CompSci 516
Database Systems

Lecture 17
Transactions – Recovery
(ARIES)

Instructor: Sudeepa Roy

Announcements

• Midterm report due on Wednesday, 11/01
• HW3 to be released soon
  – will be due in ~2 weeks after it is released

Reading Material

• Main reading
  “Concurrency Control and Recovery”
  Michael Franklin, 1997
  – Chapters 2.2, 3.2
  – A very good read for other topics on transactions as well
  – pdf on the website

• [RG]
  – Chapter 18.1-18.6

Acknowledgement: Slides by Prof. Magda Balazinska were consulted to prepare some of these slides.

Last Lecture

• UNDO
• REDO
• UNDO/REDO
• Checkpointing
• Recovery

• The main concepts of these logs were discussed

Today

• Last topic in recovery

• Physical/Logical/Physiological Logging

• ARIES protocol by IBM
  – An efficient implementation of UNDO/REDO log
  – Including the data structures that are used for the recovery

Log: old/new concepts

• Each entry in the log is called a log record
  – e.g. recall <T, X, u, v> for undo/redo log

• When a log record is created, it is assigned a unique Log Sequence Number (LSN)
  – typically, monotonically increasing to provide relative position in the log

• Update to a data item in buffer:
  – a log record is created
  – Many systems write the LSN of this log record into the page containing the data item
  • relates the state of a data page to logged updates

• How do these log records look in practice?
WAL for UNDO/REDO log

- Write Ahead Log (WAL)
  - from Lecture 16

- All log records for an update are first written to disk before the update (the modified page) is written to disk

- A transaction is not considered committed, until all its log records + COMMIT record are on disk

- Allows STEAL + NO FORCE (good!)

Physical Logging

- Physical log records indicate location of modified data in the database
  - e.g. position on a particular page
  - if a new tuple is inserted in a relation, log records may contain changes for
    - space allocation
    - index updates
    - reorganization etc.

Logical Logging

- Logical log records indicate high level info about operations performed
  - if a new tuple is inserted in a relation, log records may indicate
    - the insertion has taken place
    - value of the inserted tuple

Physical vs. Logical Logging

Logical logging

- Advantages
  - Minimizes the amount of data that must be written to the log
  - Hides many implementation details and complex operations under UNDO/REDO logic

- Disadvantages
  - Difficult to implement, as logging operations may not be atomic

Physiological Logging

- Log records are constrained to refer to a single page
  - but may reflect logical operations on that page

- e.g. for insert on a page,
  - specify the value of the tuple that is inserted
  - do not specify any free-space manipulation

- Tradeoff between physical and logical
  - atomic like physical and less logging records like logical

ARIES protocol

- A detailed but simple implementation details of logging protocol
  - Developed in IBM, but now used in many DBMS
ARIES: Main Ideas

- **Write Ahead Logging (WAL)**
  - Before an update goes to the disk, the log record for that update must go to the disk

- **Physiological Logging**

- **Page-oriented REDO**
  - REDO operations involve pages
  - The affected page is specified in the log record

- **Logical UNDO**
  - Operations performed to undo an update do not need to be the exact inverses of the operations of the original update
  - E.g., if T1 updates an index entry in page P1, later P1 is split (dynamic hashing!) and entry is moved to P2, we do not want to do UNDO operation on P1, but on the new location P2

ARIES Data Structures

**Buffer Pool**

- Contains multiple pages
- Each page contains a pageLSN = the LSN of the log record for the latest update to the page
  - Used during recovery to determine whether or not an update for a page has to be UNDONE
  - Also determines the point in the log from which the REDO pass must commence during recovery

**Transaction Table**

- Contains status information about each transaction that is currently running
  - transID (later tID)
    - Unique transaction ID
  - lastLSN for each transaction
    - LSN of the most recent log record written by the transaction
  - Status
    - Running/Committed/Aborted/…
    - Unknown (while recovery)

**Dirty Page Table**

- Contains an entry for each dirty page
- Dirty page = contains an update that is not written to disk yet
  - recoveryLSN
    - LSN of the earliest log record that might need to be "REDO"ne for the page during restart

- Recall: We care about dirty pages in memory only for REDO, not for UNDO

**Log**

- Contains an entry for each log record
- Log entry
  - Actual changes
    - E.g., WRITE A: "ab" -> "cd"

- Note: ARIES is UNDO/REDO
  - Maintains both previous and new value
  - I.e., everything in \(<T, A, u, v>\) is being maintained
ARIES Data Structures: Log

prevLSN
- Log records belonging to the same transaction are linked backwards in time using a field in each log record
- when a new log record is written
  - the value of the lastLSN field from the Transaction table is written as prevLSN
  - new record’s LSN is entered as lastLSN in the Transaction table

LSN

Transaction table

Log entry

ARIES Data Structures: Log

Type
- Update
  - e.g. WRITE A : "ab" -> "cd"
- Commit
- Abort
- END
- CLR
- Details later

Checkpoints in ARIES

- Checkpoints are periodically taken
- ARIES uses a form of fuzzy checkpoint that is extremely inexpensive
- When a checkpoint is taken
  - a checkpoint record is constructed
  - includes the contents of the Transaction Table and Dirty Page Table
- Checkpoints are efficient
  - no operation is quiesced (stalled)
  - no database pages are flushed to disk from memory!
- But the log that has to be maintained is not much reduced
  - limited in part by the earliest recoveryLSN of the dirty pages at the checkpointing time
  - writing dirty pages periodically to disk might help

Running Example: Maintaining Data Structures

Example actions

Example.
1. T1000 changes the value of A from "abc" to "def" on page P500
2. T1000 changes the value of B from "hij" to "klm" on page P600
3. T1000 changes the value of D from "mnp" to "qrs" on page P505
4. T1000 commits and the END log record is written
5. T1000 changes the value of E from "pq" to "rs" on page P700
6. P600 is flushed to disk
7. Crash!!

Example is adapted from [RG]
### First operation:
1. $T_{200}$ changes the value of $A$ from "abc" to "def" on page P500?

#### Dirty page table
- **PageID**: P500, P600, P700
- **recoveryLSN**: P505
- **PageLSN**: P500, P600, P700
- **Dirty**: True

#### Transaction table
- **transID**: T100
- **lastLSN**: P500
- **Dirty**: True

#### Log
- **pageID**: P500, P600, P700
- **ID**: T100
- **Log entry**: A = def
- **Type**: Update
- **timestamp**: Running

#### Buffer Pool
- **page P500**: B = hij
- **page P600**: B = hij
- **page P700**: E = pq

#### Disk
- **Page P500**: A = def
- **Page P600**: A = def
- **Page P700**: A = def

### Next:
2. $T_{200}$ changes the value of $B$ from "hij" to "klm" on page P600?

#### Dirty page table
- **PageID**: P500, P600, P700
- **recoveryLSN**: P505
- **PageLSN**: P500, P600, P700
- **Dirty**: True

#### Transaction table
- **transID**: T100
- **lastLSN**: P500
- **Dirty**: True

#### Log
- **pageID**: P500, P600, P700
- **ID**: T100
- **Log entry**: A = def
- **Type**: Update
- **timestamp**: Running

#### Buffer Pool
- **page P500**: B = hij
- **page P600**: B = klm
- **page P700**: E = pq

#### Disk
- **Page P500**: A = def
- **Page P600**: A = klm
- **Page P700**: A = def

### Next:
3. $T_{200}$ changes the value of $D$ from "mnp" to "qrs" on page P500?

#### Dirty page table
- **PageID**: P500
- **recoveryLSN**: P505
- **PageLSN**: P500
- **Dirty**: True

#### Transaction table
- **transID**: T100
- **lastLSN**: P500
- **Dirty**: True

#### Log
- **pageID**: P500
- **ID**: T100
- **Log entry**: A = def
- **Type**: Update
- **timestamp**: Running

#### Buffer Pool
- **page P500**: B = klm
- **page P600**: B = klm
- **page P700**: E = pq

#### Disk
- **Page P500**: A = def
- **Page P600**: A = klm
- **Page P700**: A = def

### Changes:
1. $T_{200}$ changes the value of $A$ from "abc" to "def" on page P500

#### Dirty page table
- **PageID**: P500
- **recoveryLSN**: P505
- **PageLSN**: P500
- **Dirty**: True

#### Transaction table
- **transID**: T100
- **lastLSN**: P500
- **Dirty**: True

#### Log
- **pageID**: P500
- **ID**: T100
- **Log entry**: A = def
- **Type**: Update
- **timestamp**: Running

#### Buffer Pool
- **page P500**: B = hij
- **page P600**: B = hij
- **page P700**: E = pq

#### Disk
- **Page P500**: A = def
- **Page P600**: A = def
- **Page P700**: A = def

### Changes:
2. $T_{200}$ changes the value of $B$ from "hij" to "klm" on page P600

#### Dirty page table
- **PageID**: P500, P600, P700
- **recoveryLSN**: P505
- **PageLSN**: P500, P600, P700
- **Dirty**: True

#### Transaction table
- **transID**: T100
- **lastLSN**: P500
- **Dirty**: True

#### Log
- **pageID**: P500, P600, P700
- **ID**: T100
- **Log entry**: A = def
- **Type**: Update
- **timestamp**: Running

#### Buffer Pool
- **page P500**: B = hij
- **page P600**: B = klm
- **page P700**: E = pq

#### Disk
- **Page P500**: A = def
- **Page P600**: A = klm
- **Page P700**: A = def

### Changes:
3. $T_{200}$ changes the value of $D$ from "mnp" to "qrs" on page P500

#### Dirty page table
- **PageID**: P500
- **recoveryLSN**: P505
- **PageLSN**: P500
- **Dirty**: True

#### Transaction table
- **transID**: T100
- **lastLSN**: P500
- **Dirty**: True

#### Log
- **pageID**: P500
- **ID**: T100
- **Log entry**: A = def
- **Type**: Update
- **timestamp**: Running

#### Buffer Pool
- **page P500**: B = klm
- **page P600**: B = klm
- **page P700**: E = pq

#### Disk
- **Page P500**: A = def
- **Page P600**: A = klm
- **Page P700**: A = def

---

**Note:** The diagrams and tables are placeholders for the actual figures. The text describes the changes made by the transaction and the corresponding updates in the log and page cache.
Next: 4. $T_{TD}$ changes the value of $C$ from "tuv" to "wxy" on page P505?

Dirty page table

<table>
<thead>
<tr>
<th>pageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>transactionSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td>103</td>
<td>101</td>
<td>TID005</td>
<td>P600</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
<tr>
<td>P600</td>
<td>102</td>
<td>102</td>
<td>TID005</td>
<td>P500</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
</tbody>
</table>

Transaction table

<table>
<thead>
<tr>
<th>transID</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>100</td>
<td>Running</td>
</tr>
</tbody>
</table>

Changes: 4. $T_{TD}$ changes the value of $C$ from "tuv" to "wxy" on page P505?

Dirty page table

<table>
<thead>
<tr>
<th>pageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>transactionSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td>103</td>
<td>101</td>
<td>TID005</td>
<td>P600</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
<tr>
<td>P600</td>
<td>102</td>
<td>102</td>
<td>TID005</td>
<td>P500</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
</tbody>
</table>

Transaction table

<table>
<thead>
<tr>
<th>transID</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>100</td>
<td>Running</td>
</tr>
</tbody>
</table>

Next: 5. $T_{TD}$ commits and the end log record is written

Dirty page table

<table>
<thead>
<tr>
<th>pageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>transactionSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td>103</td>
<td>101</td>
<td>TID005</td>
<td>P600</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
<tr>
<td>P600</td>
<td>102</td>
<td>102</td>
<td>TID005</td>
<td>P500</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
</tbody>
</table>

Transaction table

<table>
<thead>
<tr>
<th>transID</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>100</td>
<td>Running</td>
</tr>
</tbody>
</table>

Changes: 5. $T_{TD}$ commits and the end log record is written — step 1

Dirty page table

<table>
<thead>
<tr>
<th>pageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>transactionSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td>103</td>
<td>101</td>
<td>TID005</td>
<td>P600</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
<tr>
<td>P600</td>
<td>102</td>
<td>102</td>
<td>TID005</td>
<td>P500</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
</tbody>
</table>

Transaction table

<table>
<thead>
<tr>
<th>transID</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>100</td>
<td>Running</td>
</tr>
</tbody>
</table>

Changes: 5. $T_{TD}$ commits and the end log record is written — step 2

Dirty page table

<table>
<thead>
<tr>
<th>pageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>transactionSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td>103</td>
<td>101</td>
<td>TID005</td>
<td>P600</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
<tr>
<td>P600</td>
<td>102</td>
<td>102</td>
<td>TID005</td>
<td>P500</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
</tbody>
</table>

Transaction table

<table>
<thead>
<tr>
<th>transID</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>100</td>
<td>Running</td>
</tr>
</tbody>
</table>

Changes: 5. $T_{TD}$ commits and the end log record is written — step 3

Dirty page table

<table>
<thead>
<tr>
<th>pageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>transactionSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td>103</td>
<td>101</td>
<td>TID005</td>
<td>P600</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
<tr>
<td>P600</td>
<td>102</td>
<td>102</td>
<td>TID005</td>
<td>P500</td>
<td>PageLSN -</td>
<td>A = def</td>
<td>D = mnp</td>
<td>B = hij</td>
</tr>
</tbody>
</table>

Transaction table

<table>
<thead>
<tr>
<th>transID</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>100</td>
<td>Running</td>
</tr>
</tbody>
</table>

Note: Page P505 is removed from the transaction table.
• Whenever a transaction commits, log is flushed to the disk: i.e the “log-tail” (whatever is not on disk) is written to disk. Assume a “force-write” of log after “commit” is written

The dirty pages are not needed to be flushed to disk (NO-FORCE)

NOTE:
1. The “Commit” record is required to be flushed (i.e. all logs up to and including that commit record)
2. The “End” record is not required to be flushed, in this case we are only assuming that it has been flushed as well (so that we have a good example while doing recovery)

Log Record “Types”

• Update: standard
• Commit: log-tail forced-written to disk, up to & including commit (note that still no-force, the actual modified pages may not be written, and much smaller cost)
• Abort: abort type log record is written + undo is initiated for this transaction
• End: when a transaction is aborted or committed, some additional actions are performed, after that an end record is written
• CLR: (later)
  Undoing updates (during abort or recovery from crash), for every update record undone, write a CLR (Compensation Log Record)

---

**Log Table**

<table>
<thead>
<tr>
<th>TransID</th>
<th>PageID</th>
<th>PrevLSN</th>
<th>PID</th>
<th>Log Entry</th>
<th>Type</th>
<th>NextLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1000</td>
<td>P700</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Dirty Pages**

<table>
<thead>
<tr>
<th>PageID</th>
<th>PrevLSN</th>
<th>PID</th>
<th>Log Entry</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>P700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Transaction Table**

<table>
<thead>
<tr>
<th>TransID</th>
<th>PrevLSN</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1000</td>
<td></td>
<td>Running</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Buffer Pool**

Page P700: In use
Page P500: In use
Page P600: In use

**Diagram**

The diagram shows the state of the buffer pool and disk after the transactions have been committed, with the dirty pages flushed to their respective locations.
**Checkpointing at ARIES**

- Like before -- <START CKPT> and <END CKPT>
- Writes
  - Transaction table
  - Dirty page table
- Called “fuzzy checkpointing”
  - Non-quiet: new transactions can start
  - Does not require pages in buffer pool to be written
  - But effectiveness limited to earliest possible "recoveryLSN" in the dirty page table – has to start REDO from there
- Periodically writing dirty pages to disk helps
- After checkpointing, both transaction table and dirty page tables are empty

---

**Crash Recovery: Three Phases - Big Picture**

1. **Analysis**
   - Start from last checkpoint (from C)
   - Go forward until the last log record
   - Figure out which trans. committed since checkpoint, which failed
   - Redo history (redo dirty page table)

2. **UNDO**
   - undo effects of active transactions at crash
   - go in backward direction (until A)

---

**8. CRASH!!**
1. Analysis Pass

- It has a threefold job:
  1. Determines the point in the log at which to start the REDO pass
  2. Determines which pages "could have been" dirty at the time of the crash to avoid unnecessary I/O in the REDO pass — a conservative superset of actual dirty pages
  3. Determines which transactions had not committed at the time of the crash and will therefore need to be UNDOne.

Analysis Pass: Details

- Begin at the most recent checkpoint
- Reconstruct Dirty Page Table and Transaction Table
  - to determine the state of the system at the time of crash
- Scan forward to the end of the log
  - Contents of these two tables are modified according to the log records encountered in the forward scan

Analysis Pass: More Details

- When a log record for a transaction that does not appear in the Transaction Table is encountered
  - that transaction is added to the transaction table
- When an END record is encountered
  - that transaction is removed from the transaction table
- When an UPDATE log record for a page not in the Dirty Page Table is encountered
  - that page is added to the dirty page table
  - LSN of the record is recorded as recoveryLSN for that page
  - LastLSN is modified
- All like before!

Checkpointing in the example

- This example has no checkpointing == Checkpointing at the beginning
- Analysis phase in the recovery starts with empty Dirty Page table and empty Transaction Table
  - If checkpoint was available, the latest copies of these tables have to be read from disk from the last checkpoint

Running Example: Analysis Pass
### Analysis Pass

#### Log

<table>
<thead>
<tr>
<th>PageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>PID</th>
<th>PID entry</th>
<th>Type</th>
<th>comment/Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>P505</td>
<td></td>
<td>101</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>102</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>103</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>104</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>105</td>
<td>T1000</td>
<td>P500</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>106</td>
<td>T2000</td>
<td>P500</td>
<td>End</td>
<td></td>
</tr>
</tbody>
</table>

#### Dirty page table

<table>
<thead>
<tr>
<th>PageID</th>
<th>PID</th>
<th>PageLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P505</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
</tbody>
</table>

### Analysis Pass

#### Log

<table>
<thead>
<tr>
<th>PageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>PID</th>
<th>PID entry</th>
<th>Type</th>
<th>comment/Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>P505</td>
<td></td>
<td>101</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>102</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>103</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>104</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>105</td>
<td>T1000</td>
<td>P500</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>106</td>
<td>T2000</td>
<td>P500</td>
<td>End</td>
<td></td>
</tr>
</tbody>
</table>

#### Dirty page table

<table>
<thead>
<tr>
<th>PageID</th>
<th>PID</th>
<th>PageLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P505</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
</tbody>
</table>

### Analysis Pass

#### Log

<table>
<thead>
<tr>
<th>PageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>PID</th>
<th>PID entry</th>
<th>Type</th>
<th>comment/Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>P505</td>
<td></td>
<td>101</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>102</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>103</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>104</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>105</td>
<td>T1000</td>
<td>P500</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>106</td>
<td>T2000</td>
<td>P500</td>
<td>End</td>
<td></td>
</tr>
</tbody>
</table>

#### Dirty page table

<table>
<thead>
<tr>
<th>PageID</th>
<th>PID</th>
<th>PageLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P505</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
</tbody>
</table>

### Analysis Pass

#### Log

<table>
<thead>
<tr>
<th>PageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>PID</th>
<th>PID entry</th>
<th>Type</th>
<th>comment/Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>P505</td>
<td></td>
<td>101</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>102</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>103</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>104</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>105</td>
<td>T1000</td>
<td>P500</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>106</td>
<td>T2000</td>
<td>P500</td>
<td>End</td>
<td></td>
</tr>
</tbody>
</table>

#### Dirty page table

<table>
<thead>
<tr>
<th>PageID</th>
<th>PID</th>
<th>PageLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P505</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
</tbody>
</table>

### Analysis Pass

#### Log

<table>
<thead>
<tr>
<th>PageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>PID</th>
<th>PID entry</th>
<th>Type</th>
<th>comment/Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>P505</td>
<td></td>
<td>101</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>102</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>103</td>
<td>T1000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>104</td>
<td>T2000</td>
<td>P500</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td>105</td>
<td>T1000</td>
<td>P500</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td>106</td>
<td>T2000</td>
<td>P500</td>
<td>End</td>
<td></td>
</tr>
</tbody>
</table>

#### Dirty page table

<table>
<thead>
<tr>
<th>PageID</th>
<th>PID</th>
<th>PageLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P505</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P500</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
<tr>
<td>P600</td>
<td></td>
<td></td>
<td>Disk</td>
</tr>
</tbody>
</table>

Write A or Abort if you see an Abort log instead

---

CompSci 516: Database Systems

Page 10/29/17
### Analysis Pass

#### Log

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>undoNextLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>T1000</td>
<td>P500</td>
<td></td>
<td>Write A</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td></td>
<td>P505</td>
<td></td>
<td>Write D</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td></td>
<td>P500</td>
<td></td>
<td>Write B</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td></td>
<td>P500</td>
<td></td>
<td>Write C</td>
<td>Update</td>
<td></td>
</tr>
</tbody>
</table>

#### Transaction Table

<table>
<thead>
<tr>
<th>transID</th>
<th>timestamp</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1000</td>
<td>2010-01-01</td>
<td>Running</td>
</tr>
<tr>
<td>T1001</td>
<td>2010-01-02</td>
<td>Running</td>
</tr>
</tbody>
</table>

#### Data Page

- Page ID: P500
- Page SN: 103
- A = def
- D = mnp

### Analysis Pass

#### Log

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>ID</th>
<th>pID</th>
<th>Log entry</th>
<th>Type</th>
<th>undoNextLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>T1000</td>
<td>P500</td>
<td></td>
<td>Write A</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td></td>
<td>P505</td>
<td></td>
<td>Write D</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td></td>
<td>P500</td>
<td></td>
<td>Write B</td>
<td>Update</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td></td>
<td>P500</td>
<td></td>
<td>Write C</td>
<td>Update</td>
<td></td>
</tr>
</tbody>
</table>

#### Transaction Table

<table>
<thead>
<tr>
<th>transID</th>
<th>timestamp</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1000</td>
<td>2010-01-01</td>
<td>Running</td>
</tr>
<tr>
<td>T1001</td>
<td>2010-01-02</td>
<td>Running</td>
</tr>
</tbody>
</table>

#### Data Page

- Page ID: P500
- Page SN: 103
- A = def
- D = mnp

### ARIES: REDO Pass

#### REDO in ARIES = Repeating History

- REDO updates for all transactions
  - committed as well as for transactions to be aborted in UNDO
  - both "UPDATE" and "CLR" records (later)

- at the end of REDO, the database would be in the same state w.r.t. logged updates at the time of crash
**When REDO is NOT needed**

1. If the affected page is not in the Dirty Page Table
   - all changes to this page was written to disk
2. Otherwise, the page's recoveryLSN > LSN of the record being checked
   - RecoverLSN is the first update to a page that may not have been written to disk
   - current update has gone to disk
3. Otherwise, the pageLSN >= LSN of the record being checked
   - may need to load the page from disk (stored on the page)
   - checked last because it needs a page I/O
   - either this update or a later update was already written to disk

**ARIES: REDO Pass - details**

- Work with the Dirty Page Table
- Find the smallest recoveryLSN in the dirty page table = firstLSN
  - from the analysis phase
- Redo the "Update" (and "CLR" - later) actions, unless (in this order):
  - Affected page is not in the dirty page table
  - OR, recoveryLSN > LSN being checked
  - OR, pageLSN >= LSN being checked
- End/Commit/Abort LSNs are "skipped"

**State After Analysis Pass**

**Running Example: REDO Pass**

**REDO Pass: find firstLSN**

**REDO: Step 1**
**ARIES: UNDO Pass**

- Scan backward from the end of the log
- All transactions that have not committed at the time of the crash, should be undone
- UNDO is an unconditional operation on ARIES
  - i.e. the pageLSN s not checked because always the UNDO has to be done
  - Can do this because of the prior REDO phase – applied all logged updates to the page

**Compensation Log Record (CLR) and UndoNxtLSN**

- CLR is added after an update is undone
  - so that no “Undo” action is undone
  - e.g. as the result of a system crash during an abort
- UndoNxtLSN
  - additional field for CLR
  - LSN of the next log record that must be undone for the transaction
  - set to the value of the prevLSN of the log record being undone
Recall: ARIES Method Illustration

![Image of ARIES Method Illustration]

Running Example: UNDO Pass

**UND0 Pass**

- Efficient implementation:
  - Maintain a set ToUndo

- Initialize to lastLSNs of all “U” (unknown) transactions at Transaction Table
- undo the “largest LSN” in ToUndo at each step (the latest one in bottom-up order)

**Dirty page table**

<table>
<thead>
<tr>
<th>pageID</th>
<th>recoveryLSN</th>
<th>LSN</th>
<th>prevLSN</th>
<th>HD</th>
<th>pD</th>
<th>Log entry</th>
<th>Type</th>
<th>undoNextLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P505</td>
<td>101</td>
<td>104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P506</td>
<td>101</td>
<td>104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P507</td>
<td>101</td>
<td>104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Log**

<table>
<thead>
<tr>
<th>LSN</th>
<th>prevLSN</th>
<th>HD</th>
<th>pD</th>
<th>Log entry</th>
<th>Type</th>
<th>undoNextLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>107</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ToUndo = [101]**

- A CLR is written
- PageLSN = LSN (CLR)
- Value of C is undone

**Disk**

ToUndo = [101]
When a CLR is encountered during backward scan...

- No operation is performed on the page
  - backward scan continues to the log record pointed to by UndoNxtLSN
  - “jump over” undone update and all other updates for the transactions already undone
  - does not undo an “UNDO”

Use of CLR in UNDO

See the details in Franklin97-Transaction.pdf (3.2.4)

If some CLR records are written to disk during an UNDO phase, then a crash happens (e.g. here LSN 40, 50 are written to disk before the second crash), then the next UNDO phase will skip undoing those CLRs.

Note: REDO re-does CLRs!