461 Tutorial Week 10 Lab 1 Concurrency Control Rosa KarimiAdl

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 Ti 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.
   Answer- 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. Strict 2PL with deadlock detection. (Show the waits-for graph in case of deadlock.)
   Answer- 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. Conservative (and Strict, i.e., with locks held until end-of-transaction) 2PL.
   Answer- 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.
4. Optimistic concurrency control.

Answer- For both S1 and S2: each transaction will execute, read values from the database and write to a private workspace; they then acquire a timestamp to enter the validation phase. The timestamp of transaction T_i is i.
Sequence S1: Since T1 gets the earliest timestamp, it will commit without problem; but when validating T2 against T1, one of the three conditions must hold:

a) T_1 completes before T_2 begins.
b) T_1 completes before T_2 starts its write phase, and T_1 does not write any database object read by T_2.
c) T_1 completes its read phase before T_2 completes its write phase , and T_1 does not write any database object that is either read or written by T_2.

Since none of the three conditions hold, so T2 will be aborted and restarted later; so is T3 (same as T2).
Sequence S2: The fate is the same as in Sequence S1.

5. Timestamp concurrency control with buffering of reads and writes (to ensure recoverability) and the Thomas Write Rule.
Answer- Initiate the timestamp of objects X and Y to 0.
Sequence S1:
T1: R(X) is allowed since TS(T1)=1 and WTS(X)=0 => TS(T1) ≥ WTS(X). RTS(X) is set to 1.
T2: W(X) is allowed since TS(T2)=2, RTS(X) =1, and WTS(X)=0 => TS(T2) ≥ RTS(X) and TS(T2) ≥ WTS(X). WTS(X) is set to 2.
T2: W(Y) is allowed since TS(T2)=2, RTS(Y) =0, and WTS(Y)=0 => TS(T2) ≥ RTS(X) and TS(T2) ≥ WTS(X). WTS(Y) is set to 2.
T3: W(Y) is allowed since TS(T3)=3, RTS(Y) =0, and WTS(Y)=2 => TS(T2) ≥ RTS(X) and TS(T2) ≥ WTS(X). WTS(Y) is set to 3.
T1: W(Y) is ignored since TS(T1)=1, RTS(Y) =0, and WTS(Y)=3 => TS(T2) ≥ RTS(X) and TS(T1) < WTS(Y).

Sequence S2: Same as above.

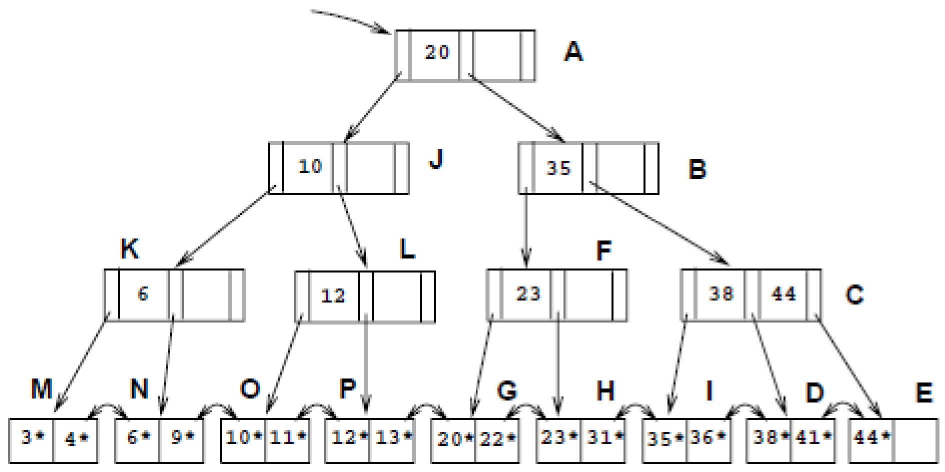
6. Multiversion concurrency control.
Answer- Sequence S1: T1 reads X, so RTS(X) = 1;
T2 is able to write X, since TS(T2) ≥ RTS(X); and RTS(X) and WTS(X) are set to 2;
T2 writes Y, RTS(Y) and WTS(Y) are set to 2;
T3 is able to write Y as well, so RTS(Y) and WTS(Y) are set to 3;
Now when T1 tries to write Y, since TS(T1) < RTS(Y), T1 needs to be aborted and restarted later with larger timestamp.
Sequence S2: The fate is similar to the one in Sequence S1.

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- As a general rule, if a protocol is defined such that transaction T can commit only after all transactions whose changes it reads commit, the protocol is recoverable. If the protocol is defined such

that each transaction can only read the changes of committed transactions, it will not cause any cascading abort.

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

Exercise 17.10 Consider the following tree.



Describe the steps involved in executing each of the following operations according to the tree-index concurrency control algorithm, in terms of the order in which nodes are locked, unlocked, read, and written. Be specific about the kind of lock obtained and answer each part independently of the others, always starting with the original tree.

1. Search for data entry 40*.
2. Search for all data entries k∗ with k ≤ 40.
3. Insert data entry 62*.
4. Insert data entry 40*.

Answer- note the abbreviation used:
S(A): Obtains shared lock on A.
X(A): Obtains exclusive lock on A.
RS(A): Release shared lock on A.
RX(A): Release exclusive lock on A.
R(A): Read node A.
W(A): Write node A.

1.S(A),R(A),S(B),RS(A),R(B),S(C),RS(B),R(C),S(D),RS(C),R(D),...(obtain other locks needed for further operation), then release lock on D when the transaction is going to commit(Strict 2PL).

2. S(A), R(A), S(J), S(B), RS(A), R(J), R(B),S(K), S(L), RS(J), S(F), S(C), RS(B), R(K), R(L), R(F), R(C),
S(M), S(N), S(O), S(P), S(G), S(H), S(I), S(D),
RS(K), RS(L), RS(F), RS(C),
R(M), R(N), R(O), R(P), R(G), R(H), R(I), R(D), ..., then release locks on M, N, O, P, G, H, I, D when the transaction is going to commit.

3. X(A), R(A), X(B), R(B), RX(A), X(C), R(C), X(E), R(E), RX(B), RX(C) W(E), ..., then release lock on E when the transaction is going to commit.

4. X(A), R(A), X(B), R(B), RX(A), X(C), R(C), X(D), R(D), New Node Z,
X(Z), X(I), R(I), W(I), W(Z), W(D),
New Node Y (For a split on C), X(Y), W(Y), W(C), W(B), RX(B), RX(C), RX(Y) ..., then release locks on I, Z, D when the transaction is going to commit.