Lecture #21

Database Recovery

SPRING 2024 Prof. Jignesh Patel
HW #5 is released. Due Sat April 20, 2024 @ 11:59pm

Project #4 is released. Due Sun April 28, 2024 @ 11:59pm
   → Checkpoint to keep you on track and not wait till the last minute

No OH next week: Spring Carnival

Final Exam
   → Thu May 2, 2024 @ 05:30pm-08:30pm

Lectures #23 and #24: Recorded lectures and will be posted next week

Lecture #26: Guest Speaker from Snowflake
   → Devin Petersohn on “Beyond SQL: Dataframes in the Database”
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.
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.
THE BIG PICTURE

Operating with **STEAL + NO-FORCE**

Atomicity: Txns may abort/fail.

Durability: Changes of committed txns should survive system failure.

Desired behavior after the system restarts (i.e., the contents of volatile memory are lost):
→ $T_1$ and $T_2$ should be durable.
→ $T_3$ & $T_4$ should be aborted.
ARIES: 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 DBMS 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).

→ LSNs represent the physical order that txns make changes to the database.

Various components in the system keep track of LSNs that pertain to them…
WAL & THE LOG

- Log
- Disk
- DRAM

LSNs
pageLSNs
flushedLSN
WAL & THE LOG

Log records flushed to disk

“Log tail” in DRAM

pageLSN

A page

Another page

pageLSN

pageLSNs

flushedLSN

DRAM

Disk

LSNs
WAL & THE LOG

Log Sequence Number (LSN).

→ Unique and monotonically increasing.
WAL & THE LOG

Log Sequence Number (LSN).
→ Unique and monotonically increasing.

Each *data page* contains a *pageLSN*.
→ The LSN of the most recent log record that updated the page.

Log records flushed to disk

“Log tail” in DRAM
WAL & THE LOG

Log Sequence Number (LSN).
→ Unique and monotonically increasing.

Each data page contains a pageLSN.
→ The LSN of the most recent log record that updated the page.

System keeps track of flushedLSN.
→ The max LSN flushed so far.
**WAL & THE LOG**

Log Sequence Number (**LSN**).
→ Unique and monotonically increasing.

Each **data page** contains a **pageLSN**.
→ The LSN of the most recent log record that updated the page.

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

**WAL:** *Before* a page is written, \( \text{pageLSN}_x < \text{flushedLSN} \)
**WAL & THE LOG**

Log Sequence Number (LSN).
→ Unique and monotonically increasing.

Each *data page* contains a *pageLSN*.
→ The LSN of the most recent log record that updated the page.

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

**WAL:** Before a page is written, $\text{pageLSN}_x < \text{flushedLSN}$
## LOG SEQUENCE NUMBERS: THE FULL PICTURE

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</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>DPT†</td>
<td>Oldest update to page_x since it was last flushed</td>
</tr>
<tr>
<td>lastLSN</td>
<td>ATT*</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>

† DPT = Dirty Page Table.  
* ATT = Active Transaction Table.
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**

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

**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_1 BEGIN>`
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**

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

**MasterRecord**

flushedLSN
WRITING LOG RECORDS

Log Sequence Numbers

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

Buffer Pool

pageLSN
A=9 B=5 C=2

flushedLSN

Log Sequence Numbers

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_3 BEGIN>
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

WAL (Tail)

001 <T_1, A, 1, 2>
003 <T_1 COMMIT>
004 <T_2 BEGIN>
005 <T_2, A, 2, 3>
006 <T_3 BEGIN>
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>

MasterRecord

flushedLSN
A=9 B=5 C=2

pageLSN
WAL (Tail)

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

Buffer Pool

pageLSN
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: <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
A=9 B=5 C=2

MasterRecord

flushedLSN

writing log records

WAL
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

pageLSN
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

pageLSN
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

pageLSN
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

pageLSN
A=9 B=5 C=2

MasterRecord

flushedLSN

⋮
WRITING LOG RECORDS

WAL (Tail)

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

Buffer Pool

Safe to evict 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

MasterRecord

flushedLSN

pageLSN

A=9 B=5 C=2

Safe to evict 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

pageLSN
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: <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>

MasterRecord

flushedLSN
A=9  B=5  C=2

pageLSN
A=9  B=5  C=2
WAL (Tail)

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

Buffer Pool

Not safe to evict 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

A=9 | B=5 | C=2

MasterRecord

flushedLSN

pageLSN

A=9 | B=5 | C=2

Not safe to evict because pageLSN > flushedLSN
WRITING LOG RECORDS

All log records have an \textit{LSN}.

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

Update the \textit{flushedLSN} in memory every time the DBMS writes 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 Strong Strict 2PL.
→ STEAL + NO-FORCE buffer management with WAL.
**TRANSACTION COMMIT**

When a txn commits, the DBMS writes a **COMMIT** record to log and guarantees that 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.
→ Indicates that no new log record for a txn will appear in the log ever again.
→ This does not need to be flushed immediately.
**TRANSACTION COMMIT**

**WAL (Tail)**

012: \(\text{T}_4\) BEGIN

013: \(\text{T}_4\), A, 9, 8

014: \(\text{T}_4\), B, 5, 1

015: \(\text{T}_4\) COMMIT

**Buffer Pool**

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

**flushedLSN**

**WAL**

- 001: \(\text{T}_1\) BEGIN
- 002: \(\text{T}_1\), A, 1, 2
- 003: \(\text{T}_1\) COMMIT
- 004: \(\text{T}_2\) BEGIN
- 005: \(\text{T}_2\), A, 2, 3
- 006: \(\text{T}_2\) COMMIT
- 007: CHECKPOINT
- 008: \(\text{T}_2\) COMMIT
- 009: \(\text{T}_3\), A, 3, 4
- 010: \(\text{T}_3\), B, 4, 2
- 011: \(\text{T}_3\), COMMIT

**Database**

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

**MasterRecord**
TRANSACTION COMMIT

WAL (Tail)

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

Buffer Pool

pageLSN
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:<T2 COMMIT>
007:<CHECKPOINT>
008:<T3 BEGIN>
009:<T3, A, 3, 4>
010:<T3, B, 4, 2>
011:<T3, COMMIT>

Database

pageLSN
A=9 B=5 C=2
MasterRecord
The diagram illustrates the transaction commit process in a database system, specifically showing the WAL (Write-Ahead Log) and the buffer pool. The WAL logs are used to recover the database state in case of a system failure.

Here are the key points highlighted in the diagram:

**WAL (Tail):**
- **012:** <T4 BEGIN>
- **013:** <T4, A, 9, 8>
- **014:** <T4, B, 5, 1>
- **015:** <T4 COMMIT>

The **flushedLSN** is set to 015.

**Buffer Pool:**
- The pageLSN is shown as 9, 5, and 2.

**WAL:**
- The WAL contains entries for each transaction.

**Database:**
- The **MasterRecord** is shown with the **flushedLSN** set to 015.

The diagram visually connects these elements, demonstrating how transactions are logged and how the system manages page-level flushing and recovery.
TRANSACTION COMMIT

WAL (Tail)

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

Buffer Pool

pageLSN
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:<T₂ COMMIT>
007:<CHECKPOINT>
008:<T₂ 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>

Database

pageLSN
A=9 B=5 C=2
MasterRecord
TRANSACTION COMMIT

WAL (Tail)

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

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

Buffer Pool

pageLSN
A=9 B=5 C=2
flushedLSN

Database

pageLSN
A=9 B=5 C=2

MasterRecord

099: <T4 TXN-END>
We cantrim 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.
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

pageLSN
A=9 B=5 C=2
flushedLSN

WAL

pageLSN
A=9 B=5 C=2

Database

MasterRecord
Transaction Abort

Transaction Log (WAL)

- LSN
- prevLSN

Buffer Pool

- A=9
- B=5
- C=2

Database

- A=9
- B=5
- C=2

LSN: 012|nil

prevLSN: 012

Transaction Log Entries:

- \( T_4 \) BEGIN
- \( T_4, A, 9, 8 \)
- \( T_4, B, 5, 1 \)

Transaction Aborted: ABORT
Transaction Abort

WAL (Tail)

```
012|nil:<T_4 BEGIN>
013|012:<T_4, A, 9, 8>
014|013:<T_4, B, 5, 1>
015|014:<T_4 ABORT>
```

Buffer Pool

- `pageLSN`: A=9 B=5 C=2
- `flushedLSN`

WAL

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

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

Database

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

MasterRecord

Database

WAL
**TRANSACTION ABORT**

Important: Need to record what steps we took to undo the txn.
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 **undoNextLSN** 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 thetxn 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>UndoNextLSN</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>NextLSN</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>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</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>
# 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>UndoNextLSN</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>UndoNextLSN</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**
## 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>UndoNextLSN</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>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</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>UndoNextLSN</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>
<tr>
<td>027</td>
<td>026</td>
<td>$T_1$</td>
<td>TXN-END</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>nil</td>
</tr>
</tbody>
</table>

TIME

TRANSACTION ABORT – CLR EXAMPLE
### 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>UndoNextLSN</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>

**prevLSN**

**undoNextLSN**

**CLR**

**Time**

**U1**

**U2**

**U3**

**CLR3**

**CLR2**

**CLR1**

**Time**
ABORT ALGORITHM

First write an **ABORT** record to log for the txn.

Then analyze the txn’s updates in reverse order. For each update record:
- Write a **CLR** entry to the log.
- Restore old value.

Lastly, write a **TXN-END** record and release locks.

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 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.
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 (ATT)

One entry per currently active txn.

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

Remove entry after the **TXN-END** record.

Txn Status Codes:

- **R** → Running
- **C** → Committing
- **U** → Candidate for Undo
DIRTY PAGE TABLE (DPT)

Keep track of which pages in the buffer pool contain changes 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.
SLIGHTLY BETTER CHECKPOINTS

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

```
<T_1  BEGIN>
<T_2  BEGIN>
<T_1,  A→P_{11},  100, 120>
<T_1  COMMIT>
<T_2,  C→P_{22},  100, 120>
<T_2  COMMIT>
<T_3  BEGIN>
<T_2,  A→P_{11},  120, 130>
<T_2  COMMIT>
<T_3,  B→P_{33},  200, 400>
<T_3  COMMIT>
<T_3  TXN-END >
<T_3  CHECKPOINT
    ATT={T_2},
    DPT={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 ($P_{22}$).
SLIGHTLY BETTER CHECKPOINTS

At the first checkpoint, assuming $P_{11}$ was flushed, $T_2$ is still running and there is only one dirty page ($P_{22}$).
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 >
<CHECKPOINT
   ATT={T_2},
   DPT={P_{22}}> 
<T_3 BEGIN>
<T_2 A→P_{11}, 120, 130>
<T_2 COMMIT>
<T_3 B→P_{33}, 200, 400>
<CHECKPOINT
   ATT={T_2,T_3},
   DPT={P_{11},P_{33}}> 
<T_3 B→P_{33}, 400, 600>```
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 ($P_{11}$, $P_{33}$).
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...

WAL

- $\langle T_1 \text{ BEGIN} \rangle$
- $\langle T_2 \text{ BEGIN} \rangle$
- $\langle T_1, A\rightarrow P_{11}, 100, 120 \rangle$
- $\langle T_1 \text{ COMMIT} \rangle$
- $\langle T_2, C\rightarrow P_{22}, 100, 120 \rangle$
- $\langle T_1 \text{ TXN-END} \rangle$
- $\langle \text{CHECKPOINT} \rangle$
  
  $\text{ATT} = \{T_2\}$,
  
  $\text{DPT} = \{P_{22}\}$

- $\langle T_3 \text{ BEGIN} \rangle$
- $\langle T_2, A\rightarrow P_{11}, 120, 130 \rangle$
- $\langle T_2 \text{ COMMIT} \rangle$
- $\langle T_3, B\rightarrow P_{33}, 200, 400 \rangle$
- $\langle \text{CHECKPOINT} \rangle$
  
  $\text{ATT} = \{T_2, T_3\}$,
  
  $\text{DPT} = \{P_{11}, P_{33}\}$

- $\langle T_3, B\rightarrow P_{33}, 400, 600 \rangle$
FUZZY CHECKPOINTS

A fuzzy checkpoint is where the DBMS allows active txns to continue to 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

Assume the DBMS flushes \( P_{11} \) before the first checkpoint starts.

Any txn that begins after the checkpoint starts is excluded from the ATT in the CHECKPOINT-END record.

The \textit{LSN} of the CHECKPOINT-BEGIN record is written to the \texttt{MasterRecord} when it completes.
FUZZY CHECKPOINTS

Assume the DBMS flushes $P_{11}$ before the first checkpoint starts.

Any txn that begins after the checkpoint starts is excluded from the ATT in the **CHECKPOINT-END** record.

The **LSN** of the **CHECKPOINT-BEGIN** record is written to the **MasterRecord** when it completes.
FUZZY CHECKPOINTS

Assume the DBMS flushes $P_{11}$ before the first checkpoint starts.

Any txn that begins after the checkpoint starts is excluded from the ATT in the **CHECKPOINT-END** record.

The **LSN** of the **CHECKPOINT-BEGIN** record is written to the **MasterRecord** when it completes.
FUZZY CHECKPOINTS

Assume the DBMS flushes $P_{11}$ before the first checkpoint starts.

Any txn that begins after the checkpoint starts is excluded from the ATT in the `CHECKPOINT-END` record.

The **LSN** of the `CHECKPOINT-BEGIN` record is written to the `MasterRecord` when it completes.
FUZZY CHECKPOINTS

Assume the DBMS flushes $P_{11}$ before the first checkpoint starts.

Any txn that begins after the checkpoint starts is excluded from the ATT in the `CHECKPOINT-END` record.

The LSN of the `CHECKPOINT-BEGIN` record is written to the `MasterRecord` when it completes.
ARIES – RECOVERY PHASES

Phase #1 – Analysis

→ Examine the WAL in forward direction starting at 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.
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.
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.

---

**Diagram:**

- **TIME**
- **WAL**
- **Start of last checkpoint**
- **CRASH!**
- **A**
- **1**
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.
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.
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 the DBMS finds a **TXN-END** record, remove its corresponding txn from ATT.

All other records:
→ If txn not in ATT, add it with status **UNDO**.
→ On commit, change txn status to **COMMIT**.

For update log 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_{96}, A\rightarrow P_{33}, 10, 15>

030:<CHECKPOINT-END
ATT=\{T_{96}, T_{97}\},
DPT=\{P_{20}\}>

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: `<T₉₆, A→ₚ₃₃, 10, 15>`

...  
030: `<CHECKPOINT-END`

ATT={T₉₆, T₉₇},

DPT={P₂₀}`

...  
040: `<T₉₆ COMMIT>`

...  
050: `<T₉₆ 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></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}\}>
  ...
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}\}>
... 
040: <T_{96} COMMIT>
... 
050: <T_{96} TXN-END>
... 
CRASH!

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

(TxnId, Status)
ANALYSIS PHASE EXAMPLE

Modify A in page $P_{33}$

<table>
<thead>
<tr>
<th>ATT</th>
<th>DPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>($T_{96}, U$)</td>
</tr>
<tr>
<td>030</td>
<td></td>
</tr>
<tr>
<td>040</td>
<td></td>
</tr>
<tr>
<td>050</td>
<td></td>
</tr>
</tbody>
</table>

010: <CHECKPOINT-BEGIN>

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

030: <CHECKPOINT-END
ATT={$T_{96}, T_{97}$},
DPT={$P_{20}$}>

040: <$T_{96}$ COMMIT>

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

CRASH!

CRASH!

LSN

ATT={$T_{96}, T_{97}$},
DPT={$P_{20}$}
ANALYSIS PHASE EXAMPLE

010: <CHECKPOINT>

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

030: <CHECKPOINT-END
ATT={T_{96}, T_{97}},
DPT={P_{20}}>

040: <T_{96} COMMIT>

050: <T_{96} TXN-END>

CRASH!

Modify A in page P_{33}

<table>
<thead>
<tr>
<th>ATT</th>
<th>DPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>010</td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>(T_{96}, U)</td>
</tr>
<tr>
<td></td>
<td>(P_{33}, 020)</td>
</tr>
<tr>
<td>030</td>
<td></td>
</tr>
<tr>
<td>040</td>
<td></td>
</tr>
<tr>
<td>050</td>
<td></td>
</tr>
</tbody>
</table>

(PageId, RecLSN)
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}}>
   ...
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></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}}>
...  
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}, 008)
040
050
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}\}>

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}, 008)
040  (T_{96}, C), (T_{97}, U)  (P_{33}, 020), (P_{20}, 008)
050
ANALYSIS PHASE EXAMPLE

WAL

<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>$T_{96}$, $A \rightarrow P_{33}$, 10, 15</td>
<td>($P_{33}, 020$)</td>
</tr>
<tr>
<td>030</td>
<td></td>
<td>($P_{33}, 020$), ($P_{20}, 008$)</td>
</tr>
<tr>
<td>040</td>
<td></td>
<td>($P_{20}, 008$)</td>
</tr>
<tr>
<td>050</td>
<td>$T_{97}$, $U$</td>
<td>($P_{33}, 020$), ($P_{20}, 008$)</td>
</tr>
</tbody>
</table>

CRASH!
REDO PHASE

The goal is to repeat history to reconstruct the database state at the moment of the crash:
→ Reapply all updates (even aborted txns!) and redo CLRss.

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 \texttt{recLSN} in \texttt{DPT}.

For each update log record or \texttt{CLR} with a given \texttt{LSN}, redo the action unless:

→ The affected page is not in \texttt{DPT}, or

→ The affected page is in \texttt{DPT}, but that log record’s LSN is less than the page’s \texttt{recLSN}. (The update was propagated to disk.)

→ Log record’s LSN ≤ pageLSN;

Note, we must fetch the page from the disk to read the page value.
REDO PHASE

To redo an action:
→ Reapply logged update.
→ 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.
→ At each step, pick the largest lastLSN across all transactions in the ATT.
→ Traverse the lastLSNs in the same order, but in reverse, for how the updates happened originally.

Write a CLR for every modification.
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
45
50
60
FULL EXAMPLE

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

<table>
<thead>
<tr>
<th>Time (LSN)</th>
<th>Event</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>
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

TIME

<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!

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

<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!

### 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>

Flushed LSN: 60

---

**LOG**

- <CHECKPOINT-BEGIN>, <CHECKPOINT-END>
- <T₁, A→P₅, 1, 2>
- <T₂, B→P₃, 2, 3>
- <T₁ ABORT>
- <CLR: Undo T₁ LSN 10>, <T₁ TXN-END>
- <T₃, C→P₁, 4, 5>
- <T₂, D→P₅, 6, 7>

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

### LSN

- **00,05**: \(<\text{CHECKPOINT-BEGIN}>\), \(<\text{CHECKPOINT-END}>\)
- **10**: \(<\text{T}_1, A\rightarrow P_5, 1, 2>\)
- **20**: \(<\text{T}_2, B\rightarrow P_3, 2, 3>\)
- **30**: \(<\text{T}_1 \text{ ABORT}>\)
- **40,45**: \(<\text{CLR}: \text{Undo T}_1 \text{ LSN 10}>\), \(<\text{T}_1 \text{ TXN-END}>\)
- **50**: \(<\text{T}_3, C\rightarrow P_1, 4, 5>\)
- **60**: \(<\text{T}_2, D\rightarrow P_5, 6, 7>\)
- **70**: \(<\text{CLR}: \text{Undo T}_2 \text{ LSN 60}, \text{UndoNext 20}>\)

### LOG

- **ATT**
  - | TxnId | Status | lastLSN |
  - | T2 | U | 60 |
  - | T3 | U | 50 |
  - | - | - | - |

- **DPT**
  - | PageId | recLSN |
  - | P1 | 50 |
  - | P3 | 08 |
  - | P5 | 10 |

**CRASH! RESTART!**

**flushedLSN**
FULL EXAMPLE

\[\begin{array}{c|c|c}
\text{TxnId} & \text{Status} & \text{lastLSN} \\
\hline
T_2 & U & 60 \\
T_3 & U & 50 \\
\hline
\end{array}\]

DPT

\begin{array}{c|c}
\text{PageId} & \text{recLSN} \\
\hline
P_1 & 50 \\
P_3 & 08 \\
P_5 & 10 \\
\hline
\end{array}

\text{flushedLSN}

\text{LOG}

\begin{align*}
00,05 & \langle \text{CHECKPOINT-BEGIN} \rangle, \langle \text{CHECKPOINT-END} \rangle \\
10 & \langle T_1, A \rightarrow P_5, 1, 2 \rangle \\
20 & \langle T_2, B \rightarrow P_3, 2, 3 \rangle \\
30 & \langle T_1 \text{ ABORT} \rangle \\
40,45 & \langle \text{CLR: Undo } T_1 \text{ LSN 10} \rangle, \langle T_1 \text{ TXN-END} \rangle \\
50 & \langle T_3, C \rightarrow P_1, 4, 5 \rangle \\
60 & \langle T_2, D \rightarrow P_5, 6, 7 \rangle \\
70 & \langle \text{CLR: Undo } T_2 \text{ LSN 60}, \text{UndoNext 20} \rangle \\
\end{align*}

\text{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**

- Lifetime: 70,85

- 70: `<CLR: Undo T_2 LSN 60, UndoNext 20>`
- 80,85: `<CLR: Undo T_3 LSN 50>, <T_3 TXN-END>`
**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>`

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

**ATT**
- TxnId | Status | lastLSN
- T₂ | U | 60
- T₃ | U | 50
- - | - | -

**DPT**
- PageId | recLSN
- P₁ | 50
- P₃ | 08
- P₅ | 10

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>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>

Flush dirty pages + WAL to disk!

**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

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

Flush dirty pages + WAL to disk!
FULL EXAMPLE

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

<table>
<thead>
<tr>
<th>DPT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PageId</td>
<td>recLSN</td>
</tr>
<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>

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

CRASH! RESTART!

flushedLSN
FULL EXAMPLE

**LSN 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;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>
<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>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>

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>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

<table>
<thead>
<tr>
<th>00,05</th>
<th>&lt;CHECKPOINT-BEGIN&gt;, &lt;CHECKPOINT-END&gt;</th>
</tr>
</thead>
<tbody>
<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 20&gt;</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>
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

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     CRASH! RESTART!
80,85  <CLR: Undo T2 LSN 60, UndoNext 20>
90,95  CRASH! RESTART!

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.
NEXT CLASS

You now know how to build a single-node DBMS.

So now we can talk about distributed databases!