Shiyi Li

ID: 4576123

CSCI 4707

HW4

A.

B.

1.

a. True, {T1, T2} and {T1, T3} have conflicts

b. True

c. False

2.

a.

T1: S(X) R(X) X(Y)W(Y) C

T2: X(X) W(X) X(Y)W(Y) C

T3: X(Y) W(Y) C

b.

This schedule causes deadlock.

T2

T1

T3

3.

a.

T1

T2

T3

This is conflict serializable, the equivalent schedule is {T3, T2, T1}.

b.

T3

T1

T2

This is conflict serializable, the equivalent schedule is {T1, T2, T3}.

c.

T3

T1

T2

This is conflict serializable because it has cycle.

d.

T2

T1

This is conflict serializable because it has cycle.

e.

T3

T4

T1

T2

This is conflict serializable because it has cycle.

4.

T1: W(X) W(Y) C

T2: W(X) W(Y) C

This schedule is conflict-serializable, and it is conflict equivalent {T1, T2}. This cannot use 2PL to achieve this schedule. T1 first obtains an exclusive lock on X, and then T2 wants to write X. but it has a lock on X, so T2 is blocked and go to the action W(Y) in T1. In this way, the sequence of the actual schedule isn’t the same as the sequence of the schedule shown above.

C

1.

a. False

b. True

c. True

d. False

2.

a.

|  |  |  |  |
| --- | --- | --- | --- |
| LSN | LOG | prevLSN | undonextLSN |
| 00 | Update: T1 write P2 | null | null |
| 10 | Update: T1 write P1 | 00 | 00 |
| 20 | Update: T2 write P5 | null | null |
| 30 | Update: T3 write P3 | null | null |
| 40 | T3 commit | 30 | Not update |
| 50 | Update: T2 write P5 | 20 | 20 |
| 60 | Update: T2 write P3 | 50 | 50 |
| 70 | T2 abort | 60 | Not update |

b.

The transaction table, we can see that the lastLSN for T2 is 70, and the action described by LSN 70 is abort T2, which has prevLSN 60. It adds the LSN 60 to the ToUndo set. Undo chooses LSN 60 in the ToUndo set. It undoes the action with LSN 60, write a CLR with undonextLSN 50 to record the Undo, and add the prevLSN of LSN 60 to the ToUndo set. Then Undo chooses LSN 50 in the ToUndo set. It undoes the action with LSN 50, write a CLR with undonextLSN 20 to record the Undo, and add the prevLSN of LSN 50 to the ToUndo set. Finally, Undo chooses LSN 20 in the ToUndo set. It undoes the action with LSN 20, write a CLR with undonextLSN null to record the Undo. Since LSN is the first update action for T2, after undo the action described by LSN 20, we have undone the whole T2. Then we write a T2 end record to the log to state that we have finished the aborting of T2.

c.

|  |  |  |  |
| --- | --- | --- | --- |
| LSN | LOG | prevLSN | undonextLSN |
| 80 | CLR:Undo T2 LSN60 | 70 | 50 |
| 90 | CLR:Undo T2 LSN50 | 80 | 20 |
| 100 | CLR:Undo T2 LSN20 | 90 |  |
| 110 | T2 ends | 100 |  |

3.

a.

Dirty page table Transaction table

|  |  |
| --- | --- |
| PageID | RecLSN |
| P1 | 40 |
| P2 | 30 |
| P3 | 60 |
| P5 | 50 |

|  |  |
| --- | --- |
| TransID | LAST\_LSN |
| T1 | 40 |
| T2 | 110 |

b.

Dirty page table Transaction table

|  |  |
| --- | --- |
| PageID | RecLSN |
| P1 | 40 |
| P2 | 30 |
| P3 | 60 |
| P5 | 50 |

|  |  |
| --- | --- |
| TransID | LAST\_LSN |
| T1 | 40 |
| T2 | 110 |

c.

|  |
| --- |
| LOG |
| CLR: Undo T2 LSN 90 |
| CLR: Undo T2 LSN 80 |
| CLR: Undo T2 LSN 50 |
| T2 end |
| CLR: Undo T1 LSN 40 |
| CLR: Undo T2 LSN 30 |
| T1 end |