UPCOMING DATABASE TALKS

Confluent ksqlDB (Kafka) → Monday Nov 23rd @ 5pm ET

Microsoft SQL Server Optimizer → Monday Nov 30th @ 5pm ET

Snowflake Lecture → Monday Dec 7th @ 3:20pm ET
CRASH RECOVERY

Recovery algorithms are techniques to ensure database consistency, transaction atomicity, and durability despite failures.

Recovery algorithms have two parts:
→ Actions during normal txn processing to ensure that the DBMS can recover from a failure.
→ Actions after a failure to recover the database to a state that ensures atomicity, consistency, and durability.
Algorithms for Recovery and Isolation Exploiting Semantics

Developed at IBM Research in early 1990s for the DB2 DBMS.

Not all systems implement ARIES exactly as defined in this paper but they're close enough.
ARIES – MAIN IDEAS

Write-Ahead Logging:
→ Any change is recorded in log on stable storage before the database change is written to disk.
→ Must use STEAL + NO-FORCE buffer pool policies.

Repeating History During Redo:
→ On restart, retrace actions and restore database to exact state before crash.

Logging Changes During Undo:
→ Record undo actions to log to ensure action is not repeated in the event of repeated failures.
TODAY’S AGENDA

Log Sequence Numbers
Normal Commit & Abort Operations
Fuzzy Checkpointing
Recovery Algorithm
We need to extend our log record format from last class to include additional info.

Every log record now includes a globally unique log sequence number (LSN).

Various components in the system keep track of LSNs that pertain to them...
## LOG SEQUENCE NUMBERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Where</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>flushedLSN</td>
<td>Memory</td>
<td>Last LSN in log on disk</td>
</tr>
<tr>
<td>pageLSN</td>
<td>page__x</td>
<td>Newest update to page__x</td>
</tr>
<tr>
<td>recLSN</td>
<td>page__x</td>
<td>Oldest update to page__x since it was last flushed</td>
</tr>
<tr>
<td>lastLSN</td>
<td>T_i</td>
<td>Latest record of txn T_i</td>
</tr>
<tr>
<td>MasterRecord</td>
<td>Disk</td>
<td>LSN of latest checkpoint</td>
</tr>
</tbody>
</table>
WRITING LOG RECORDS

Each data page contains a pageLSN.
→ The LSN of the most recent update to that page.

System keeps track of flushedLSN.
→ The max LSN flushed so far.

Before page x can be written to disk, we must flush log at least to the point where:
→ pageLSN$_x$ ≤ flushedLSN
WRITING LOG RECORDS

Log Sequence Numbers

<table>
<thead>
<tr>
<th>No</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>017</td>
<td>(&lt;T_5) BEGIN</td>
</tr>
<tr>
<td>018</td>
<td>(&lt;T_5, A, 9, 8&gt;</td>
</tr>
<tr>
<td>019</td>
<td>(&lt;T_5, B, 5, 1&gt;</td>
</tr>
<tr>
<td>020</td>
<td>(&lt;T_5) COMMIT</td>
</tr>
</tbody>
</table>

Buffer Pool

- pageLSN: A=9  B=5  C=2
- flushable: A=9  B=5  C=2

Log Sequence Numbers

<table>
<thead>
<tr>
<th>No</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>(&lt;T_1) BEGIN</td>
</tr>
<tr>
<td>002</td>
<td>(&lt;T_1, A, 1, 2&gt;</td>
</tr>
<tr>
<td>003</td>
<td>(&lt;T_1) COMMIT</td>
</tr>
<tr>
<td>004</td>
<td>(&lt;T_2) BEGIN</td>
</tr>
<tr>
<td>005</td>
<td>(&lt;T_2, A, 2, 3&gt;</td>
</tr>
<tr>
<td>006</td>
<td>(&lt;T_1) BEGIN</td>
</tr>
<tr>
<td>007</td>
<td>(&lt;OPENPOINT&gt;</td>
</tr>
<tr>
<td>008</td>
<td>(&lt;T_2) COMMIT</td>
</tr>
<tr>
<td>009</td>
<td>(&lt;T_3, A, 3, 4&gt;</td>
</tr>
<tr>
<td>010</td>
<td>(&lt;T_4) BEGIN</td>
</tr>
<tr>
<td>011</td>
<td>(&lt;T_4, X, 5, 6&gt;</td>
</tr>
<tr>
<td>012</td>
<td>(&lt;T_4, Y, 9, 7&gt;</td>
</tr>
<tr>
<td>013</td>
<td>(&lt;T_3, B, 4, 2&gt;</td>
</tr>
<tr>
<td>014</td>
<td>(&lt;T_3) COMMIT</td>
</tr>
<tr>
<td>015</td>
<td>(&lt;T_4, B, 2, 3&gt;</td>
</tr>
<tr>
<td>016</td>
<td>(&lt;T_4, C, 1, 2&gt;</td>
</tr>
</tbody>
</table>

Master Record

- pageLSN: A=9  B=5  C=2
- flushable: A=9  B=5  C=2

Database
WRITING LOG RECORDS

WAL (Tail)

017: <T5 BEGIN>
018: <T5, A, 9, 8>
019: <T5, B, 5, 1>
020: <T5 COMMIT>

 Buffer Pool

pageLSN recLSN

Not safe to unpin because pageLSN > flushedLSN

WAL

001: <T1 BEGIN>
002: <T1, A, 1, 2>
003: <T1 COMMIT>
004: <T2 BEGIN>
005: <T2, A, 2, 3>
006: <T3 BEGIN>
007: <CHECKPOINT>
008: <T2 COMMIT>
009: <T3, A, 3, 4>
010: <T4 BEGIN>
011: <T4, X, 5, 6>
012: <T4, Y, 9, 7>
013: <T3, B, 4, 2>
014: <T3 COMMIT>
015: <T4, B, 2, 3>
016: <T4, C, 1, 2>

Database

pageLSN recLSN

A=9 B=5 C=2

MasterRecord

flushedLSN

pageLSN recLSN

A=9 B=5 C=2

MasterRecord
WRITING LOG RECORDS

All log records have an $LSN$. 

Update the $\text{pageLSN}$ every time a txn modifies a record in the page.

Update the $\text{flushedLSN}$ in memory every time the DBMS writes out the WAL buffer to disk.
NORMAL EXECUTION

Each txn invokes a sequence of reads and writes, followed by commit or abort.

Assumptions in this lecture:
→ All log records fit within a single page.
→ Disk writes are atomic.
→ Single-versioned tuples with Strict 2PL.
→ **STEAL + NO-FORCE** buffer management with WAL.
Write **COMMIT** record to log.

All log records up to txn's **COMMIT** record are flushed to disk.
→ Log flushes are sequential, synchronous writes to disk.
→ Many log records per log page.

When the commit succeeds, write a special **TXN-END** record to log.
→ This does not need to be flushed immediately.
WAL (Tail)

012: <T₄ BEGIN>
013: <T₄, A, 9, 8>
014: <T₄, B, 5, 1>
015: <T₄ COMMIT>

Buffer Pool

pageLSN  recLSN
A=9  B=5  C=2

flushedLSN

WAL

001: <T₁ BEGIN>
002: <T₁, A, 1, 2>
003: <T₁ COMMIT>
004: <T₂ BEGIN>
005: <T₂, A, 2, 3>
006: <T₁ BEGIN>
007: <CHECKPOINT>
008: <T₂ COMMIT>
009: <T₃, A, 3, 4>
010: <T₃, B, 4, 2>
011: <T₃ COMMIT>
012: <T₂ BEGIN>
013: <T₄, A, 9, 8>
014: <T₄, B, 5, 1>
015: <T₄ COMMIT>

Database

MasterRecord

pageLSN  recLSN
A=9  B=5  C=2

Transaction Commit
TRANSACTION COMMIT

WAL (Tail)

012: <T₄ BEGIN>
013: <T₄, A, 9, 8>
014: <T₄, B, 5, 1>
015: <T₄ COMMIT>

Buffer Pool

flushedLSN

A=9 B=5 C=2

WAL

001: <T₁ BEGIN>
002: <T₁, A, 1, 2>
003: <T₁ COMMIT>
004: <T₂ BEGIN>
005: <T₂, A, 2, 3>
006: <T₂ COMMIT>
007: <CHECKPOINT>
008: <T₃ COMMIT>
009: <T₃, A, 3, 4>
010: <T₃, B, 4, 2>
011: <T₃ COMMIT>
012: <T₄ BEGIN>
013: <T₄, A, 9, 8>
014: <T₄, B, 5, 1>
015: <T₄ COMMIT>

flushedLSN = 015

Database

MasterRecord

pageLSN recLSN

A=9 B=5 C=2
We can trim the in-memory log up to flushedLSN

```
012:<T4 BEGIN>
013:<T4, A, 9, 8>
014:<T4, B, 5, 1>
015:<T4 COMMIT>
...  
099:<T4 TXN- END>
```
We can trim the in-memory log up to flushedLSN.
Abort atxn is a special case of the ARIES undo operation applied to only one txnn.

We need to add another field to our log records:
- *prevLSN*: The previous *LSN* for the txnn.
- This maintains a linked-list for each txnn that makes it easy to walk through its records.
TRANSACTION ABORT

Buffer Pool:
- pageLSN: A=9, B=5, C=2
- recLSN: flushedLSN

WAL (Tail):
- LSN | prevLSN
- 012 | nil
- 013 | 012
- 014 | 013

WAL:
- pageLSN: A=9, B=5, C=2
- recLSN: MasterRecord

Transaction:
- <T4, BEGIN>
- <T4, A, 9, 8>
- <T4, B, 5, 1>

Database:
- LSN | prevLSN
- nil
TRANSACTION ABORT

WAL (Tail)

012|nil:<T₄ BEGIN>
013|012:<T₄, A, 9, 8>
014|013:<T₄, B, 5, 1>
015|014:<T₄ ABORT>
???
099|098:<T₄ TXN-END>

Buffer Pool

pageLSN recLSN
A=9 B=5 C=2
flushedLSN

WAL

pageLSN recLSN
A=9 B=5 C=2

MasterRecord

Database
Important: Need to record what steps we took to undo thetxn.
COMPENSATION LOG RECORDS

A **CLR** describes the actions taken to undo the actions of a previous update record.

It has all the fields of an update log record plus the **undoNext** pointer (the next-to-be-undone LSN).

**CLRs** are added to log other records but the DBMS does not wait for them to be flushed before notifying the application that the txn aborted.
## TRANSACTION ABORT – CLR EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>TxnId</th>
<th>Type</th>
<th>Object</th>
<th>Before</th>
<th>After</th>
<th>UndoNext</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>T₁</td>
<td>BEGIN</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>T₁</td>
<td>UPDATE</td>
<td>A</td>
<td>30</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## TRANSACTION ABORT – CLR EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>TxnId</th>
<th>Type</th>
<th>Object</th>
<th>Before</th>
<th>After</th>
<th>Undo</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>T₁</td>
<td>BEGIN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>T₁</td>
<td>UPDATE</td>
<td>A</td>
<td>30</td>
<td>40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>011</td>
<td>T₁</td>
<td>CLR-002</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
<td></td>
</tr>
</tbody>
</table>

TIME
### TRANSACTION ABORT – CLR EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>TxnId</th>
<th>Type</th>
<th>Object</th>
<th>Before</th>
<th>After</th>
<th>UndoNext</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>T₁</td>
<td>BEGIN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>T₁</td>
<td>UPDATE</td>
<td>A</td>
<td>30</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>011</td>
<td>T₁</td>
<td>CLR-002</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
</tr>
</tbody>
</table>

**TIME**
### TRANSACTION ABORT – CLR EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>TxnId</th>
<th>Type</th>
<th>Object</th>
<th>Before</th>
<th>After</th>
<th>UndoNext</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>T₁</td>
<td>BEGIN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>T₁</td>
<td>UPDATE</td>
<td>A</td>
<td>30</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>011</td>
<td>T₁</td>
<td>CLR-002</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
</tr>
</tbody>
</table>

TIME
## TRANSACTION ABORT – CLR EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>TxnId</th>
<th>Type</th>
<th>Object</th>
<th>Before</th>
<th>After</th>
<th>UndoNext</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>T₁</td>
<td>BEGIN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>T₁</td>
<td>UPDATE</td>
<td>A</td>
<td>30</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>011</td>
<td>T₁</td>
<td>CLR-002</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
</tr>
</tbody>
</table>

The LSN of the next log record to be undone.
### TRANSACTION ABORT – CLR EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>TxnId</th>
<th>Type</th>
<th>Object</th>
<th>Before</th>
<th>After</th>
<th>UndoNext</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>T₁</td>
<td>BEGIN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>T₁</td>
<td>UPDATE</td>
<td>A</td>
<td>30</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>011</td>
<td>T₁</td>
<td>CLR-002</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
</tr>
<tr>
<td>027</td>
<td>026</td>
<td>T₁</td>
<td>TXN-END</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>nil</td>
</tr>
</tbody>
</table>
ABORT ALGORITHM

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.
At end, write a **TXN-END** log record.

Notice: **CLRs** never need to be undone.
TODAY’S AGENDA

- Log-Sequence Numbers
- Normal Commit & Abort Operations
- Fuzzy Checkpointing
- Recovery Algorithm
NON-FUZZY CHECKPOINTS

The DBMS halts everything when it takes a checkpoint to ensure a consistent snapshot:
→ Halt the start of any new txns.
→ Wait until all active txns finish executing.
→ Flushes dirty pages on disk.

This is bad for runtime performance but makes recovery trivially easy.
SLIGHTLY BETTER CHECKPOINTS

Pause modifying txns while the DBMS takes the checkpoint.
→ Prevent queries from acquiring write latch on table/index pages.
→ Don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause modifying txns while the DBMS takes the checkpoint.
→ Prevent queries from acquiring write latch on table/index pages.
→ Don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause modifying txns while the DBMS takes the checkpoint.

→ Prevent queries from acquiring write latch on table/index pages.

→ Don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause modifying txns while the DBMS takes the checkpoint.
→ Prevent queries from acquiring write latch on table/index pages.
→ Don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause modifying txns while the DBMS takes the checkpoint.
→ Prevent queries from acquiring write latch on table/index pages.
→ Don't have to wait until all txns finish before taking the checkpoint.

We must record internal state as of the beginning of the checkpoint.
→ Active Transaction Table (ATT)
→ Dirty Page Table (DPT)
ACTIVE TRANSACTION TABLE

One entry per currently active txn.

→ **txnId**: Unique txn identifier.
→ **status**: The current "mode" of the txn.
→ **lastLSN**: Most recent *LSN* created by txn.

Entry removed after the txn commits or aborts.

Txn Status Codes:

→ **R** → Running
→ **C** → Committing
→ **U** → Candidate for Undo
DIRTY PAGE TABLE

Keep track of which pages in the buffer pool contain changes from uncommitted transactions.

One entry per dirty page in the buffer pool:

→ \texttt{recLSN}: The \texttt{LSN} of the log record that first caused the page to be dirty.
At the first checkpoint, $T_2$ is still running and there are two dirty pages ($P_{11}, P_{22}$).
**SLIGHTLY BETTER CHECKPOINTS**

At the first checkpoint, $T_2$ is still running and there are two dirty pages ($P_{11}, P_{22}$).

At the second checkpoint, $T_3$ is active and there are two dirty pages ($P_{11}, P_{33}$).

This still is not ideal because the DBMS must stall txns during checkpoint...
Fuzzy Checkpoints

A fuzzy checkpoint is where the DBMS allows active txns to continue the run while the system flushes dirty pages to disk.

New log records to track checkpoint boundaries:
→ CHECKPOINT-BEGIN: Indicates start of checkpoint
→ CHECKPOINT-END: Contains ATT + DPT.
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.
ARIES – RECOVERY PHASES

Phase #1 – Analysis
→ Read WAL from last **CHECKPOINT-END** 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.
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**.
ANALYSIS PHASE

At end of the Analysis Phase:
→ **ATT** identifies which txns were active at time of crash.
→ **DPT** identifies which dirty pages might not have made it to disk.
ANALYSIS PHASE EXAMPLE

WAL

010:<CHECKPOINT-BEGIN>
   ...
020:<T<sub>96</sub>, A→P<sub>33</sub>, 10, 15>
   ...
030:<CHECKPOINT-END
    ATT={T<sub>96</sub>, T<sub>97</sub>},
    DPT={P<sub>20</sub>, P<sub>33</sub>}>
   ...
040:<T<sub>96</sub> COMMIT>
   ...
050:<T<sub>96</sub> TXN-END>
   ...
CRASH!

<table>
<thead>
<tr>
<th>LSN</th>
<th>ATT</th>
<th>DPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>050</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 | (T₉₆, U) |  |
030 |  |  |
040 |  |  |
050 |  |  |

(TxnId, Status)
ANALYSIS PHASE EXAMPLE

Modify A in page $P_{33}$

010: <CHECKPOINT-

020: $<T_{96}, A \rightarrow P_{33}, 10, 15>$

030: <CHECKPOINT-END

ATT={$T_{96}, T_{97}$},

DPT={$P_{20}, P_{33}$}>

040: $<T_{96}$ COMMIT>

050: $<T_{96}$ TXN-END>

CRASH!
ANALYSIS PHASE EXAMPLE

WAL

\[010: \text{<CHECKPOINT-BEGIN>}
\]
\[020: \text{<T}_{96}, A \rightarrow P_{33}, 10, 15>\]
\[030: \text{<CHECKPOINT-END}
\]
\[\text{ATT} = \{T_{96}, T_{97}\},
\text{DPT} = \{P_{20}, P_{33}\}\]
\[040: \text{<T}_{96} \text{ COMMIT>}
\]
\[050: \text{<T}_{96} \text{ TXN-END>}
\]

CRASH!

<table>
<thead>
<tr>
<th>LSN</th>
<th>ATT</th>
<th>DPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>(T_{96}, U)</td>
<td>(P_{33}, 020)</td>
</tr>
<tr>
<td>030</td>
<td>(T_{96}, U), (T_{97}, U)</td>
<td>(P_{33}, 020), (P_{20}, 022)</td>
</tr>
<tr>
<td>040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>050</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**ANALYSIS PHASE EXAMPLE**

**WAL**

```
010: <CHECKPOINT-BEGIN>

020: <T_{96}, A \rightarrow 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**

<table>
<thead>
<tr>
<th>LSN</th>
<th>ATT</th>
<th>DPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>(T_{96}, U)</td>
<td>(P_{33}, 020)</td>
</tr>
<tr>
<td>030</td>
<td>(T_{96}, U), (T_{97}, U)</td>
<td>(P_{33}, 020), (P_{20}, 022)</td>
</tr>
<tr>
<td>040</td>
<td>(T_{96}, C), (T_{97}, U)</td>
<td>(P_{33}, 020), (P_{20}, 022)</td>
</tr>
<tr>
<td>050</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ANALYSIS PHASE EXAMPLE

WAL

010: <CHECKPOINT-BEGIN>
   ...
020: <T_{96}, A\rightarrow 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  (T_{97}, U)  (P_{33}, 020), (P_{20}, 022)
The goal is to repeat history to reconstruct state at the moment of the crash:
→ Reapply all updates (even aborted txns!) and redo **CLR**s.

There are 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 \text{recLSN} in \text{DPT}.

For each update log record or \text{CLR} with a given \text{LSN}, redo the action unless:
→ Affected page is not in \text{DPT}, or
→ Affected page is in \text{DPT} but that record's \text{LSN} is less than the page's \text{recLSN}. 
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.
FULL EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>&lt;CHECKPOINT-BEGIN&gt;</td>
</tr>
<tr>
<td>05</td>
<td>&lt;CHECKPOINT-END&gt;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;T₁, A→P₅, 1, 2&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;T₂, B→P₃, 2, 3&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;T₁ ABORT&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;CLR: Undo T₁ LSN 10&gt;</td>
</tr>
<tr>
<td>45</td>
<td>&lt;T₁ TXN-END&gt;</td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

prevLSNs

CMU-DB
15-445/645 (Fall 2020)
FULL EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>&lt;CHECKPOINT-BEGIN&gt;</td>
</tr>
<tr>
<td>05</td>
<td>&lt;CHECKPOINT-END&gt;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;T₁, A→P₅, 1, 2&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;T₂, B→P₃, 2, 3&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;T₁ ABORT&gt;</td>
</tr>
<tr>
<td>40</td>
<td>&lt;CLR: Undo T₁ LSN 10&gt;</td>
</tr>
<tr>
<td>45</td>
<td>&lt;T₁ TXN-END&gt;</td>
</tr>
<tr>
<td>50</td>
<td>&lt;T₃, C→P₁, 4, 5&gt;</td>
</tr>
<tr>
<td>60</td>
<td>&lt;T₂, D→P₅, 6, 7&gt;</td>
</tr>
</tbody>
</table>

X CRASH!
FULL EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>00,05</td>
<td>&lt;CHECKPOINT-BEGIN&gt;, &lt;CHECKPOINT-END&gt;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;T₁, A→P₅, 1, 2&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;T₂, B→P₃, 2, 3&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;T₁ ABORT&gt;</td>
</tr>
<tr>
<td>40,45</td>
<td>&lt;CLR: Undo T₁ LSN 10&gt;, &lt;T₁ TXN-END&gt;</td>
</tr>
<tr>
<td>50</td>
<td>&lt;T₃, C→P₁, 4, 5&gt;</td>
</tr>
<tr>
<td>60</td>
<td>&lt;T₂, D→P₅, 6, 7&gt;</td>
</tr>
</tbody>
</table>

CRASH! RESTART!

flushedLSN

ATT

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₂</td>
<td>U</td>
<td>60</td>
</tr>
<tr>
<td>T₃</td>
<td>U</td>
<td>50</td>
</tr>
</tbody>
</table>

DPT

<table>
<thead>
<tr>
<th>PageId</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>50</td>
</tr>
<tr>
<td>P₃</td>
<td>08</td>
</tr>
<tr>
<td>P₅</td>
<td>10</td>
</tr>
</tbody>
</table>
FULL EXAMPLE

**LSN**

- **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>
- **70**: <CLR: Undo T₂ LSN 60, UndoNext 20>

**LOG**

- ** CLR**: Undo T₁ LSN 10, <T₁ TXN-END>

**ATT**

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₂</td>
<td>U</td>
<td>60</td>
</tr>
<tr>
<td>T₃</td>
<td>U</td>
<td>50</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**DPT**

<table>
<thead>
<tr>
<th>PageId</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>50</td>
</tr>
<tr>
<td>P₃</td>
<td>08</td>
</tr>
<tr>
<td>P₅</td>
<td>10</td>
</tr>
</tbody>
</table>

flushedLSN

CRASH! RESTART!
**FULL EXAMPLE**

### Att

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₂</td>
<td>U</td>
<td>60</td>
</tr>
<tr>
<td>T₃</td>
<td>U</td>
<td>50</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### DPT

<table>
<thead>
<tr>
<th>PageId</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>50</td>
</tr>
<tr>
<td>P₃</td>
<td>08</td>
</tr>
<tr>
<td>P₅</td>
<td>10</td>
</tr>
</tbody>
</table>

### LSN

<table>
<thead>
<tr>
<th>LSN</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>00,05</td>
<td>&lt;CHECKPOINT-BEGIN&gt;, &lt;CHECKPOINT-END&gt;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;T₁, A→P₅, 1, 2&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;T₂, B→P₃, 2, 3&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;T₁ ABORT&gt;</td>
</tr>
<tr>
<td>40,45</td>
<td>&lt;CLR: Undo T₁ LSN 10&gt;, &lt;T₁ TXN-END&gt;</td>
</tr>
<tr>
<td>50</td>
<td>&lt;T₃, C→P₁, 4, 5&gt;</td>
</tr>
<tr>
<td>60</td>
<td>&lt;T₂, D→P₅, 6, 7&gt;</td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>CRASH! RESTART!</td>
</tr>
<tr>
<td>70</td>
<td>&lt;CLR: Undo T₂ LSN 60, UndoNext</td>
</tr>
<tr>
<td>80,85</td>
<td>&lt;CLR: Undo T₃ LSN 50&gt;, &lt;T₃ TXN-END&gt;</td>
</tr>
</tbody>
</table>

Flush dirty pages + WAL to disk!
FULL EXAMPLE

**ATT**

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₂</td>
<td>U</td>
<td>60</td>
</tr>
<tr>
<td>T₃</td>
<td>U</td>
<td>50</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**DPT**

<table>
<thead>
<tr>
<th>PageId</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>50</td>
</tr>
<tr>
<td>P₃</td>
<td>08</td>
</tr>
<tr>
<td>P₅</td>
<td>10</td>
</tr>
</tbody>
</table>

**LSN**

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>
70      →  <CLR:  Undo  T₂  LSN  60,  UndoNext>
80,85   →  <CLR:  Undo  T₃  LSN  50>,  <T₃  TXN-END>

CRASH! RESTART!

Flush dirty pages + WAL to disk!
FULL EXAMPLE

<table>
<thead>
<tr>
<th>LSN</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>00,05</td>
<td>&lt;CHECKPOINT-BEGIN&gt;, &lt;CHECKPOINT-END&gt;</td>
</tr>
<tr>
<td>10</td>
<td>&lt;T₁, A→P₅, 1, 2&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;T₂, B→P₃, 2, 3&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;T₁ ABORT&gt;</td>
</tr>
<tr>
<td>40,45</td>
<td>&lt;CLR: Undo T₁ LSN 10&gt;, &lt;T₁ TXN-END&gt;</td>
</tr>
<tr>
<td>50</td>
<td>&lt;T₃, C→P₁, 4, 5&gt;</td>
</tr>
<tr>
<td>60</td>
<td>&lt;T₂, D→P₅, 6, 7&gt;</td>
</tr>
<tr>
<td></td>
<td><strong>CRASH! RESTART!</strong></td>
</tr>
<tr>
<td>70</td>
<td>&lt;CLR: Undo T₂ LSN 60, UndoNext</td>
</tr>
<tr>
<td>80,85</td>
<td>&lt;CLR: Undo T₃ LSN 50&gt;, &lt;T₃ TXN-END&gt;</td>
</tr>
<tr>
<td></td>
<td><strong>CRASH! RESTART!</strong></td>
</tr>
</tbody>
</table>

Flush dirty pages + WAL to disk!
**FULL EXAMPLE**

**ATT**

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td>U</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DPT**

<table>
<thead>
<tr>
<th>PageId</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>50</td>
</tr>
<tr>
<td>P3</td>
<td>08</td>
</tr>
<tr>
<td>P5</td>
<td>10</td>
</tr>
</tbody>
</table>

**LSN**

- 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>

  **CRASH! RESTART!**

- 70: 
  - <CLR: Undo T₂ LSN 60, UndoNext 20>

- 80, 85: 
  - <CLR: Undo T₃ LSN 50>, <T₃ TXN-END>

  **CRASH! RESTART!**
**FULL EXAMPLE**

### LSN

<table>
<thead>
<tr>
<th>LastLSN</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>00,05</td>
<td><strong>&lt;CHECKPOINT-BEGIN&gt;</strong>, <strong>&lt;CHECKPOINT-END&gt;</strong></td>
</tr>
<tr>
<td>10</td>
<td><strong>&lt;T₁, A→P₅, 1, 2&gt;</strong></td>
</tr>
<tr>
<td>20</td>
<td><strong>&lt;T₂, B→P₃, 2, 3&gt;</strong></td>
</tr>
<tr>
<td>30</td>
<td><strong>&lt;T₁ ABORT&gt;</strong></td>
</tr>
<tr>
<td>40,45</td>
<td><strong>&lt;CLR: Undo T₁ LSN 10&gt;</strong>, <strong>&lt;T₁ TXN-END&gt;</strong></td>
</tr>
<tr>
<td>50</td>
<td><strong>&lt;T₃, C→P₁, 4, 5&gt;</strong></td>
</tr>
<tr>
<td>60</td>
<td><strong>&lt;T₂, D→P₅, 6, 7&gt;</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X CRASH! RESTART!</strong></td>
</tr>
<tr>
<td>70</td>
<td><strong>&lt;CLR: Undo T₂ LSN 60, UndoNext 20&gt;</strong></td>
</tr>
<tr>
<td>80,85</td>
<td><strong>&lt;CLR: Undo T₃ LSN 50&gt;</strong>, <strong>&lt;T₃ TXN-END&gt;</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X CRASH! RESTART!</strong></td>
</tr>
</tbody>
</table>
FULL EXAMPLE

---

**ATT**

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₂</td>
<td>U</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**DPT**

<table>
<thead>
<tr>
<th>PageId</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>50</td>
</tr>
<tr>
<td>P₃</td>
<td>08</td>
</tr>
<tr>
<td>P₅</td>
<td>10</td>
</tr>
</tbody>
</table>

---

**LSN**

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>

70      <CLR: Undo T₂ LSN 60, UndoNext 20>

80,85   <CLR: Undo T₃ LSN 50>, <T₃ TXN-END>

---

**LOG**

---

CRASH! RESTART!

flushedLSN
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?**

→ 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

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

Log Sequence Numbers:
→ **LSNs** identify log records; linked into backwards chains per transaction via **prevLSN**.
→ **pageLSN** allows comparison of data page and log records.
You now know how to build a single-node DBMS.

So now we can talk about distributed databases!