CS 186 - Fall 2024
Exam Prep 9
Recovery

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.

    Answer: a, b.

    - A is correct because during the REDO phase of recovery, some UPDATE log records that reflect writes that never made it to disk will be skipped.
    - Similarly, B is correct, because some CLR's that reflect UNDO's that never made it to disk will be skipped.
    - C is incorrect because even if REDO begins at a later LSN, the system does not add any new transactions to the transaction table during REDO.

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

    Answer: c.

    In general, we redo everything from the earliest recLSN in the DPT to get back unflushed changes from before crash. Since we can guarantee that all changes up to a checkpoint have been flushed, all unflushed changes from before the crash happened after the checkpoint. Therefore, we can redo starting from the checkpoint.

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

    Answer: UNDO isn't necessary in terms of undoing operations on disk. REDO is still necessary.

    Having a buffer pool large enough to hold all uncommitted data means we don't have to STEAL (allow an uncommitted transaction to overwrite the most recent committed value of an object on disk). Since all the updates will be sitting in the buffer pool at the time of crash, no changes will be made to disk, so no operations need to be undone.

    REDO is needed to get back unflushed changes from before the crash. If everything is held in the buffer, this must be redone

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

   Answer: Only UNDO is necessary.

   UNDO is necessary because we still have to finish aborting transactions that were in progress and weren't committing. The updates that these transactions have gotten through so far must be rolled back. As for REDOs, ARIES might perform some redoes because there may be transactions still in progress at the time of a crash, but these will be undone in the UNDO phase.

Recovery Practice

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?

   Answer: LSN 50 - the begin-checkpoint record.

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?

LSN	Record
100	T1 end
110	T2 abort

Transaction Table

Txn ID	Last LSN	Txn status
T2	110	aborting

Dirty Page Table

Page	recLSN
P1	10
P2	30
P4	60

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

   Answer: LSN 10 - the oldest recLSN in the dirty page table.

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?

   Answer:

   - LSN 20 is not redone; the recLSN of P2 is already higher than 20, so the effects of LSN 20 are already on disk.
   - LSN 40 is not redone; page P3 is not in the dirty page table, and thus also already on disk.
   - LSNs 50, 70, 80, 100, and 110 are not update operations, so we don't do anything for them.

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!).

LSN	Record
120	T2 CLR nextLSN: 20

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 log 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?

   Answer: Same as the checkpoint from before! We never made another checkpoint.

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

Transaction Table

Txn ID	Last LSN	Txn status
T2	120	aborting

Dirty Page Table

Page	recLSN
P1	10
P2	30
P4	60

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

   Answer: 10, 30, 60, 90, 120.

   Importantly, note that we redid T1 even though it committed, and that we ARE redoing CLRs.

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

LSN	Record
130	T2 CLR nextLSN: null
140	T2 end