Database Recovery
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.
ARIES

Algorithms for Recovery and Isolation Exploiting Semantics

Developed at IBM Research in early 1990s.

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

Write-Ahead Logging:
→ Any change is recorded in log on stable storage before the database change is written to disk.
→ Has to be STEAL + NO-FORCE.

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 log sequence number (LSN).

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Where</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>flushedLSN</td>
<td>Memory</td>
<td>Last LSN in log on disk</td>
</tr>
<tr>
<td>pageLSN</td>
<td>page&lt;sub&gt;x&lt;/sub&gt;</td>
<td>Newest update to page&lt;sub&gt;x&lt;/sub&gt;</td>
</tr>
<tr>
<td>recLSN</td>
<td>page&lt;sub&gt;x&lt;/sub&gt;</td>
<td>Oldest update to page&lt;sub&gt;x&lt;/sub&gt; since it was last flushed</td>
</tr>
<tr>
<td>lastLSN</td>
<td>T&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Latest action of txn T&lt;sub&gt;i&lt;/sub&gt;</td>
</tr>
<tr>
<td>MasterRecord</td>
<td>Disk</td>
<td>LSN of latest checkpoint</td>
</tr>
</tbody>
</table>
WRITING LOG RECORDS

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

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

Before page \( x \) can be written to disk, we must flush log at least to the point where:
→ \( \text{pageLSN}_x \leq \text{flushedLSN} \)
WRITING LOG RECORDS

- **Log Sequence Numbers**
  - WAL (Tail)
  - Buffer Pool
  - Database

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

- **Log Sequence Numbers**
  - pageLSN: A=9, B=5, C=2
  - recLSN:

- **Database**
  - MasterRecord
  - flushedLSN

- **Example Log Records**
  - 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>
WRITING LOG RECORDS

WAL (Tail)

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

Buffer Pool

pageLSN recLSN
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: <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>

MasterRecord

pageLSN recLSN
A=9 B=5 C=2
WRITING LOG RECORDS

WAL (Tail)

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

Buffer Pool

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

CMU 15-445/645 (Fall 2018)
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>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

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

<table>
<thead>
<tr>
<th>WAL</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>001:</td>
<td>&lt;T_1 BEGIN&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002:</td>
<td>&lt;T_1, A, 1, 2&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>003:</td>
<td>&lt;T_1 COMMIT&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>004:</td>
<td>&lt;T_2 BEGIN&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005:</td>
<td>&lt;T_2, A, 2, 3&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006:</td>
<td>&lt;T_1 BEGIN&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007:</td>
<td>&lt;CHECKPOINT&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008:</td>
<td>&lt;T_2 COMMIT&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>009:</td>
<td>&lt;T_3, A, 3, 4&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010:</td>
<td>&lt;T_4 BEGIN&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011:</td>
<td>&lt;T_4, X, 5, 6&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>012:</td>
<td>&lt;T_4, Y, 9, 7&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>013:</td>
<td>&lt;T_3, B, 4, 2&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>014:</td>
<td>&lt;T_3 COMMIT&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>015:</td>
<td>&lt;T_4, B, 2, 3&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>016:</td>
<td>&lt;T_4, C, 1, 2&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MasterRecord

flushedLSN

pageLSN
recLSN
A=9
B=5
C=2
WRITING LOG RECORDS

WAL (Tail)

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

Buffer Pool

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

MasterRecord
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

Safe to unpin because pageLSN \leq 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_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

MasterRecord

pageLSN recLSN
A=9 B=5 C=2

Safe to unpin because pageLSN \leq flushedLSN
Writing Log Records

WAL (Tail)

017: \langle T_5 \text{ BEGIN} \rangle
018: \langle T_5, A, 9, 8 \rangle
019: \langle T_5, B, 5, 1 \rangle
020: \langle T_5 \text{ COMMIT} \rangle

Buffer Pool

Not safe to unpin because pageLSN > flushedLSN

WAL

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

MasterRecord

PageLSN recLSN
A=9 B=5 C=2

[Not safe to unpin because pageLSN > flushedLSN]
WRITING LOG RECORDS

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

Eachtxn 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.
TRANSACTION COMMIT

Write **COMMIT** record to log.

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

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

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

Buffer Pool

pageLSN  recLSN
A=9  B=5  C=2

flushedLSN = 015

TRANSACTION COMMIT

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: <T1 BEGIN>
013: <T4, A, 9, 8>
014: <T4, B, 5, 1>
015: <T4 COMMIT>

flushedLSN = 015

Database

MasterRecord

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

TRANSACTION 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>
</tbody>
</table>

flushedLSN

Database

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>

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

We can trim the in-memory log up to flushedLSN

**CMU 15-445/645 (Fall 2018)**
We can trim the in-memory log up to flushedLSN.
TRANSACTION ABORT

Aborting a txn is actually a special case of the ARIES undo operation applied to only one transaction.

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 COMMIT

Buffer Pool

<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>

Transaction Log:

- T_4: BEGIN
- T_4: A, 9, 8
- T_4: B, 5, 1

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>

Flushed LSN:

- flushedLSN

Master Record
WAL (Tail)

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

Buffer Pool

flushedLSN

A=9 B=5 C=2

TRANSACTION COMMIT
TRANSACTION COMMIT

WAL (Tail)

012|nil:<T₄ BEG>
013|012:<T₄, A, 9, 8>
014|013:<T₄, B, 5, 1>
015|014:<T₄ ABORT

???
099|098:<T₄ TXN-END

Important: Need to record what steps we took to undo the txn.

Buffer Pool

pageLSN recLSN
A=9 B=5 C=2

flushedLSN

Database

pageLSN recLSN
A=9 B=5 C=2

MasterRecord
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 like any other record.
## 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</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
</tr>
</tbody>
</table>

(Time progression indicated by downward arrow.)
### 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>$T_1$</td>
<td>UPDATE</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</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>$T_1$</td>
<td>CLR</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
</tr>
</tbody>
</table>

- **TIME**

- **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>Undo</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>T₁</td>
<td>BEGIN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>002</td>
<td>001</td>
<td>T₁</td>
<td>UPDATE</td>
<td>A</td>
<td>30</td>
<td>40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>002</td>
<td>T₁</td>
<td>ABORT</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>011</td>
<td>T₁</td>
<td>CLR</td>
<td>A</td>
<td>40</td>
<td>30</td>
<td>001</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>TxnId</th>
<th>Type</th>
<th>Object</th>
<th>Before</th>
<th>After</th>
<th>UndoNext</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>nil</td>
<td>$T_1$</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</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>
ABORT ALGORITHM

First write an **ABORT** record to log.
Then play back updates in reverse order. For each update:
→ Write a **CLR** entry.
→ Restore old value.
At end, write a **TXN-END** log record.

Notice: **CLR**s 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 obviously bad…
SLIGHTLY BETTER CHECKPOINTS

Pause txns while the DBMS takes the checkpoint.
→ We don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause txns while the DBMS takes the checkpoint.
→ We don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause txns while the DBMS takes the checkpoint.
→ We don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause txns while the DBMS takes the checkpoint.
→ We don't have to wait until all txns finish before taking the checkpoint.
SLIGHTLY BETTER CHECKPOINTS

Pause txns while the DBMS takes the checkpoint.
→ We don't have to wait until all txns finish before taking the checkpoint.

We have to 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 when txn commits or aborts.

Status Codes:

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

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

One entry per dirty page:
→ recLSN: The LSN of the log record that first caused the page to be dirty.
SLIGHTLY BETTER CHECKPOINTS

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

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

This still isn't ideal because we have to stall all txns during checkpoint...
FUZZY CHECKPOINTS

A **fuzzy checkpoint** is where the DBMS allows other txns to continue the run.

New log records to track checkpoint boundaries:
- **CHECKPOINT-BEGIN**: Indicates start of checkpoint
- **CHECKPOINT-END**: Contains ATT + DPT.
The LSN of the CHECKPOINT-BEGIN record is written to the database's MasterRecord entry on disk.

Any txn that starts after the checkpoint is excluded from the txn table listing.
The LSN of the `CHECKPOINT-BEGIN` record is written to the database's `MasterRecord` entry on disk.

Any txn that starts after the checkpoint is excluded from the txn table listing.
Fuzzy Checkpoint

The \textit{LSN} of the \textsc{Checkpoint-Begin} record is written to the database's \texttt{MasterRecord} entry on disk.

Any txn that starts after the checkpoint is excluded from the \texttt{txn} table listing.
The LSN of the CHECKPOINT-BEGIN record is written to the database's MasterRecord entry on disk.

Any txn that starts after the checkpoint is excluded from the txn table listing.
ARIES – RECOVERY PHASES

Phase #1 – Analysis
→ Read the WAL 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.

Phase #3 – Undo
→ Reverse the actions of txns that did not commit before the crash.
ARIES – OVERVIEW

Start from last **BEGIN-CHECKPOINT** found via **MasterRecord**.

**Analysis**: Figure out which txns committed or failed since checkpoint.

**Redo**: Repeat all actions.

**Undo**: Reverse effects of failed txns.
ANALYSIS PHASE

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

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

WAL

010: <CHECKPOINT-BEGIN>

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

...<CHECKPOINT-END

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

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

...<T_{96} COMMIT>

...<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,P33}>
  ...
040: <T96 COMMIT>
  ...
050: <T96 TXN-END>
  ...
CRASH!

LSN  ATT          DPT
     (TxnId, Status)
010
020  (T96, U)
030
040
050

(TxnId, Status)
## ANALYSIS PHASE EXAMPLE

**Modify A in page P\textsubscript{33}**

<table>
<thead>
<tr>
<th>PageID</th>
<th>RecLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T\textsubscript{96}</td>
<td>U</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PageID</th>
<th>RecLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T\textsubscript{97}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PageID</th>
<th>RecLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P\textsubscript{20}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PageID</th>
<th>RecLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P\textsubscript{33}</td>
<td>020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PageID</th>
<th>RecLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P\textsubscript{20}</td>
<td>022</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PageID</th>
<th>RecLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T\textsubscript{96}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PageID</th>
<th>RecLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T\textsubscript{97}</td>
<td></td>
</tr>
</tbody>
</table>

**ATT** = \{T\textsubscript{96}, T\textsubscript{97}\}

**DPT** = \{P\textsubscript{20}, P\textsubscript{33}\}

### CHECKPOINT - BEGIN

010: \langle \text{CHECKPOINT-END} \rangle

020: \langle T\textsubscript{96}, A\rightarrow P\textsubscript{33}, 10, 15 \rangle

030: \langle \text{CHECKPOINT-END} \rangle

\text{ATT}=\{T\textsubscript{96}, T\textsubscript{97}\}, \text{DPT}=\{P\textsubscript{20}, P\textsubscript{33}\}

040: \langle T\textsubscript{96}, \text{COMMIT} \rangle

050: \langle T\textsubscript{96}, \text{TXN-END} \rangle

---

**CRASH!**
WAL

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

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

WAL

010: <CHECKPOINT-BEGIN>

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

030: <CHECKPOINT-END
ATT={{T_{96}, T_{97}}},
DPT={P_{20}, P_{33}}>

040: <T_{96} COMMIT>

050: <T_{96} TXN-END>

CRASH!

LSN  ATT               DPT
010
020  \((T_{96}, U)\)       \((P_{33}, 020)\)
030  \((T_{96}, U), (T_{97}, U)\) \((P_{33}, 020), (P_{20}, 022)\)
040  \((T_{96}, C), (T_{97}, U)\) \((P_{33}, 020), (P_{20}, 022)\)
050  \((T_{97}, U)\)       \((P_{33}, 020), (P_{20}, 022)\)
The goal is to repeat history to reconstruct state at the moment of the crash:

→ Reapply all updates (even aborted txns!) and redo CLRs.

There techniques that allow the DBMS to avoid unnecessary reads/writes, but we will ignore that in this lecture...
REDO PHASE

Scan forward from the log record containing smallest \texttt{recLSN} in \texttt{DPT}.

For each update log record or \texttt{CLR} with a given \texttt{LSN}, redo the action unless:
→ Affected page is not in the \texttt{DPT}, or
→ Affected page is in \texttt{DPT} but that record's \texttt{LSN} is greater than smallest \texttt{recLSN}, or
→ Affected \texttt{pageLSN} (on disk) $\geq$ record's \texttt{LSN}
REDO PHASE

To redo an action:
→ Reapply logged action.
→ Set \texttt{pageLSN} to log record's \texttt{LSN}.
→ No additional logging, no forcing!

At the end of Redo Phase, write \texttt{TXN-END} log records for all txns with status \texttt{C} and remove them from the \texttt{ATT}.
UNDO PHASE

Undo all txns that were active at the time of crash and therefore will never commit.

→ These are all txns with U status in the ATT after the Analysis Phase.

Process them in reverse LSN order using the lastLSN to speed up traversal.

Write a CLR for every modification.
FULL EXAMPLE

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 | <T₃, C→P₁, 4, 5>
60 | <T₂, D→P₅, 6, 7>

prevLSNs

CRASH!
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!

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

CMU 15-445/645 (Fall 2018)
### 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  
10     
20     
30     
40,45  
50     
60     
70

**LOG**

- **<CHECKPOINT-BEGIN>, <CHECKPOINT-END>**
- **<T_1, A→P_5, 1, 2>**
- **<T_2, B→P_3, 2, 3>**
- **<T_1 ABORT>**
- **<CLR: Undo T_1 LSN 10>, <T_1 TXN-END>**
- **<T_3, C→P_1, 4, 5>**
- **<T_2, D→P_5, 6, 7>**
- **<CLR: Undo T_2 LSN 60, UndoNext 20>**

**CRASH! RESTART!**
**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>

**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>

**Flush WAL to disk!**
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>

Flush WAL to disk!
**FULL EXAMPLE**

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

Flush WAL to disk!
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  | <CLR: Undo T₂ LSN 60, UndoNext 20>
80,85 | <CLR: Undo T₃ LSN 50>, <T₃ TXN-END>

ATT

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

CRASH! RESTART!
FULL EXAMPLE

LSN

LOG

<CHECKPOINT-BEGIN>, <CHECKPOINT-END>

<T_1, A→P_5, 1, 2>
<T_2, B→P_3, 2, 3>
<T_1 ABORT>

<T_2, C→P_1, 4, 5>
<T_2, D→P_5, 6, 7>

CRASH! RESTART!

<T_2, Undo T_2 LSN 10>, <T_1 TXN-END>
<T_3, C→P_1, 4, 5>
<T_2, D→P_5, 6, 7>

CRASH! RESTART!

<T_3, Undo T_3 LSN 50>, <T_3 TXN-END>

<T_2, Undo T_2 LSN 20>, <T_2 TXN-END>

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>

flushedLSN

<CLR: Undo T_2 LSN 60, UndoNext 20>
<CLR: Undo T_3 LSN 50>, <T_3 TXN-END>
<CLR: Undo T_2 LSN 20>, <T_2 TXN-END>
ADDITIONAL CRASH ISSUES (1)

What does the DBMS do if it crashes during recovery in the Analysis Phase?
→ Nothing. Just run recovery again.

What does the DBMS do if it crashes during recovery in the Redo Phase?
→ Again nothing. Redo everything again.
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 **CLR**s 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!