Recovery Notes (Richard)

11/23/15 1:05 AM

Notes include text from *Database Management Systems*, 2/e by Raghu Ramakrishnan and Johannes Gehrke.

There is a lot of text below, but I think it should easy to read and follow.

General Recovery Information:

When our system crashes, our buffer pool is empty; we lost all our temporary data. The only things we have from disk. Our **recovery manager** takes care of us getting back up and running.

There are two decisions we need to make: Steal vs No-Steal, Force vs No-force. Below is an outline of the choices we have.

Force/No Force:

- Force: When a transaction commits, it makes sure all of the pages it updated (dirty pages) are flushed into disk (so that the data is persisted).
- No Force: When a transaction commits, it does not care if the pages it updated are flushed into disk (so that the data is persisted).

Steal/No Steal

- Steal: When a transaction is running, it can move dirty pages from the buffer pool to disk.
- No Steal: When a transaction is running, it cannot move dirty pages from the buffer pool to disk.

Most systems use a steal/no-force approach. ARIES uses Steal and No Force.

Log Records:

MUST HAVE THE FOLLOWING: LSN, prevLSN, TID, Type, (maybe more)

See textbook for more details.

Type	**Other Notes**
ABORT	After logging this, Rollback (UNDO) will be initiated.
COMMIT	See textbook. TLDR begins writing some part log to disk.
END	See textbook. TLDR ends transaction.
UPDATE	The pageLSN of the page is also set to the LSN of the update log record.
CLR (Compensation Log Record)	SEE TEXTBOOK FOR THIS – VERY DESCRIPTIVE. (TLDR, contains both the LSN of the update it is reverting AND has a undoNextLSN field telling what is the next thing to undo. Note how undoNextLSN builds *backwards*)
BEGIN_CHECKPOINT	Takes snapshot of DP table, Transaction Table
END_CHECKPOINT	Includes the current contents of the transaction table and the dirty page table, and appended to the log.

Table 1 Log Record Types

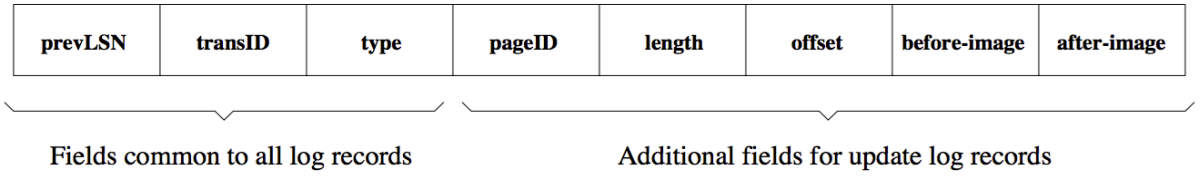

Figure 20.2 Contents of an Update Log Record

Figure 1 Update Record. Before-image is value of changed bytes before change.

ARIES Algorithm

ARIES Key Features:

- WAL: Any change to a database object is first recorded in the log; the record in the **log** must be written to stable storage **before** the change to the database **object** is written to disk.
- Repeats history during REDO: Self-explanatory
- Logging changes during UNDO: *(Think about what this prevents)*

There are three phases to ARIES – Analysis, Redo, and Undo.

Data Structures used (See worksheet):

Dirty Page Table
Transaction Table

Analysis:

1. Find most recent BEGIN_CHECKPOINT log record
   a. Initialize the dirty page table and transaction table from end_checkpoint record
   b. What point of time do these table entries correspond to?
2. Scan Log forward until reaches end of log:
   a. END log record for a transaction T: T is removed from Xact Table
   b. Log record other than an end record for a transaction T: an entry for T is added to the transaction table if it is not already there. Further, the entry for T is modified:
      i. The lastLSN field is set to the LSN of this log record.
      ii. If the log record is a commit record, the status is set accordingly
   c. If (UPDATE/CLR) affecting page P, and P is not in the dirty page table:
      i. an entry is inserted into this table with page id of P and recLSN equal to the LSN of this log record.
      ii. This LSN identifies the oldest change to page P that may not have been written to disk.

Redo Phase:

1. After Analysis, look at the dirty page table and find smallest recLSN.
   a. Identifies the oldest update that may not have been written to disk prior to crash.
2. Scan Log forward until reaching end of log, iterating through log entries:
   a. For each (UPDATE/CLR) encountered, **DO NOT REDO IF**:
      i. The affected page is not in the dirty page table
      ii. The affected page is in the dirty page table, but the recLSN for the entry is greater than the LSN of the log record being checked,
      iii. The pageLSN (stored on the page, which must be retrieved to check this condition) is greater than or equal to the LSN of the log record being checked.
   b. **ELSE** we redo:
      i. The logged action is reapplied.
      ii. The pageLSN on the page is set to the LSN of the redone log record. No additional log record is written at this time.
3. Remove all committed transactions from the transaction table and write an "END".

Undo Phase:

1. Begin with transaction table – all transactions left are loser transactions.
2. Consider the set of lastLSN values for all loser transactions = "ToUndo" (a set).
3. While "ToUndo" is not empty:
   a. Choose the largest LSN value in this set and process the corresponding log record.
   b. To process a log record:
      i. If CLR:
         1. If undoNextLSN value is not null, the undoNextLSN value is added to "ToUndo";
         2. If the undoNextLSN is null, an end record is written for the transaction because it is completely undone, and the CLR is discarded.
      ii. If UPDATE:
         1. a CLR is written and the corresponding action is undone, and the prevLSN value in the update log record is added to "ToUndo".
4. When the set ToUndo is empty, the Undo phase is complete.

Restart is now complete, and the system can proceed with normal operations.