Nalongsone Danddank Student ID : 14958950 StarID: jf3893pd

Email: [nalongsone.danddank@my.metrostate.edu](mailto:nalongsone.danddank@my.metrostate.edu)\

**Metropolitan State University**

**ICS 311 —Database Management Systems**

**Homework #7 Total: 18 Points**

**Question 1 (4 Points):**

For each of the following schedules, draw the serlializability graph (also known as **precedence graph** or **conflict graph)** and determine whether the schedule is conflict serializable or no:

1.1)

**Solution:**

|  |  |  |
| --- | --- | --- |
| T4 | T5 | T6  5  6  4 |
| R(X) |  |  |
|  | R(Y) |  |
|  |  | W(X)  The precedence graph has **no cycle**.  So the schedule is **conflict serializable**. |
|  | R(X) |  |
| R(Y) |  |  |

1.2) **Solution:**

|  |  |  |
| --- | --- | --- |
| T4 | T5 | T6  5  4 |
| R(X) |  |  |
| R(Y) |  |  |
| W(X) |  |  |
|  | R(Y) | 6 |
|  |  | W(Y) |
| W(X) |  |  |
|  | R(Y) | The precedence graph has **a cycle** between T5 and T6.  So the schedule is **Not** **conflict serializable**. |

**Question 2 (6 Points):**

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

**Answer:**

**Deadlock prevention:** to ensure the system will never enter deadlock state.

**Deadlock detection and recovery**: when we allow the system enter a deadlock state, we try to recover by using their scheme, both methods may result in transaction rollback.

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

**Answer:**

The **wait-die** scheme is a **non-preemptive** technique. When transaction Ti requests a data item currently held by Tj, Ti is allowed to **wait** only if it has a time-stamp smaller than that of Tj Otherwise, Ti is rolled back(**dies**)

The **wound-wait** scheme is a **preemptive** technique. It is a **counterpart** to the wait-die scheme technique. When transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has a tiemstamp larger than that of Tj. Otherwise, Tj is rolled back (Tj is wounded by Ti).

* 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) |  |

**Solution:**

2

1

4

3

This wait-for graph has a **cycle** on T1 -> T2 -> T3 -> T1. So there is a **deadlock**.

* 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.**

**Solution:**

**Schedule :**

|  |  |
| --- | --- |
| T1 | T2 |
| Lock-S(X) | 2  1 |
| R(X) |  |
|  | Lock-X(Y)  Deadlock Wait-for Graph |
|  | R(Y) |
|  | W(Y) |
| Lock-X(X) |  |
| W(X) |  |
|  | Lock-S(X) |
|  | R(X) |
| Lock-X(Y) |  |
| R(Y) |  |
| W(Y) |  |
|  | Lock-X(X) |
|  | W(X) |

**Question 3 (2 Points):**

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

**Answer:**

To ensure atomicity of transactions, the recovery manager has either finished its execution and has saved all the changes that made to the database system or incomplete executions are completely undone.

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

**Answer:**

To ensure durability of transactions, the recovery manager achieves by saving all the changes that made to the system in a log and whenever a crash occurs all the committed transactions are survived.

**Question 4 (3 Points):**

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

**Solution:**

|  |  |  |
| --- | --- | --- |
|  | **Redo Phase:** | **Undo Phase (undo T3)** |
| <T1 start> |  |  |
| <T3 start> |  | 1. **T3 start found.**   **Add<T3 abort> to log** |
| <checkpoint T1, T2, T3> | 1. **L = T1, T2, T3** |  |
|
| <T1,p5,200,300> | 1. **p5 = 300** |  |
| <T2,p3,400,500> | 1. **p3 = 500** |  |
| <T2 commit> | 1. **L = T1, T3**   **(remove T2, commit found)** |  |
| <T3,p3,500,600> | 1. **p3 = 600** | 1. **p3 = 500**   **Add <T3, p3, 500> to log** |
|
| <T1,p5,200> | 1. **p5 = 200** |  |
| <T1 abort> | 1. **L = T3**   **(remove T1, abort found)** |  |
|  |  |  |
| **<T3, p3, 500>** |  |  |
| **<T3 abort>** |  |  |

**Question 5 (3 Points):**

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

**Solution:**

|  |  |
| --- | --- |
| **Log** | **Undo Phase (undo T1)** |
| <T1 start> | **3. T1 abort** |
| <T2 start> |  |
| <T1,p1,100,200> | **2. p1 = 100** |
| <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,p1,100>** |  |
| **<T1,abort>** |  |