**1. Introduction to Database Management System (DBMS)**

* DBMS is software used to store, manage, and retrieve data from databases.
* Ensures data organization and management for efficient operations in organizations.
* Key functions: data storage, data retrieval, security, integrity, and backup.

**1.1 Data**

* Data refers to raw facts and figures with no inherent meaning.
* Examples: "25", "Ajit", "$50".
* Represented in alphabets, digits, and special characters.

**1.2 Information**

* Information is processed data that is organized and meaningful.
* Example: “The age of Ravi is 25.”
* Information is used for decision-making and action-taking.

**1.3 File**

* A file is a collection of related data stored in secondary memory.
* Files store data for long-term use and retrieval.

**1.3.1 File-Oriented Approach**

* File-Oriented Approach: Traditional system where each application has separate files to store data.
* Data is stored and processed by separate programs.
* Files are dependent on programs and vice versa.

**1.3.2 Disadvantages of File-Oriented Approach**

1. Data Redundancy and Inconsistency:
   * Same data stored in multiple files, leading to higher storage costs and inconsistent data.
2. Difficulty in Accessing Data:
   * Hard to retrieve data efficiently as files lack advanced querying mechanisms.
3. Data Isolation:
   * Data stored in different files/formats, making it difficult to integrate and share.
4. Integrity Problems:
   * Hard to enforce data constraints across multiple programs.
5. Atomicity Issues:
   * Difficult to ensure atomicity during transaction failures (e.g., power loss).
6. Concurrency Control:
   * Simultaneous access to files by multiple users is not well-supported.
7. Security Problems:
   * Lack of security controls leads to unauthorized access to data.

**1.4 Database**

A **database** is an organized collection of related data stored in a structured format, allowing multiple users to access it efficiently. It is designed to provide easy and secure data storage and retrieval. The data in a database can be **persistent**, **integrated**, and **shared**.

**Key Features of Data in a Database:**

1. **Well-Organized**:  
   Data in a database must be structured and stored in an organized manner. This organization ensures that data can be accessed, updated, and managed efficiently.
2. **Related**:  
   Data in a database is typically related to each other in some logical way. For example, student data (name, roll number, and address) is related to the student record.
3. **Accessible in a Logical Order**:  
   Data must be accessible in a way that makes sense to users, such as retrieving records in alphabetical or chronological order without difficulty.
4. **Stored Only Once**:  
   In a database, data should only be stored once to avoid redundancy and inconsistency. Redundant storage of data can lead to inefficiency and errors.

**Types of Data in a Database:**

1. **Persistent Data**:  
   Data in a database is **persistent**, meaning it remains stored until explicitly deleted by the user. This ensures that data is available for future use unless removed.
   * Example: A student's record (name, roll number, address) is stored until explicitly deleted.
2. **Integrated Data**:  
   **Integrated data** refers to data collected from different files or sources but stored together in a database after removing redundancy.
   * Example: If a database includes customer information from multiple systems (sales, customer service, billing), it integrates all the data, removing duplication.
3. **Shared Data**:  
   Data in a database is **shared**, allowing multiple users to access the data at the same time without causing conflicts or affecting its correctness.
   * Example: Multiple users (students, faculty, administrators) can access the same student information database, but the data remains consistent and accurate for everyone.

**Example:**

* **Student Database**:  
  A student database may contain data like:
  + Roll number
  + Name
  + Address

This data is related (it belongs to a student), well-organized, and accessible logically (sorted by roll number or name), and stored only once to prevent duplication.

**Advantages of Database Systems Over Traditional Paper-Based Methods**

1. **Compactness**:
   * Reduces the need for physical paper files.
   * Data is stored digitally, saving space and reducing storage costs.
2. **Speed**:
   * Machines retrieve and modify data faster than humans can handle paper files.
   * Quick search and update operations.
3. **Less Drudgery**:
   * Eliminates manual maintenance of files and repetitive tasks.
   * Data updates and modifications are automated.
4. **Accuracy**:
   * Ensures accurate, up-to-date information retrieval.
   * Reduces human errors in data processing and record-keeping.

**1.5 Database Management System (DBMS)**

A **DBMS** is a collection of related data and a set of programs designed to manage, define, create, and manipulate databases.

**1.5.1 Functions of DBMS:**

1. **Defining Database Schema**:
   * Defines the structure of the database and specifies access rights to authorized users.
2. **Manipulation of the Database**:
   * Includes functions for inserting, updating, deleting, and retrieving data.
3. **Sharing of Database**:
   * Facilitates data sharing for multiple users while maintaining data consistency.
4. **Protection of Database**:
   * Ensures protection from unauthorized users through security mechanisms.
5. **Database Recovery**:
   * Enables recovery of data in case of system failure.

**1.5.2 Advantages of DBMS:**

1. **Reduction of Redundancies**:
   * Centralized data control reduces data duplication, saving storage space and improving consistency.
2. **Sharing of Data**:
   * Multiple users or applications can share the same data under DBMS control.
3. **Data Integrity**:
   * Ensures data accuracy and consistency. Data values are validated for correctness and range.
4. **Data Security**:
   * Database administrators (DBA) manage access controls and ensure sensitive data is protected.
5. **Conflict Resolution**:
   * DBAs resolve conflicts between user requirements, choosing optimal file structures for performance.
6. **Data Independence**:
   * **Physical Data Independence**: Changes in physical storage don’t affect the conceptual view or applications.
   * **Logical Data Independence**: Changes in the conceptual schema don’t affect external schemas or applications.

**1.5.3 Disadvantages of DBMS:**

1. **High Cost**:
   * DBMS software and hardware costs, including networking installation, can be expensive.
2. **Processing Overhead**:
   * DBMS consumes additional resources to implement security, integrity, and data sharing.
3. **Centralized Database Control**:
   * Centralized control may lead to system bottlenecks.
4. **Complex Setup**:
   * Setting up a DBMS requires specialized knowledge, skills, and significant time and financial investment.
5. **Performance Issues**:
   * Complex databases may suffer from performance degradation.

**DBMS vs File Processing System**

| **Feature** | **DBMS** | **File Processing System** |
| --- | --- | --- |
| **Data Redundancy** | Minimal | Exists |
| **Data Inconsistency** | Does not exist | Exists |
| **Accessing Database** | Easier | Difficult |
| **Data Isolation** | Not found | Exists |
| **Transactions** | Possible (Insert, Update, Delete, View) | Not possible |
| **Concurrent Access & Recovery** | Supported | Not supported |
| **Data Security** | Strong | Weak |
| **Relationship Storage** | Stored in structural tables | Stored in directories |

**2. Database Models**

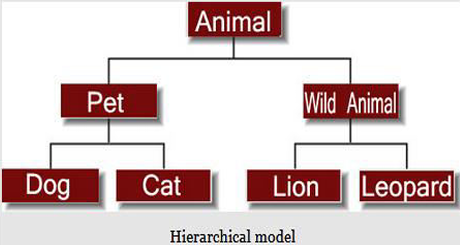
A **Database Model** defines the structure, **logical design**, **Structure of DB**, and how data is stored, accessed, and updated in a DBMS. The main database models include:

1. **Hierarchical Model**
2. **Network Model**
3. **Entity-Relationship Model (E-R)**
4. **Relational Model**
5. **Object-Oriented Database Model**

**2.1 Hierarchical Database Model**

Hierarchical database model arranges data in a structure similar to a tree.

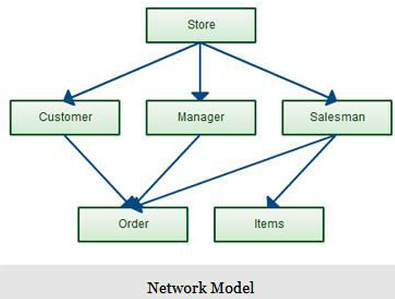
* **Structure**: Data is arranged in a tree-like structure with a **root** node at the top. Each parent node can have multiple child nodes, but each child node has only one parent.
* **Relationship**: Represents **one-to-many** relationships.
* **Example**: Organizational charts, directory structures.



**2.2 Network Model**

In this model data is organised more like a graph, and are allowed to have more than one parent node.

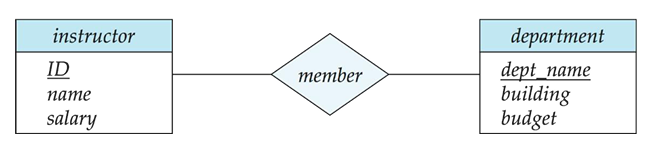
* **Extension of Hierarchical Model**: Data is organized in a **graph-like** structure, allowing a **many-to-many** relationship between entities.
* **Advantages**: More flexible, supports more relationships, faster data access.
* **Example**: Used in systems where multiple relationships need to be established, such as telecommunications and transportation networks.



**2.3 Entity-Relationship (E-R) Model**

data is organised in two-dimensional tables and the relationship is maintained by storing a common field.

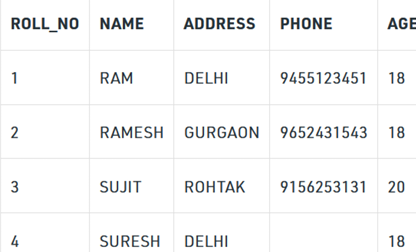
* **Entities**: Objects of interest (e.g., students, employees).
* **Attributes**: Characteristics of entities (e.g., name, age).
* **Relationships**: Associations between entities (e.g., student enrolled in a course).
* **Model Representation**: E-R diagrams help visually represent the database structure.
* **Example**: In a **School Database**, "Student" is an entity, with attributes like name, age, address, and relationships to "Course" entities.



**2.4 Relational Model**

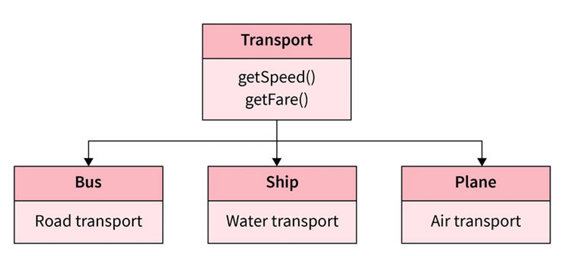
In this model, data is organised in two-dimensional tables and the relationship is maintained by storing a common field.

* **Structure**: Data is organized in **tables (relations)**. Each row (tuple) represents a record, and each column (attribute) represents data characteristics.
* **Key Features**:
  + **Tuple**: Each row in a table.
  + **Degree**: The number of attributes (columns) in a table.
  + **Cardinality**: The number of tuples (rows) in a table.
  + **Relation Schema**: Defines the structure of a relation (table) with its attributes.
  + **NULL Values**: Represents missing or unknown data in a table.
* **Example**: A **STUDENT** table with columns like ROLL\_NO, NAME, AGE, and ADDRESS.



**2.5 Object-Oriented Data Model**

* **Real-World Representation**: Data is represented as objects, similar to object-oriented programming.
* **Objects**: Entities with attributes and methods that define behavior.
* **Inheritance**: Objects can inherit properties from other objects (e.g., **Transport** object with derived objects like **Bus**, **Ship**, and **Plane**).
* **Example**: A **Transport** object with attributes like **road**, **water**, or **air**, and derived objects like **Bus**, **Ship**, and **Plane**.



| **Sr. No.** | **Feature** | **Hierarchical Data Model** | **Network Data Model** | **Relational Data Model** |
| --- | --- | --- | --- | --- |
| **1** | **Usage** | Oldest and not in use today. | Used in older systems but replaced by relational model | Most widely used in modern systems today. |
| **2** | **Data Organization** | Organizes data in a tree structure. | Organizes data in a directed graph. | Organizes data in tables (relations). |
| **3** | **Relationships Supported** | Implements 1:1 and 1:n relationships. | Implements 1:1, 1:n, and many-to-many relationships. | Implements 1:1, 1:n, and many-to-many relationships. |
| **4** | **Relationship Representation** | Uses pointers to establish relationships physically. | Uses linked lists to establish relationships. | Uses logical representations (rows and columns) to depict relationships. |
| **5** | **Insertion Anomaly** | Exists, i.e., child node cannot be inserted without the parent. | No insertion anomaly. | No insertion anomaly. |
| **6** | **Deletion Anomaly** | Exists, i.e., it is difficult to delete the parent node. | No deletion anomaly. | No deletion anomaly. |
| **7** | **Update Anomaly** | Leads to inconsistency due to multiple instances of child records. | No such problem as only one instance of records exists. | No update anomaly, normalization resolves redundant data. |
| **8** | **Data Independence** | Lacks data independence. | Partial data independence. | Provides full data independence. |
| **9** | **Querying Support** | No facility for querying the database. | No facility for querying the database. | Supports SQL-based declarative querying. |
| **10** | **Data Complexity** | Used to access complex and asymmetric data. | Used to access complex and symmetric data. | Used to access complex and symmetric data. |
| **11** | **Design Complexity** | Difficult to design due to its complexity. | Difficult to design and manipulate due to complexity. | Easy to comprehend due to abstraction of physical details. |
| **12** | **Flexibility** | Less flexible. | More flexible than the hierarchical model. | Most flexible among the three models. |
| **13** | **Common Uses** | Used by XML and XAML. | Used by VAX-DBMS | Used by Oracle, MySQL, PostgreSQL, etc. |

**3. Database System Architecture**

The architecture of a database system depends on the underlying computer system. Common types include:

**1. Centralized Architecture**

* **Description**: The database is stored on a single central server. All users connect to this server for data access.
* **Pros**: Easier to manage.
* **Cons**: Can become a bottleneck due to heavy traffic.

**2. Client-Server Architecture**

* **Description**: The database resides on a server, while clients access it remotely. The server handles data tasks, and clients manage the interface.
* **Pros**: Scalable, flexible.
* **Cons**: More complex than centralized systems.

**3. Parallel (Multi-Processor) Architecture**

* **Description**: Multiple processors handle data in parallel for faster processing and retrieval.
* **Pros**: High performance, suitable for large datasets.
* **Cons**: Expensive and requires specialized hardware.

**4. Distributed Architecture**

* **Description**: Data is stored across multiple locations or servers. Communication between servers is needed for queries.
* **Pros**: High fault tolerance, scalable.
* **Cons**: Complex to manage, requires synchronization between servers.

**Summary of Key Differences:**

| **Architecture** | **Data Location** | **Management** | **Scalability** |
| --- | --- | --- | --- |
| **Centralized** | Single location/server | Single-point management | Limited scalability |
| **Client-Server** | Server (centralized), Clients (remote) | Server manages data, clients interact | Moderate scalability |
| **Parallel (Multi-Processor)** | Multiple processors handling tasks | Distributed processing | High performance, costly |
| **Distributed** | Multiple distributed locations | Data synchronization across servers | High scalability, complex |

**Database Users**

* **Application Programmers**:  
  Write application programs using various tools. Use RAD tools for quick forms/reports without coding.
* **Sophisticated Users**:  
  Use query languages directly without programming. Queries are processed by the query processor.
* **Specialized Users**:  
  Create special database applications like CAD systems, expert systems, or multimedia databases.
* **Naïve Users**:  
  Use pre-written applications without coding knowledge (e.g., a bank teller using a money transfer app).

**Database Administrator (DBA)**

* Manages all database system activities and ensures data availability and security.
* Responsibilities:
  + Define schema and storage structures.
  + Modify schema/organization when needed.
  + Manage user permissions.
  + Enforce integrity constraints.
  + Monitor and optimize performance.

**Query Processor**

* Accepts user queries and processes them.
* Components:
  + **DDL Interpreter**: Interprets schema definitions into the data dictionary.
  + **DML Compiler**: Translates query language into low-level instructions; optimizes query plans.
  + **Query Evaluation Engine**: Executes low-level instructions on the database.

**Storage Manager**

* Acts as an interface between the database and application programs.
* Responsibilities:
  + **Authorization and Integrity Manager**: Checks permissions and constraints.
  + **Transaction Manager**: Maintains database consistency during failures.
  + **File Manager**: Manages space and structures on disk.
  + **Buffer Manager**: Transfers data between disk and memory efficiently.
* **Data Structures**:
  + **Data Files**: Actual database content.
  + **Data Dictionary**: Metadata about database structure.
  + **Indices**: Faster access paths to data.

**4. Data Abstraction**

* **Data Abstraction** hides irrelevant details from users, showing only necessary parts of the database.
* It simplifies user interaction, enhances security, and supports data independence.
* Complex internal structures are hidden; users access only the needed parts.

**4.1 Levels of Abstraction in DBMS**

There are **three levels**:

* **Physical (Internal) Level**:
  + Lowest level, describes *how* data is stored (data structures, storage blocks).
  + Decided by developers.
  + Example: Data stored in bytes, blocks, etc.
* **Logical (Conceptual) Level**:
  + Middle level, describes *what* data is stored and *relationships* among data.
  + Used by developers/DBAs.
  + Example: Tables, fields, relationships.
* **View (External) Level**:
  + Highest level, shows only selected parts of the database to users.
  + Provides multiple views for different users.
  + Example: Users filling a form via GUI without knowing backend details.

**5. Data Independence**

* **Data Independence** means **changing the database schema** at one level without affecting programs at higher levels.
* Main goal of **three levels of abstraction** is to **achieve data independence**.
* Saves time and cost when the database evolves.

**5.1 Physical Data Independence**

* Changes at **physical level** **do not affect** the **logical level**.
* Changes could include:
  + Using a new storage device (e.g., SSD instead of HDD),
  + Changing storage structures,
  + Changing file organization methods,
  + Relocating database files.

**5.2 Logical Data Independence**

* Changes at **logical level** **do not affect** the **view (external) level**.
* **Logical View** is how users see and interact with data (tables, attributes, relationships).
* Changes could include:
  + Adding or deleting tables, fields, or relationships,
  + Modifying data definitions.

**6.** **DBMS Languages**

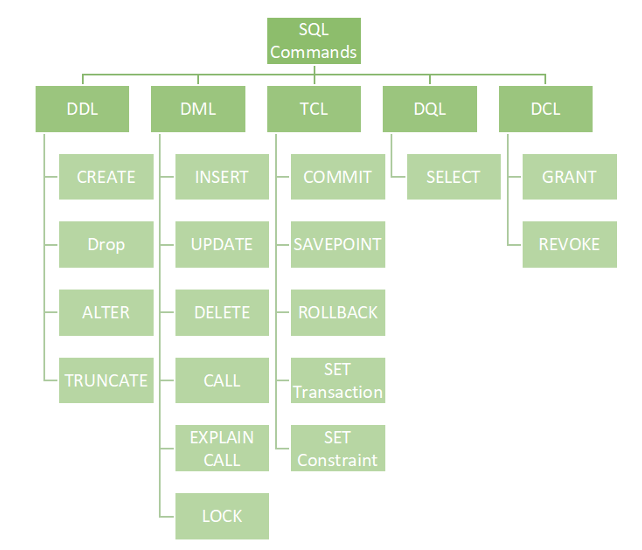
1. DDL – Data Definition Language

2. DQL – Data Query Language

3. DML – Data Manipulation Language

4. DCL – Data Control Language

5. TCL – Transaction Control Language



**6.1 Data Definition Language (DDL)**

Deals with **database schema structure**, not data.

**Commands:**

* CREATE → Create database or objects (tables, views, etc.)
* DROP → Delete objects.
* ALTER → Modify existing database structure.
* TRUNCATE → Remove all records from a table (but not the table itself).
* COMMENT → Add comments to data dictionary.
* RENAME → Rename database objects.

**6.2 Data Query Language (DQL)**

Deals with **fetching data**.

**Command:**

* SELECT → Retrieve data from one or more tables.

**6.3 Data Manipulation Language (DML)**

Deals with **manipulating (inserting, updating, deleting) data**.

**Commands:**

* INSERT → Add new data.
* UPDATE → Modify existing data.
* DELETE → Remove data.
* LOCK → Control concurrency of table access.
* CALL → Execute stored procedures or programs.
* EXPLAIN PLAN → Show how SQL statements are executed.

**6.4 Data Control Language (DCL)**

Deals with **permissions and access control**.

**Commands:**

* GRANT → Give access rights to users.
* REVOKE → Remove access rights from users.

**6.5 Transaction Control Language (TCL)**

Deals with **managing transactions**.

**Commands:**

* BEGIN → Start a transaction.
* COMMIT → Save all changes.
* ROLLBACK → Undo changes if errors occur.
* SAVEPOINT → Set a checkpoint to roll back partially.
* SET TRANSACTION → Set transaction properties (like isolation levels).

**7. Integrity Constraints**

In relational database design, **integrity constraints** are rules that ensure the accuracy and consistency of data. These restrictions control what values can be inserted, modified, or deleted.

There are **four main types** of integrity constraints:

**1. Domain Constraints**

* Every domain must contain **atomic values** (smallest indivisible units).
* **Composite** and **multi-valued attributes** are not allowed.
* **Data type checks** ensure the values are appropriate (e.g., age must be an integer).

**Example:**  
If Name is a composite attribute or Phone has multiple values, domain constraints are violated.

**2. Key Constraints (Uniqueness Constraints)**

* Ensures every tuple (row) in a relation is **unique**.
* A relation can have multiple **candidate keys**; one is chosen as the **primary key**.
* **Primary keys cannot have NULL values** (implies NOT NULL constraint).

**Example:**  
If two employees have the same EID = 01, it violates key constraint because primary keys must be unique.

**3. Entity Integrity Constraints**

* No **primary key** can have a **NULL** value.
* Each tuple must be uniquely identifiable.

**Example:**  
If a record has NULL in the primary key field (like EID = NULL), it violates the entity integrity constraint.

**4. Referential Integrity Constraints**

* Deals with **foreign keys**.
* A foreign key in one relation must refer to a **valid primary key** in another relation.
* Foreign key can be **NULL** but not an **invalid** reference.

**Example:**  
If DNO = 22 is present as a foreign key but does not exist as a primary key in the referenced table, referential integrity is violated.

**8. DML (Data Manipulation Language) Operations**

DML operations are used to **access and manipulate data** in existing database objects (tables).

**1. SELECT Command**

* Retrieves data from a table.

**Syntax:**

sql

CopyEdit

SELECT \* FROM table\_name;

**Example:**

sql

CopyEdit

SELECT \* FROM students;

SELECT \* FROM students WHERE due\_fees <= 20000;

**2. INSERT Command**

* Adds new records to a table.

**Syntax:**

sql

CopyEdit

INSERT INTO table\_name (column1, column2) VALUES (value1, value2);

**Example:**

sql

CopyEdit

INSERT INTO students (stu\_id, stu\_name, city)

VALUES (1, 'Nirmit', 'Gorakhpur');

**3. UPDATE Command**

* Modifies existing records in a table.

**Syntax:**

sql

CopyEdit

UPDATE table\_name

SET column\_name = value

WHERE condition;

**Example:**

sql

CopyEdit

UPDATE students

SET due\_fees = 20000

WHERE stu\_name = 'Mini';

**4. DELETE Command**

* Removes records from a table.

**Syntax:**

sql

CopyEdit

DELETE FROM table\_name

WHERE condition;

**Example:**

sql

CopyEdit

DELETE FROM students

WHERE stu\_id = '001';

**Advantages of DML**

* Alters data stored in the database.
* Provides effective human-machine interaction.
* Allows users to specify exactly what data is required.
* Supports variety and flexibility across different database systems.

**Disadvantages of DML**

* Cannot change the database structure (only data).
* Limited table views (may hide some columns).
* Can't access data outside stored objects.
* Can't create or delete tables using DML.

