

Sample Solution – Assignment 6
CIS 451/551, Fall 2007

PROBLEM 1 (24 points)

Exercise 17.2 Consider the following classes of schedules: serializable, conflict-serializable, view-serializable, recoverable, avoids-cascading-aborts, and strict. For each of the following schedules, state which of the preceding classes it belongs to. If you cannot decide whether a schedule belongs in a certain class based on the listed actions, explain briefly. The actions are listed in the order they are scheduled and prefixed with the transaction name. If a commit or abort is not shown, the schedule is incomplete; assume that abort or commit must follow all the listed actions.

1. T1:R(X), T2:R(X), T1:W(X), T2:W(X)
2. T1:W(X), T2:R(Y), T1:R(Y), T2:R(X)
3. T1:R(X), T2:R(Y), T3:W(X), T2:R(X), T1:R(Y)
4. T1:R(X), T1:R(Y), T1:W(X), T2:R(Y), T3:W(Y), T1:W(X), T2:R(Y)
5. T1:R(X), T2:W(X), T1:W(X), T2:Abort, T1:Commit
6. T1:R(X), T2:W(X), T1:W(X), T2:Commit, T1:Commit
7. T1:W(X), T2:R(X), T1:W(X), T2:Abort, T1:Commit
8. T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Commit
9. T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Abort
10. T2: R(X), T3:W(X), T3:Commit, T1:W(Y), T1:Commit, T2:R(Y), T2:W(Z), T2:Commit
11. T1:R(X), T2:W(X), T2:Commit, T1:W(X), T1:Commit, T3:R(X), T3:Commit
12. T1:R(X), T2:W(X), T1:W(X), T3:R(X), T1:Commit, T2:Commit, T3:Commit

Answer 17.2 For simplicity, we assume the listed transactions are the only ones active currently in the database and if a commit or abort is not shown for a transaction, we'll assume a commit will follow all the listed actions.

1. Not serializable, not conflict-serializable, not view-serializable; It is recoverable and avoid cascading aborts; not strict.
2. It is serializable, conflict-serializable, and view-serializable; It does NOT avoid cascading aborts, is not strict; We can not decide whether it's recoverable or not, since the abort/commit sequence of these two transactions are not specified.
3. It is the same with number 2 above.
4. It is NOT serializable, NOT conflict-serializable, NOT view-serializable; It is NOT avoid cascading aborts, not strict. We can not decide whether it's recoverable or not, since the abort/commit sequence of these transactions are not specified.
5. It is serializable, conflict-serializable, and view-serializable; It is recoverable and avoid cascading aborts; It is not strict.
6. It is serializable and view-serializable, not conflict-serializable; It is recoverable and avoid cascading aborts; It is not strict.
7. It is not serializable, not view-serializable, not conflict-serializable; It is not recoverable, therefore not avoid cascading aborts, not strict.
8. It is not serializable, not view-serializable, not conflict-serializable; It is not recoverable, therefore not avoid cascading aborts, not strict.
9. It is serializable, view-serializable, and conflict-serializable; It is not recoverable, therefore not avoid cascading aborts, not strict.
10. It belongs to all above classes.
11. (assume the 2nd T2:Commit is instead T1:Commit). It is serializable and view-serializable, not conflict-serializable; It is recoverable, avoid cascading aborts and strict.
12. It is serializable and view-serializable, not conflict-serializable; It is recoverable, but not avoid cascading aborts, not strict.

PROBLEM 2 (18 points)

Exercise 17.4 Consider the following sequences of actions, listed in the order they are submitted to the DBMS:

- ◆ Sequence S1: T1:R(X), T2:W(X), T2:W(Y), T3:W(Y), T1:W(Y), T1:Commit, T2:Commit, T3:Commit
- ◆ Sequence S2: T1:R(X), T2:W(Y), T2:W(X), T3:W(Y), T1:W(Y), T1:Commit, T2:Commit, T3:Commit

For each sequence and for each of the following concurrency control mechanisms, describe how the concurrency control mechanism handles the sequence. Assume that the timestamp of transaction T_i is i . For lock-based concurrency control mechanisms, add lock and unlock requests to the previous sequence of actions as per the locking protocol. The DBMS processes actions in the order shown. If a transaction is blocked, assume that all its actions are queued until it is resumed; the DBMS continues with the next action (according to the listed sequence) of an unblocked transaction.

1. Strict 2PL with timestamps used for deadlock prevention.
2. Strict 2PL with deadlock detection. (Show the waits-for graph in case of deadlock.)
3. Conservative (and Strict, i.e., with locks held until end-of-transaction) 2PL.

Answer 17.4 The answer to each question is given below.

1. Assume we use Wait-Die policy.

Sequence S1: T1 acquires shared-lock on X;

When T2 asks for an exclusive lock on X, since T2 has a lower priority, it will be aborted;

T3 now gets exclusive-lock on Y;

When T1 also asks for an exclusive-lock on Y which is still held by T3, since T1 has higher priority, T1 will be blocked waiting;

T3 now finishes write, commits and releases all the lock;

T1 wakes up, acquires the lock, proceeds and finishes;

T2 now can be restarted successfully.

Sequence S2: The sequence and consequence are the same with Sequence S1, except T2 was able to advance a little more before it gets aborted.

2. In deadlock detection, transactions are allowed to wait, they are not aborted until a deadlock has been detected. (Compared to prevention schema, some transactions may have been aborted prematurely.)

Sequence S1: T1 gets a shared-lock on X;

T2 blocks waiting for an exclusive-lock on X;

T3 gets an exclusive-lock on Y;

T1 blocks waiting for an exclusive-lock on Y;

T3 finishes, commits and releases locks;

T1 wakes up, gets an exclusive-lock on Y, finishes up and releases lock on X and Y;

T2 now gets both an exclusive-lock on X and Y, and proceeds to finish.

No deadlock.

Sequence S2: There is a deadlock. T1 waits for T2, while T2 waits for T1.

3. Sequence S1: With conservative and strict 2PL, the sequence is easy. T1 acquires lock on both X and Y, commits, releases locks; then T2; then T3.

Sequence S2: Same as Sequence S1.

PROBLEM 3 (8 points)

Exercise 17.5 For each of the following locking protocols, assuming that every transaction follows that locking protocol, state which of these desirable properties are ensured: serializability, conflict-serializability, recoverability, avoidance of cascading aborts.

1. Always obtain an exclusive lock before writing; hold exclusive locks until end-of-transaction. No shared locks are ever obtained.
2. In addition to (1), obtain a shared lock before reading; shared locks can be released at any time.
3. As in (2), and in addition, locking is two-phase.
4. As in (2), and in addition, all locks held until end-of-transaction.

Answer 17.5

	Serializable	Conflict serializable	Recoverable	Avoid cascading aborts
1	No	No	No	No
2	No	No	Yes	Yes
3	Yes	Yes	Yes	Yes
4	Yes	Yes	Yes	Yes

TOTAL 50 points