[Date]

Advance Data Base Management System:

Written By SAUD KHAN MARWAT

Introduction of Advanced Database Management System (ADBMS):

An **Advanced Database Management System** is an enhanced version of a traditional DBMS that handles complex, large-scale, and diverse data more efficiently. It supports advanced features like object-orientation, distributed processing, and real-time access.

Explanation:

An Advanced DBMS is a **more powerful database system** that can store and manage **not just text and numbers**, but also **images, videos, locations, and real-time data**. It is **faster**, **more secure**, and can be used over the internet or in big companies where huge amounts of data are handled.

Examples of ADBMS:

- **Oracle Database**
- **IBM Db2**
- **Microsoft SQL Server (with analytics)**
- **PostgreSQL (with extensions)**
- **MongoDB (NoSQL)**

Advantages of ADBMS:

1. **Handles complex and large datasets efficiently**
2. **Supports advanced data types (e.g., multimedia, spatial, temporal)**
3. **Improves query performance through optimization techniques**
4. **Allows distributed and cloud-based data storage and processing**
5. **Offers strong security and real-time data access capabilities**

Daily Life Example:

Online Shopping Apps (e.g., Amazon): ADBMS is used to manage customer data, product inventories, multimedia images, real-time order tracking, and recommendation systems. It ensures that millions of users can browse, search, and place orders at the same time without errors or delays.

Data Modeling (in ADBMS):

Introduction:

Data Modeling means planning and designing **how data will be stored** in a database. It shows **what kind of data** will be saved and **how different data items are connected** to each other.

Explanation :

Before we build a database, we need a **blueprint (design)** — just like we need a map before building a house. Data modeling helps us decide **what tables to make**, **what data to store** in each table, and **how the tables are linked**. This helps avoid confusion and mistakes later.

Types of Data Models (by Structure):

1. **Entity-Relationship (ER) Model**
    - Uses **entities** (like Student, Teacher) and **relationships** (like Enrolls, Teaches)
    - Helps design the database before building it
    - Very common for **drawing ER diagrams**
2. **Relational Model**
    - Data is stored in **tables (rows and columns)**
    - Most commonly used model today
    - Used in databases like **MySQL, SQL Server, Oracle**
3. **Object-Based Data Model**

- Represents data as **objects** (like in programming languages)
- Each object has **attributes** (data) and **methods** (functions)
- Used in **object-oriented databases**

4. **Object-Oriented Model**
   - A more advanced version of object-based
   - Combines features of object-oriented programming with databases
   - Useful for **multimedia**, **CAD**, and **complex apps**
5. **Hierarchical Model**
   - Data is stored in a **tree-like structure** (parent → child)
   - One parent can have many children, but each child has only one parent
   - Used in **old systems** like IBM's early databases
6. **Network Model**
   - Like hierarchical, but more flexible
   - One child can have **multiple parents**
   - Allows **many-to-many relationship.**

Data Integrity & Security:

Introduction:

Data Integrity means **correctness, accuracy, and consistency** of data in a database. It ensures that the data stored is **reliable**, **not changed by mistake**, and remains **trustworthy over time**.

1. SQL Injection

SQL Injection is a **type of attack** where a hacker **enters harmful SQL code** into input fields (like a login box) to access or destroy data in the database.

Why is this related?
Because **SQL Injection breaks data integrity and security** by allowing attackers to **change or steal** the data.

2: Man-in-the-Middle (MITM) Attack – In Easy Words

What is it?
A **Man-in-the-Middle (MITM) attack** is like someone secretly **listening or watching** your conversation with someone else — **without you knowing**.

OR

A **Man-in-the-Middle attack** is when a hacker secretly comes between you and the website or app you're using, to **steal or change your data**

Where it can happen?

- On **public Wi-Fi** (like in cafés or airports)
- On **unsecure websites** (those without "https")
- On **networks with weak security**

What Can the Attacker Do?

- **Steal passwords**
- **Read your private messages**
- **Change the messages** you send or receive
- **Pretend to be the website** or app you're using

How to Stay Safe:

1. **Use websites with HTTPS** (lock icon in address bar)
2. **Never use public Wi-Fi without VPN**
3. **Use strong passwords and 2-step verification**

4. **Install security updates** on your phone and computer

3: Data Tampering.

This means **changing data in the database without permission** — either by a hacker or by a mistake. It's a **big violation of data integrity**.

🧠 **Why is this important?**
Because if someone **changes your marks, bank balance, or identity** in a database, it can cause serious problems.

TRANSACTION:

Advanced Database Management Systems (ADBMS), a **transaction** is a **sequence of one or more database operations** (such as read, write, insert, delete, or update) that are **executed as a single logical unit of work**.

Key Characteristics of a Transaction:

A transaction must satisfy the **ACID properties** to ensure reliability and consistency in the database:

1. **Atomicity**: All operations within the transaction are completed successfully, or none of them are.
2. **Consistency**: The transaction brings the database from one valid state to another valid state.
3. **Isolation**: Transactions are executed independently of one another. Intermediate results are hidden from other transactions.
4. **Durability**: Once a transaction is committed, its changes are permanent, even in the case of a system failure.

Example:

BEGIN TRANSACTION;

UPDATE accounts SET balance = balance - 1000 WHERE account_id = 1;

UPDATE accounts SET balance = balance + 1000 WHERE account_id = 2;

COMMIT;

In this example:

- Money is transferred from account 1 to account 2.
- If either update fails, the transaction is rolled back to maintain consistency.

In Advanced DBMS:

In addition to basic transactions, **advanced systems handle**:

- **Nested transactions**
- **Distributed transactions**
- **Long-duration transactions**
- **Multiversion concurrency control (MVCC)**

What is Concurrency Control:

Concurrency Control is the management of **simultaneous operations** (transactions) on a database **without conflicting** with each other, ensuring **data integrity and consistency**.

- In multi-user database systems, multiple transactions may access the same data at the same time.
- Concurrency control ensures that **interleaved execution** of transactions does **not produce incorrect results**.

How Does Concurrency Control Work?

Concurrency control mechanisms **synchronize access** to data by:

1. **Preventing conflicts** (like two writes at the same time).
2. **Ensuring isolation** as per the ACID properties.

3. **Scheduling transactions** so that the final result is **equivalent to some serial execution** (called **serializability**).

Concurrency Control Issues (Problems):

When concurrency is not properly controlled, several issues can occur:

1. Lost Update

Two transactions overwrite each other's updates.

Example:

- T1: reads balance = 100
- T2: reads balance = 100
- T1: adds 10 → writes 110
- T2: adds 20 → writes 120 (T1's update is lost)

2. Dirty Read (Uncommitted Dependency)

A transaction reads data written by another **uncommitted** transaction.

Example:

- T1: writes balance = 150 (but not committed)
- T2: reads balance = 150
- T1: aborts → T2 read invalid data

3. Non-repeatable Read

A transaction reads the **same data twice** and gets different results due to another transaction's update in between.

4. Phantom Read

A transaction reads a **set of rows**, then another transaction inserts/deletes a row, and the first transaction reads again and sees **extra or missing rows**.

Concurrency Control Methods:

1. Lock-Based Protocols:

These methods use **locks** to control access to data items.

- **Shared Lock (S-lock):** Allows multiple transactions to read a data item.
- **Exclusive Lock (X-lock):** Only one transaction can write to the data item; no other transaction can read or write.
- **Two-Phase Locking (2PL):**
  - **Growing phase:** Transaction acquires all locks.
  - **Shrinking phase:** Transaction releases locks and cannot acquire new ones.
  - Guarantees serializability but can cause deadlocks.

2. Timestamp-Based Protocols:

Each transaction is given a **timestamp**. The system uses these timestamps to decide the serial order of transactions.

- Ensures **serializability** by allowing only the oldest transaction to write.
- Prevents **conflicts** but can lead to unnecessary transaction aborts.

What is data base recovery technique?

Database Recovery Techniques are methods used in a **Database Management System (DBMS)** to **restore the database to a correct, consistent state** after a failure, such as system crash, power failure, or software bug. The main goal is to ensure **Atomicity** and **Durability**—two key properties of ACID.

What is Database Recovery?

Database recovery is the process of **restoring the database to the last consistent state** before the failure occurred, using backups and logs.

Types of Database Recovery Techniques in DBMS:

1. **Rollback (Undo) Recovery Technique**
   - Used when a **transaction fails before commit**.
   - All changes made by the transaction are **undone**.
   - Brings the database back to its **previous consistent state**.
2. **Commit Recovery Technique**
   - Ensures that **committed transactions** are not lost during a system crash.
   - If a transaction has **committed but changes were not written to disk**, the system **re-applies (redo)** the changes after recovery.
3. **Checkpoint Recovery Technique**
   - A **checkpoint** is a saved state of the database.
   - During recovery, the system starts from the **last checkpoint** instead of going through the entire log.
   - Makes recovery **faster and more efficient**.
4. **Deferred Database Modification Technique**
   - Changes made by transactions are **not applied immediately**.
   - Changes are written to the database **only after the transaction commits**.
   - If a transaction fails → **no need to undo**, because no changes were applied yet.

What is a File in DBMS?

In DBMS, a **file** is a collection of **related records stored on a secondary storage device** (like hard disk).
Each file represents a **table or dataset** and contains rows (records) and fields (attributes).

What is File Organization in DBMS?

File Organization refers to the **method of arranging records in a file**.
It determines **how data is stored, accessed, and managed** on disk.

In simple terms: It's **how records are physically placed inside a file**, so they can be **retrieved efficiently**.

Objectives of File Organization in DBMS:

1. ✅ **Efficient Data Access**
    - Reduce time taken to **search, insert, update, or delete** records.
2. ✅ **Minimize Storage Space**
    - Use storage efficiently to avoid unnecessary space usage.
3. ✅ **Faster Retrieval of Records**
    - Organize data to allow quick searching (e.g., using indexes).
4. ✅ **Ease of Record Addition and Deletion**
    - Make it easy to add new records or remove existing ones without disturbing others.
5. ✅ **Data Integrity and Consistency**
    - Ensure correct and reliable storage of data.
6. ✅ **Support for Various Access Methods**
    - Allow **sequential**, **random**, or **indexed** access to records as needed.
7. ✅ **Minimize Data Redundancy**
    - Prevent duplication of data by proper organization.

What is Database Administrator (DBA) Role Management?

Database Administrator (DBA) is a person (or role) responsible for the **installation, configuration, management, maintenance, security, and performance** of a database system.

In short: A DBA is the **guardian** of the database who ensures it works correctly, is secure, and runs efficiently.

Key Responsibilities of a DBA in DBMS:

Responsibility	Explanation
1:**Security Management:**	Controls **user access**, prevents unauthorized usage, and **protects data** from breaches.
2: **Database Installation & Configuration:**	Installs DBMS software, sets up databases, and configures them for use.
3: **Backup and Recovery:**	Regularly **backs up** the database and restores it in case of failure.
4: **Performance Tuning:**	Monitors and optimizes database **speed, queries, and resource usage**.
5 **Database Design & Implementation:**	Helps design the structure of tables, relationships, keys, and indexes.
6: **User Account Management:**	Creates user roles, grants or revokes privileges, and manages roles.
7: **Storage Management:**	Manages how data is stored, disk usage, and memory allocation.
8: **Monitoring & Troubleshooting:**	Keeps an eye on system logs, errors, and **fixes issues** before they cause major problems.
9:Data Integrity Management:	Ensures **accuracy and consistency** of stored data.

Responsibility	Explanation
10: **Migration and Upgrades:**	Handles **upgrading** database systems and **migrating** data to new environments.

Query Processing & Optimizing:

What is a Query in DBMS?

A **query** is a **request for data** or information from a database.

It is written using a **query language**, such as **SQL (Structured Query Language)**, and is used to **retrieve, insert, update, or delete** data.

✅ Example of a Query (in SQL):

SELECT name, age FROM students WHERE grade = 'A';

🔷 This query retrieves the **name** and **age** of students who have a grade 'A' from the **students** table.

Query Processing in DBMS

Query Processing is the series of steps taken by the DBMS to **execute a query** and return the correct result efficiently.

Phases of Query Processing:

Phase	Description
1: **Parsing and Translation**	The query is **checked for syntax errors** and converted into an **internal representation** (like a query tree).
2: **Optimization**	The system chooses the **most efficient execution plan** (e.g., using indexes or best join order).
3: **Evaluation/Execution**	The DBMS **executes the query plan** using the chosen strategy and accesses the actual data.
4: **Result Output**	The final **result is returned** to the user or application that made the query