CSC 742
Database Management Systems

Topic #15: Recovery

Recovery Concepts

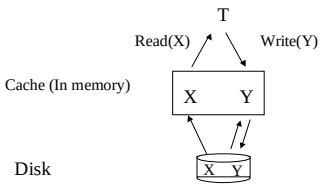
- Recovery
  - Restore the database to the most recent consistent state before the time of failure.
- Two categories
  - Catastrophic failures
    - Loose data from disk
    - Bring backup from archives and redo committed transactions after backup.
  - Non-catastrophic failures
    - Loose data from memory
    - Our focus

Recovery Concepts (Cont'd)

- A hint
  - Two choices of recovery algorithms
    - Undo/no-undo
    - Redo/no-redo

Caching in DBMS

- Disk pages that include the data items are cached in main memory buffers.
  - Update: disk pages are read into cache before being written back.



Caching in DBMS (Cont'd)

- Two strategies to write modified buffer to disk
  - In-place updating
    - Write the back back to the same original disk location
    - Overwrite the old value
  - Shadowing
    - Write an updated buffer at a different disk location.
    - Multiple versions of the same data item may be maintained.

Recovery Concepts (Cont'd)

- Two main techniques
  - Deferred update
    - Don't update the database on disk *until after* the transaction commits
    - What's the benefit?
    - Disadvantage?

Two Main Techniques (Cont'd)

- Immediate update
  - The database *may* be updated before the transaction commits.
  - These operations must be recorded in the log on disk by *force writing* before they are applied to the database.
  - Advantages?
  - Disadvantages?
  - How about redo?

Recovery Concepts (Cont'd)

- Before image (BFIM)
  - The old value of a data item before updating
- After image (AFIM)
  - The new value of a data item after updating
- Write-ahead logging (WAL)
  - The process to ensure that before the AFIM is recorded in the database on disk
    - the BFIM of the data item is recorded in the log,
    - and the log entry is written to disk.
  - Necessary when in-place updating is used
    - Why?

Recovery Concepts (Cont'd)

- Recovery log entries
  - Undo-type log entry
    - Include the BFIM of a data item being updated.
    - Needed for undo.
  - Redo-type log entry
    - Include the AFIM of a data item being updated.
    - Needed for redo.

Recovery Concepts (Cont'd)

- Steal/no-steal approach
  - Steal: updated pages are allowed to be written to disk before the transaction commits
    - Immediate update
  - No-steal: updated pages cannot be written to disk before the transaction commits.
    - Deferred update

Recovery Concepts (Cont'd)

- Force/no-force approach
  - Force: all pages updated by a transaction are immediately written to disk when the transaction commits.
    - Advantage?
    - Disadvantage?
  - No-force: otherwise.
    - Advantage?
    - Disadvantage?

Recovery Concepts (Cont'd)

- Checkpoints
  - A type of log entry.
  - Written into the log when the DBMS writes out to the database on disk all DBMS buffers that have been modified.
  - No redo for committed transactions.
  - How about undo?

Checkpoint (Cont'd)

- Procedures
  - Suspend execution of transactions temporarily.
  - Force-write all main memory buffers that have been modified to disk.
  - Write a [checkpoint] record to the log, and force the log to disk.
  - Resume executing transactions.
- What about deferred update?

Fuzzy Checkpointing

- Maintain a [valid check point] record.
- Procedures
  - Suspend execution of transactions temporarily.
  - Concurrently execute:
    - Force-write all main memory buffers that have been modified to disk.
    - Write a [checkpoint] record to the log, and force the log to disk.
  - Resume executing transactions.
  - Update the [valid check point] record when all of the above finish.

Recovery Concepts (Cont'd)

- Transaction rollback (abort)
  - Any data items changed by the transaction must be restored to their BFIMs.
  - Undo-type log entries are for this purpose.
  - Cascading rollback
    - All transactions that read these items must be aborted as well.
  - Avoid cascading abort

Rollback (Cont'd)

- For strict schedules,
  - no need for reads to be logged (reads are only needed to determine cascade)
  - can extract BFIMs easily from individual entries

Idempotence of Recovery

- Recovery ops must have no additional effect if redone
- Entire recovery procedure must be restartable repeatedly in case of failure during recovery

No-Undo/Redo

- Deferred update
- No-undo/Redo protocol
  - A transaction cannot change the database on disk until it commits.
  - A transaction does not reach its commit point until all its update operations are in the log *and* the log is force-written to disk
- Intuition
  - Write operations on a log
  - Forget if aborted
  - Copy over if committed

No-undo/Redo In A Single-User Environment

- Use two lists of transactions
  - Committed Transactions: since the last check point.
  - Active Transactions: at most one
  - Redo all the write operations of the committed transactions in the order in which they were in the log.

```
[start_transaction, T1]
[write_item, T1, D, 20]
[commit, T1]
[start_transaction, T2]
[write_item, T2, B, 10]
[write_item, T2, D, 25]   ← System crash
```


No-undo/Redo In A Multi-User Environment

- Recovery process depends on concurrency control protocol.
- Assume that strict 2PL is used
  - Recovery process remains the same.


```
[start_transaction, T1]
[write_item, T1, D, 20]
[commit, T1]
[checkpoint]
[start_transaction, T4]
[write_item, T4, B, 15]
[write_item, T4, A, 20]
[commit, T4]
[start_transaction, T2]
[write_item, T2, B, 12]
[start_transaction, T3]
[write_item, T3, A, 30]
[write_item, T2, D, 25]   ← System crash
```

Committed:
Active:

Improvement

- Observation: If a data item X has been updated more than once by committed transactions since the last checkpoint, it is only necessary to redo the last update of X.
- Recovery:
  - Start from the end of the log
  - Whenever an item is redone, it is added to a list of redone items.
  - Before an item is redone, the list is checked.
    - If the item appears on the list, it is ignored.


```
[start_transaction, T1]
[write_item, T1, D, 20]
[commit, T1]
[checkpoint]
[start_transaction, T4]
[write_item, T4, B, 15]
[write_item, T4, A, 20]
[commit, T4]
[start_transaction, T2]
[write_item, T2, B, 12]
[start_transaction, T3]
[write_item, T3, A, 30]
[commit, T3]
[write_item, T2, D, 25]   ← System crash
```

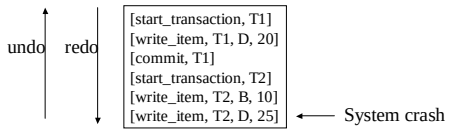

Undo/Redo

Immediate update

- Write on main database
- During recovery,
  - Undo if aborted
  - Redo if committed but not yet recorded

Undo/Redo In A Single-User Environment

- Use two lists of transactions
  - Committed Transactions: since the last check point.
  - Active Transactions: at most one
  - Undo all the write operations of active transactions (in the reverse order in which the operations were written in the log).
  - Redo all the write operations of the committed transactions in the order in which they were in the log.



Undo/Redo In A Multi-User Environment

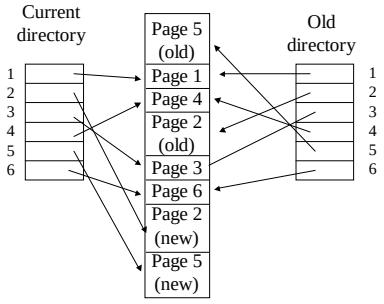
- Recovery process depends on concurrency control protocol.
- Assume that strict 2PL is used
  - Recovery process remains the same.

Undo/No-Redo

- Just like Undo/Redo
- But write all operations before commit (thus no need to redo)
  - A force approach

No-undo/No-redo:

- Shadow paging
  - Consider the database to be made up of a number of fixed-size disk pages
  - A transaction accesses the database through a directory.
    - Current directory
    - Shadow directory



ARIES Recovery Algorithm

- Uses a steal/no-force approach for writing
  - Why?
- Based on three concepts
  - Write-ahead logging
  - Repeating history during redo
    - Retrace all actions of the DBMS prior to the crash to reconstruct the database state when the crash occurred.
    - Uncommitted transactions are undone.
  - Logging changes during undo
    - Prevent ARIES from repeating completed undo operations when a failure occurs during recovery.

ARIES (Cont'd)

- Recovery: three steps
  - Step 1: Analysis
    - Identify updated pages in the buffer
    - Identify active transactions when the crash occurred
    - Identify the point in the log where redo should start
  - Step 2: Redo
    - Redo operations are applied until he end of the log
    - Include writes from uncommitted transactions.
    - Only necessary redo operations are applied.

ARIES Recovery (Cont'd)

- Three steps (Cont'd)
  - Step 3: Undo
    - Log is scanned backward
    - Updates from active transactions are undone.

ARIES Recovery (Cont'd)

- Information sources for ARIES recovery
  - Log
  - Transaction table
  - Dirty page table
- Checkpoint
  - Transaction and dirty page tables are stored in the log at checkpoints.

ARIES Log Entries

- Each log record has a log sequence number (LSN)
  - Monotonically increasing
  - Indicates the address of the log record on disk.
- Logging actions
  - **Write**
  - **Commit**
  - **Abort**
  - **Undo**
  - **Ending a transaction**
- Each log record has the previous LSN for that transaction.
  - Link the log records of the same transaction (in reverse order)

ARIES Recovery (Cont'd)

- Each page is associated with the LSN of the last log record corresponding to a change for that page.
- Transaction table
  - Contains an entry for each active transaction.
  - Rebuild during recovery
- Dirty page table
  - Contains an entry for each dirty page in the buffer.
  - Include the page ID and the LSN corresponding to the earliest update to that page.

ARIES Recovery Process

- Analysis phase
  - Start from the last checkpoint
  - Reconstruct the transaction and the dirty page tables

LSN	LAST_LSN	TRAN_ID	TYPE	PAGE_ID
1	0	T1	UPDATE	C
2	0	T2	UPDATE	B
3	1	T1	COMMIT	
4	Begin_checkpoint			
5	End_checkpoint			
6	0	T3	UPDATE	A
7	2	T2	UPDATE	C
8	7	T2	COMMIT	

Transaction table

TRAN_ID	LAST LSN	STATUS
T1	3	COMMIT
T2	2	ACTIVE

Dirty page table

PAGE_ID	LSN
C	1
B	2

Reconstruct these tables

ARIES Recovery Process (Cont'd)

- Redo phase
  - Determine the starting point for redo
  - The smallest LSN in the dirty page table.
  - Only redo the pages in the dirty page table that with a LSN not greater than the LSN of the log record.
- Undo phase
  - Undo_set: the set of active transactions when the crash occurred.
  - Scan the log backward, undo each update for the transactions in undo_set.

LSN	LAST_LSN	TRAN_ID	TYPE	PAGE_ID
1	0	T1	UPDATE	C
2	0	T2	UPDATE	B
3	1	T1	COMMIT	
4	Begin_checkpoint			
5	End_checkpoint			
6	0	T3	UPDATE	A
7	2	T2	UPDATE	C
8	7	T2	COMMIT	

Transaction table

TRAN_ID	LAST LSN	STATUS
T1	3	COMMIT
T2	2	ACTIVE

Dirty page table

PAGE_ID	LSN
C	1
B	2

Redo phase and undo phase?

Catastrophic Failure

- To account for catastrophic failure
  - backup entire database
  - backup system log more frequently
  - in case of failure, recover the last back and reapply the latest version of the log