**DBMS Recovery**

**1. Introduction to Database Recovery**

Database recovery is the process of restoring the database to a correct state in the event of a failure. This ensures that data remains consistent and reliable, even after crashes or errors.

- Objective: Ensure the database can recover from errors or failures and continue operating correctly.

- Importance:

- Safeguard data from corruption.

- Maintain consistency after system failures.

- Ensure the database can handle failures gracefully.

**2. Need for Recovery**

Database systems face potential disruptions due to hardware failures, software bugs, or human errors. Without recovery mechanisms, these events could lead to data loss, corruption, or inconsistencies. Hence, recovery is essential to:

- Maintain Data Integrity: Ensure the accuracy and consistency of data.

- Ensure Availability: After a failure, the system must return to a usable state quickly.

- Consistency: Ensure the database adheres to ACID (Atomicity, Consistency, Isolation, Durability) properties.

- Atomicity: Ensures transactions are fully completed or rolled back.

- Durability: Once a transaction is committed, it remains in the system, even if there's a failure.

**3. Types of Errors**

Database errors can be categorized into different types based on their cause and severity:

# a. Transaction Failure

Occurs due to issues within the transaction itself, like input errors or logical inconsistencies. This leads to incomplete or incorrect transactions that may need to be rolled back.

- Reasons:

- Input data violations.

- Deadlocks.

- Violating database constraints (e.g., foreign key violations).

# b. System Failure

The system crashes due to hardware or software issues, such as operating system crashes or power failures, leaving transactions incomplete and in an uncertain state.

- Reasons:

- Hardware malfunctions.

- Software bugs.

- Power outages.

# c. Media Failure

Occurs when storage devices (disks, drives) become damaged or corrupt, leading to the loss of critical data.

- Reasons:

- Disk crashes.

- Hardware wear and tear.

- Catastrophic events (e.g., fire, floods).

# d. Logical Errors

These errors occur when the database produces incorrect results due to application bugs or bad data input, requiring a rollback to a consistent state.

- Reasons:

- Incorrect queries.

- Poor data manipulation logic.

# e. Concurrency Control Failures

When multiple transactions attempt to access the same data concurrently, without proper control, it can lead to inconsistencies or deadlocks.

- Reasons:

- Race conditions.

- Deadlocks.

**4. Recovery Techniques**

To handle these failures, various recovery techniques have been developed. These aim to restore the database to a consistent state while minimizing data loss.

# a. Log-Based Recovery

- Purpose: Maintain a log file (also called transaction log) to keep a record of all transactions and changes made to the database.

- Log Components:

- Before-Image: The state of data before a transaction.

- After-Image: The state of data after a transaction.

- Types of Logging:

- Write-Ahead Logging (WAL): Log entries are written before changes are made to the actual database. This ensures that all changes can be rolled back or replayed as necessary.

- Steps:

- If a system failure occurs, the recovery process checks the log file.

- Redo Operations: Reapply committed transactions using after-images.

- Undo Operations: Rollback incomplete or uncommitted transactions using before-images.

# b. Checkpointing

- Purpose: Minimize recovery time by periodically saving the current state of the database (called a checkpoint).

- Process:

- A checkpoint flushes all the committed changes to the disk and records the checkpoint in the log.

- During recovery, the system can start from the last checkpoint instead of scanning the entire log.

# c. Shadow Paging

- Purpose: This method maintains two versions of the data: one active and one shadow. Changes are made to the shadow pages, and only after the transaction is successfully committed, the shadow pages are swapped with the active ones.

- Process:

- Shadow Pages: These contain the old state of the data before any changes.

- Commit Operation: Once a transaction completes, the system swaps the shadow page table with the current page table.

- Recovery: In case of failure, the system rolls back to the shadow pages, ensuring that uncommitted changes are not applied.

# d. Deferred Update

- Purpose: Ensures that updates are deferred until a transaction has reached its commit point.

- Process:

- Changes are made in a local buffer but are not applied to the database until the transaction is committed.

- In the case of failure, since no changes were made to the database, no rollback is required.

# e. Immediate Update

- Purpose: Updates are made immediately, but the system maintains a log for undo operations in case of failure.

- Process:

- Changes are written to the database as soon as a transaction starts, but log entries ensure that uncommitted transactions can be undone.

# f. Two-Phase Commit Protocol (2PC)

- Purpose: This is used in distributed databases to ensure atomicity across multiple systems.

- Process:

- Phase 1 (Preparation): The coordinator asks all participants if they can commit the transaction.

- Phase 2 (Commit): If all participants agree, the coordinator sends a commit message. Otherwise, a rollback is issued.

# g. Backups

- Full Backup: Creates a complete copy of the database, which can be used for restoration.

- Incremental Backup: Only copies data that has changed since the last backup.

- Differential Backup: Copies all the data changed since the last full backup.

---