Aries Example

A database contains two pages P1 and P2. P1 contains two elements A and B. P2 contains two elements C and D.

Consider the following sequence of operations on the database:

- Transaction T1 writes A.
- Transaction T2 writes B.
- Transaction T2 writes C.
- The system flushes the log to disk and also flushes page P2 to disk.
- Transaction T1 writes D.
- Transaction T1 commits. The system writes a commit log record and flushes the tail of the log to disk.
- Transaction T2 writes B.
- The system writes an END log record for T1.
- The system crashes.

1 State of the System Before Crash

The state of the system right before the crash is as follows:

Transaction Table:

transID	lastLSN	status
T2	6	Running (i.e. In Progress)

Transaction T1 gets removed from the table after we force-write (i.e., write and flush to disk) the commit log record. Right before being removed, T1 had an entry in the table with lastLSN 5 and status committed.

Dirty Page Table:

pageID	recLSN
P1	1
P2	4

Log:

LSN	transID	prevLSN	type	pageID	log entry	undoNextLSN
1	T1	-	Update	P1	Write A (A → A1)	-
2	T2	-	Update	P1	Write B (B → B2)	-
3	T2	2	Update	P2	Write C (C → C3)	-
4	T1	1	Update	P2	Write D (D → D4)	-
5	T1	4	Commit	-	-	-
6	T2	3	Update	P1	Write B (B2 → B6)	-
7	T1	5	End	-	-	-

Pages in memory:

- P1: A has value A1. B has value B6. The pageLSN is 6.
- P2: C has value C3. D has value D4. The pageLSN is 4.

2 Analysis Phase

When the system crashes and restarts, only the part of the log that was flushed to disk remains. Everything else (transactions table, dirty pages table, tail of the log) is gone. When the system restarts, it thus finds on disk the log file up to LSN 5. The analysis phase starts at the beginning of the log. It reads the log forward and rebuilds the transactions table and dirty pages table. Their contents at the end of the phase are as shown below.

The rules of the analysis phase are as follows:

- END record removes transaction from the Transaction Table.
- Other records update lastLSN for corresponding transaction.
- Commit record changes status from Unknown to Committed.
- For redoable records, update the Dirty Page Table.

Transaction Table:

transID	lastLSN	status
T1	5	Committed
T2	3	Unknown

Dirty Page Table:

pageID	recLSN
P1	1
P2	3

3 Redo Phase

The REDO phase starts at the firstLSN, which is the smallest LSN in the Dirty Page Table. In this example, it's LSN 1.

For each redoable log record (update or CLR), the redo phase redoes the change if necessary. To check if the system needs to reapply the change to the page, it first checks if the page is in the Dirty Page Table. If it is in the table, then it checks that the recLSN for the page is lower or equal to the LSN of the change under consideration. If that is the case, then finally the system reads the page from disk and checks if the pageLSN is strictly smaller than the current LSN. If that is the case then it redoes the change. Otherwise, it skips the change. In the example, we get the following:

- LSN 1: Redone.
- LSN 2: Redone
- LSN 3: No need to redo because PageLSN is 3 already.
- LSN 4: Redone
- LSN 5: Skipped

At this point, the system is back in the state at the time when the tail of the log was last flushed to disk. The system can now write an END type record for T1. This record will have LSN 6. T1 can then be removed from the Transaction Table.

4 Undo Phase

The system must undo T2. It's the only transaction in the Transaction Table. Its lastLSN is 3. So we start undo at this LSN 3. There is no need to write an abort log record during undo. The system can write CLRs directly:

LSN	transID	prevLSN	type	pageID	log entry	undoNextLSN
1	T1	-	Update	P1	Write A (A → A1)	-
2	T2	-	Update	P1	Write B (B → B2)	-
3	T2	2	Update	P2	Write C (C → C3)	-
4	T1	1	Update	P2	Write D (D → D4)	-
5	T1	4	Commit	-	-	-
6	T1	5	End	-	-	
7	T2	-	CLR	-	Unto T2 LSN 3	2
8	T2	-	CLR	-	Unto T2 LSN 2	-
9	T2	8	End	-	-	

Pages in memory:

- P1: A has value A1. B has its initial value. The pageLSN is 8.
- P2: C has its initial value C. D has value D4. The pageLSN is 7.

5 Second Crash

Let's consider several scenarios for a second crash:

- System crashes without having written any of the new log entries or pages to disk: In that case, the system will redo the analysis, redo, and undo phases in exactly the same way as we did above.
- System flushes the log to disk, except for T2's END log record: In that case, at the end of the analysis phase, only T2 is in the Transaction Table. We have the same Dirty Page Table. The redo phase is the same as above until LSN 5. For LSN 6, the system does nothing. For LSN 7, if we previously flushed the page to disk, then we skip it. Otherwise, we reapply the change. Same for LSN 8. Finally, the undo phase will undo T2. It finds a CLR for T2 without any undoNextLSN. So it writes an END log record directly.
- System flushed the entire new log to disk: In that case, the transactions table at the end of the analysis phase is empty and the system would not undo anything.
- System crashes after flushing CLR with LSN 7. In that case, the undo phase starts from that CLR when undoing T2. It directly follows the undoNextLSN pointer. It adds the second CLR and END log records.

6 Checkpoint

Aries uses *fuzzy* checkpoints, which proceed in three steps:

1. Write a begin_checkpoint log record.
2. Construct an end_checkpoint log record with current content of the Transaction Table and Dirty Page Tables. Append the record to the log. Flush tail of the log to disk.
3. Write the LSN of the begin_checkpoint log record in a special place.

The analysis phase starts from the begin_checkpoint and initializes the Transaction Table and Dirty Page Table with the value in the end_checkpoint log record.

In our example, imagine that we checkpoint after flushing P2 to disk and before $w1[D]$. Then after the analysis phase, the Dirty Pages table contains P1 with recLSN 1 and P2 with recLSN 4. Redo still needs to start at LSN 1 but it skips over LSN 3 without loading P2 from disk to check that the change need not be applied.

In another scenario, if the system also flushed P1 to disk before the checkpoint, the Dirty Page Table after analysis would only contain P2 with LSN 4. Redo would start at LSN 4.

7 Transaction Aborts

Finally, consider the scenario where instead of crashing T2 aborts.

Transaction Table: Empty since both transactions ended.

Dirty Page Table:

pageID	recLSN
P1	1
P2	4

Log:

LSN	transID	prevLSN	type	pageID	log entry	undoNextLSN
1	T1	-	Update	P1	Write A (A → A1)	-
2	T2	-	Update	P1	Write B (B → B2)	-
3	T2	2	Update	P2	Write C (C → C3)	-
4	T1	1	Update	P2	Write D (D → D4)	-
5	T1	4	Commit	-	-	-
6	T2	3	Update	P1	Write B (B2 → B6)	-
7	T1	5	End	-	-	-
8	T2	6	Abort	-	-	-
9	T2	-	CLR	-	Undo T2 LSN6	3
10	T2	-	CLR	-	Undo T2 LSN3	2
11	T2	-	CLR	-	Undo T2 LSN2	-
12	T2	11	End	-	-	-

Pages in memory:

- P1: A has value A1. B has its initial value. The pageLSN is 11.
- P2: C has its initial value. D has value D4. The pageLSN is 10.