Database Recovery
ADMINISTRIVIA

Project #3 is due Sun Nov 14\textsuperscript{nd} @ 11:59pm.
Additional office hour on Saturday.

Homework #4 is due Wed Nov 10\textsuperscript{th} @ 11:59pm.
UPCOMING DATABASE TALK

Fluree - Cloud-Native Ledger Graph Database

→ Mon Nov 15th @ 4:30pm ET
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.
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.
CHECKPOINTS

The WAL will grow forever.

After a crash, the DBMS must replay the entire log, which will take a long time.

The DBMS periodically takes a checkpoint where it flushes all buffers out to disk.
CHECKPOINTS

Output onto stable storage all log records currently residing in main memory.

Output to the disk all modified blocks.

Write a `<CHECKPOINT>` entry to the log and flush to stable storage.
CHECKPOINTS

WAL

<T1 BEGIN>
<T1, A, 1, 2>
<T1 COMMIT>
<T2 BEGIN>
<T2, A, 2, 3>
<T3 BEGIN>
<T3 COMMIT>
<T3, A, 3, 4>

: 

CRASH!
CHECKPOINTS

WAL

\langle T1 \text{ BEGIN} \rangle
\langle T1, A, 1, 2 \rangle
\langle T1 \text{ COMMIT} \rangle
\langle T2 \text{ BEGIN} \rangle
\langle T2, A, 2, 3 \rangle
\langle T3 \text{ BEGIN} \rangle
\langle \text{CHECKPOINT} \rangle
\langle T2 \text{ COMMIT} \rangle
\langle T3, A, 3, 4 \rangle
\langle T3, A, 3, 4 \rangle
\text{CRASH!}
Any txn that committed before the checkpoint is ignored ($T_1$).
CHECKPOINTS

Any txn that committed before the checkpoint is ignored ($T_1$).

$T_2 + T_3$ did not commit before the last checkpoint.
CHECKPOINTS

Any txn that committed before the checkpoint is ignored ($T_1$).

$T_2 + T_3$ did not commit before the last checkpoint.

→ Need to **redo** $T_2$ because it committed after checkpoint.

→ Need to **undo** $T_3$ because it did not commit before the crash.
CHECKPOINTS – CHALLENGES

The DBMS must stall txns when it takes a checkpoint to ensure a consistent snapshot.

Scanning the log to find uncommitted txns can take a long time.

Not obvious how often the DBMS should take a checkpoint...
CHECKPOINTS – FREQUENCY

Checkpointing too often causes the runtime performance to degrade.
→ System spends too much time flushing buffers.

But waiting a long time is just as bad:
→ The checkpoint will be large and slow.
→ Makes recovery time much longer.
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.
A R I E S  –  M A I N  I D E A S

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
WAL RECORDS

We need to extend our log record format from last class to include additional info.

Every log record now includes a globally unique \textit{log sequence number} (LSN).

Various components in the system keep track of \textit{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>
Each data page contains a \textit{pageLSN}.  
\[\rightarrow\] The \textit{LSN} of the most recent update to that page.

System keeps track of \textit{flushedLSN}.  
\[\rightarrow\] The max \textit{LSN} flushed so far.

Before page \textit{x} can be written to disk, we must flush log at least to the point where:  
\[\rightarrow \textit{pageLSN}_x \leq \textit{flushedLSN}\]
Writing Log Records

WAL (Tail)

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

Buffer Pool

<table>
<thead>
<tr>
<th>pageLSN</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
<tr>
<td>C=2</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>pageLSN</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
<tr>
<td>C=2</td>
<td></td>
</tr>
</tbody>
</table>

MasterRecord

flushedLSN
WRITING LOG RECORDS

Log Sequence Numbers

WAL (Tail)

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

Buffer Pool

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

Log Sequence Numbers

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 <T₂ COMMIT>
009 <T₃, A, 3, 4>
100 <T₃ BEGIN>
101 <T₃, X, 5, 6>
102 <T₄, Y, 9, 7>
103 <T₄, B, 4, 2>
104 <T₄ BEGIN>
105 <T₄, B, 2, 3>
106 <T₄ C, 1, 2>

Database

pageLSN recLSN
A=9 B=5 C=2

MasterRecord

⋮
Writing Log Records

WAL (Tail)

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

Buffer Pool

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: <T3, BEGIN>
007: <CHECKPOINT>
008: <T2, COMMIT>
009: <T2, 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

flushedLSN

pageLSN  recLSN

A=9  B=5  C=2

MasterRecord
WAL (Tail)

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

Buffer Pool

pageSN | 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: <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

pageSN | recLSN
A=9    | B=5    | C=2
MasterRecord

Writing Log Records
**WRITING LOG RECORDS**

**WAL (Tail)**

017: \(<T_5 \text{BEGIN}>\)
018: \(<T_5, A, 9, 8>\)
019: \(<T_5, B, 5, 1>\)
020: \(<T_5 \text{ COMMIT}>\)

**Buffer Pool**

```
<table>
<thead>
<tr>
<th>page.LSN</th>
<th>rec.LSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
</tbody>
</table>
```

**WAL**

```
001: \(<T_1 \text{ BEGIN}>\)
002: \(<T_1, A, 1, 2>\)
003: \(<T_1 \text{ COMMIT}>\)
004: \(<T_2 \text{ BEGIN}>\)
005: \(<T_2, A, 2, 3>\)
006: \(<T_3 \text{ BEGIN}>\)
007: \(<\text{CHECKPOINT}>\)
008: \(<T_2 \text{ COMMIT}>\)
009: \(<T_3, A, 3, 4>\)
010: \(<T_4 \text{ BEGIN}>\)
011: \(<T_4, X, 5, 6>\)
012: \(<T_4, Y, 9, 7>\)
013: \(<T_3, B, 4, 2>\)
014: \(<T_3 \text{ COMMIT}>\)
015: \(<T_4, B, 2, 3>\)
016: \(<T_4, C, 1, 2>\)
```

**Database**

```
<table>
<thead>
<tr>
<th>page.LSN</th>
<th>rec.LSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
</tbody>
</table>
```

**MasterRecord**

flushedLSN
Writing Log Records

WAL (Tail)
017: <T_5 BEGIN>
018: <T_5, A, 9, 8>
019: <T_5, B, 5, 1>
020: <T_5 COMMIT>

Buffer Pool
pageSN  recLSN
A=9  B=5  C=2
flushedLSN

WAL
001: <T_1 BEGIN>
002: <T_1, A, 1, 2>
003: <T_1 COMMIT>
004: <T_2 BEGIN>
005: <T_2, A, 2, 3>
006: <T_2 COMMIT>
007: <CHECKPOINT>
008: <T_2 COMMIT>
009: <T_3, A, 3, 4>
010: <T_4 BEGIN>
011: <T_4, X, 5, 6>
012: <T_4, Y, 9, 7>
013: <T_3, B, 4, 2>
014: <T_3 COMMIT>
015: <T_4, B, 2, 3>
016: <T_4, C, 1, 2>

Database
pageSN  recLSN
A=9  B=5  C=2
MasterRecord
WAL (Tail)

017: <T_5, BEGIN>
018: <T_5, A, 9, 8>
019: <T_5, B, 5, 1>
020: <T_5, COMMIT>

Buffer Pool

pageLSN  recLSN
A=9  B=5  C=2

flushedLSN

WAL

001: <T_1, BEGIN>
002: <T_1, A, 1, 2>
003: <T_1, COMMIT>
004: <T_2, BEGIN>
005: <T_2, A, 2, 3>
006: <T_2, COMMIT>
007: <CHECKPOINT>
008: <T_2, COMMIT>
009: <T_3, A, 3, 4>
100: <T_4, BEGIN>
101: <T_4, X, 5, 6>
102: <T_4, Y, 9, 7>
103: <T_3, B, 4, 2>
104: <T_3, COMMIT>
105: <T_4, B, 2, 3>
106: <T_4, C, 1, 2>

Database

pageLSN  recLSN
A=9  B=5  C=2

MasterRecord

Writing Log Records
WRITING LOG RECORDS

WAL (Tail)

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

Buffer Pool

Safe to unpin because pageLSN ≤ flushedLSN

WAL

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

Database

MasterRecord

A=9 | B=5 | C=2

Safe to unpin because pageLSN ≤ flushedLSN
WRITING LOG RECORDS

WAL (Tail)

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

Buffer Pool

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: <T₂ COMMIT>
007: <CHECKPOINT>
008: <T₂ COMMIT>
009: <T₃, A, 3, 4>
010: <T₃ 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

MasterRecord

pageLSN reclSN

A=9 B=5 C=2

pageLSN reclSN

A=9 B=5 C=2

Not safe to unpin because pageLSN > flushedLSN
All log records have an **LSN**.

Update the **pageLSN** every time a txn modifies a record in the page.

Update the **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.
TRANSACTION COMMIT

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

Database

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

MasterRecord

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: <T3, B, 4, 2>
011: <T3, COMMIT>
TRANSACTION COMMIT

WAL (Tail)

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

flushedLSN = 015

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:<T3, BEGIN>
013:<T4, A, 9, 8>
014:<T4, B, 5, 1>
015:<T4 COMMIT>

flushedLSN = 015

Buffer Pool

pageLSN  recLSN
A=9  B=5  C=2

Database

pageLSN  recLSN
A=9  B=5  C=2

MasterRecord
TRANSACTION COMMIT

WAL (Tail)
012: \(<T_4 \text{ BEGIN}>\)
013: \(<T_4, A, 9, 8>\)
014: \(<T_4, B, 5, 1>\)
015: \(<T_4 \text{ COMMIT}>\)
... 
099: \(<T_4 \text{ TXN-END}>\)

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

WAL
001: \(<T_1 \text{ BEGIN}>\)
002: \(<T_1, A, 1, 2>\)
003: \(<T_1 \text{ COMMIT}>\)
004: \(<T_2 \text{ BEGIN}>\)
005: \(<T_2, A, 2, 3>\)
006: \(<T_2 \text{ BEGIN}>\)
007: \(<\text{CHECKPOINT}>\)
008: \(<T_2 \text{ COMMIT}>\)
009: \(<T_3, A, 3, 4>\)
010: \(<T_3, B, 4, 2>\)
011: \(<T_3 \text{ COMMIT}>\)
012: \(<T_4 \text{ BEGIN}>\)
013: \(<T_4, A, 9, 8>\)
014: \(<T_4, B, 5, 1>\)
015: \(<T_4 \text{ COMMIT}>\)

Database
pageLSN  recLSN
A=9  B=5  C=2
MasterRecord
We can trim the in-memory log up to flushedLSN
We can trim the in-memory log up to flushedLSN
Aborting a txn is a special case of the ARIES undo operation applied to only one txn.

We need to add another field to our log records:

→ **prevLSN**: The previous *LSN* for the txn.
→ This maintains a linked-list for each txn that makes it easy to walk through its records.
TRANSACTION ABORT

WAL (Tail)

012|nil:<T4 BEGIN>
013|012:<T4, A, 9, 8>
014|013:<T4, B, 5, 1>

Buffer Pool

pageSN | recSN
A=9  B=5  C=2
flushedSN

WAL

pageSN | recSN
A=9  B=5  C=2

Database

MasterRecord
TRANSACTION ABORT

Buffer Pool

<table>
<thead>
<tr>
<th>pageLSN</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
</tbody>
</table>

flushedLSN

WAL (Tail)

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>012</td>
<td>nil</td>
</tr>
<tr>
<td>013</td>
<td>012</td>
</tr>
<tr>
<td>014</td>
<td>013</td>
</tr>
</tbody>
</table>

WAL

<table>
<thead>
<tr>
<th>pageLSN</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
</tbody>
</table>

prevLSN

Database

MasterRecord
TRANSACTION ABORT

WAL (Tail)

<table>
<thead>
<tr>
<th>Page</th>
<th>LSN</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>012</td>
<td>nil:</td>
<td>(&lt;T_4 \text{ BEGIN}&gt;)</td>
</tr>
<tr>
<td>013</td>
<td>012:</td>
<td>(&lt;T_4, A, 9, 8&gt;)</td>
</tr>
<tr>
<td>014</td>
<td>013:</td>
<td>(&lt;T_4, B, 5, 1&gt;)</td>
</tr>
</tbody>
</table>

Buffer Pool

<table>
<thead>
<tr>
<th>Page</th>
<th>LSN</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\text{flushedLSN})</td>
</tr>
</tbody>
</table>

Database

<table>
<thead>
<tr>
<th>Page</th>
<th>LSN</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\text{MasterRecord})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(A=9, B=5, C=2)</td>
</tr>
</tbody>
</table>
TRANSACTION ABORT

WAL (Tail)

012|nil:<T\text{\_4} \text{BEGIN}>
013|012:<T\text{\_4}, A, 9, 8>
014|013:<T\text{\_4}, B, 5, 1>
015|014:<T\text{\_4} \text{ABORT}>

Buffer Pool

\begin{array}{c|c|c|c}
\text{pageLSN} & \text{recLSN} & \\
A=9 & B=5 & C=2 & \\
\end{array}

\text{flushedLSN}

WAL

\begin{array}{c|c|c|c}
\text{pageLSN} & \text{recLSN} & \\
A=9 & B=5 & C=2 & \\
\end{array}

Database

\text{MasterRecord}
**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**

<table>
<thead>
<tr>
<th>pageSN</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
</tbody>
</table>

**Database**

<table>
<thead>
<tr>
<th>pageSN</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=9</td>
<td>B=5</td>
</tr>
</tbody>
</table>
TRANSACTION ABORT

**Important**: Need to record what steps we took to undo the txn.
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 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>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>
<tr>
<td></td>
<td></td>
<td></td>
<td></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>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>
## 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>$T_1$</td>
<td>BEGIN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>$T_1$</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>$T_1$</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>$T_1$</td>
<td>CLR-002</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
</tr>
</tbody>
</table>
### TRANSACTION ABORT – CLR EXAMPLE

The LSN of the next log record to be undone is **001**.
## 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
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 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.
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.
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.
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.
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.
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.
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-END message.

**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 transactions that have not been flushed to disk.

One entry per dirty page in the buffer pool:

→ recLSN: The LSN of the log record that first caused the page to be dirty.
At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$),
At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$),
At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$),
At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$),
At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$),
At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$),
SLIGHTLY BETTER CHECKPOINTS

At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$).

At the second checkpoint, assuming $P_{22}$ was flushed, $T_2$ and $T_3$ are active and the dirty pages are ($P_{11}$, $P_{33}$).

```
<T_1 BEGIN>
<T_2 BEGIN>
<T_1, A→P_{11}, 100, 120>
<T_1 COMMIT>
<T_2, C→P_{22}, 100, 120>
<T_1 TXN-END >
<T_3 START>
<T_2, A→P_{11}, 120, 130>
<T_2 COMMIT>
<T_3, B→P_{33}, 200, 400>
<T_1 TXN-END>
<T_3 COMMIT>
<T_3, B→P_{33}, 400, 600>
```
At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$),

At the second checkpoint, assuming $P_{22}$ was flushed, $T_2$ and $T_3$ are active and the dirty pages are ($P_{11}$, $P_{33}$).
SLIGHTLY BETTER CHECKPOINTS

At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$).

At the second checkpoint, assuming $P_{22}$ was flushed, $T_2$ and $T_3$ are active and the dirty pages are ($P_{11}$, $P_{33}$).
SLIGHTLY BETTER CHECKPOINTS

At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$).

At the second checkpoint, assuming $P_{22}$ was flushed, $T_2$ and $T_3$ are active and the dirty pages are ($P_{11}$, $P_{33}$).

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 writes the log records for checkpoint. → No attempt to force dirty pages to disk.

New log records to track checkpoint boundaries:
→ CHECKPOINT-BEGIN: Indicates start of checkpoint
→ CHECKPOINT-END: Contains ATT + DPT.
FUZZY CHECKPOINTS

The \textit{LSN} of the \texttt{CHECKPOINT-BEGIN} record is written to the database's \texttt{MasterRecord} entry on disk when the checkpoint successfully completes.

Any txn that starts \underline{after} the checkpoint is excluded from the ATT in the \texttt{CHECKPOINT-END} record.
FUZZY CHECKPOINTS

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.

WAL

<\text{T}_1 \text{ BEGIN}>
<\text{T}_2 \text{ BEGIN}>
<\text{T}_1, A\rightarrow P_{11}, 100, 120>
<\text{T}_1 \text{ COMMIT}>
<\text{T}_2, C\rightarrow P_{22}, 100, 120>
<\text{T}_1 \text{ TXN-END }>
<\text{CHECKPOINT-BEGIN}>
<\text{T}_3 \text{ START}>
<\text{T}_2, A\rightarrow P_{11}, 120, 130>
<\text{CHECKPOINT-END}>
\text{ ATT} = \{\text{T}_2\},
\text{ DPT} = \{P_{22}\}
<\text{T}_2 \text{ COMMIT}>
<\text{T}_3, B\rightarrow P_{33}, 200, 400>
<\text{CHECKPOINT-BEGIN}>
<\text{T}_3, B\rightarrow P_{33}, 10, 12>
<\text{CHECKPOINT-END}>
\text{ ATT} = \{\text{T}_2, \text{T}_3\},
\text{ DPT} = \{P_{11}, P_{33}\>
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.
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.
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.
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.
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.
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 AT&T in the **CHECKPOINT-END** record.
ARIES – RECOVERY PHASES

Phase #1 – Analysis
→ Read WAL from last MasterRecord 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.
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.
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.
A R I E S – O V E R V I E W

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.
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.
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.
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.
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.
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.
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 \( \text{recLSN}=\text{LSN} \).
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_{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!

<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: `<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!

---

**Table:**

<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!

---

<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: <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!

<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></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

010: <CHECKPOINT-

020: <T₉₆, A→P₃₃, 10, 15>

030: <CHECKPOINT-END
  ATT={T₉₆, T₉₇},
  DPT={P₂₀, P₃₃}>

040: <T₉₆ COMMIT>

050: <T₉₆ TXN-END>

CRASH!

ATT={T₉₆, T₉₇}, DPT={P₂₀, P₃₃}>

020: (T₉₆, U)

030

040

050

Modify A in page P₃₃
Modify A in page $P_{33}$

\begin{tabular}{|c|c|c|}
\hline
\textbf{ATT} & \textbf{DPT} \\
\hline
010 &  \\
020 & ($T_{96}, U$) ($P_{33}, 020$) \\
030 &  \\
040 &  \\
050 &  \\
\hline
\end{tabular}

CRASH!
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 |                   |     
040 |                   |     
050 |                   |     

CRASH!
ANALYSIS PHASE EXAMPLE

WAL

010: <CHECKPOINT-BEGIN>
020: <T_{96}, A\to 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!

<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}, 008)</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) | (P33, 020) 
030 | (T96, U), (T97, U) | (P33, 020), (P20, 008) 
040 | (T96, C), (T97, U) | (P33, 020), (P20, 008) 
050 |     |     

(CMUIDB) 15-445/645 (Fall 2021)
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 | P33, 020
030 | T96, U, T97, U | P33, 020, P20, 008
040 | T96, C, T97, U | P33, 020, P20, 008
050 | T97, U | P33, 020, P20, 008
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 are techniques that allow the DBMS to avoid unnecessary reads/writes, but we will ignore that in this lecture...
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:

$\rightarrow$ Affected page is not in $\text{DPT}$, or

$\rightarrow$ 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 \( \text{lastLSN} \) 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>
</tbody>
</table>
FULL EXAMPLE

LSN | LOG
---|---
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 | 
50 | 
60 | 

TIME

LSN LOG
FULL EXAMPLE

LSN  | LOG
--- | ---
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  | 
60  | 

TIME

LSN

LOG
FULL EXAMPLE

LSN | LOG
---|---
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
60

prevLSNs

TIME

log

15-445/645 (Fall 2021)
FULL EXAMPLE

LSN | LOG
---|---
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  |
60  |
FULL EXAMPLE

**LSN**

- 00,05:  <CHECKPOINT-BEGIN>,  <CHECKPOINT-END>
- 10:  <T<sub>1</sub>, A→P<sub>5</sub>, 1, 2>
- 20:  <T<sub>2</sub>, B→P<sub>3</sub>, 2, 3>
- 30:  <T<sub>1</sub>  ABORT>
- 40,45:  <CLR: Undo T<sub>1</sub> LSN 10>,  <T<sub>1</sub>  TXN-END>
- 50:  <T<sub>3</sub>, C→P<sub>1</sub>, 4, 5>
- 60:  <T<sub>2</sub>, D→P<sub>5</sub>, 6, 7>

**LOG**

**ATT**

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T&lt;sub&gt;2&lt;/sub&gt;</td>
<td>U</td>
<td>60</td>
</tr>
<tr>
<td>T&lt;sub&gt;3&lt;/sub&gt;</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>P&lt;sub&gt;1&lt;/sub&gt;</td>
<td>50</td>
</tr>
<tr>
<td>P&lt;sub&gt;3&lt;/sub&gt;</td>
<td>08</td>
</tr>
<tr>
<td>P&lt;sub&gt;5&lt;/sub&gt;</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>T_2</td>
<td>U</td>
<td>60</td>
</tr>
<tr>
<td>T_3</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>P_1</td>
<td>50</td>
</tr>
<tr>
<td>P_3</td>
<td>08</td>
</tr>
<tr>
<td>P_5</td>
<td>10</td>
</tr>
</tbody>
</table>

**LSN**

- 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>

**LOG**

- CRASH! RESTART!

FlushedLSN
FULL EXAMPLE

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>

CRASH! RESTART!

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

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>70</td>
<td>&lt;CLR: Undo T₂ LSN 60, UndoNext 20&gt;</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>

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>

FlushedLSN

CRASH! RESTART!
## FULL EXAMPLE

### 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>
</tbody>
</table>

### CRASH! RESTART!

<table>
<thead>
<tr>
<th>LSN</th>
<th>LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>&lt;CLR: Undo T₂ LSN 60, UndoNext 20&gt;</td>
</tr>
<tr>
<td>80,85</td>
<td>&lt;CLR: Undo T₃ LSN 50&gt;, &lt;T₃ TXN-END&gt;</td>
</tr>
</tbody>
</table>

### 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
**FULL EXAMPLE**

### LSN

- **00,05**: <CHECKPOINT-BEGIN>, <CHECKPOINT-END>
- **10**: <T<sub>1</sub>, A→P<sub>5</sub>, 1, 2>
- **20**: <T<sub>2</sub>, B→P<sub>3</sub>, 2, 3>
- **30**: <T<sub>1</sub> ABORT>
- **40,45**: <CLR: Undo T<sub>1</sub> LSN 10>, <T<sub>1</sub> TXN-END>
- **50**: <T<sub>3</sub>, C→P<sub>1</sub>, 4, 5>
- **60**: <T<sub>2</sub>, D→P<sub>5</sub>, 6, 7>
- **70**: <CLR: Undo T<sub>2</sub> LSN 60, UndoNext>
- **80,85**: <CLR: Undo T<sub>3</sub> LSN 50>, <T<sub>3</sub> TXN-END>

### LOG

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

#### 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>70</td>
<td>&lt;CLR: Undo T₂ LSN 60, UndoNext&gt;</td>
</tr>
<tr>
<td>80,85</td>
<td>&lt;CLR: Undo T₃ LSN 50&gt;, &lt;T₃ TXN-END&gt;</td>
</tr>
</tbody>
</table>

#### 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>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>
<tr>
<td></td>
<td>CRASH! RESTART!</td>
</tr>
</tbody>
</table>

Flush dirty pages + WAL to disk!
FULL EXAMPLE

AT&T

<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  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>

CRASH! RESTART!

70     <CLR: Undo T2 LSN 60, UndoNext 20>
80,85  <CLR: Undo T3 LSN 50>, <T3 TXN-END>

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>70</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 | 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>

CRASH! RESTART!

70 | <CLR: Undo T₂ LSN 60, UndoNext 20>
80,85 | <CLR: Undo T₃ LSN 50>, <T₃ TXN-END>

CRASH! RESTART!
FULL EXAMPLE

ATT

<table>
<thead>
<tr>
<th>TxnId</th>
<th>Status</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_2</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>P_1</td>
<td>50</td>
</tr>
<tr>
<td>P_3</td>
<td>08</td>
</tr>
<tr>
<td>P_5</td>
<td>10</td>
</tr>
</tbody>
</table>

LSN

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!

LOG

<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;T_1, A→P_5, 1, 2&gt;</td>
</tr>
<tr>
<td>20</td>
<td>&lt;T_2, B→P_3, 2, 3&gt;</td>
</tr>
<tr>
<td>30</td>
<td>&lt;T_1 ABORT&gt;</td>
</tr>
<tr>
<td>40,45</td>
<td>&lt;CLR: Undo T_1 LSN 10&gt;, &lt;T_1 TXN-END&gt;</td>
</tr>
<tr>
<td>50</td>
<td>&lt;T_3, C→P_1, 4, 5&gt;</td>
</tr>
<tr>
<td>60</td>
<td>&lt;T_2, D→P_5, 6, 7&gt;</td>
</tr>
<tr>
<td>70</td>
<td>&lt;CLR: Undo T_2 LSN 60, UndoNext 20&gt;</td>
</tr>
<tr>
<td>80,85</td>
<td>&lt;CLR: Undo T_3 LSN 50&gt;, &lt;T_3 TXN-END&gt;</td>
</tr>
</tbody>
</table>

flushedLSN
FULL EXAMPLE

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>
70  | X CRASH! RESTART!
80,85 | <CLR: Undo T₂ LSN 60, UndoNext 20>
      | <CLR: Undo T₃ LSN 50>, <T₃ TXN-END>
      | X CRASH! RESTART!

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>

flushedLSN
**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>
</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>`
- **90,95**: `<CLR: Undo T2 LSN 20>, <T2 TXN-END>`

**CRASH! RESTART!**
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?
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.
How can the DBMS improve performance during recovery in the Redo Phase?

How can the DBMS improve performance during recovery in the Undo Phase?
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?
**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!