CS 186 - Fall 2020
Exam Prep Section 4
Transactions & Concurrency, Recovery

Friday, October 30, 2020

1 Transactions and Concurrency

In this question, we will explore the key topics of transactions and concurrency: serializability, types of locks, two-phase locking, and deadlocks.

We will do this by actually running multiple transactions at the same time on a database, and seeing what happens. You may find it helpful (but not necessary) to draw some graphs:

- For the lock type questions, you may wish to draw a graph representing the whole database and which resources are being locked.
- For serializability questions, you may wish to draw a graph with a node for each transaction, and arrows if there are *conflicts* between transactions.
- For deadlock questions, you may wish to draw a graph with a node for each transaction, and arrows if a transaction is *waiting* for a lock held by another transaction.

We will use a database with tables A, B, C, ... and table A holds rows A1, A2, A3, ... and so on.

Consider the following sequence of operations:

```
Txn 1:  IX-Lock(Database)      (1)
Txn 1:  IX-Lock(Table A)       (2)
Txn 1:  X-Lock(Row A1)         (3)
Txn 1:  Write(Row A1)          (4)
Txn 1:  Unlock(Row A1)         (5)
Txn 1:  S-Lock(Row A2)         (6)
Txn 2:  IX-Lock(Database)      (7)
Txn 2:  IX-Lock(Table A)       (8)
Txn 2:  X-Lock(Row A1)         (9)
Txn 2:  Write(Row A1)          (10)
Txn 2:  S-Lock(Row A2)         (11)
Txn 2:  Read(Row A2)           (12)
```

1. Is Transaction 1 doing Two-Phase Locking so far?

2. Is Transaction 1 doing Strict Two-Phase Locking?

3. Is this schedule conflict-serializable so far? If not, what is the cycle?

4. Is this schedule serial so far?

Continuing with all operations so far:

```
Txn 2:  SIX-Lock(Table B)      (13)
Txn 2:  X-Lock(Row B1)         (14)
Txn 2:  Write(Row B1)          (15)
Txn 2:  Unlock(Row B1)         (16)
Txn 2:  Unlock(Table B)        (17)
Txn 1:  SIX-Lock(Table B)      (18)
Txn 1:  X-Lock(Row B1)         (19)
Txn 1:  Read(Row B1)           (20)
```

5. Is Transaction 2 doing Two-Phase Locking so far?

6. Is Transaction 2 doing Strict Two-Phase Locking?

7. Is this schedule conflict-serializable so far? If not, what is the cycle?

8. Suppose we start a new transaction, Transaction 3. What kind of locks can Transaction 3 acquire on the whole database?

9. Given the above answers, what kind of locks can Transaction 3 acquire on Table A?

10. Given the above answers, what kind of locks can Transaction 3 acquire on Row A3?

11. Given the above answers, what kind of locks can Transaction 3 acquire on Table B?

12. Given the above answers, what kind of locks can Transaction 3 acquire on Row B2?

13. What kind of locks can Transaction 1 acquire on Row B2?

```
Txn 2:  IX-Lock(Table B)          (21)
Txn 3:  IX-Lock(Database)         (22)
Txn 3:  X-Lock(Table C)           (23)
Txn 3:  IX-Lock(Table A)          (24)
Txn 3:  X-Lock(Row A1)            (25)
Txn 1:  IX-Lock(Table C)          (26)
```

14. We have now entered a deadlock. What is the waits-for cycle between the transactions?

15. We can end this deadlock by aborting the youngest transaction. Which transaction do we abort?

    Alternatively, we could have avoided this deadlock in the first place by using *wound-wait* or *wait-die*. Recall from lecture that these methods cause transactions to sometimes abort, according to a priority order, when they try to acquire locks.

    Let's say that the priority of the transaction is its number (Txn 1 is highest priority).

16. If we were using *wound-wait*, what is the **first operation** in this sequence that would cause a transaction to get aborted, and which transaction gets aborted?

17. If we were using *wait-die*, what is the **first operation** in this sequence that would cause a transaction to get aborted, and which transaction gets aborted?

2 Miscellaneous Transactions & Concurrency

Consider a database with objects X and Y and two transactions. Transaction 1 reads X and Y and then writes X and Y. Transaction 2 reads and writes X then reads and writes Y.

1. Create a schedule for these transactions that is **not** serializable. Explain why your schedule is not serializable.

2. Would your schedule be allowed under strict two-phase locking? Why or why not?

Now consider the following schedule.

	1	2	3	4	5	6	7	8
T1				X(B)	X(A)			S(D)
T2		X(D)	S(E)					
T3	X(E)						S(B)	
T4						X(A)		

3. Draw the waits-for graph for this schedule.

4. Are there any transactions involved in deadlock?

5. Next, assume that T1 priority > T2 > T3 > T4. You are the database administrator and one of your users purchases an exclusive plan to ensure that their transaction, Transaction 2, runs to completion. Assuming the same schedule, what deadlock avoidance policy would you choose to make sure Transaction 2 commits?

3 Conceptual Recovery

1. Consider a scenario where we update the recLSN in the dirty page table to reflect each update to a page, regardless of when the page was brought into the buffer pool. What bugs might you see after recovery? Select all that apply. Explain your reasoning.
   (a) Some writes of committed transactions would be lost.
   (b) Some writes of aborted transactions would be visible in the database.
   (c) The system tries to commit or abort a transaction that is not in the transaction table.

2. Suppose that you are forced to flush pages in the DPT to disk upon making a checkpoint. Which of the following cases are now guaranteed? There is one correct answer. Explain your reasoning.
   (a) We can skip one of the three phases (analysis/redo/undo) completely
   (b) We must start analysis from the beginning of the log
   (c) Redo will start at the checkpoint.
   (d) Redo must start from the beginning of the log
   (e) Undo can start at the checkpoint
   (f) Undo must run until the beginning of the log

3. If the buffer pool is large enough that uncommitted data are never forced to disk, is UNDO still necessary? How about REDO? Explain your reasoning.

4. If updates are always forced to disk when a transaction commits, is UNDO still necessary? Will ARIES perform any REDOs? Explain your reasoning.

4 Recovery Practice

For this question, you will want to have the details of the ARIES protocol handy, so we suggest you have the slides or some notes to look at while doing this question.

The year is 2029. Power outages in Berkeley are so common now that PG&E does not even send out warnings anymore - instead, they just pull the plug whenever they want.

Our database has just restarted from one such power outage. You look at the logs on disk, and this is what you see:

LSN	Record		
10	T1	update	P1
20	T2	update	P2
30	T1	update	P2
40	T1	update	P3
50		begin-checkpoint	
60	T1	update	P4
70		end-checkpoint	
80	T1	commit	
90	T2	update	P1

You load up the checkpoint and see:

Transaction Table

Txn ID	Last LSN	Txn status
T1	40	running
T2	20	running

Dirty Page Table

Page	recLSN
P1	10
P2	30

1. What is the latest LSN that this checkpoint is guaranteed to be up-to-date to?

2. What do the transaction table and dirty page table look like at the end of analysis, and what log records do we write during analysis?

3. The next phase of ARIES is redo. What LSN do we start the redo from?

4. From that record, we will redo the effects of all the following records, except we will not redo certain records. What are the LSNs of the records we do NOT redo?

5. The last phase of ARIES is undo. What do we do for this phase? Answer this question by writing out the log records that will be recorded for each step.

   Stop after you write your first CLR record (make sure your CLR record specifies the nextLSN!).

Click! The lights go out, and you realize PG&E has pulled the power yet again... during ARIES recovery no less!

Five minutes later, the power comes back online. You inspect the log, and are glad to see that **all the logp records you wrote have made it to disk**.

6. You load up the checkpoint. What does the transaction table and dirty page table look like?

7. You run the analysis phase. What do the transaction table and dirty page look like at the end of analysis?

8. You run the redo phase. In order, what are the LSNs that we redo?

9. Now we run the undo phase. What do we do? (Answer again with the log records that you have to add.)