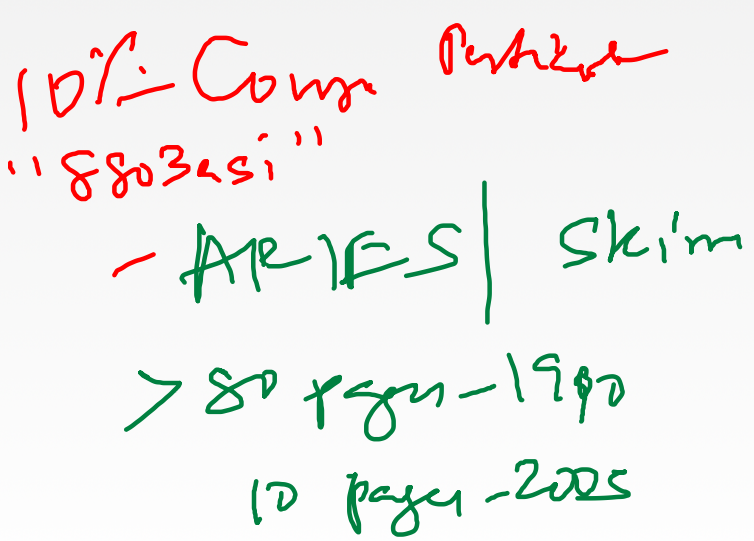
Georgia Tech

Lecture 8: Recovery (Part 2)

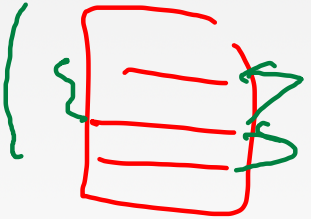
CREATING THE NEXT®

Today's Agenda



Recap

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

eager vs lazy

- 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.

dirty pgs
no need to flush (sync)

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.

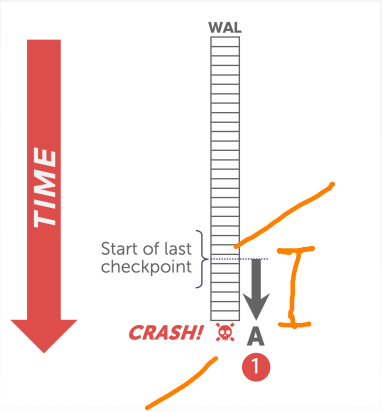
Phases of ARIES

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.

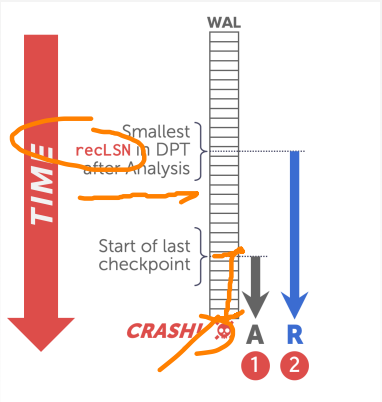
ARIES – Overview

- 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.



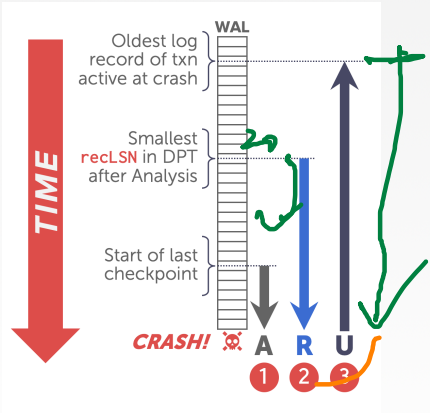
ARIES – Overview

- 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.



Analysis Phase

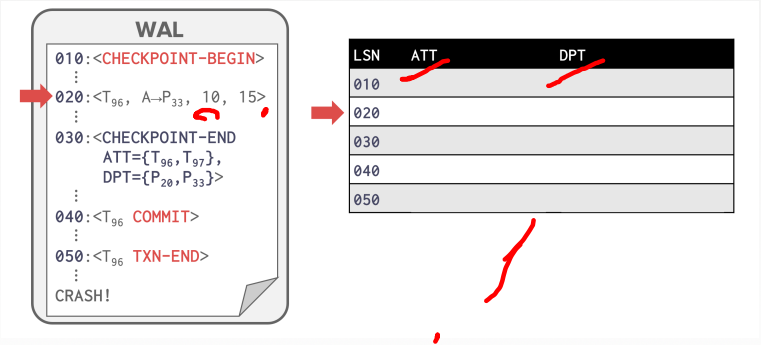
Analysis Phase

- 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

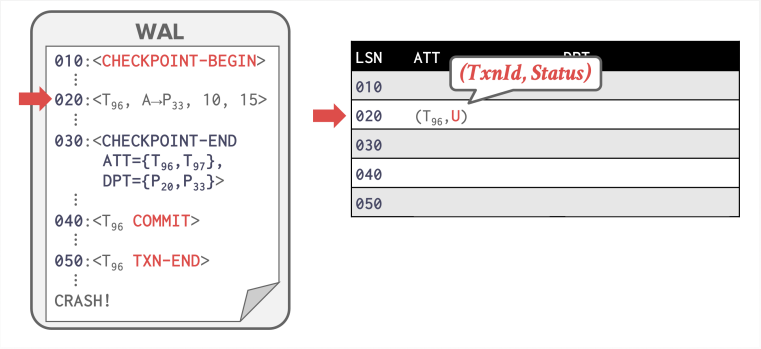
Analysis Phase

- 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.

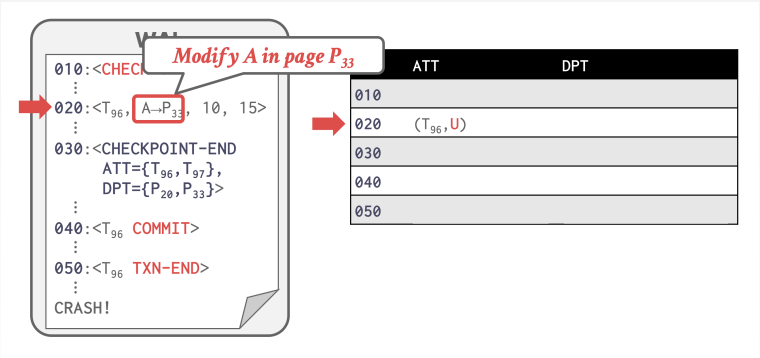
Analysis Phase: Example



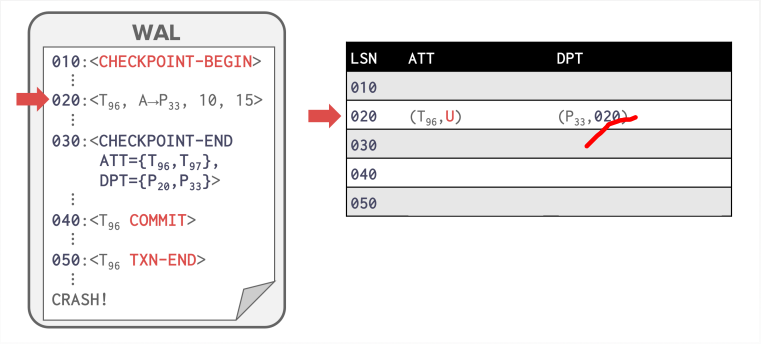
Analysis Phase: Example



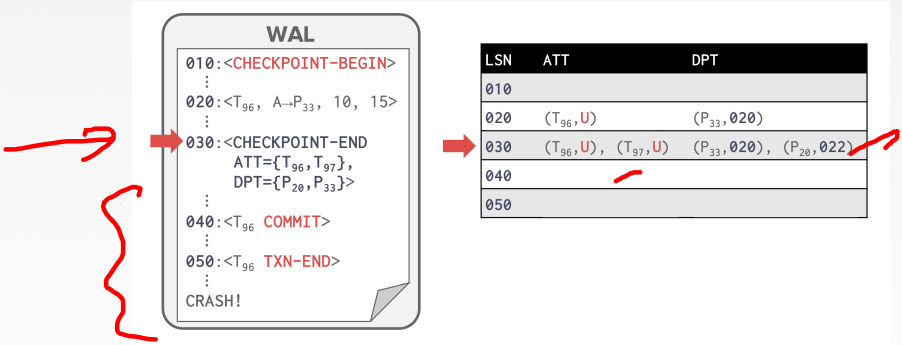
Analysis Phase: Example



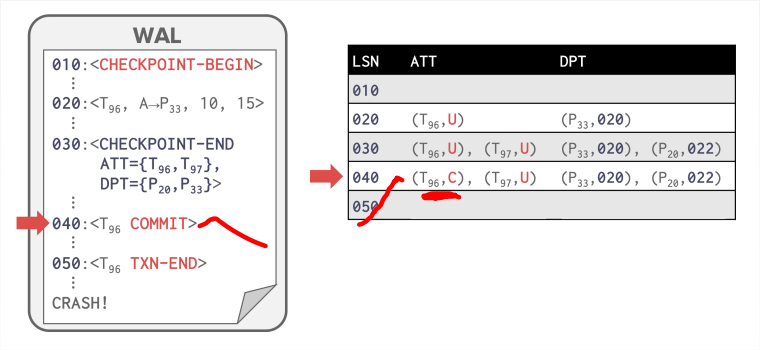
Analysis Phase: Example



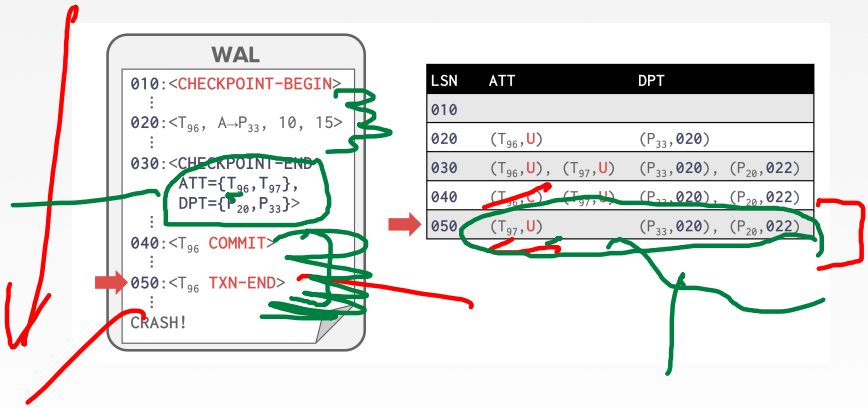
Analysis Phase: Example



Analysis Phase: Example



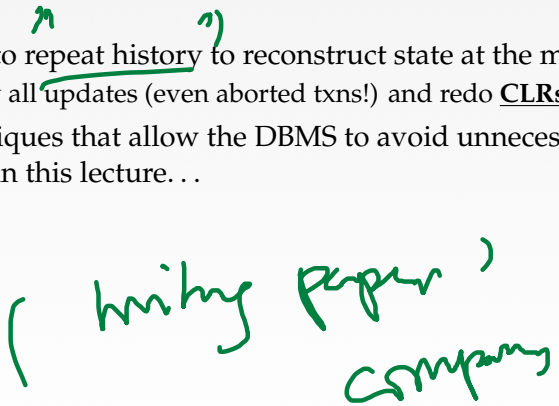
Analysis Phase: Example



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**

Subtle

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

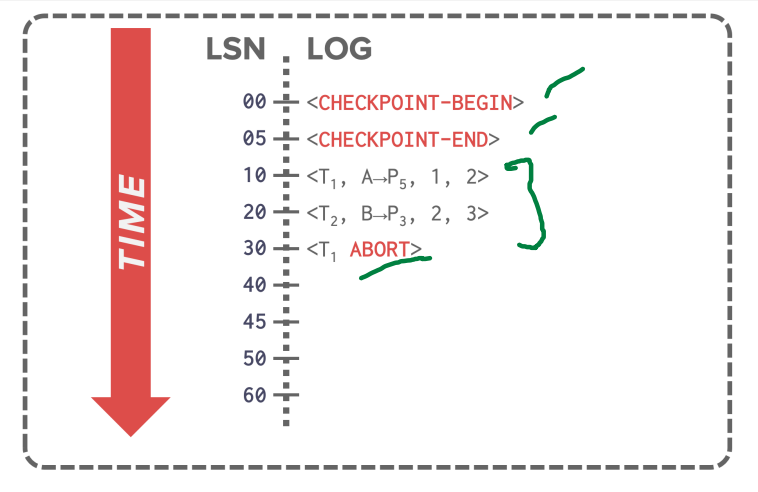
- Undo all txns that were active at the time of crash and therefore will never commit.
  - These are all the txns with **U** status in the ATT after the Analysis Phase.
- Process them in **reverse** LSN order using the **lastLSN** to speed up traversal.
- Write a **CLR** for every modification.

Undo Phase

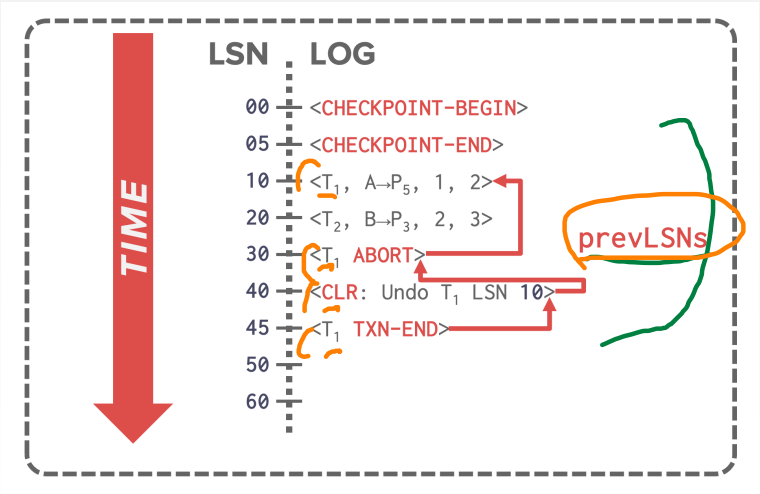
- ToUndo= **lastLSN** of "loser" txns
- Repeat until ToUndo is empty:
  - Pop largest LSN from ToUndo.
  - If this LSN is a CLR and **undoNext** = nil, then write an **TXN-END** record for this txn.
  - If this LSN is a CLR, and **undoNext** != nil, then add **undoNext** to ToUndo
  - Else this LSN is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

Full Example

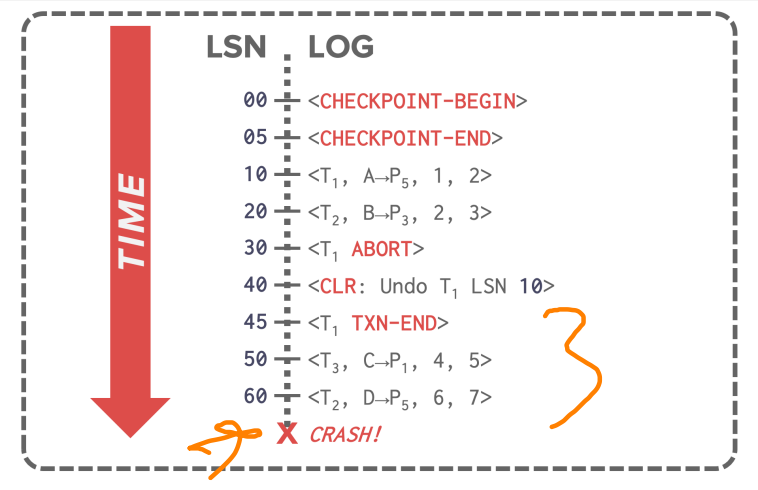
Full Example



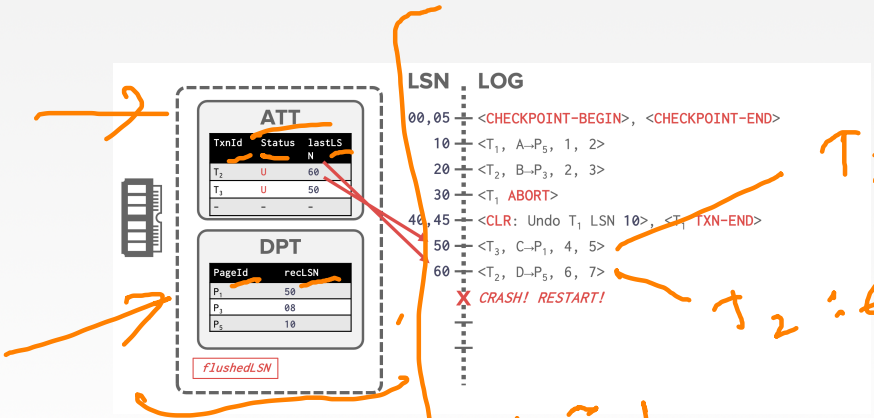
Full Example



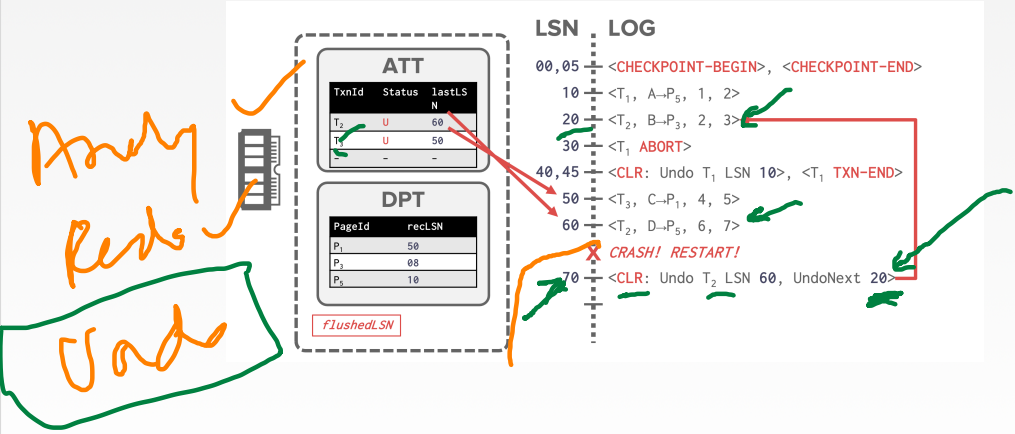
Full Example



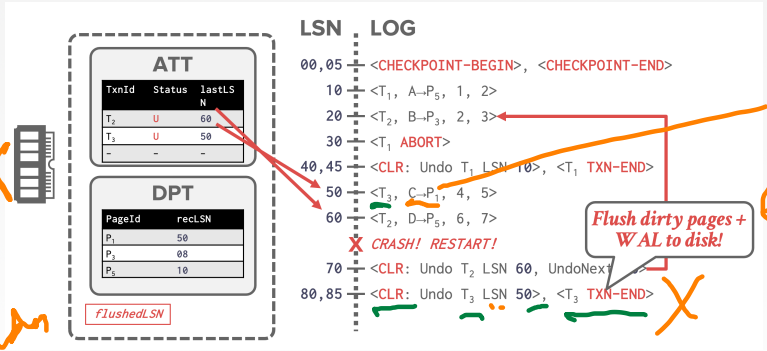
Full Example



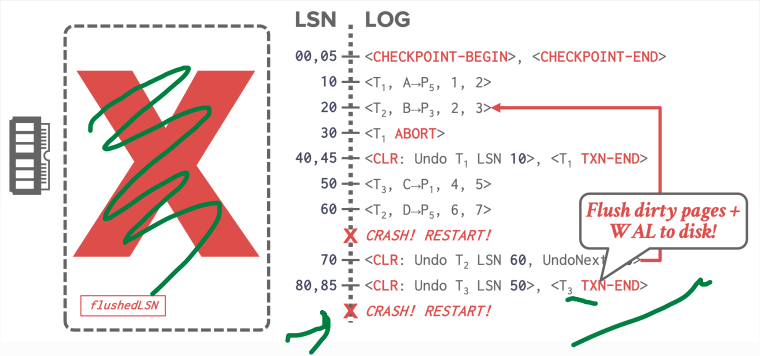
Full Example



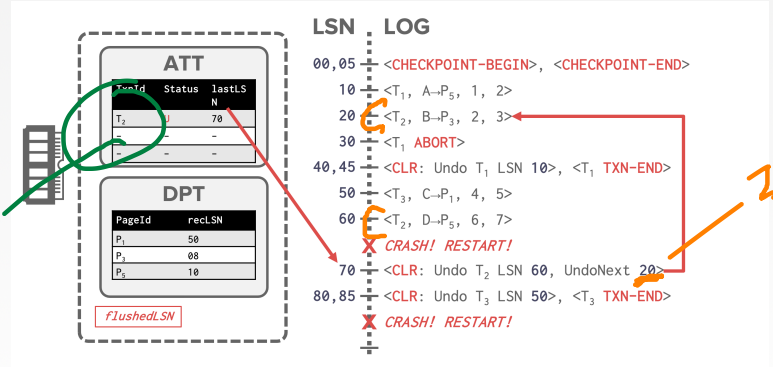
Full Example



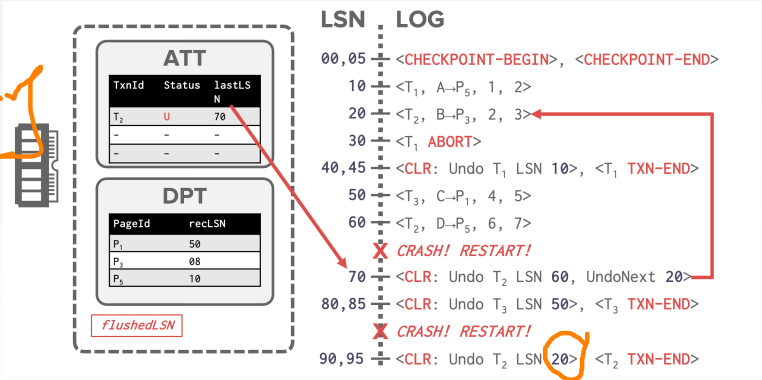
Full Example



Full Example



Full Example



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)

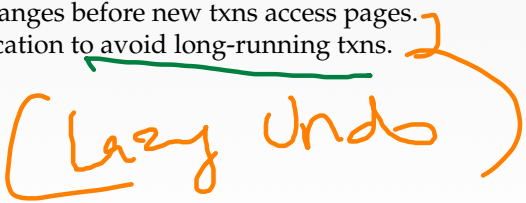
- What does the DBMS do if it crashes during recovery in the Analysis Phase?
  - Nothing. Just run recovery again.
- What does the DBMS do if it crashes during recovery in the Redo Phase?
  - Again nothing. Redo everything again.

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