Quiz 3

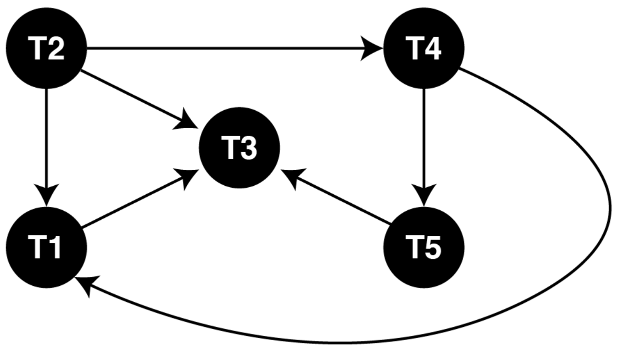

Figure 1: Q1

1 MCQs

1. T/F: Two phase locking prevents transactions deadlock.
2. T/F: Strict two phase locking guarantees cascadelessness.
3. T/F: If a schedule is view-serializable, it is conflict-serializable.
4. T/F: During recovery, we process the undo list in forward order, then the redo list in reverse order.
5. T/F: Recovery undo actions are logged.
6. T/F: Isolation means that transactions execute serially, one after the other.

1.1 Solution

1. False. It only ensures serializability.
2. True. See textbook.
3. False. Many examples in prior homeworks. VS is a superset of CS.
4. False. Undo list in reverse order, then redo list in forward order.
5. True. This is to ensure that if we fail during recovery, we can recover again.
6. False. Isolation means that transactions *seem* to execute serially, but interleaving of actual operations is allowed.

2 Question 1

Given the following precedence graph, determine whether this schedule is conflict-serializable and provide a topological sort order.

2.1 Solution

Since there are no cycles in this precedence graph, there are no conflicts, so we can make a conflict serializable schedule. A potential sort order is: T2, T4, T1, T5, T3. We can actually swap T1 and T5 if we wanted.

3 Question 2

Consider the following schedules, where R is read, W is write, D is display:

Schedule 1:

T1	T2
W(A)	
	R(A)
	W(A)
W(B)	
	R(B)

Schedule 2:

T1	T2
R(A)	
W(A)	
	R(A)
R(A)	
W(B)	
	R(B)

Schedule 3:

T1	T2
R(A)	
D(A)	
	W(B)
	R(A)
W(B)	
	W(B)
	D(A)
R(B)	
D(B)	

1. Which of the above are conflict serializable? For those that are conflict serializable, give the serial schedule they are equivalent with.
2. If possible, make each of the above recoverable by adding a single commit; explain where the commit should go and why it makes the schedule recoverable. If impossible, explain why.
3. If possible, make each of the above cascadeless by adding a single commit; explain where the commit should go and why it makes the schedule cascadeless. If impossible, explain why.
4. If possible, add locks so that each of the above adheres to two phase locking. Write where the locks are taken and released. Avoid deadlocks. If impossible, explain why.

3.1 Solution

1. Which of the above are conflict serializable? For those that are conflict serializable, give the serial schedule they are equivalent with.

Schedules 1 and 2 are conflict serializable, equivalent to T1, T2. Schedule 3 is not conflict serializable - conflict cycles with T1 and T2.

2. If possible, make each of the above recoverable by adding a single commit; explain where the commit should go and why it makes the schedule recoverable. If impossible, explain why.

We note that a recoverable schedule is one in which all transactions T commit after any transaction T' on which T depends commits.

Schedule 1: Possible. T1 commits first.

Schedule 2: Possible. T1 commits first.

Schedule 3: Possible. T2 commits first.

3. If possible, make each of the above cascadeless by adding a single commit; explain where the commit should go and why it makes the schedule cascadeless. If impossible, explain why.

We note that a cascadeless schedule is one in which no transaction T reads a value that is modified by another transaction T' before T' commits.

Schedule 1: Impossible. T2 reads the value of A (which T1 modified) before T1 ever gets a chance to commit.

Schedule 2: Impossible. T2 reads the value of A (which T1 modified) before T1 ever gets a chance to commit.

Schedule 3: Possible. T2 commits first.

4. If possible, add locks so that each of the above adheres to two phase locking. Write where the locks are taken and released. Avoid deadlocks. If impossible, explain why.

Schedule 1: Possible:

T1	T2
X-lock(A)	
W(A)	
	X-lock(A)
	R(A)
	W(A)
X-lock(B)	
W(B)	
unlock(A)	
unlock(B)	
	S-lock(B)
	R(B)
	unlock(A)
	unlock(B)

Schedule 2: Possible:

T1	T2
X-lock(A)	
R(A)	
W(A)	
	S-lock(A)
	R(A)
R(A)	
X-lock(B)	
W(B)	
unlock(A)	
unlock(B)	
	S-lock(B)
	R(B)
	unlock(A)
	unlock(B)

Schedule 3: Possible:

T1	T2
S-lock(A)	
R(A)	
D(A)	
	X-lock(B)
	W(B)
	S-lock(A)
	R(A)
X-lock(B)	
W(B)	
	W(B)
	D(A)
	unlock(A)
	unlock(B)
R(B)	
D(B)	
unlock(A)	
unlock(B)	

4 Question 3

Consider a simple checkpointing protocol and the following set of operations in the log. A crash happens at the indicated time.

```
<T4, start>
<T4, y, 2, 3>
<T1, start>
<T1, z, 0, 5>
<T1, commit>
<T2, start>
<T2, z, 5, 7>
<checkpoint, {??}>
```

```
<T2, x, 1, 9>
<T4, commit>
<T3, start>
<T3, z, 7, 2>
--- CRASH ---
```

1. What should be the value(s) in the set indicated by the question marks?
2. What are the contents of the undo list?
3. What are the contents of the redo list?
4. Show the recovery actions pertaining to the undo log.
5. What is the value of the variable z at the end of the undo actions?
6. Show the recovery actions pertaining to the redo log.
7. What is the value of the variable z at the end of the redo actions?

4.1 Solution

1. What should be the value(s) in the set indicated by the question marks?

The checkpoint should contain a list of the transactions active at the time of checkpointing. We see that 1, 2, 4 have started, but 1 has commited, so it should contain: T4, T2.

2. What are the contents of the undo list?

We want to undo any transactions that have started by the time of the checkpoint but not committed before the crash. This includes: T2, T3.

3. What are the contents of the redo list?

We want to redo any transactions that were started and committed before the crash, but after the checkpoint. This includes: T4.

4. Show the recovery actions pertaining to the undo log.

We want to do the following: (set z to 7 from 2), (set x to 1 from 9), (set z to 5 from 7).

5. What is the value of the variable z at the end of the undo actions?

z should be set to 5.

6. Show the recovery actions pertaining to the redo log.

We want to do the following: (commit T4).

7. What is the value of the variable z at the end of the redo actions?

z should be set to 5.

Feedback

As this is a new course, we appreciate any feedback you have for us! If you enjoyed this assignment, hated this assignment, or have other thoughts to share, please do so here!