**Database Systems & Web**

**Tutorial 13**

Q1. Consider histories H1 and H2 given below:

**H1 = r1(x); r2(z); r1(z); r3(x); r3(y);w1(x);w3(y); r2(y);w2(z);w2(y)**

**H2 = r1(x); r2(z); r3(x); r1(z); r2(y); r3(y);w1(x);w2(z);w3(y);w2(y)**

These histories are generated by the following transactions:

**T1 = r1(x); r1(z);w1(x)**

**T2 = r2(z); r2(y);w2(z);w2(y)**

**T3 = r3(x); r3(y);w3(y)**

(a) Draw the serialization graph for H1 and state whether or not it is serializable. If it is serializable, give the equivalent serial history.

(b) State whether H2 is or not serializable. If it is serializable, give the equivalent serial history.

Q2. For each of the following transaction schedules, draw the precedence (conflict) graph and decide if the schedule is conflict--serializable. If the schedule is conflict

serializable, , give an equivalent serial schedule. If the schedule is not conflict-serializable, explain why not.



Q3. Consider the following two transactions and schedule (time goes from top to bottom).



1. Is this schedule conflict-serializable? Explain why or why not.
2. Show how 2PL can ensure a conflict-serializable schedule for the same transactions above.
3. Show how the use of locks without 2PL can lead to a schedule that is NOT conflict serializable.

Q4. For each one of the following schedules decide whether they can be produced by a Two Phase Lock (2PL) scheduler





Q1. Consider schedules S1, S2 & S3 given below:

**S1: r1(x); r2(z); r1(z); r3(x); r3(y); w1(x); c1; w3(y); c3; r2(y); w2(z); w2(y);c2**

**S2: r1(x); r2(z); r1(z); r3(x); r3(y); w1(x); w3(y); r2(y); w2(z); w2(y); c1; c2; c3;**

**S3: r1(x); r2(z); r3(x); r1(z); r2(y); r3(y); w1(x); w2(z); w3(y); w2(y); c3; c2;**

Determine whether each schedule is strict, cascadeless, recoverable, or nonrecoverable.

Q2. State whether H1 and H2 are view serializable.

H1: R2(A); R1(A); W1(C); R3(C); W1(B); R4(B); W3(A); R4(C); W2(D); R2(B); W4(A); W4(B)

H2: W3(Z), R2(X), W2(Y), R1(Z), W3(Y), W1(Y)

Q3. For the lock requests in Tables below, determine which lock will be granted or blocked by the lock manager.

Does there exist a deadlock in the lock requests in Tables, explain why or why not.

To prevent deadlock, we use a lock manager that adopts the Wait-Die policy. We assume that in terms of priority:T1> T2> T3.

Determine which lock request will be granted, blocked or aborted.





Q4. **Recovery**

Assume a system having a system log with immediate updates has the following log entries, ending with a system crash:

<start\_transaction, R>

<write\_item, R, X, 1, 5>

<write\_item, R, Y, -1, 0>

<commit, R>

<start\_transaction, S>

<write\_item, S, Z, 8, 12>

<checkpoint record>

<write\_item, S, X, 5, 10>

<start\_transaction, T>

<write\_item, T, Y, 0, 15>

<commit, S>

--- System Crash ---

i. a. Which transactions, if any, need to be redone?

b. Which transactions, if any, need to be undone?

c. Which transactions, if any, are not affected by the crash ?

ii. Now assume the system uses deferred updates.

a. Rewrite the log entries for the transactions in (i) for this logging method.

b. Which transactions, if any, need to redone after the failure ?

c. Which transactions, if any, need to be undone?

d. Which transaction, if any, are not affected by the crash?