**Metropolitan State University**

**ICS 311 —Database Management Systems**

**Homework #7 Answers**

Due: see syllabus

**Homework #7 Total: 18 Points**

## Question 1 (4 Points):

For each of the following schedules, draw the serlializability graph and determine whether the schedule is conflict serializable or no:

1.1)

|  |  |  |
| --- | --- | --- |
| T4 | T5 | T6 |
| R(X) |  |  |
|  | R(Y) |  |
|  |  | W(X) |
|  | R(X) |  |
| R(Y) |  |  |

T4

T5

T6

The schedule is conflict serializable because there are no cycles in the serializability graph.

1.2)

T4

T5

T6

R(X)

R(Y)

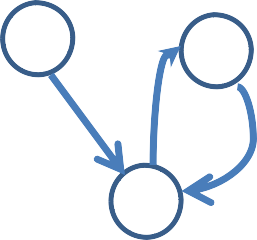
W(X)

R(Y)

W(Y)

W(X)

R(Y)



T4

T5

T6

The schedule is not conflict serializable because there is a cycle in the serializability graph.

## Question 2 (6 Points):

* 1. (1 Points) Explain the difference between “deadlock prevention” and “deadlock detection and recovery”.

***Deadlock prevention*** protocols ensure that the system will *never* enter into a deadlock state.

***Deadlock detection and recovery*** allows system to enter into a deadlock state, and then tries to recover using deadlock retention and recovery

* 1. (1 Points) Explain the difference between “Wait/Die” and “Wound/Wait” techniques for deadlock prevention.

**wait-die** scheme — non-preemptive

older transaction may wait for younger one to release data item. Younger transactions never wait for older ones; they are rolled back instead.

**wound-wait** scheme — preemptive

older transaction *wounds* (forces rollback) of younger transaction instead of waiting for it. Younger transactions may wait for older ones.

See book Section 15.2 (pages 674 to 679)

* 1. (2 Points) Given the following schedule, draw the wait-for graph. **Is there a deadlock?**

T1

T2

T3

T4

Lock-S(A)

R(A)

Lock-X(B)

W(B)

Lock-S(B)

Lock-S(C)

R(C)

Lock-X(C)

Lock-x(B)

Lock-x(A)



T1

T2

T4

T3

Yes. The schedule suffers from a deadlock (T1 is waiting for T2 to release the lock on B, T2 is waiting for T3 to release the lock on C, T3 is waiting for T1 to release the lock on A).

* 1. (2 Points) Consider the following two transactions T1 and T2: T1: R(X) W(X) R(Y) W(Y) Commit

T2: R(Y) W(Y) R(X) W(X) Commit

Draw a schedule for T1 and T2 operations that leads to a deadlock when you use exclusive/shared locking (i.e., 3-state locking). **Draw the wait-for graph for your schedule. Is there a deadlock?**

|  |  |
| --- | --- |
| **T1** | **T2** |
| Lock-x(X) |  |
| R(X) |  |
| W(X) |  |
|  | Lock-x(Y) |
|  | R(Y) |
|  | W(Y) |
|  | Lock-x(X) |
| Lock-x(Y) |  |

T1

T2

There is a deadlock because T1 is waiting for T2 to release the lock on Y and T2 is waiting for T1 to release the lock on X.

## Question 3 (2 Points):

* 1. Explain how does the recovery manager ensure atomicity of transactions?

The Atomicity property of a transaction is violated when a transaction starts to work but the transaction is not completed because of a transaction/system failure. Upon recovering from the failure, the Recovery Manger performs UNDO for all the operations of an incomplete transactions hence ensuring atomicity (either all or none).

* 1. Explain how does the recovery manager ensure durability of transactions?

The durability property of a transaction is violated if a system failure takes place after the transactions is committed in memory but before the transaction is committed on disk. Upon recovering from a failure, the recovery manager performs a REDO for all operations of any transaction that is committed in memory but not committed on disk.

## Question 4 (3 Points):

Given the following log, **show the steps** that are taken by the recovery manger to recover from the crash.

|  |  |  |
| --- | --- | --- |
| **Log** | **Redo Phase** | **Undo Phase (undo T3)** |
| <T1 start> |  |  |
| <T3 start> |  | **9- add <T3 abort> to log** |
| <checkpoint T1, T2, T3> | **1- L = T1 T2 T3** |  |
|
| <T1,p5,200,300> | **2- p5 = 300** |  |
| <T2,p3,400,500> | **3- p3 = 500** |  |
| <T2 commit> | **4- L = T1 T3 (remove T2)** |  |
| <T3,p3,500,600> | **5- p3 = 600** | **8 – p3 = 500 add <T3,p3,500> to log** |
|
| <T1,p5,200> | **6- P5 = 200** |  |
| <T1 abort> | **7- L = T3 (remove T1)** |  |
| **<T3, p3, 500>** |  |  |
| **<T3 abort>** |  |  |

## Question 5 (3 Points):

Given the following log, **show the steps** that are taken by the recovery manager to abort transaction T1.

|  |  |
| --- | --- |
| **Log** | **Undo Phase (undo T1)** |
| <T1 start> | **4- T1 abort** |
| <T2 start> |  |
| <T1,p1,100,200> | **2- p1 = 200** |
| <T1,p2,50,60> | **1-  P2 = 50** |
| <T2,p1,100,200> |  |
| <T3,p4,100,200> |  |
| <T3 commit> |  |
| <T2,p5,200,300> |  |
| <T2,p3,200,300> |  |
| <T2 commit> |  |
| **T1 Crash** |  |
| **<T1,p2,50>** |  |
| **<T1,p1,200>** |  |
| **<T1 abort>** |  |