INM370 – Advanced Databases
Tutorial 4 - Relational Database Management Issues
Model Answers

1)

a) Transaction T2 is executed with isolation level SERIALIZABLE

In this case, the transaction execution must be equivalent to a Serial schedule. Two such schedules exist, given that we have 2 transactions. Thus, either T1 executed completely before T2, or vice versa:

B1, O1, O2, C1, B2, O3, O4, C2 – (a1, a2) = (40, 40)
B2, O3, O4, C2, B1, O1, O2, C1 – (a1, a2) = (25, 25)

b) Transaction T2 is executed with isolation level REPEATABLE READ

In this case, firstly it is not possible to allow values to be accessed before being committed and secondly there is a guarantee that the data read cannot change for the duration of the transaction (i.e. the read is repeatable). Therefore, the results for (a1, a2) need to be equivalent to the situation in which T1 is completely executed before T2, or T2 is completely executed before T1 (whole transactions need to be executed).
It is interesting to note that phantom reads due to the INSERT of T1 cannot ensue, because T1 cannot commit in between the executions of O3 and O4 due to the UPDATE of T1 clashing with the reads of T2.

B1, O1, O2, C1, B2, O3, O4, C2 – (a1, a2) = (40, 40)
B2, O3, O4, C2, B1, O1, O2, C1 – (a1, a2) = (25, 25)

c) Transaction T2 is executed with isolation level READ COMMITTED

In this case, a transaction has access to committed data and each operation in a transaction only sees data that was committed before the operation started. There are three possibilities for the execution of the operations in T1 and T2, as follows.

B1, O1, O2, C1, B2, O3, O4, C2 – (a1, a2) = (40, 40)
B2, O3, B1, O1, O2, C1, O4, C2 – (a1, a2) = (25, 40)
B2, O3, O4, C2, B1, O1, O2, C1 – (a1, a2) = (25, 25)

d) Transaction T2 is executed with isolation level READ UNCOMMITTED

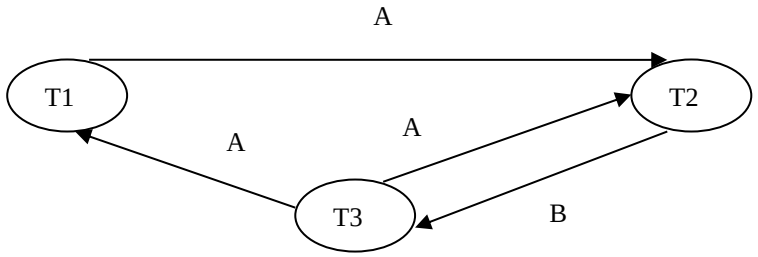
In this case, it is possible to have access to uncommitted data. The possibilities are as follows:

B1, O1, O2, C1, B2, O3, O4, C2 – (a1, a2) = (40, 40)
B1, O1, B2, O3, O2, C1, O4, C2 – (a1, a2) = (30, 40)
B1, O1, B2, O3, O4, C2, O2, C1 – (a1, a2) = (30, 30)
B2, O3, O4, C2, B1, O1, O2, C1 – (a1, a2) = (25, 25)
B2, O3, B1, O1, O4, C2, O2, C1 – (a1, a2) = (25, 30)
B2, O3, B1, O1, O2, C1, O4, C2 – (a1, a2) = (25, 40)

2)

a) r1(A); r2(A); r3(B); r1(C); w1(A); r2(B); w2(B); w3(A);

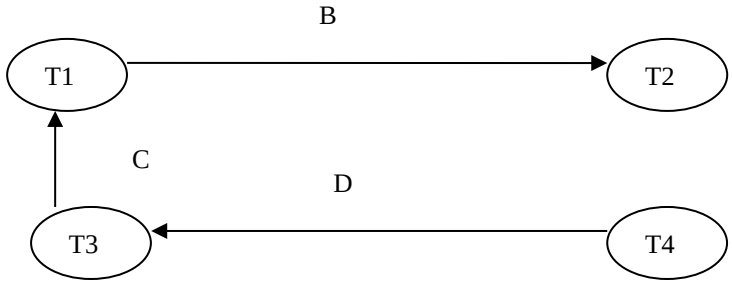
T1 holds shared lock on A
T2 holds shared lock on A
T3 holds shared lock on B
T1 holds shared lock on C
T1 denied exclusive lock on A
T2 holds shared lock on B
T2 denied exclusive lock on B
T3 denied exclusive lock on A (NB: both T1 and T2 have a conflicting lock)



Deadlock!!!!

b) r2(A); w2(B); r1(C); r1(B); r3(D); w3(C); w4(D)

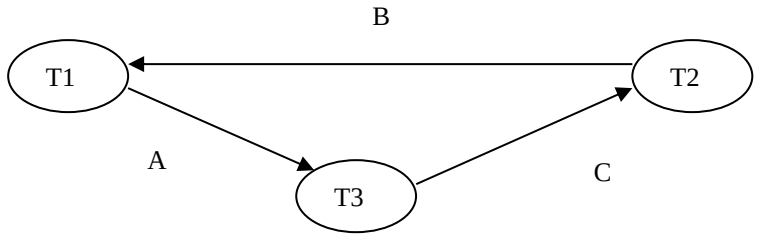
T2 holds shared lock on A
T2 holds exclusive lock on B
T1 holds shared lock on C
T1 denied shared lock on B
T3 holds shared lock on D
T3 denied exclusive lock on C
T4 denied exclusive lock on D



Deadlock free

c) w3(A); w1(B); r1(A); w2(C); r2(B); r3(C)

T3 holds exclusive lock on A
T1 holds exclusive lock on B
T1 denied shared lock on A
T2 holds exclusive lock on C
T2 denied shared lock on B
T3 denied shared lock on C



Deadlock!!!!

3)

a) This schedule is not possible under 2PL protocol, since transaction T1 will have to release its (exclusive) lock on A to allow T2 to have a (shared) lock on A. However, if this happens, transaction T1 will not be able to acquire a shared lock on A (to read the data item), since it will already be in the "shrinking phase" of rigorous 2PL approach.

b) Yes, since:
   a. No reads attempt to read an item that has been already updated by a later/younger transaction.
   b. No writes attempt to either write an item that has been already read by a younger transaction, or write an item that has been already updated by a younger transaction.

   Or equivalently, in more detail:
   - T1 writes the item A initially;
   - T2 reads the item A that has been updated by an older transaction – T1;
   - T2 writes an item B, which is written by T2 only (and no other transaction reads it).
   - T1 reads A, which has been last written by itself, and ONLY read by the younger transaction – T2.

4)

```
CREATE VIEW EmployeeView AS
  SELECT *
  FROM Employee E1
  WHERE Salary < 50000
  AND (SELECT COUNT (*) AS EmployeeCount
      FROM Employee E2
      WHERE E2.Department = E1.Department) < 10;

GRANT SELECT ON EmployeeView TO John;
```

5)

```
SELECT (A)
```

Note: This privilege has been obtained by Eve through Carol (and David). The other privilege(s) that had been originally granted has/have been revoked, since Amy issued

REVOKE to Bob's privileges, and this REVOKE cascaded to other users, including Eve (see CASCADE in Amy's REVOKE statement).
In more detail: Amy revoked `SELECT, DELETE` from Bob. This cascaded to the following privileges being revoked from David: `SELECT (A,B), DELETE.` This in turn resulted in the following privileges being revoked from Eve: `SELECT (A), DELETE`
But, David and thus Eve have been granted `SELECT (A)` privilege via Carol. Amy did not revoke privileges from Carol. Thus Eve still has the following privilege: `SELECT (A).`