X_1(B), W_1(B), X_2(A), W_2(A), S_2(B), R_2(B), S_1(A), R_1(A), T_1.Commit, T_2.Commit.$
   Using **strict 2PL**

   (b) $X_1(B), W_1(B), X_2(A), W_2(A), S_2(B), R_2(B), S_1(A), R_1(A), T_1.Commit, T_2.Commit.$
   Using **non-strict 2PL**

   (c) $X_1(B), W_1(B), S_1(A), S_2(B), R_2(B), R_1(A), X_2(A), W_2(A), T_1.Commit, T_2.Commit.$
   Using **non-strict 2PL**