CS 5300 module6

Name Student ID

Problem #1 (10 Points)

a) Consider the three transactions T_1, T_2, and T_3, and the schedules S_1 and S_2.

T_1: $r_1(X); r_1(Z); w_1(X);$

T_2: $r_2(Y); r_2(Z); w_2(Y);$

T_3: $w_3(X); r_3(Y); w_3(Z);$

S_1: $w_3(X); r_1(X); r_3(Y); r_2(Y); w_3(Z); r_2(Z); r_1(Z); w_2(Y); w_1(X);$

S_2: $w_3(X); r_3(Y); w_3(Z); r_2(Y); r_2(Z); w_2(Y); r_1(X); r_1(Z); w_1(X);$

Fill out the **best response** and **justify** your answer:

1) S_2 is a _____serial_____ schedule (why)?

Actions of different transactions are not interleaved

2) S_1 and S_2 are _____ equivalent _____ schedules (why)?

1) Every read operation reads from the same write operation in both schedule, and
2) Both schedules have the same final writes (results).

3) S_1 is a _____ serializable _____ schedule (why)?

Because it is equivalent to S_2

or

Its graph does not have a cycle (it is acyclic).

b) Improper scheduling of transactions may results in unpleasant course of events such as, “Dirty read”, “unrepeatable read”, … For the following transactions and schedules, name the nature of the anomalies (justify your answer):

```
T1:                     T2
Read (x);               Read (x);
x  := x – N;            x  := x + M;
Write (x);              Write (x);
Read (y);
y  := y + N;
Write (y);
```

a) Schedule1:

```
T1                      T2
Read (x);
x  := x – N;
                        Read (x);
                        x  := x + M;
Write (x);
Read (y);
                        Write (x);
y  := y + N;
Write (y);
```

Lost update; X has an incorrect value since its update by T1 is lost. Assume initially X is 80 and N is 5 and M is 4,

b) Schedule2:

```
T1                      T2
Read (x);
x  := x – N;
Write (x);
                        Read (x);
                        x  := x + M;
                        Write (x);
Read(y);
•
•
•
```

This is a dirty read. Assume T1 fails after Read(y), so its effect has to be undone, however, T2 has already read the temporary incorrect value of X,

Problem #2 (8 Points)

1) Define the term transaction within the scope of databases.

A query does not change the data in base sources (e.g., relations), however, a transaction my do so.
It is a set of operations that transfer databases from one consistence state to another consistent state.

2) Relative to query processing, name issues that complicate transaction processing.

Dependence between transactions, and system failure during the execution of a request.

3) ACID property stands for:

Atomicity, consistency, isolation, and durability

4) Define the term concurrency control:

The concurrency control is a protocol to generate a serializable schedule for execution of a sequence of transactions

Problem #3 (10 Points)

a) Define the following:

a. A schedule is **Recoverable** if:

A recoverable schedule is a schedule that for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, T_i commits before T_j commits.

Ti writes x, Tj reads x and Ti commits before Tj.

b. A schedule is **Cascadeless** if:

A schedule is a cascadeless schedule where each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before read operation of T_j.

Ti writes x and Ti commits, before Tj reads x.

c. A schedule is **Strict** if:

A schedule is a strict schedule in which transactions cannot read or write an item X until the last transaction that wrote X is committed (or aborted).

b) Consider the following schedules. Determine whether each schedule is recoverable, cascadeless, or strict (**make sure to justify and support your answers**).

S_1: $r_1(x); r_2(z); r_1(z); r_3(x); r_3(y); w_1(x); c_1; w_3(y); c_3; r_2(y); w_2(z); w_2(y); c_2;$

S_2: $r_1(x); r_2(z); r_1(z); r_3(x); r_3(y); w_1(x); w_3(y); r_2(y); w_2(z); w_2(y); c_1; c_2; c_3;$

S_3: $r_1(x); r_2(z); r_3(x); r_1(z); r_2(y); r_3(y); w_1(x); c_1; w_2(z); w_3(y); w_2(y); c_3; c_2;$

Recoverable schedule: A schedule is recoverable if the following condition is satisfied:

Tj commits after Ti if Tj has read any data item written by Ti.

- If A1>C3>C2, then S1 is recoverable because rolling back of T1 does not affect T2 and T3. If C1>A3>C2. S1 is not recoverable because T2 read the value of Y (r2(Y)) after T3 wrote X (w3(Y)) and T2 committed but T3 rolled back. Thus, T2 used non- existent value of Y. If C1>C3>A3, then S1 is recoverable because roll back of T2 does not affect T1 and T3. Strictest condition of S3 is C3>C2.
- If A1>C2>C3, then S2 is recoverable because roll back of T1 does not affect T2 and T3. If C1>A2>C3, then S2 is recoverable because the roll back of T2 will restore the value of Y that was read and written to by T3 (w3(Y)). It will not affect T1. If C1>C2>A3, then S2 is not recoverable because T3 will restore the value of Y which was not read by T2. Strictest condition of S4 is C3>C2, but it is not satisfied by S2.
- If A1>C3>C2, then S3 is recoverable because neither T2 nor T3 writes to X, which is written by T1. If C1>A3>C2, then S3 is not recoverable because T3 will restore the value of Y, which was not read by T2. Thus, T2 committed with a non-existent value of Y. If C1>C3>A2, then S3 is recoverable because it will restore the value of Y to the value, which was read by T3. Thus, T3 committed with the right value of Y. Strictest condition of S3 is C3>C2, but it is not satisfied by S3.
- **Cascadeless schedule : A schedule is cascadeless if the following condition is satisfied:**

  Tj reads X only after Ti has written to X and terminated (aborted or committed).
- Schedule S1 is not cascadeless because T3 reads X (r3(X)) before T1 commits.

- Schedule S2 is not cascadeless because T3 reads X (r3(X)) before T1 commits.
- Schedule S3 is not cascadeless because T3 reads X (r3(X)) before T1 commits or T2 reads Y (r2(Y)) before T3 commits.
- **Strict schedule: A schedule is strict if it satisfies the following conditions:**

  1. Tj reads a data item X after Ti has written to X and Ti is terminated (aborted or committed)

  2. Tj writes a data item X after Ti has written to X and Ti is terminated (aborted or committed)
- Schedule S1 is not strict because T3 reads X (r3(X)) before T1 has written to X (w1(X)) but T3 commits after T1. In a strict schedule T3 must read X after C1.
- Schedule S2 is not strict because T3 reads X (r3(X)) before T1 has written to X (w1(X)) but T3 commits after T1. In a strict schedule T3 must read X after C1.
- Schedule S3 is not strict because T3 reads X (r3(X)) before T1 has written to X (w1(X)) but T3 commits after T1. In a strict schedule T3 must read X after C1.

Problem #4 (7 Points)

a) Lock-based protocol has been proposed as a mechanism to allow concurrent execution of transactions, what is it (**detailed discussion, make sure to address its shortcomings**).

to ensure serializability is to require data items to be accessed in a mutual exclusive fashion ─ while a transaction is accessing the data item, no other transaction can access that data item, i.e., being Locked.

To access a data item, transaction T_i must first request for a lock on that data item. If the data item is already locked by another transaction in an incompatible mode, the concurrency control manager will not grant the lock until all incompatible locks held by other transactions are released.

b) Define the term two-phase lock-based protocol, and compare and contrast it with lock-based protocol.

This protocol ensures serializability, however, it requires that each transaction issue lock and unlock requests in two phases:

- Growing Phase: A transaction may obtain locks, but may not release any lock.
- Shrinking phase: A transaction may release locks, but may not obtain any new locks.

Initially, a transaction is in the growing phase. It acquires locks as needed. Once it releases a lock, it enters the shrinking phase, and can issue no more lock requests.