Unit 9
Backup and Recovery

9.1. Failure Classifications

9.2. Recovery and Atomicity

9.4. Log based Recovery

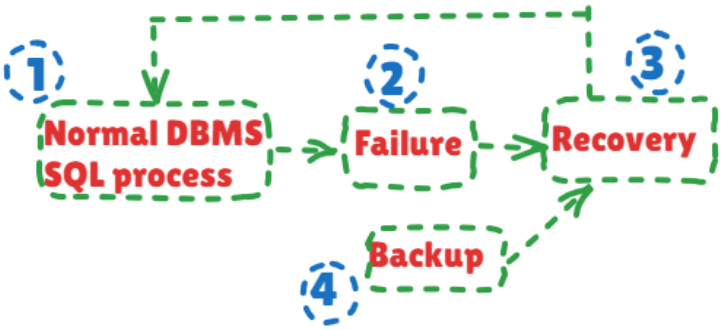
9.5. Shadow Paging

9.6. Local Recovery Manager

9.7. Backup system

Introduction:

- Backup and Recovery are crucial for ensuring data integrity and availability in database systems. These mechanisms safeguard against data loss and help restore systems to a consistent state after failures.



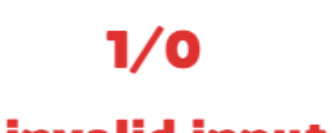
- A Computer system can be *failed due to variety of reasons like disk crash, power fluctuation, software error, sabotage and fire or flood at computer site.* These failures result in the lost of information.
- Thus, database must be *capable of handling such situations to ensure that the atomicity and durability* properties of transactions are preserved.
- An integral part of a database system is the *recovery manager that is responsible for recovering the data*. It ensures atomicity by undoing the actions of transactions that do not commit and durability by making sure that all actions of committed transactions survive system crashes and media failures.

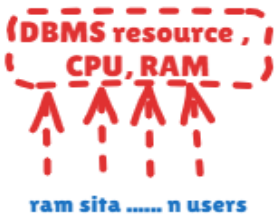
9.1. Failure Classifications

Failure refers to the state when system can no longer continue with its normal execution and that results in loss of information.

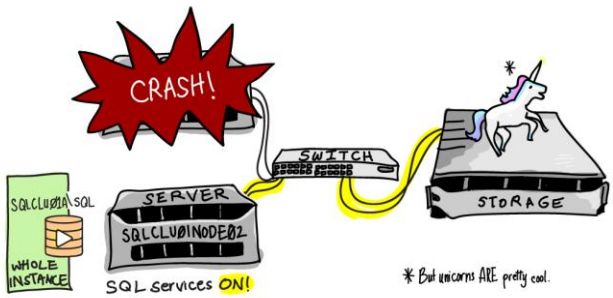
Different types of failure are as follows:

1. **Transaction Failures:**
   - **Causes:** Logical errors, such as invalid input, or system issues, like deadlocks.
   - **Example:** Transaction transferring funds between two accounts is aborted due to insufficient funds in the source account.





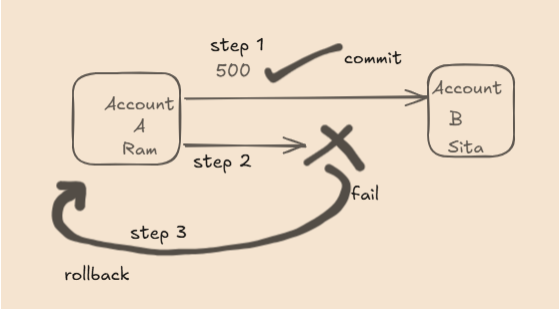
2. **System Failures:**
   - **Causes:** Unrecoverable errors in hardware or software, resulting in the database stopping unexpectedly.
   - **Example:** A sudden power outage halts an ongoing database operation.



3. **Media Failures:**
    - **Causes:** Physical issues with storage devices, such as disk corruption or drive failures.
    - **Example:** A database file becomes unreadable due to hard drive damage.

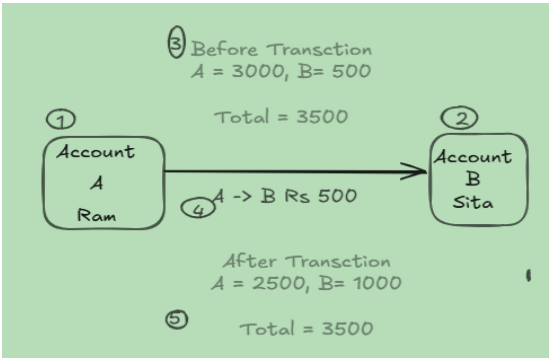
4. **Catastrophic Failures:**
    - **Causes:** Severe disruptions like natural disasters or major system crashes.
    - **Example:** A fire destroys the data center hosting the database serv

9.2. Recovery and Atomicity

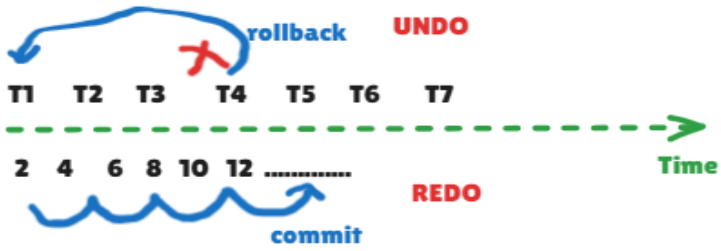


- Recovery from failure state refers to the method by which system restore its most recent consistent state just before the time of failure. There are several methods by which you can recover database from failure state.

- Recovery mechanisms ensure the database maintains **atomicity** and **consistency** despite failures:
    - **Atomicity:** Ensures that a transaction is either fully completed or fully rolled back, leaving the database in a consistent state.
    - **Consistency:** Guarantees that the database transitions from one valid state to another.



Key Techniques for Recovery:



1. **Undo Operations:** Reverse changes made by incomplete transactions.
2. **Redo Operations:** Reapply changes made by committed transactions to ensure durability.

Example:

- A bank transaction involves debiting $500 from Account A and crediting $500 to Account B.
    - If the system crashes after debiting Account A but before crediting Account B, the recovery mechanism will roll back the debit to Account A to maintain atomicity.
    - Conversely, if the crash occurs after both actions, the system ensures the transaction is redone to maintain consistency.

Steps in Recovery Process:

1. Identify incomplete and committed transactions using logs.
2. Apply undo for incomplete transactions to revert partial changes.
3. Apply redo for committed transactions to ensure all changes persist.

Types of recovery

- Recovery methods in a Database Management System (DBMS) are techniques used to restore a database to a consistent state after a failure.
- These methods ensure data integrity and durability by addressing failures in transactions, systems, or storage.
- Below are the primary types of recovery methods:
    - ***Log-Based Recovery***
    - ***Checkpointing***
    - ***Shadow Paging***
    - ***ARIES (Algorithm for Recovery and Isolation Exploiting Semantics)***
    - ***Recovery Using Backup***
    - ***Transaction Rollback***

- ***Deferred Update***
- ***Immediate Update***
- ***Remote Backup Systems***

Choosing a Recovery Method

The choice of recovery method depends on:

- The nature of the application.
- System requirements (speed, reliability).
- Resource availability (storage, processing power).

9.4. Log based Recovery

https://github.com/sanjeevlcc/notes_2081/blob/main/DBMS_BIM_BSCIT_BCA/DBMS%20ADVANCE/sql_logs.txt

```
nano /etc/mysql/mysql.conf.d/mysqld.cnf
        general_log = 1
        general_log_file = /var/log/mysql/general.log
```

- In log-based recovery, every database transaction writes a log entry before or after executing an operation.
- Logs are stored in stable storage and help recover data after a failure.
- *Logs are useful for system and disk failures. The logs must be in stable storage.*
- *Log contains complete records of DB activities.*

```
controlplane $ pwd
/var/log/mysql
controlplane $ ls
error.log  general.log
```

```
2025-01-11T10:00:31.773361Z         8 Query     create database db1
2025-01-11T10:00:36.050478Z         8 Query     SELECT DATABASE()
2025-01-11T10:00:36.051302Z         8 Init DB   db1
2025-01-11T10:00:36.052439Z         8 Query     show databases
2025-01-11T10:00:36.063866Z         8 Query     show tables
2025-01-11T10:01:04.782090Z         8 Query     create table t1(id int(4), name varchar(11))
2025-01-11T10:01:51.966760Z         8 Query     insert into t1() values (1,"ram")
```

```
Terminal 1 ×
mysql> create database db1;
Query OK, 1 row affected (0.00 sec)

mysql> use db1;
Database changed
mysql> create table t1(id int(4), name varchar(11));
Query OK, 0 rows affected, 1 warning (0.11 sec)

mysql> insert into t1() values (1,"ram");
Query OK, 1 row affected (0.05 sec)

mysql>
```

Example:

- A transaction updates a record's salary from 10,000 to 12,000. The log records:

  START T1

  WRITE (EMP, Salary, 10,000, 12,000)

  COMMIT T1

- *If a failure occurs, the log replays these steps to ensure consistency.*

Case Study:

In a banking system, if a system crashes during a fund transfer, log-based recovery ensures that the *transaction either fully completes or is fully rolled back.*

Types of Logging:

- **Undo Logging**: Records the old value before updating.
- **Redo Logging**: Records the new value after updating.

Example:

A transaction updates balance in an account table:

- Initial state: balance = 500.
- Update: balance = balance + 200.

Undo Logging Example:

Log entries:

1. ***<T1, Start>***
2. ***<T1, A, 500> (Before updating)***
3. ***<T1, Commit>***

If failure occurs: Undo changes using <T1, A, 500> to restore the value to 500.

Redo Logging Example:

Log entries:

1. *<T1, Start>*
2. *<T1, A, 700> (After updating, partial execution or commit, RAM)*
3. *<T1, Commit>(DISK save)*

If failure occurs:

- Redo the operation to set the value to 700.

SQL Scenario:

```
BEGIN TRANSACTION;
    UPDATE accounts SET balance = balance + 200
    WHERE account_id = 1;
COMMIT;
```

Example:

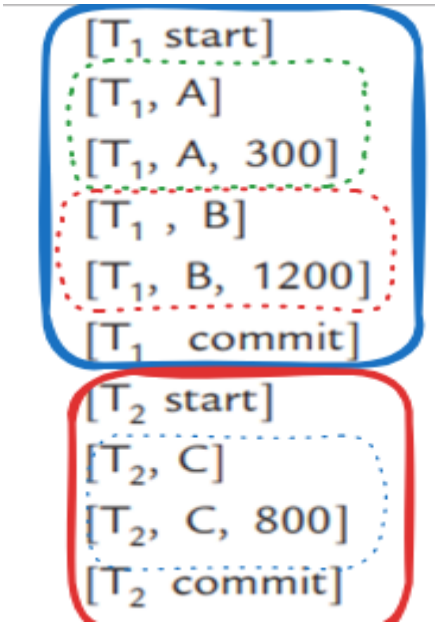
Consider the example of Banking system. Suppose you want to transfer Rs 200 from Account A to B in Transaction T1 and deposit Rs 200 to Account C in T2. The transactions T1 and T2 are shown in fig.

T_1	T_2

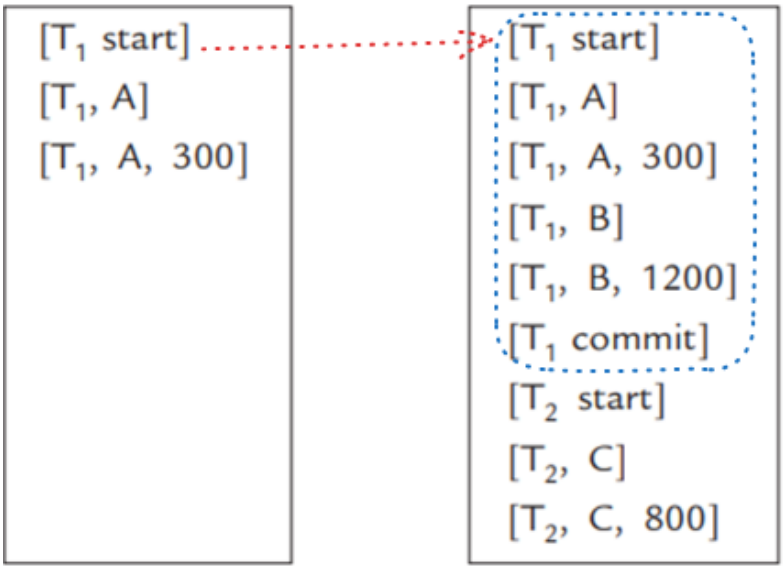
read(A);
A = A – 200;
write(A);
read(B);
B = B + 200;
write(B);
	read(C);
	C = C + 200;
	write(C);

Suppose, the initial values of A, B and C as accounts are Rs 500, Rs 1,000 and Rs 600 respectively,

Various log records for T1 and T2



For a redo operation, log must contain [Ti start] and [Ti commit] log records.



Crash will happen at any time of execution of transactions. Suppose crash happened

(i) After write (A) of T1:

At that time log records in log. There is no need to redo operation because no commit record appears in the log. Log records of T1 can be deleted.

(ii) After write (C) of T2:

At that time log records in log are. In this situation, you have to redo T1 because both [T1 start] and [T1 commit] appears in log. After redo operation, value of A and B are 300 and 1200 respectively. Values remain same because redo is idempotent.

(iii) During recovery:

If system is crashed at the time of recovery, simply starts the recovery again

9.5. Shadow Paging

Description:

- Shadow Paging is an alternative technique for recovery to overcome the disadvantages of log-based recovery techniques.

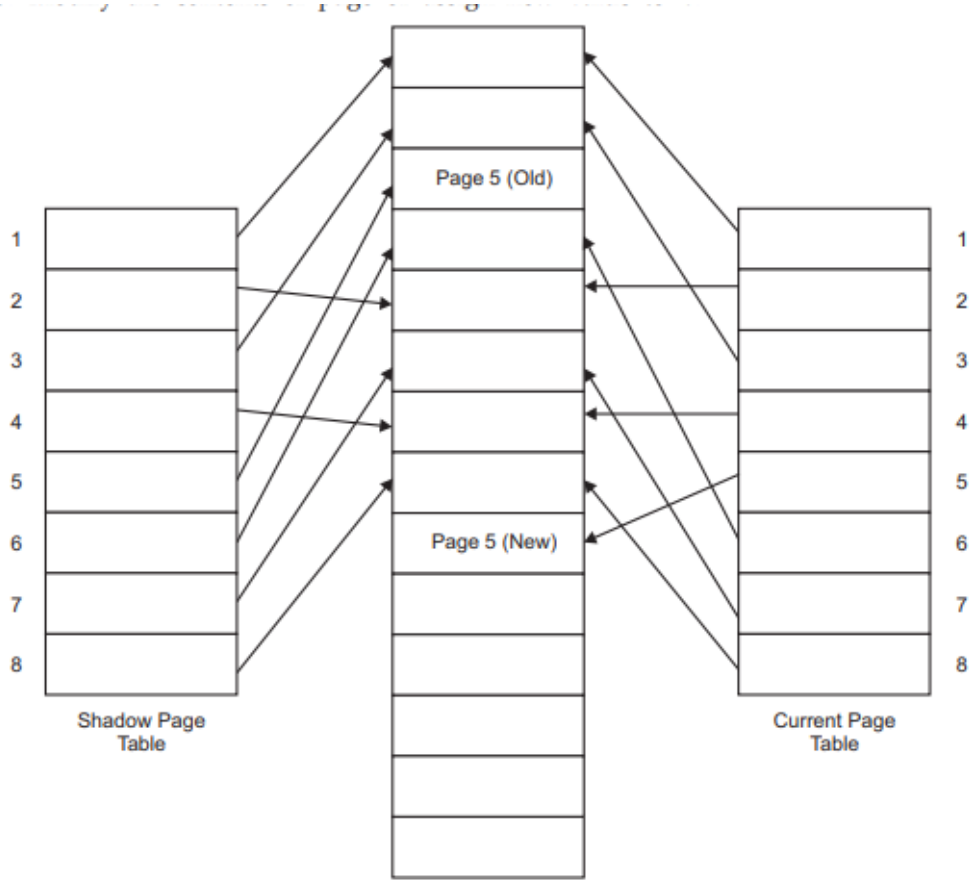
- The main idea behind the shadow paging is to keep two-page tables in database, one is used for current operations and other is used in case of recovery. Database is partitioned into fixed length blocks called pages. For memory management, adopt any paging technique.
- Maintains two-page tables (current and shadow). Updates are made to the current table while the shadow table remains unchanged. During recovery, the shadow table is used.

Page Table

- To keep record of each page in database, maintain a page table. Total number of entries in page table is equal to the number of pages in database. Each entry in page table contains a pointer to the physical location of pages.
- The two-page tables in Shadow Paging are:
  - **Shadow copy (original):** Used as a fallback during recovery. This table cannot be changed during any transaction.
  - **Current copy:** Updated during transactions. This table may be changed during transaction.

➢ Both page tables are identical at the start of transaction. Suppose a transaction Ti performs a write operation on data item V, that resides in page j. The write operation procedure is as follows:

1. If j th page is in main memory then k otherwise first transfer it from secondary memory to main memory by instruction input (V).
2. Suppose page is used first time then:
   a. System first finds a free page on disk and delete its entry from free page list.
   b. Then modify the current page table so that it points to the page found in step 2(a).
3. Modify the contents of page or assign new value to V.



How it Works:

- Changes are applied to the current page.
- The shadow copy remains unaltered.
- After a successful transaction, the current page table replaces the shadow page table.
- In case of failure, the system reverts to the shadow copy.

Advantages:

- No logging overhead.
- Simple and efficient for read-heavy systems.

Disadvantages:

- High disk space usage due to duplicate pages.
- Inefficient for frequent updates.

Example:
Consider a banking application:

- The shadow copy maintains the previous state of customer accounts.
- If an update fails, the system restores balances from the shadow copy.

Case Study:

A real-time analytics system uses shadow paging to ensure rapid recovery during hardware failures without log processing.

9.6. Local Recovery Manager

- **Description**:
  The Local Recovery Manager (LRM) is responsible for handling failures at the transaction or local database level. It ensures database consistency and manages recovery for localized issues without affecting the entire system.

- **Functions of LRM**:
  1. **Transaction Recovery**: Handles rollback or commit of transactions.
  2. **Log Management**: Maintains and processes transaction logs.
  3. **Lock Management**: Ensures concurrent transactions do not conflict.

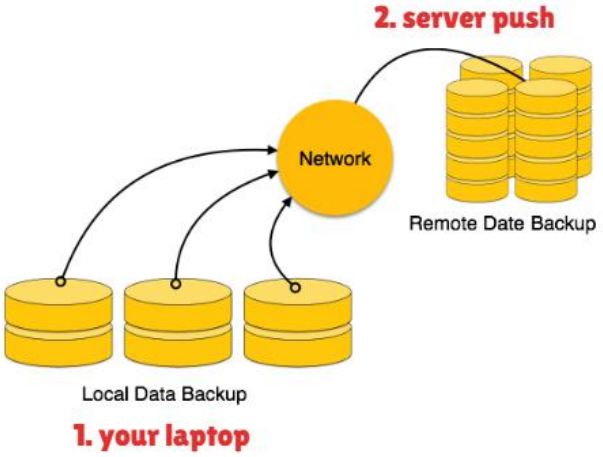
- **Steps**:
  - Identify the failed transaction.
  - Undo incomplete changes using logs.
  - Restore the database to a consistent state.

- **Example**:
  A payroll system processes employee salaries. If the system fails while calculating, the Local Recovery Manager rolls back the affected transaction without interrupting others.

- **Case Study**:
  In a retail Point-of-Sale (POS) system, the LRM ensures consistent transaction records for a specific store during network outages.

9.7. Backup system

- **Description**:
  A Backup System creates periodic copies of the database, ensuring recovery in case of major failures like disasters or ransomware attacks.



- **Types of Backups**:
  1. **Full Backup**: A complete copy of the database.
  2. **Incremental Backup**: Copies only the changes since the last backup.
  3. **Differential Backup**: Copies changes since the last full backup.

- **Recovery Process**:
  - Restore the most recent full backup.
  - Apply incremental or differential backups to update the database.

- **Best Practices**:
  - Automate backups at regular intervals.
  - Store backups in secure, remote locations.
  - Test backups periodically for integrity.

- **Example**:
  A hospital database performs daily backups at midnight. A system crash at 10:00 AM requires restoring the midnight backup and applying transaction logs up to 10:00 AM.

- **Case Study**:
  During a cyberattack, a financial organization restores its database using remote backups stored in a cloud system, ensuring minimal downtime.

Q/A

Fill in the Blanks (20 Questions)

Multiple Choice Questions (MCQ) (20 Questions)

Short Questions (20 Questions)

Comprehensive Questions (20 Questions)

ANSWER of (MCQ)

ANSWER of Short Questions