CSIS 7102 Spring 2004
Lecture 10: ARIES

Dr. King-Ip Lin

ARIES

- State of the art recovery manager
- Developed by Mohan et al at IBM
- Characteristics:
  - Simple but flexible
  - Support operation logging (handle recovery in case of insert/delete operations)
  - Fast recovery and minimum overhead
  - Parallelism
  - Support fine granularity locking
  - Support isolation levels

Logs in ARIES

- Log sequence number (LSN)
  - Associated with each log record
  - Unique for each log record
  - Sequentially increasing
  - Typical implementation
    - Offset to the start of a log file
    - Enable each log record to be located quickly – crucial for ARIES
    - One can have multiple log files, each keep track of log at certain time. Then use file name and offset to uniquely identify the LSN

Logs in ARIES

- Each disk page is associated with a PageLSN
  - the LSN of the last log record whose effects are reflected on the page
  - How can it be used?
    - During recovery, if a given page's PageLSN > LSN of a log record that act on that page, no need to redo that log record (Why?)

Logs in ARIES

- Each log entry also store a *PrevLSN*
  - The previous LSN of the same transaction that is adding this log record
- Thus a log record in ARIES looks like this:

LSN	TransId	PrevLSN	RedoInfo	UndoInfo

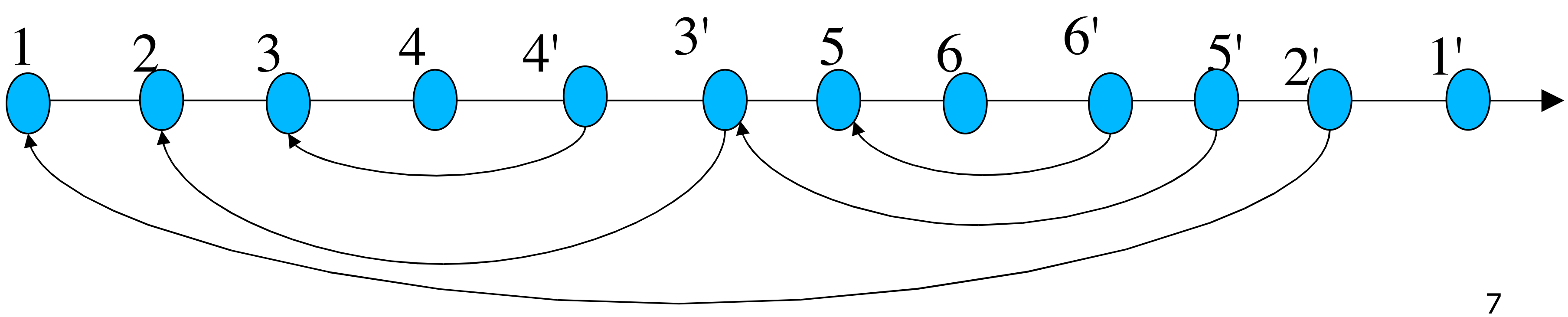
Logs in ARIES

- Compensation log records (CLR)
  - Log record generated during undo phase of recovery
  - Enable recovery mechanism to avoid undo same operation again
  - Have a field UndoNextLSN to note next (earlier) record to be undone
    - Records in between would have already been undone
    - Required to avoid repeated undo of already undone actions

LSN	TransID	UndoNextLSN	RedoInfo

Logs in ARIES

- When an undo is performed for an update log record
  - Generate a CLR containing the undo action performed (actions performed during undo are logged physicaly or physiologically).
    - CLR for record *n* noted as *n*′ in figure below
  - Set UndoNextLSN of the CLR to the PrevLSN value of the update log record
    - Arrows indicate UndoNextLSN value



Latches

- One requirement for ARIES: No updates should be in progress on a block (page) when it is output to disk
- To ensure this:
  - Before writing a data item, transaction acquires exclusive lock on block containing the data item
  - Lock can be released once the write is completed.
    - Such locks held for short duration are called **latches**.
  - Before a block is output to disk, the system acquires an exclusive latch on the block
    - Ensures no update can be in progress on the block
- Notice that latches and locks are not necessarily the same
  - One can lock at very fine granularity but in case of writing to disk, it still needs latches on the block (page)

Redo/Undo in ARIES

- **Physiological redo**
  - Affected page is physically identified, action within page can be logical
    - Used to reduce logging overheads
      - e.g. when a record is deleted and all other records have to be moved to fill hole
        - Physiological redo can log just the record deletion
        - Physical redo would require logging of old and new values for much of the page

Redo/Undo in ARIES

- Implications:
  - Requires page to be output to disk atomically
    - Easy to achieve with hardware RAID, also supported by some disk systems
    - Incomplete page output can be detected by checksum techniques,
      - But extra actions are required for recovery
      - Treated as a media failure
  - Redo/undo operations not necessary idempotent
    - Thus using LSN and compensation log records to avoid redo/undo again
  - On the other hand, this enable finer grain locking and other fancy operations to be recovered.

Data structures in ARIES

- Extra data structure maintained during normal operations
  - To enhance efficiency during recovery
  - To allow easier checkpointing
- **DirtyPageTable**
  - List of pages in the buffer that have been updated
  - Contains, for each such page
    - **PageLSN** of the page
    - **RecLSN** is an LSN such that log records before this LSN have already been applied to the page version on disk
      - Set to current end of log when a page is inserted into dirty page table (just before being updated)
      - Recorded in checkpoints, helps to minimize redo work

Data structures in ARIES

- Transaction table
  - Keep track of current active transactions
  - Maintain the prevLSN of each transaction
  - Also keep the UndoNxtLSN in case of recovery

Fuzzy checkpoints

- Asynchronous checkpointing
  - i.e. processing do not stop during checkpointing
- Start by writing a <begin chkpt> record to the log
- Then construct a <end chkpt> record containing
  - Transaction table
  - Dirty page table
- Write the <end chkpt> record to stable storage
- Then write the LSN of the <begin chkpt> record to some stable storage
- Normal processing are allowed between writing of the <begin chkpt> and <end chkpt> record

ARIES : Normal operation

- During normal operations, when updates to a record on a page occurs
  1. Record is locked
  2. Page is latched in the X mode
  3. Log record is written
  4. LSN of the log record is placed on transaction table
  5. Update is performed
  6. pageLSN of the page is updated
  7. Page is unlatched

ARIES : Normal operation

- Page latching before writing log is crucial
  - Guarantees LSN corresponds to the order of updates (if locking is at a finer level than page)
- On the other hand, in cases where lock granularity is page (or coarser) and strict 2-phase locking is used, then latches are not necessary
- Fuzzy checkpoints are made periodically

ARIES : Recovery

ARIES recovery involves three passes

- Analysis pass: Determines
  - Which transactions to undo
  - Which pages were dirty (disk version not up to date) at time of crash
  - RedoLSN: LSN from which redo should start
- Redo pass:
  - Repeats history, redoing all actions from RedoLSN
    - RecLSN and PageLSNs are used to avoid redoing actions already reflected on page
- Undo pass:
  - Rolls back all incomplete transactions
    - Transactions whose abort was complete earlier are not undone
      - Key idea: no need to undo these transactions: earlier undo actions were logged, and are redone as required

ARIES : Recovery : Analysis

Analysis pass

1. Starts from last complete checkpoint log record
   - Reads in DirtyPageTable from log record
   - Sets RedoLSN = min of RecLSNs of all pages in DirtyPageTable
     - In case no pages are dirty, RedoLSN = checkpoint record's LSN
   - Sets undo-list = list of transactions in checkpoint log record
   - Reads LSN of last log record for each transaction in undo-list from checkpoint log record

ARIES : Recovery : Analysis

1. Scans forward from checkpoint
   - If any log record found for transaction not in undo-list, adds transaction to undo-list
   - Whenever an update log record is found
     - If page is not in DirtyPageTable, it is added with RecLSN set to LSN of the update log record
   - If transaction end log record found, delete transaction from undo-list
   - Keeps track of last log record for each transaction in undo-list
     - May be needed for later undo

ARIES : Recovery : Analysis

- At end of analysis pass:
  - RedoLSN determines where to start redo pass
  - RecLSN for each page in DirtyPageTable used to minimize redo work
  - All transactions in undo-list need to be rolled back

ARIES : Recovery : Redo

Redo Pass: Repeats history by replaying every action not already reflected in the page on disk, as follows:

- Scans forward from RedoLSN. Whenever an update log record is found:
  1. If the page is not in DirtyPageTable or the LSN of the log record is less than the RecLSN of the page in DirtyPageTable, then skip the log record
  2. Otherwise fetch the page from disk. If the PageLSN of the page fetched from disk is less than the LSN of the log record, redo the log record

  NOTE: if either test is negative the effects of the log record have already appeared on the page. First test avoids even fetching the page from disk!

ARIES : Recovery : Undo

Undo pass

- Performs backward scan on log undoing all transaction in undo-list
  - Backward scan optimized by skipping unneeded log records as follows:
    - Next LSN to be undone for each transaction set to LSN of last log record for transaction found by analysis pass.
    - At each step pick largest of these LSNs to undo, skip back to it and undo it
    - After undoing a log record
      - For ordinary log records, set next LSN to be undone for transaction to PrevLSN noted in the log record
      - For compensation log records (CLRs) set next LSN to be undo to UndoNextLSN noted in the log record
        - All intervening records are skipped since they would have been undo already
- Undos performed as described earlier

ARIES : Recovery : Undo/redo

Undo/Redo pass

- Note that pageLSN is updated during recovery
  - E.g. when log record 8 is being redo, the corresponding pageLSN is set to 8.
  - When the page is flushed to the disk, the new pageLSN will denote that the page is already redone
- Important to ensure undo/redo not repeated unnecessarily.

ARIES : Crash during recovery

- Crash during analysis
  - No harm done. Restart analysis
- Crash during redo
  - Repeat whole process.
  - Use pageLSN to avoid unnecessary redo
- Crash during undo
  - During redo stage, redo both action and CLR if necessary
  - Once again, use pageLSN to see what action need to be undone

ARIES : Example

T #	Prev LSN	Undonext LSN

Transaction table

Page #	RecLSN	PageLSN

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	**1**	**0**

Transaction table

Page #	RecLSN	PageLSN
1	**0**	**1**

Dirty page table

1. **T1 write page 1**
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	1	0
2	**2**	**0**

Transaction table

Page #	RecLSN	PageLSN
1	0	1
2	**1**	**2**

Dirty page table

1. T1 write page 1
2. **T2 write page 2**
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	**3**	0
2	2	0

Transaction table

Page #	RecLSN	PageLSN
1	0	**3**
2	1	2

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. **T1 write page 1**
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
2	2	0
3	**4**	**0**

Transaction table

Page #	RecLSN	PageLSN
1	0	3
2	1	2
4	**3**	**4**

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. **T3 write page 4**

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
2	2	0
3	4	0

Transaction table

Page #	RecLSN	PageLSN
~~1~~	~~0~~	~~3~~
2	1	2
4	3	4

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

PageLSN of 1 on disk = 3

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
~~2~~	~~2~~	~~0~~
3	4	0

Transaction table

Page #	RecLSN	PageLSN
2	1	2
4	3	4

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. **T2 commits**
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
3	4	0

Transaction table

Page #	RecLSN	PageLSN
2	1	2
4	3	4

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. **Begin Checkpoint**
3. **End Checkpoint**
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
3	4	0
4	**8**	**0**

Transaction table

Page #	RecLSN	PageLSN
2	1	2
4	3	4
3	**6**	**7**

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. **T4 write page 3**

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
3	4	0
4	7	0

Transaction table

Page #	RecLSN	PageLSN
2	1	2
~~**4**~~	~~**3**~~	~~**4**~~
3	6	7

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. T1 writes page 4

Crash!

PageLSN of 4 on disk = 4

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
3	**9**	0
4	7	0

Transaction table

Page #	RecLSN	PageLSN
2	1	**9**
3	6	7

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. **T3 write page 2**
2. T3 commits
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	3	0
3	**9**	**0**
4	7	0

Transaction table

Page #	RecLSN	PageLSN
2	1	9
3	6	7

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. **T3 commits**
3. T1 writes page 4

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	**11**	0
4	7	0

Transaction table

Page #	RecLSN	PageLSN
2	1	8
4	**10**	**11**
3	6	7

Dirty page table

1. T1 write page 1
2. T2 write page 2
3. T1 write page 1
4. T3 write page 4

(Page 1 flushed to disk)

1. T2 commits
2. Begin Checkpoint
3. End Checkpoint
4. T4 write page 3

(Page 4 flushed to disk)

1. T3 write page 2
2. T3 commits
3. **T1 writes page 4**

Crash!

ARIES : Example

T #	Prev LSN	Undonext LSN
1	10	0
4	7	0

Transaction table

Page #	RecLSN	PageLSN
2	1	8
4	**9**	**10**
3	6	7

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>

Notation: <Trans#, page #, PrevLSN>

ARIES : Example (Analysis: step 1)

T #	Last LSN	Undonext LSN
1	3	
3	4	

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>

- RedoLSN = min(1, 3) = 1
- Undo-list = {T1, T3}

ARIES : Example (Analysis: step 2)

T #	Last LSN	Undonext LSN
1	3	0
3	4	0
4	**8**	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	**7**	--

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. **<T4, 3, ->**
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>

- RedoLSN = min(1, 3, 8) = 1
- Undo-list = {T1, T3, **T4**}

ARIES : Example (Analysis: step 2)

T #	Last LSN	Undonext LSN
1	3	0
3	**9**	0
4	8	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	7	--

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

١. <begin_chkpt>
٢. <end_chkpt>
٣. <T4, 3, ->
٤. **<T3, 2, 4>**
٥. <T3, commits>
٦. <T1, 4, 3>

- RedoLSN = min(1, 3, 8) = 1
- Undo-list = {T1, T3, T4}

ARIES : Example (Analysis: step 2)

T #	Last LSN	Undonext LSN
1	3	0
~~**3**~~	~~**9**~~	~~**0**~~
4	8	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	7	--

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. **<T3, commits>**
6. <T1, 4, 3>

- RedoLSN = min(1, 3, 8) = 1
- Undo-list = {T1, ~~**T3**,~~ T4}

ARIES : Example (Analysis: step 2)

T #	Last LSN	Undonext LSN
1	**11**	0
4	8	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	7	--

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. **<T1, 4, 3>**

- RedoLSN = min(1, 3, 8) = 1
- Undo-list = {T1, T4}

ARIES : Example (Redo)

T #	Last LSN	Undonext LSN
1	11	0
4	8	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	7	--

Dirty page table

Log at crash

No redo, page not in dirty page table

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>

- RedoLSN = min(1, 3, 8) = 1
- Redo start at first step

ARIES : Example (Redo)

T #	Last LSN	Undonext LSN
1	11	0
4	8	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	7	--

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, -> ← 2 > 1, thus read page 2 PageLSN 2 = 0, thus redo
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>

- RedoLSN = min(1, 3, 8) = 1
- Redo start at first step

ARIES : Example (Redo)

T #	Last LSN	Undonext LSN
1	11	0
4	8	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	7	--

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, -> ←
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>

4 > 3, read page 4
But PageLSN = 4, so NO redo

- RedoLSN = min(1, 3, 8) = 1
- Redo start at first step

ARIES : Example (Redo)

T #	Last LSN	Undonext LSN
1	11	0
4	8	0

Undo list

Page #	RecLSN	PageLSN
2	1	3
4	3	4
3	7	--

Dirty page table

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. **<T4, 3, ->**
4. **<T3, 2, 4>**
5. <T3, commits>
6. **<T1, 4, 3>**

Other operations need redo

- RedoLSN = min(1, 3, 8) = 1
- Redo start at first step

ARIES : Example (Undo)

T #	Last LSN	Undonext LSN
1	11	0
4	8	0

Undo list

- Next record to undo = max(11, 8) = 11
- CLR written
- Last LSN T1 = prevLSN of record 11 = 3

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. **<T1, 4, 3>**
7. **<CLR, T1, 4, 3>**

Notation for CLR: <CLR, transaction #, page #, nextundoLSN>

ARIES : Example (Undo)

T #	Last LSN	Undonext LSN
1	3	0
4	8	0

Undo list

- Next record to undo = max(3, 8) = 3
- CLR written
- Last LSN = "-", T4 removed from undo list

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. **<T4, 3, ->**
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>
7. <CLR, T1, 4, 3>
8. **<CLR, T4, 3, ->**

ARIES : Example (Undo)

T #	Last LSN	Undonext LSN
1	3	0

Undo list

- Next record to undo = 3
- CLR written
- Last LSN set to prevLSN of recotd = 1

Log at crash

- <T1, 1, ->
- <T2, 2, ->
- **<T1, 1, 1>**
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>
7. <CLR, T1, 4, 3>
8. <CLR, T4, 3, ->
9. **<CLR, T1, 1, 1>**

ARIES : Example (Undo)

T #	Last LSN	Undonext LSN
1	1	0

Undo list

- Next record to undo = 1
- CLR written
- Last LSN = "-". T1 removed
- No transaction left, recovery finished.

Log at crash

- **<T1, 1, ->**
- <T2, 2, ->
- <T1, 1, 1>
- <T3, 4, ->
- <T2 commits>

1. <begin_chkpt>
2. <end_chkpt>
3. <T4, 3, ->
4. <T3, 2, 4>
5. <T3, commits>
6. <T1, 4, 3>
7. <CLR, T1, 4, 3>
8. <CLR, T4, 3, ->
9. <CLR, T1, 1, 1>
10. **<CLR, T1, 1, ->**

ARIES : Other features

- Recovery Independence
  - Pages can be recovered independently of others
    - E.g. if some disk pages fail they can be recovered from a backup while other pages are being used
- Savepoints:
  - Transactions can record savepoints and roll back to a savepoint
    - Useful for complex transactions
    - Also used to rollback just enough to release locks on deadlock

ARIES : Other features

- Fine-grained locking:
  - Index concurrency algorithms that permit tuple level locking on indices can be used
    - These require logical undo, rather than physical undo, as in advanced recovery algorithm
- Recovery optimizations: For example:
  - Dirty page table can be used to prefetch pages during redo
  - Out of order redo is possible:
    - redo can be postponed on a page being fetched from disk, and performed when page is fetched.
    - Meanwhile other log records can continue to be processed