Homework 6

Yanlei Diao

Question 1. Serializability and Recoverability

Consider the following classes of schedules: *serializable*, *conflict-serializable*, *recoverable*, *avoids-cascading-aborts* (same as *cascadeless*), and *strict*. For the following schedule, state which of the preceding classes it belongs to.

(It is important to note that serializability concerns only committed transactions while recoverability concerns both committed and aborted transactions.)

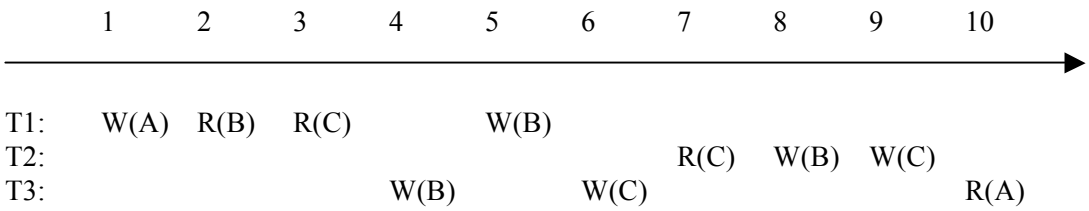
The actions are listed in the order they are scheduled and prefixed with the transaction name.

T1:R(A), T2:W(A), T1:W(A), T2:Abort, T1:Commit

Answer: [5']
1' for each statement
It is serializable, conflict-serializable;
It is recoverable and avoids cascading aborts;
It is not strict.

Question 2. Serializable Schedules

Consider the following schedule involving three transactions T1, T2 and T3:



(1) Draw the precedence graph for this schedule.

(2) Is this schedule conflict serializable? Why or why not? If it is conflict serializable, give the equivalent serial schedule (just write the order of the transactions).

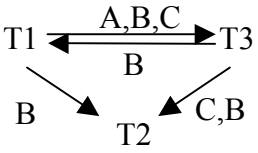
(3) Write down all instances where one transaction "reads from" another transaction.
(If T2 reads from T1, write T1 → T2.)

Now swap the 4th and 5th actions in the schedule.

(4) Is the new schedule conflict serializable? Why or why not? If it is conflict serializable, give the equivalent serial schedule (just write the order of the transactions).

Answer:

(1) [5'] 1' for each edge, 1' for all three nodes



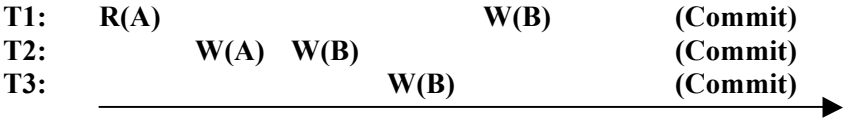
(2) [5']
No, because the precedence graph has a cycle between T1 and T3.

(3) [5'] 2 points each
T1→ T3 (A)
T3→ T2 (C)

(4) [5']
Yes. Now the precedence graph is acyclic. [2']
And an equivalent serial schedule is: [3']
T1→ T3 → T2

Question 3. Concurrency Control Protocols

Consider three transactions, T1, T2, and T3, with timestamps 1, 2, 3, respectively. Now consider the following sequences of actions, listed in the order that they are produced from user queries:



Explain how this sequence of actions will be executed using the **Strict Two Phase Locking** protocol (2PL). The following table is designed for you to specify in time order the activities that take place in the lock manager and in access to the database. The activities include:

- S(O) or X(O): *the lock requested for access to a database object, O*;
- R(O) or W(O): *the read or write access to a database object, O*;
- Ti blocked: *which transaction may be blocked under this protocol*;
- Ti resumed: *until when a blocked transaction can resume, with its lock request granted*; *and*
- Ti commits and releases locks: *when a transaction can commit and release locks*.

The first few actions are already given below. Please complete the other actions for this schedule.

Time stamp	**Action:** please choose one action at a time

1	T1: S(A)
2	T1: R(A)
3	T2: X(A)
4	T2: blocked
5	…
6	
7	
8	
9	
10	
11	
12	
13	

Answer:
(1) 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.)
T1 gets a shared-lock on A.
T2 blocks waiting for an exclusive-lock on A.
T3 gets an exclusive-lock on B.
T1 blocks waiting for an exclusive-lock on B.
T3 finishes, commits and releases locks.
T1 wakes up, gets an exclusive-lock on B, finishes up and releases lock on A and B.
T2 now gets both an exclusive-lock on A and B, and proceeds to finish.
No deadlock.

Time stamp	**Action:** please choose one action at a time
1	T1: S(A)
2	T1: R(A)
3	T2: X(A)
4	T2: blocked
5	T3: X(B)
6	T3: W(B)
7	T1: X(B)
8	T1: blocked
9	T3: commits and releases locks
10	T1: resumed (with an X lock on B)
11	T1: W(B)
12	T1: commits and releases locks
13	T2: resumed (with X locks on A and B)
14	T2: W(A)
15	T2: W(B)
16	T2: commits and releases locks

Question 4. Recovery

Examine the schedule given below. There are three transactions, T1, T2, and T3. Initially, the salary = 1 and the tax = 2. The assignments happen within the local memory space of the transactions and the effects of these assignments are not reflected in the database until the WRITE operation.

```
     T1                        T2                    T3
-------------------------------------------------------------------
0                                                     start
1                                                     READ tax
2                                                     tax := tax + 1
3     start
4     READ salary
5     salary := salary + 1
6                                                     WRITE tax
7                                                     commit
8                               start
9                               READ tax
10                              READ salary
11                              tax := tax + salary
12                              WRITE tax
13                              commit
14    READ tax
15    tax := tax + salary
16    WRITE tax, salary
17    ------------------- system crashes -------------------------
```

(1) Show the log entries that would be generated by this execution. Use the same notation used in class. For each log entry, indicate what line above generates it.

(2) When the system is rebooted and tries to recover from the crash, which transaction(s) should be undone and which may be redone?

Answer:

(1) The log entries are generated for write operations, shown in lines 6, 12, 16.

LSN LOG

1 T3 writes a new value to the object “tax” (line 6)
2 T3 commits
3 T2 writes a new value to the object “tax” (line 12)
4 T2 commits
5 T1 writes a new value to the object “tax” and a new value to the object “salary” (line 16)

(2) When the system is rebooted and tries to recover from the crash, T1 needs to be undone as it didn't finish. According to the “all or none” property, we need to undo its changes.

T2 and T3 may be redone because they committed before the crash, but their changes to the data pages might still reside in memory (due to the “no force” policy). To ensure that changes of T2 and T3 persist, we may have to redo their actions by reading the corresponding log entries.

Question 5. The ARIES Recovery Algorithm

Consider the execution shown in the following figure.

LSN	Log Record
00	Update: T1 writes P2
10	Update: T1 writes P1
20	Update: T2 writes P5
30	Update: T3 writes P3
40	T3 commits
50	Update: T2 writes P5
60	Update: T2 writes P5
70	T2 aborts

(1) For the above log records, complete the table below with

- the prevLSN value, and
- the undonextLSN value if it is a compensation log record, otherwise simply with "—".

LSN	prevLSN	undonextLSN (of a CLR)
00		
10		
20		
30		
40		
50		
60		
70		

(2) Describe the actions taken to rollback transaction T2.

(3) Assume that the system crashed right after generating the log record 70. Now the system wakes up and runs the ARIES recovery algorithm. Please show all new log entries generated in the recovery process, where each log record includs the LSN, type of log record, transaction ID, page ID, prevLSN, and undonextLSN.

LSN	Type	Xact ID	Page ID	prevLSN	undonextLSN
80					
90					
100					
110					
120					
130					
140					

Answer:

(1) The extended table is shown above.

LSN	**prevLSN**	**undonextLSN (of a CLR)**
00	---	---
10	00	--- (not a CLR)
20	---	---
30	---	---
40	30	--- (not an update log record)
50	20	--- (not a CLR)
60	50	--- (not a CLR)
70	60	--- (not an update log record

(2)
Step i) Restore P3 to the before-image stored in LSN 60.
Step ii) Restore P5 to the before-image stored in LSN 50.
Step iii) Restore P5 to the before-image stored in LSN 20.

(3) The log tail should include undo operations for both T1 and T2. It can look like this:

LSN	Type	Xact ID	Page ID	prevLSN	undonextLSN
80	CLR	T2	P5	70	50
90	CLR	T2	P5	80	20
100	CLR	T2	P5	90	---
105	end	T2	---	100	---
110	Abort	T1	--	10	--
120	CLR	T1	P1	10	00
130	CLR	T1	P2	120	---
140	End	T1	---		---

There is no penalty if students omit the Abort and End records in this exercise.