Lecture 8: Recovery (Part 2)

Log Sequence Numbers

- Log Sequence Numbers:
  - LSNs identify log records; linked into backwards chains per transaction via prevLSN.
  - pageLSN allows comparison of data page and log records.

ARIES

- Mains ideas of ARIES:
  - WAL with STEAL/NO-FORCE
  - Fuzzy Checkpoints (snapshot of dirty page ids)
  - Write CLRs when undoing, to survive failures during restarts
  - ATT tells the DBMS which txns were active at time of crash.
  - DPT tells the DBMS which dirty pages might not have made it to disk.

Fuzzy Checkpointing

- The LSN of the **<CHECKPOINT-BEGIN>** record is written to the database's MasterRecord entry on disk when the checkpoint successfully completes.
- Any txn that starts after the checkpoint is excluded from the ATT in the **<CHECKPOINT-END>** record.

TXN-END Record: Abort

- First write an **<ABORT>** record to log for the txn.
- Then play back the txn's updates in reverse order. For each update record:
  - Write a CLR entry to the log.
  - Restore old value.
- When a txn aborts, we immediately tell the application that it is aborted.
- We don't need to wait to flush the CLRs
- At end, write a **<TXN-END>** log record.
- **Notice:** CLRs never need to be undone.

TXN-END Record: Commit

- Write **<COMMIT>** Record to Log
- All log records up to the transaction's **LastLSN** are flushed.
  - Log flushes are sequential, synchronous writes to disk
- Commit() returns
- Write **<TXN-END>** record to log
- Besides flushing, **<TXN-END>** record is related to **releasing locks**

Purpose of CLR

- Before restoring the old value of a page, write a Compensation Log Record (CLR).
- Logging **continues** during UNDO processing
- CLRs contain REDO info
- CLRs are never UNDOne
  - Undo need not be idempotent (>1 UNDO won't happen)
  - But they might be Redone when repeating history (=1 UNDO guaranteed)
- By appropriate chaning of the CLRs to log records written during forward processing, a **bounded amount of logging** is ensured during rollbacks, even in the face of repeated failures during restart.

Today's Agenda

- Phases of ARIES
- Analysis Phase
- Redo and Undo Phases
- Full Example
- Additional Crash Issues

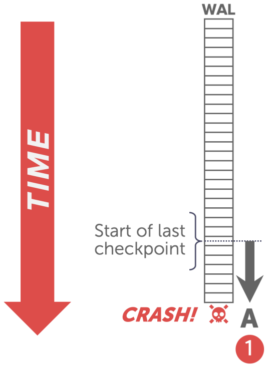
Phases of ARIES

ARIES – Phases

- **Phase 1 – Analysis**
  - Read WAL from last checkpoint to identify dirty pages in the buffer pool and active txns at the time of the crash.
- **Phase 2 – Redo**
  - Repeat **all** actions starting from an appropriate point in the log (even txns that will abort).
- **Phase 3 – Undo**
  - Reverse the actions of txns that did not commit before the crash.

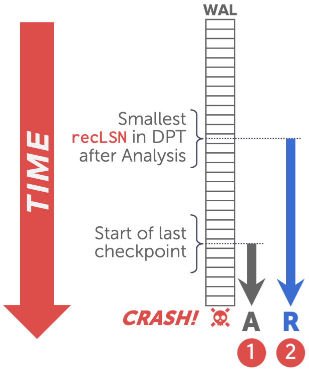
ARIES – Overview

- Start from last **<BEGIN-CHECKPOINT>** found via **MasterRecord**.
- Analysis: Figure out which txns committed or failed since checkpoint.
- Redo: Repeat all actions.
- Undo: Reverse effects of failed txns.



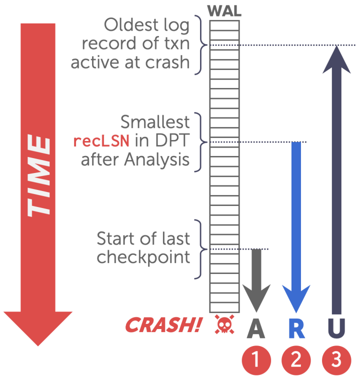
ARIES – Overview

- Start from last **<BEGIN-CHECKPOINT>** found via **MasterRecord**.
- Analysis: Figure out which txns committed or failed since checkpoint.
- Redo: Repeat all actions.
- Undo: Reverse effects of failed txns.



ARIES – Overview

- Start from last **<BEGIN-CHECKPOINT>** found via **MasterRecord**.
- Analysis: Figure out which txns committed or failed since checkpoint.
- Redo: Repeat all actions.
- Undo: Reverse effects of failed txns.


WAL
Oldest log record of txn active at crash
Smallest recLSN in DPT after Analysis
Start of last checkpoint
TIME
CRASH!
A
R
U
1
2
3

Analysis Phase

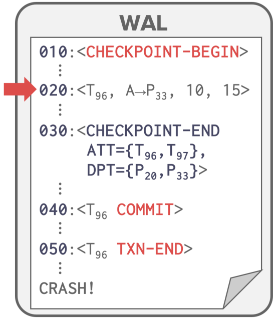
Analysis Phase

- Scan log forward from last successful checkpoint.
- If you find a **TXN-END** record, remove its corresponding txn from **ATT**.
- All other records:
  - Add txn to ATT with status **UNDO**.
  - On commit, change txn status to **COMMIT**.
- For **UPDATE** records:
  - If page P not in **DPT**, add P to DPT, set its **recLSN** = LSN.
  - **recLSN**: LSN of the log record which **first** caused the page to be dirty

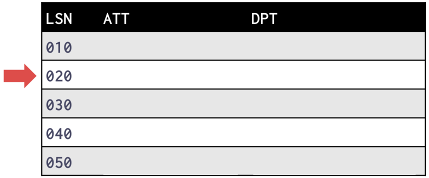
Analysis Phase

- At end of the Analysis Phase:
  - ATT tells the DBMS which txns were active at time of crash.
  - DPT tells the DBMS which dirty pages might not have made it to disk.

Analysis Phase: Example

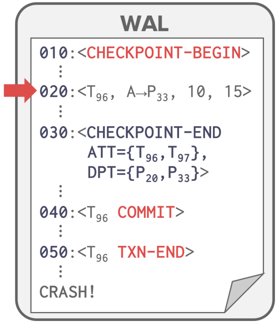


```
WAL
010:<CHECKPOINT-BEGIN>
⋮
020:<T96, A→P33, 10, 15>
⋮
030:<CHECKPOINT-END
      ATT={T96,T97},
      DPT={P20,P33}>
⋮
040:<T96 COMMIT>
⋮
050:<T96 TXN-END>
⋮
CRASH!
```

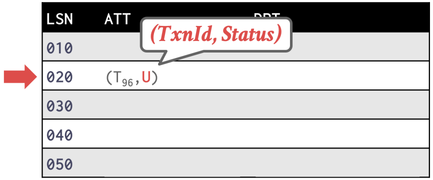


LSN	ATT	DPT
010		
020		
030		
040		
050		

Analysis Phase: Example



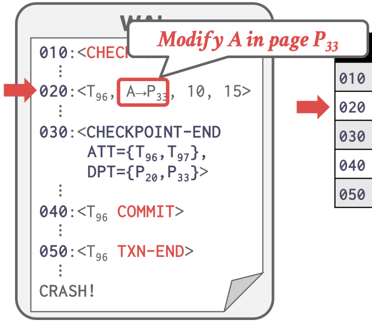
```
WAL
010:<CHECKPOINT-BEGIN>
   ⋮
020:<T96, A→P33, 10, 15>
   ⋮
030:<CHECKPOINT-END
      ATT={T96,T97},
      DPT={P20,P33}>
   ⋮
040:<T96 COMMIT>
   ⋮
050:<T96 TXN-END>
   ⋮
CRASH!
```



(TxnId, Status)

LSN	ATT	DPT
010		
020	(T_{96},U)	
030		
040		
050		

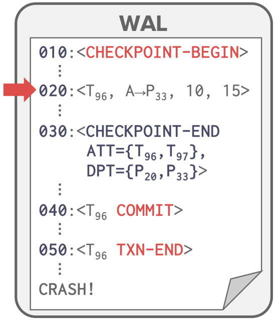
Analysis Phase: Example



```
010:<CHECK
  ⋮
020:<T96, A→P33, 10, 15>
  ⋮
030:<CHECKPOINT-END
       ATT={T96,T97},
       DPT={P20,P33}>
  ⋮
040:<T96 COMMIT>
  ⋮
050:<T96 TXN-END>
  ⋮
CRASH!
```

	ATT	DPT
010		
020	(T_{96},U)	
030		
040		
050		

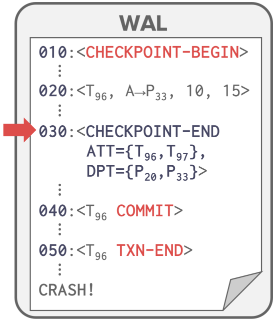
Analysis Phase: Example



```
WAL
010:<CHECKPOINT-BEGIN>
  ⋮
020:<T96, A→P33, 10, 15>
  ⋮
030:<CHECKPOINT-END
     ATT={T96,T97},
     DPT={P20,P33}>
  ⋮
040:<T96 COMMIT>
  ⋮
050:<T96 TXN-END>
  ⋮
CRASH!
```

LSN	ATT	DPT
010		
020	(T_{96}, U)	$(P_{33}, 020)$
030		
040		
050		

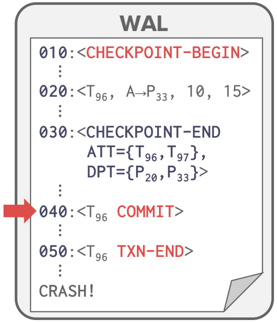
Analysis Phase: Example



```
WAL
010:<CHECKPOINT-BEGIN>
   ⋮
020:<T96, A→P33, 10, 15>
   ⋮
030:<CHECKPOINT-END
     ATT={T96,T97},
     DPT={P20,P33}>
   ⋮
040:<T96 COMMIT>
   ⋮
050:<T96 TXN-END>
   ⋮
CRASH!
```

LSN	ATT	DPT
010		
020	(T_{96}, U)	$(P_{33}, 020)$
030	(T_{96}, U), (T_{97}, U)	$(P_{33}, 020)$, $(P_{20}, 022)$
040		
050		

Analysis Phase: Example

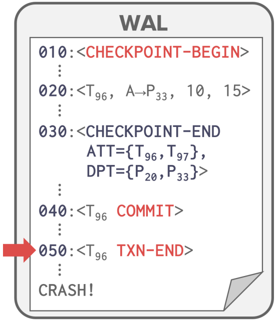


WAL

010:<CHECKPOINT-BEGIN>
⋮
020:<T_{96}, A→P_{33}, 10, 15>
⋮
030:<CHECKPOINT-END
ATT={T_{96},T_{97}},
DPT={P_{20},P_{33}}>
⋮
040:<T_{96} COMMIT>
⋮
050:<T_{96} TXN-END>
⋮
CRASH!

LSN	ATT	DPT
010		
020	(T_{96},U)	(P_{33},020)
030	(T_{96},U), (T_{97},U)	(P_{33},020), (P_{20},022)
040	(T_{96},C), (T_{97},U)	(P_{33},020), (P_{20},022)
050		

Analysis Phase: Example



```
WAL
010:<CHECKPOINT-BEGIN>
  ⋮
020:<T96, A→P33, 10, 15>
  ⋮
030:<CHECKPOINT-END
     ATT={T96,T97},
     DPT={P20,P33}>
  ⋮
040:<T96 COMMIT>
  ⋮
050:<T96 TXN-END>
  ⋮
CRASH!
```

LSN	ATT	DPT
010		
020	(T_{96},U)	(P_{33},020)
030	(T_{96},U), (T_{97},U)	(P_{33},020), (P_{20},022)
040	(T_{96},C), (T_{97},U)	(P_{33},020), (P_{20},022)
050	(T_{97},U)	(P_{33},020), (P_{20},022)

Redo and Undo Phases

Redo Phase

- The goal is to repeat history to reconstruct state at the moment of the crash:
  - Reapply all updates (even aborted txns!) and redo **CLRs**.
- There techniques that allow the DBMS to avoid unnecessary reads/writes, but we will ignore that in this lecture. . .

Redo Phase

- Scan forward from the log record containing smallest/oldest **recLSN** in DPT.
- For each update log record or CLR with a given LSN, redo the action unless:
  - Affected page is not in DPT, or
  - Affected page is in DPT but that record's **LSN** is older than page's **recLSN**.
- Apply changes for pages in DPT and **pageLSN** (in DB) < **LSN**
- Everything before the oldest **recLSN** in DPT is guaranteed to have been flushed.
- If a page's **recLSN** is newer than **LSN**, then no need to read page in from disk to check **pageLSN**

Redo Phase

- To redo an action:
  - Reapply logged action.
  - Set **pageLSN** to log record's LSN.
  - No additional logging, no forced flushes!
- At the end of Redo Phase, write **<TXN-END>** log records for all txns with status **C** and remove them from the ATT.

Undo Phase

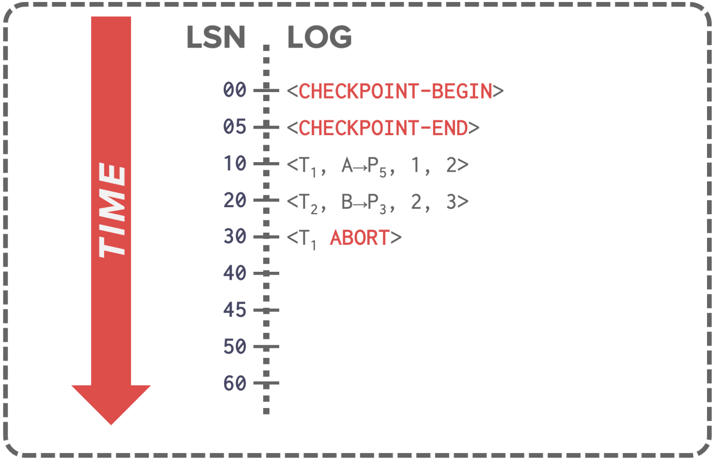
- Undo all txns that were active at the time of crash and therefore will never commit.
    - These are all the txns with **U** status in the ATT after the Analysis Phase.
- Process them in **reverse** LSN order using the **lastLSN** to speed up traversal.
- Write a **CLR** for every modification.

Undo Phase

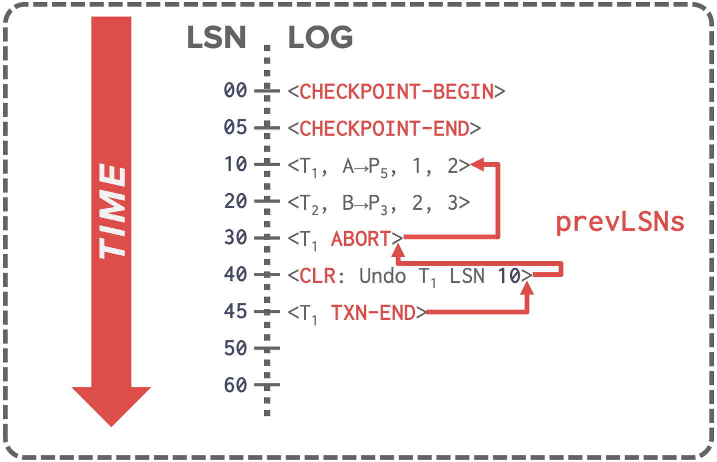
- ToUndo= <u>**lastLSN**</u> of "loser" txns
- Repeat until ToUndo is empty:
    - Pop largest LSN from ToUndo.
    - If this LSN is a CLR and <u>**undoNext**</u> = nil, then write an <u>**TXN-END**</u> record for this txn.
    - If this LSN is a CLR, and <u>**undoNext**</u> != nil, then add <u>**undoNext**</u> to ToUndo
    - Else this LSN is an update. Undo the update, write a CLR, add prevLSN to ToUndo.

Full Example

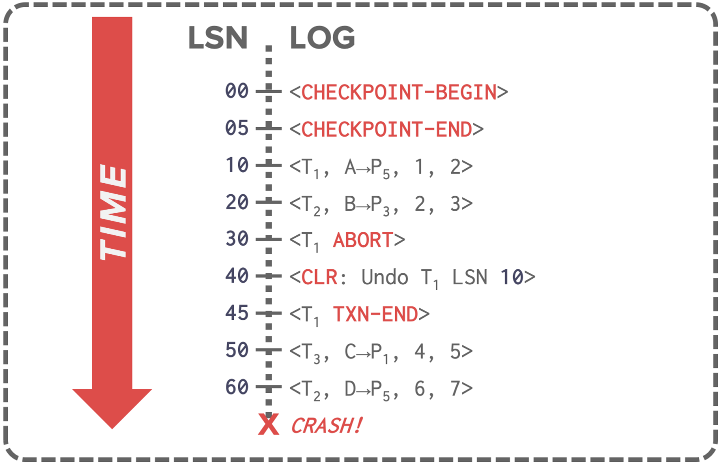
Full Example



Full Example

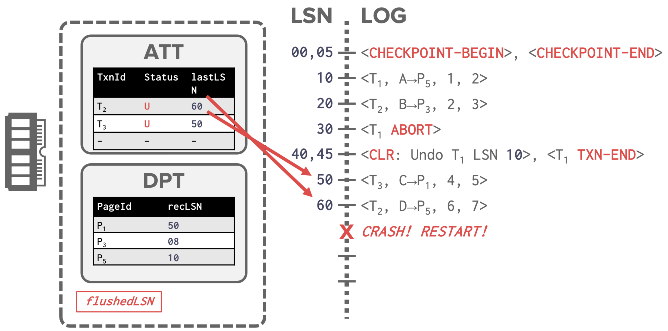


Full Example



```
00  <CHECKPOINT-BEGIN>
05  <CHECKPOINT-END>
10  <T1, A→P5, 1, 2>
20  <T2, B→P3, 2, 3>
30  <T1 ABORT>
40  <CLR: Undo T1 LSN 10>
45  <T1 TXN-END>
50  <T3, C→P1, 4, 5>
60  <T2, D→P5, 6, 7>
X CRASH!
```

Full Example



ATT

TxnId	Status	lastLSN
T_2	U	60
T_3	U	50
-	-	-

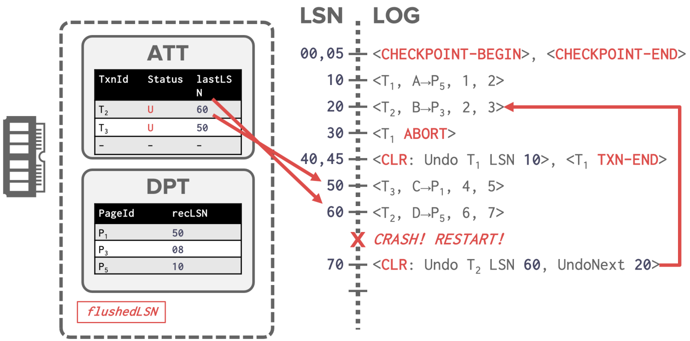
DPT

PageId	recLSN
P_1	50
P_3	08
P_5	10

flushedLSN

LSN	LOG
00,05	<CHECKPOINT-BEGIN>, <CHECKPOINT-END>
10	<T_1, $A \rightarrow P_5$, 1, 2>
20	<T_2, $B \rightarrow P_3$, 2, 3>
30	<T_1 ABORT>
40,45	<CLR: Undo T_1 LSN 10>, <T_1 TXN-END>
50	<T_3, $C \rightarrow P_1$, 4, 5>
60	<T_2, $D \rightarrow P_5$, 6, 7>
X	*CRASH! RESTART!*

Full Example



ATT

TxnId	Status	lastLSN
T_2	U	60
T_3	U	50
-	-	-

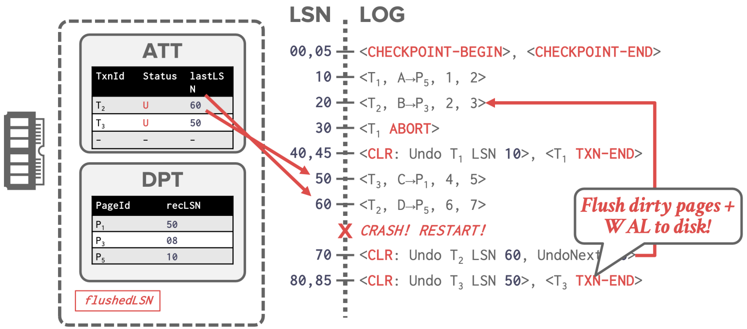
DPT

PageId	recLSN
P_1	50
P_3	08
P_5	10

flushedLSN

LSN	LOG
00,05	<CHECKPOINT-BEGIN>, <CHECKPOINT-END>
10	<T_1, A→P_5, 1, 2>
20	<T_2, B→P_3, 2, 3>
30	<T_1 ABORT>
40,45	<CLR: Undo T_1 LSN 10>, <T_1 TXN-END>
50	<T_3, C→P_1, 4, 5>
60	<T_2, D→P_5, 6, 7>
X	*CRASH! RESTART!*
70	<CLR: Undo T_2 LSN 60, UndoNext 20>

Full Example



ATT

TxnId	Status	lastLSN
T_2	U	60
T_3	U	50
-	-	-

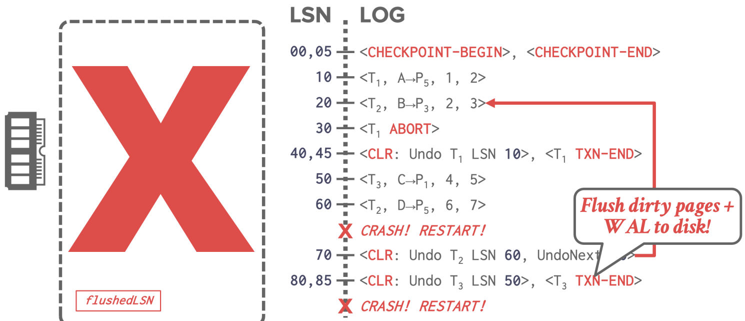
DPT

PageId	recLSN
P_1	50
P_3	08
P_5	10

flushedLSN

```
LSN     LOG
00,05   <CHECKPOINT-BEGIN>, <CHECKPOINT-END>
10      <T1, A→P5, 1, 2>
20      <T2, B→P3, 2, 3>
30      <T1 ABORT>
40,45   <CLR: Undo T1 LSN 10>, <T1 TXN-END>
50      <T3, C→P1, 4, 5>
60      <T2, D→P5, 6, 7>
   X    CRASH! RESTART!
70      <CLR: Undo T2 LSN 60, UndoNext
80,85   <CLR: Undo T3 LSN 50>, <T3 TXN-END>
```

Full Example



LSN | LOG

00,05 <CHECKPOINT-BEGIN>, <CHECKPOINT-END>

10 <T_1, A→P_5, 1, 2>

20 <T_2, B→P_3, 2, 3>

30 <T_1 ABORT>

40,45 <CLR: Undo T_1 LSN 10>, <T_1 TXN-END>

50 <T_3, C→P_1, 4, 5>

60 <T_2, D→P_5, 6, 7>

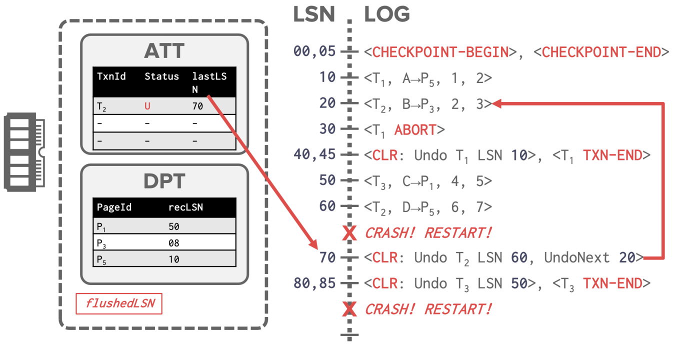
X *CRASH! RESTART!*

70 <CLR: Undo T_2 LSN 60, UndoNext

80,85 <CLR: Undo T_3 LSN 50>, <T_3 TXN-END>

X *CRASH! RESTART!*

Full Example



ATT

TxnId	Status	lastLSN
T_2	U	70
-	-	-
-	-	-

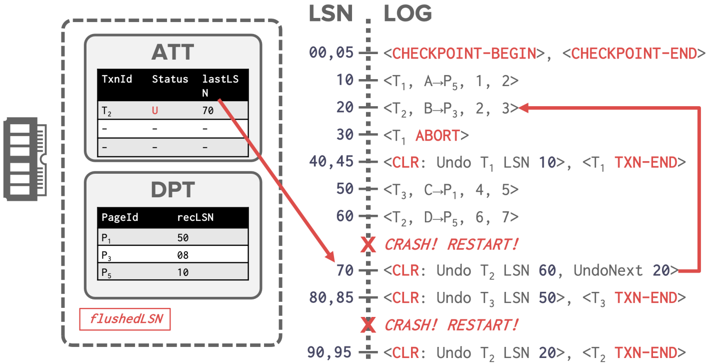
DPT

PageId	recLSN
P_1	50
P_3	08
P_5	10

flushedLSN

LSN	LOG
00,05	<CHECKPOINT-BEGIN>, <CHECKPOINT-END>
10	<T_1, A→P_5, 1, 2>
20	<T_2, B→P_3, 2, 3>
30	<T_1 ABORT>
40,45	<CLR: Undo T_1 LSN 10>, <T_1 TXN-END>
50	<T_3, C→P_1, 4, 5>
60	<T_2, D→P_5, 6, 7>
	CRASH! RESTART!
70	<CLR: Undo T_2 LSN 60, UndoNext 20>
80,85	<CLR: Undo T_3 LSN 50>, <T_3 TXN-END>
	CRASH! RESTART!

Full Example



ATT

TxnId	Status	lastLSN
T_2	U	70
-	-	-
-	-	-

DPT

PageId	recLSN
P_1	50
P_3	08
P_5	10

flushedLSN

LSN	LOG
00,05	<CHECKPOINT-BEGIN>, <CHECKPOINT-END>
10	<T_1, A→P_5, 1, 2>
20	<T_2, B→P_3, 2, 3>
30	<T_1 ABORT>
40,45	<CLR: Undo T_1 LSN 10>, <T_1 TXN-END>
50	<T_3, C→P_1, 4, 5>
60	<T_2, D→P_5, 6, 7>
X	*CRASH! RESTART!*
70	<CLR: Undo T_2 LSN 60, UndoNext 20>
80,85	<CLR: Undo T_3 LSN 50>, <T_3 TXN-END>
X	*CRASH! RESTART!*
90,95	<CLR: Undo T_2 LSN 20>, <T_2 TXN-END>

Additional Crash Issues

Additional Crash Issues (1)

- What does the DBMS do if it crashes during recovery in the Analysis Phase?
- What does the DBMS do if it crashes during recovery in the Redo Phase?

Additional Crash Issues (1)

- What does the DBMS do if it crashes during recovery in the Analysis Phase?
  - Nothing. Just run recovery again.
- What does the DBMS do if it crashes during recovery in the Redo Phase?
  - Again nothing. Redo everything again.

Additional Crash Issues (2)

- How can the DBMS improve performance during recovery in the Redo Phase?
- How can the DBMS improve performance during recovery in the Undo Phase?

Additional Crash Issues (2)

- How can the DBMS improve performance during recovery in the Redo Phase?
  - Assume that it is not going to crash again and flush all changes to disk asynchronously in the background.
- How can the DBMS improve performance during recovery in the Undo Phase?
  - Lazily rollback changes before new txns access pages.
  - Rewrite the application to avoid long-running txns.

Conclusion

Parting Thoughts

- Mains ideas of ARIES:
  - WAL with STEAL/NO-FORCE
  - Fuzzy Checkpoints (snapshot of dirty page ids)
  - Redo everything since the earliest dirty page
  - Undo txns that never commit
  - Write CLRs when undoing, to survive failures during restarts

Next Class

- Deconstruct ARIES