Recovery Subsystem

Mirunalini.P

SSNCE

June 8, 2022

Table of Contents

Session Outcome

- Understand Recovery Concepts
- Understand WAL
- Understand Checkpointing
- Understand Deferred Update
- Understand Immediate Update

Recovery Process

- Recovery process restores database to the most **recent consistent state** before the time of failure.
- The system must keep information about the changes that were applied to data items by the various transactions which were kept in system log.
  Typical recovery strategies are:
  - Restore backed-up copy of database from archival storage when the damage due to catastrophic failure
  - Identify any changes that may cause inconsistency in a noncatastrophic failure, reverse the changes by applying either of the operations like **undoing or redo**.

Caching and DBMS cache

- Recovery is considered in terms of disk pages, collection of in-memory buffers called DBMS cache.
- **Database Cache**: A set of main memory buffers; each buffer typically holds contents of one disk block.
- Stores the disk blocks that contain the data items being read and written by the database transactions.
- A directory for the cache is used to keep track of which database items are in the buffers.
- Cache Table : Table of entries of the form (buffer addr, disk block addr, modified bit, pin/unpin bit, ...) to indicate which disk blocks are currently in the cache buffers.
- For any DBMS requests, checks the cache directory if not present checks on disk and copy the disk pages into the cache.

Caching and DBMS cache

- Each buffer has several bits or flags associated with it to inform about the buffer modification
- **The dirty bit:** Indicates whether or not the buffer is modified.
  - When a page first read from the database disk to cache buffer, a new entry is inserted into directory with page address and dirty bit is set 0.
  - As the buffer is modified dirty bit is set 1.
  - When the buffer contents are replaced (flushed) from the cache, the contents must first be written back to the corresponding disk page only if its dirty bit is 1.
- **The pin-unpin bit**: Set to 1 if it cannot be written back to disk from the cache yet (usually waiting for a commit)

Strategies for Flushing

Strategies when flushing a modified buffer back to disk:

- **In-place update:** Writes the buffer back to the same original disk location
  It overwrites the old values of any changed data items
  Recovery requires a log to undo and/or redo
  Maintains a single log file.
- **Shadow update:** Write an updated buffer to a different disk location.
  Multiple versions of data items can be maintained.

Write Ahead Log [WAL] Rule

- In Inplace updating log recovery is used.
- **Undo-Type log Entries:** Includes the old value (BFIM) of the item since this is needed to **undo** the effect of the operation from the log.
- **Redo-Type Log Entries**: Includes the new value (AFIM) of the item written by the operation since this is needed to **redo** the effect of the operation from the log.
- The log entries are stored in the current log buffer in the DBMS cache.
- When data item is updated, entries are made in appropriate log entry and the log entry is flushed on to the disk before the BFIM is overwritten with AFIM in the database.
- With the help of WAL, the information needed for recovery must be first written to the log file on disk before changes are made to the database on disk.

Write Ahead Log [WAL] Rule:

Write-Ahead Logging (WAL) protocol consists of two rules for Undo and Redo operations:

- The BFIM cannot be overwritten by its AFIM in the database on disk until all UNDO - type log records for updating transaction have been force-written to disk.
- The commit operation of a transaction cannot be completed until all the REDO - type and UNDO - type log records for that transaction have been force-written to disk (STABLE STORE).

Techniques used to move a page from cache to memory

Standard DBMS recovery terminology includes the terms steal/no-steal and force/no-force which specify some rules when a page from database cache can be written to disk:

- **No-Steal Approach:** if a cache page updated by a transaction cannot be written to disk before the transaction commits.
- **Steal Approach:** if the recovery protocol allows writing an update before the transaction commits.
- **Force Approach:** If all pages updated by a transaction are immediately written to disk before the transaction commits.
- **No-Force Approach:** if all pages are updated after the commit of transactions.
  Typical database systems employ a steal/no-force (UNDO/REDO) strategy and deferred update (NO-UNDO) recovery scheme follows a no-steal approach.