UNIT V (Part 2)
Database Recovery System

Database Recovery

Purpose of Database Recovery

- To bring the database into the last consistent state, which existed prior to the failure.
- To preserve transaction properties (Atomicity, Consistency, Isolation and Durability).
- Example: fund transfer transaction

Transaction failure :

– Logical errors: transaction cannot complete due to some internal error condition

– System errors: the database system must terminate an active transaction due to an error condition (e.g. deadlock)

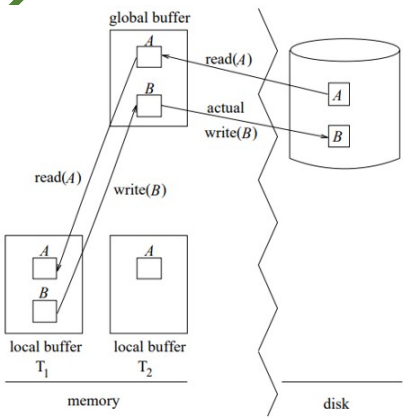
- System crash: a power failure or other hardware or software failure causes the system to crash. It is assumed that non-volatile storage contents are not corrupted.
- Disk failure: a head crash or similar failure destroys all or part of disk storage

Storage Structure

- Volatile storage:
  - does not survive system crashes
  - examples: main memory, cache memory
- Non-volatile storage:
  - survives system crashes
  - examples: disk, tape
- Stable storage:
  - a mythical form of storage that survives all failures
  - approximated by maintaining multiple copies on distinct non-volatile media

Stable-Storage Implementation

- Maintain multiple copies of each block on separate disks; copies can be at remote sites to protect against disasters such as fire or flooding.
- Protecting storage media from failure during data transfer:
  - Execute output operation as follows (assuming two copies of each block):
  - Copies of a block may differ due to failure during output operation. To recover from failure:
  - If either copy of an inconsistent block is detected to have an error (bad checksum), overwrite it by the other copy. If both have no error, but are different, overwrite the second block by the first block.



Recovery and Atomicity

- Consider transaction Ti that transfers $50 from account A to account B; goal is either to perform all database modifications made by Ti or none at all.
- Several output operations may be required for Ti (to output A and B). A failure may occur after one of these modifications have been made but before all of them are made.
- To ensure atomicity despite failures, we first output information describing the modifications to stable storage without modifying the database itself.
- When a DBMS recovers from a crash, it should maintain the following −
  - It should check the states of all the transactions, which were being executed.

- A transaction may be in the middle of some operation; the DBMS must ensure the atomicity of the transaction in this case.
- It should check whether the transaction can be completed now or it needs to be rolled back.
- No transactions would be allowed to leave the DBMS in an inconsistent state.

➢ There are two types of techniques, which can help a DBMS in recovering as well as maintaining the atomicity of a transaction −

- Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
- Maintaining shadow paging, where the changes are done on a volatile memory, and later, the actual database is updated.

Log-based Recovery

- Log is a sequence of records, which maintains the records of actions performed by a transaction. It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe.
- Log-based recovery works as follows
  - The log file is kept on a stable storage media.
  - When a transaction enters the system and starts execution, it writes a log about it.

<T_n, Start>

When the transaction modifies an item X, it write logs as follows −

<T_n, X, V1, V2>

It reads Tn has changed the value of X, from V1 to V2.

Other type of log records are:

- <T_i start>: It contains information about when a transaction T_i starts.
- <T_i commit>: It contains information about when a transaction T_i commits.
- <T_i abort>: It contains information about when a transaction T_i aborts.

Undo and Redo Operations –

- Undo: using a log record sets the data item specified in log record to old value.
- Redo: using a log record sets the data item specified in log record to new value.

The database can be modified using two approaches:

1. Deferred database modification:

- The deferred modification technique occurs if the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

2. Immediate database modification:

- The Immediate modification technique occurs if database modification occurs while the transaction is still active.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

Recovery using Log records

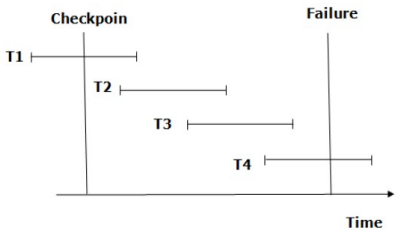
- When the system is crashed, then the system consults the log to find which transactions need to be undone and which need to be redone.
  - If the log contains the record <Ti, Start> and <Ti, Commit> or <Ti, Commit>, then the Transaction Ti needs to be redone.
  - If log contains record<Tn, Start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the Transaction Ti needs to be undone.

Recovery with Concurrent Transactions:

- When more than one transaction are being executed in parallel, the logs are interleaved. At the time of recovery, it would become hard for the recovery system to backtrack all logs, and then start recovering. To ease this situation, most modern DBMS use the concept of 'checkpoints'.

Checkpoint

- Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. As time passes, the log file may grow too big to be handled at all.
- Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk.
- Checkpoint declares a point before which the DBMS was in consistent



state, and all the transactions were committed.

ARIES Recovery Algorithm:

- A steal, no-force approach
  - Steal: if a frame is dirty and chosen for replacement, the page it contains is written to disk even if the modifying transaction is still active.
  - No-force: Pages in the buffer pool that are modified by a transaction are not forced to disk when the transaction commits.

Algorithms for Recovery and Isolation Exploiting Semantics, or ARIES is a recovery algorithm designed to work with a no-force, steal database approach.

The ARIES recovery procedure consists of three main steps:

Analysis

- The analysis step identifies the dirty (updated) pages in the buffer, and the set of transactions active at the time of the crash. The appropriate point in the log where the REDO operation should start is also determined

REDO

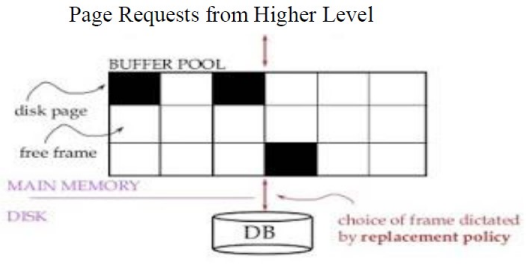
- The REDO phase actually reapplies updates from the log to the database. Generally, the REDO operation is applied to only committed transactions. However, in ARIES, this is not the case. Certain information in the ARIES log will provide the start point for REDO, from which REDO operations are applied until the end of the log is reached. In addition, information stored by ARIES and in the data pages will allow ARIES to determine whether the operation to be redone has actually been applied to the database and hence need not be reapplied. Thus only the necessary REDO operations are applied during recovery.

UNDO

- During the UNDO phase, the log is scanned backwards and the operations of transactions that were active at the time of the crash are undone in reverse order. The information needed for ARIES to accomplish its recovery procedure includes the log, the Transaction Table, and the Dirty Page Table. In addition, check pointing is used. These two tables are maintained by the transaction manager and written to the log during check pointing.

BUFFER MANAGEMENT

- The buffer manager is the software layer that is responsible for bringing pages from physical disk to main memory as needed. The buffer manages the available main memory by dividing the main memory into a collection of pages, which we called as buffer pool. The main memory pages in the buffer pool are called frames.



Buffer Management in a DBMS

- Data must be in RAM for DBMS to operate on it!
- Buffer manager hides the fact that not all data is in RAM.

Buffer Manager

- ❑ A Buffer Manager is responsible for allocating space to the buffer in order to store data into the buffer.
- ❑ If a user request a particular block and the block is available in the buffer, the buffer manager provides the block address in the main memory.
- ❑ If free space is not available, it throws out some existing blocks from the buffer to allocate the required space for the new block.

For serving the database system in the best possible way, the buffer manager uses the following methods:

- Buffer Replacement Strategy: If no space is left in the buffer, it is required to remove an existing block from the buffer before allocating the new one. The various operating system uses the LRU (least recently used) scheme. In LRU, the block that was least recently used is removed from the buffer and written back to the disk. Such type of replacement strategy is known as Buffer Replacement Strategy.
- Pinned Blocks:
  - Check
  - Dirty

Failure with Loss of Non-volatile Storage:

- The basic scheme is to dump the entire content of the database to stable storage periodically—say, once per day.

More precisely, no transaction may be active during the dump procedure, and a procedure similar to checkpointing must take place:

1. Output all log records currently residing in main memory onto stable storage.
2. Output all buffer blocks onto the disk.
3. Copy the contents of the database to stable storage.
4. Output a log record <dump> onto the stable storage.

Remote backup definition and strategies.

- Remote backup as a part of data protection in companies has a vital importance since it guarantees the organization its operational continuity and functionality in the occurrence of loss or damage to critical data that may affect them, such as:
  - Loss of the DB of company personnel
  - Corruption of the DB and / or application that control automated production processes
  - Loss of historical organizational data, while the company is in the process of auditing or evaluating quality standards
  - Loss of information on the balance sheets and management indicators of the different organizational processes