Problem 1: Serializability

Consider the following schedules. The actions are listed in the order they are scheduled, and prefixed with the transaction name.

S1: T1:R(X), T2:R(X), T1:W(Y), T2:W(Y), T1:R(Y), T2:R(Y)

S2: T3:W(X), T1:R(X), T1:W(Y), T2:R(Z), T2:W(Z), T3:R(Z)

For each of the schedules, answer the following questions:

- Is the schedule conflict-serializable? If so, what are all the conflict equivalent serial schedules?

- Is the schedule view-serializable? If so, what are all the view equivalent serial schedules?

Answers:

- S1 is not conflict-serializable since the dependency graph has a cycle. S2 is conflict-serializable as the dependency graph is acylic. The order T2-T3-T1 is the only equivalent serial order.

- S1 is not view serializable. S2 is trivially view-serializable as it is conflict serializable. The only serial order allowed is T2-T3-T1.

Problem 2: Two-Phase Locking

Consider the following schedules:

S1: T1:R(X), T2:W(Y), T2:R(X), T1:W(Y), T1:Commit, T2:Commit

S2: T1:R(X), T2:R(Y), T1:W(Z), T1:Commit, T3:R(Y), T3:R(Z), T2:W(Y), T3:W(X), T2:Commit, T3:Commit

For each schedule, state which of the following concurrency control protocols allows it, that is, allows the actions to occur in exactly the order shown: 2PL, Strict 2PL. Please provide a brief explanation for your answer...If YES, show where the lock requests could have happened; If NO, explain briefly

Answers:

- S1 will be permitted by 2PL, but not by Strict 2PL. This is because when T1 tries to request an exclusive lock to write Y, T2 has not committed yet, and therefore has not released its exclusive lock on Y. It will be permitted by 2PL because then T2 can release the lock on Y immediately after it acquires a shared lock for X (prior to reading X). Then T1 will be able to acquire its exclusive lock on Y.

- S2 will be permitted by 2Pl, but not by strict 2PL. It will not be permitted by strict 2PL because when T2 tries acquire an x-lock prior to writing Y, it will not be able to get this lock since T3 has a shared lock on Y. Since T2 makes the request before T3 commits, T2 will not be able to get the lock. However, in 2PL, after reading Z, T3 can immediately acquire an x-lock for x and release its lock for Y. So the given schedule will follow 2PL.

Question 3: Deadlock Management

Consider the following sequence of actions, listed in the order it is submitted to the DBMS (S is a shared lock, X is an exclusive lock):

S1: T1:S(A), T2:X(A), T3:X(B), T1:X(B), T3:S(A)

S1: T1:X(A), T2:S(C), T1:S(B), T2:S(B), T3:X(B), T2:X(A)

For both the sequences S1 and S2 given above: Show the waits-for graph and indicate whether there will be a deadlock or not at the end of each sequence.

Answers:

- Yes, there's a deadlock. T2->T1; T1->T3; T3->T2.

- No deadlock. T3->T2. T2->T1; T3->T1.

NOTE: If this question appears in the exam, please draw the waits-for graph.