Subject - Database Management System

Unit I

Introduction to Database System

1.1 Database concepts -

Data - Data refers to raw, unorganized facts and figures that can be processed to generate meaningful information.

Database - A database is a structured system used to store, manage, and retrieve data efficiently for multiple users and applications.

It's important due to following features →

1) Scalability - Handles large volumes of data efficiently.
2) Data Integrity - Maintains accuracy using rules and constraints.
3) Security - Protects data through controlled access and permissions.
4) Analytics - Helps analyze data to support better decision-making.

Database Management System - It is a software system that manages, stores and retrieves data efficiently in a structured format.

DBMS acts as a bridge between a central database and multiple clients, including apps and users.

1) It allows users to create, update and query databases efficiently.
2) Ensures data integrity, consistency, and security across multiple users and applications.
3) Reduces data redundancy and inconsistency through centralized control.
4) Supports concurrent data access, transaction management and automatic backups.

File system vs DBMS -

File system	DBMS
1) The file system is a way of arranging the files in a storage medium within a computer.	1) DBMS is a software for managing the database.
2) Redundant data can be present in a file system.	2) In DBMS there is no redundant data.
3) There is no efficient query processing in the file system.	3) Efficient query processing is there in DBMS.
4) There is less data consis-tency in the file system.	4) There is more data consistency because of the process of normalization.
5) It is less complex as compared to DBMS.	5) It has more complexity in handling as compared to the file system.
6) File systems provide less security in comparison to DBMS.	6) DBMS has more security mechanisms as compared to file systems.
7) It is less expensive than DBMS.	7) It has a comparatively higher cost than a file system.
8) There is no data indepen-dence.	8) In DBMS data independence exists, mainly of two types → 1) Logical Data Inde-pendence 2) Physical Data Independence.
9) Only one user can access data at a time.	9) Multiple users can access data at a time.
10) The users are not required to write procedures.	10) The user has to write procedures for managing databases.

Application of DBMS →

1) Railway Reservation System →
In the rail route reservation framework, the information base is needed to store the record or information of ticket appointments, status of train's appearance, and flight.

2) Library management system →
There are many books in the library so; it is difficult to store the record of the relative multitude of books in a register or duplicate. Along these lines, the data set administration framework (DBMS) is utilized to keep up all the data identified with the name of the book, issue date, accessibility of the book and its writer.

3) Banking →
Database the executive's framework is utilized to store the exchange data of the client in the information base.

4) Accounting and finance →
The information base administration framework is utilized for putting away data about deals, holding and acquistion of monetary instruments, for example - stocks and bonds in a data set.

5) Healthcare system →
DBMS is used in healthcare to manage patient data, medical records, and billing information.

6) Security →
DBMS provides security features to ensure that only authorized users have access to the data.

7) Telecommunication →
DBMS are essential to the telecommunication industry because they manage enormous volume of data on billing; customer information and network optimization.

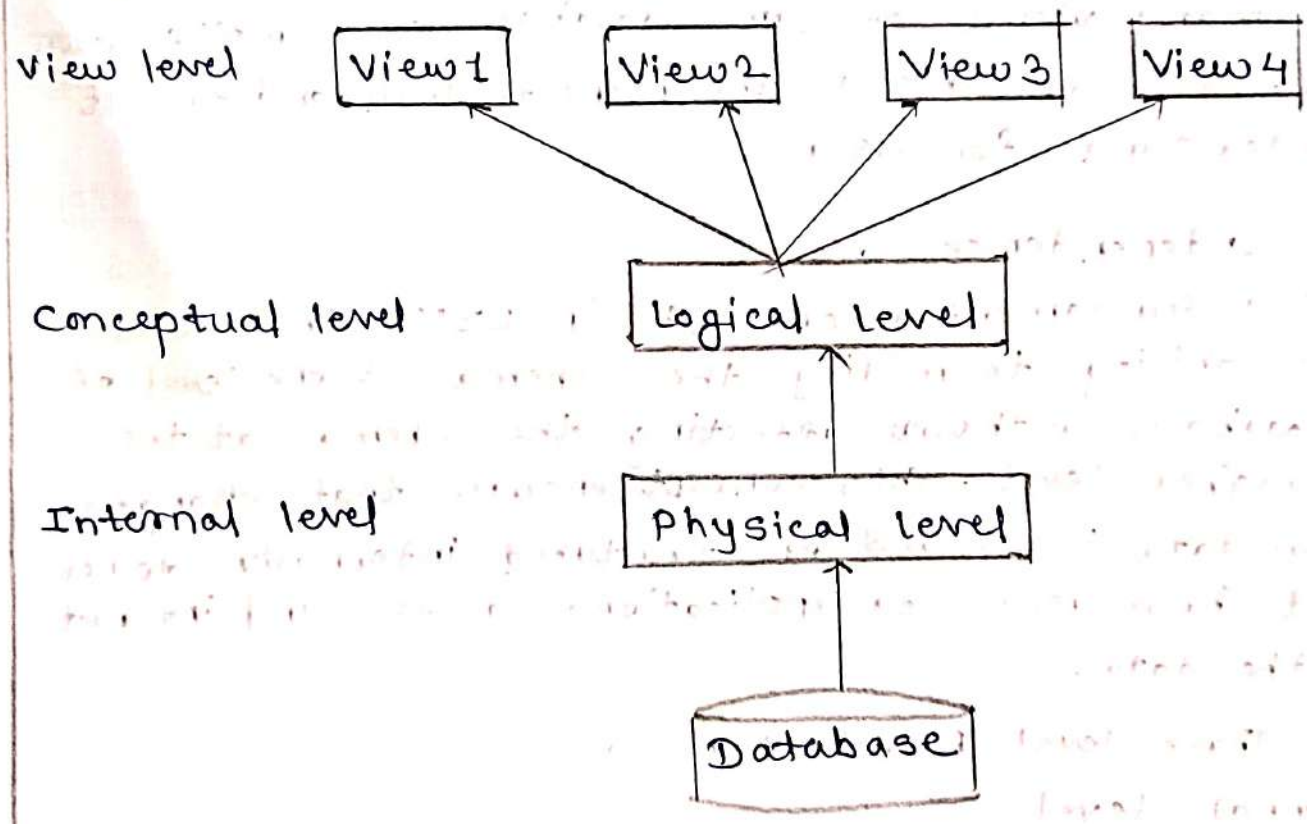
Data Abstraction →

Data Abstraction is the process of hiding unwanted and irrelevant details from the end user. It helps to store information in such a way that the end user can access data which is necessary, the user will not be able to see what data is stored or how it is stored in a database.

Levels of Abstraction in DBMS →

There are three levels of data abstraction →

1) Physical or Internal level
2) Logical or conceptual level
3) View or external level



Physical or Internal level →

It is the lowest level of data abstraction which defines how data is stored in database. It defines data structures used to store data and methods to access data in database. It is very complex to understand and hence kept hidden from user. Physical level deals with actual storage details like data organization, disk space allocation and data access methods.

Logical or Conceptual level →

It is intermediate level present next to physical level. It defines what data is present in database and their relationships between them. It is less complex as compared to physical level. Programmers generally work at this level and depending on data, structure of tables, relationships and their constraints is decided at this level.

View or external level →

It is the highest level in abstraction. There are different levels of views and each view defines only a part of whole data required to user. This level de-fines many views of same database for simplification of view to user. This is the highest level and easiest to understand for user.

Data Independence →

It is a fundamental concept in DBMS that refers to the ability to modify the schema at one level of the database without affecting the schema at the next higher level. This concept ensures that changes in how data is stored or structured internally do not impact how users or applications access and interact with the data.

DBMS Three-level Architecture →

1) Internal level
2) Conceptual level
3) View level.

Types of Data Independence →

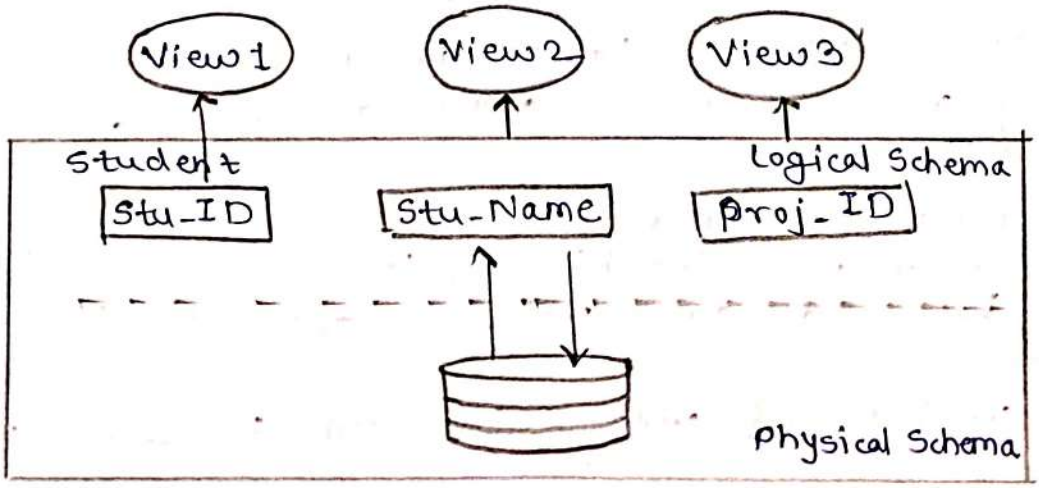
Data Independence is the ability to change the data-base schema at one level without affecting the schema at other levels. It helps in maintaining flexibility, reducing maintenance and ensuring that applications continue to work despite internal

changes in the database.

1) Logical Data Independence →
Ability to change the logical structure (tables, columns, relationships) without affecting external views or application programs. Purpose of this to allow the database structure to evolve without impacting user access or requiring changes in application code.

2) Physical Data Independence →
The ability to change how data is physically stored without affecting the logical schema or user-facing applications. Purpose is to improve performance, storage efficiency, or hardware configurations without changing how the data is structured logically.

Database Schemas →
A database schemas is the design or structure of a database that defines how data is organized and how different data elements relate to each other. It acts as a blueprint, outlining tables, fields, relationships, and rules that govern the data..
In simple terms, the schema provides the framework that makes it easier to understand, manage and use data in a database. It's created by database designers to ensure the data is consistent and efficiently organized.



Types of Database Schema →

1) Conceptual Database schema →

A conceptual schema describes the overall structure of the entire database at a high level.

It focuses on the meaning of data and captures how different entities relate to each other - without worrying about how data is stored physically.

2) Physical Database Schema →

- The physical schema defines how the data is actually stored on disk.
- Specifies files, indexes, partitions, storage blocks, access paths, etc.

It represents the lowest level of abstraction, focusing on performing, storage optimization and data retrieval efficiency.

- Determines the physical layout of tables & indexes.
- Depends on the specific DBMS (eg. MySQL, Oracle).
- Designed mainly by the database administrator (DBA).

3) Logical Database Schema →

- The logical schema describes the logical structure of data as it appears to the database designers and developers.
- Defines tables, columns, primary keys, foreign keys, relationships and integrity rules.

It ensures →

- Data is structured properly.
- Relationships between tables are meaningful.
- Integrity rules maintain data accuracy during operations.

4) View Database Schema →

- The view schema is the highest level of abstraction, showing how individual users or applications see the database.
- Defines customized views tailored to user's needs.

Users interact with these views without knowing about the underlying logical or physical structures.

The Codd's rules →

Rule 1: The Information Rule →
All information, whether it is user information or metadata, that is stored in a database must be entered as a value in a cell of a table. It is said that everything within the database is organized in a table layout.

Rule 2: The Guaranteed Access Rule →
Each data element is guaranteed to be accessible logic-ally with a combination of the table name, primary key and attribute name.

Rule 3: Systematic Treatment of NULL Values →
Every Null value in a database must be given a systematic and uniform treatment

Rule 4: Active Online Catalog Rule →
The database catalog, which contains metadata about the database, must be stored and accessed using the same relational database management system.

Rule 5: The Comprehensive Data Sublanguage Rule →
A crucial component of any efficient database system is its ability to offer an easily understandable data manipulation language that facilitates defining, querying and modifying information within the database.

Rule 6: The View Updating Rule →
All views that are theoretically updatable must also be updatable by the system.

Rule 7: High-level Insert, Update and Delete →
A successful database system must prossess the feature of facilitating high-level insertions, updates, and deletions that can grant users the ability to conduct these operations with ease through a single query.

Rule 8: Physical Data Independence →
Application programs and activities should remain unaffected when changes are made to the physical storage structured or methods.

Rule 9: Logical Data Independence →
Application programs and activities should remain unaffected when changes are made to the logical struc-ture of the data, such as adding or modifying tables.

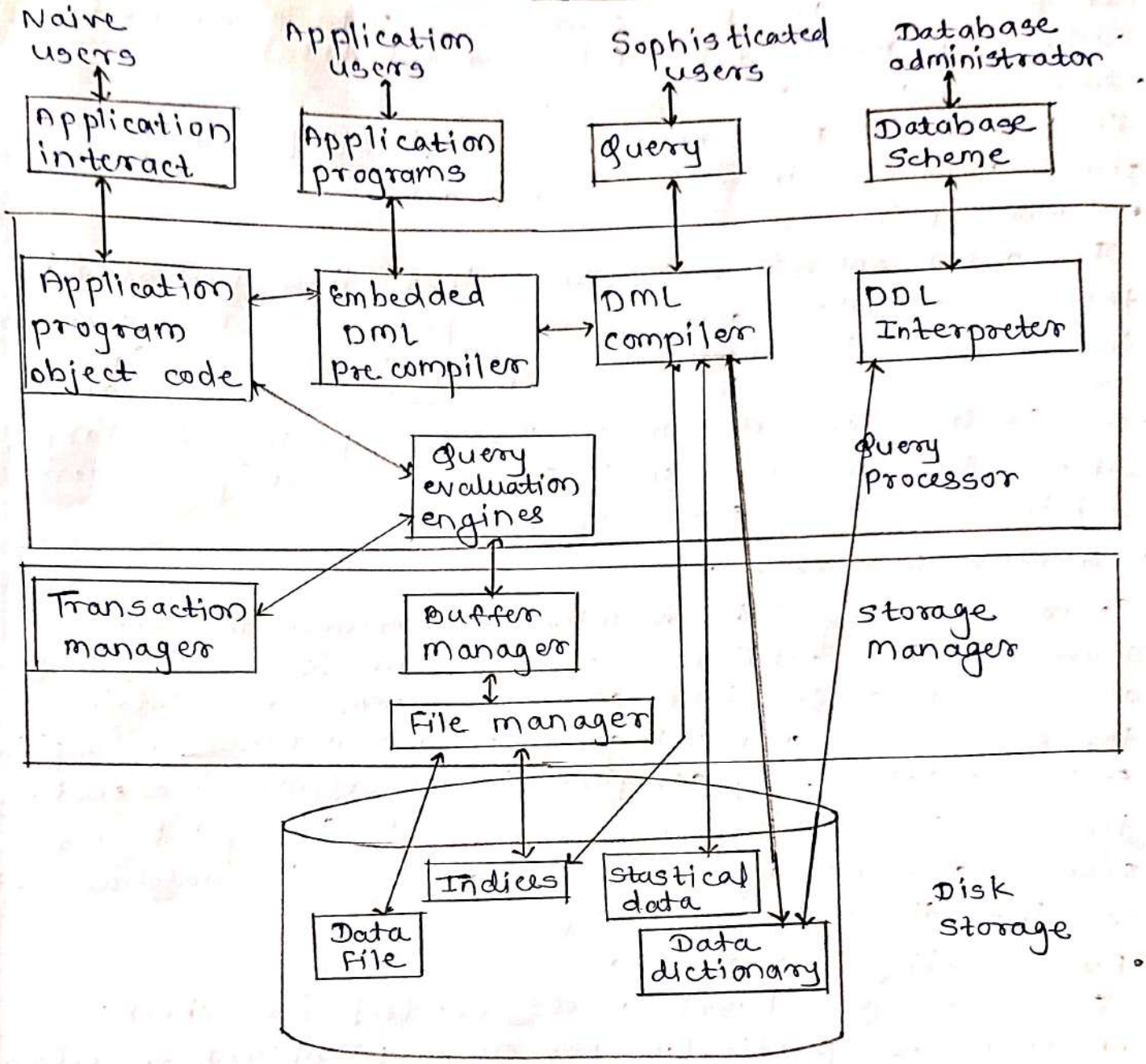
Rule 10: Integrity Independence →
Integrity constraints should be specified separately from application programs and stored in the catalog. They should be automatically enforced by the database system.

Rule 11: Distribution Independence →
The distribution of data across multiple locations should be invisible to users, and the database system should handle the distribution transparently.

Rule 12: Non-Subversion Rule →
If the interface of the system is providing access to low-level records, then the interface must not be able to damage the system and bypass security and integrity constraints.

Overall structure of DBMS →
In today's data driven world, DBMS are essential for applications such as banking systems, e-commerce platforms, education and medical systems. They not only store and manage large amount of data, but also provide functionality that provides performance, security and scalability for multiple users with multiple access levels.

Components of a Database system →



1) Query Processor →

It interprets the requests (queries) received from end user via an application program into instructions. It also executes the user request which is received from DML compiler. query processor contains the following components →

- DML compiler →

It processes the DML statements into low level instruction (machine language), so that they can be executed.

- DDL interpreter →
It processes the DDL statements into a set of table containing meta data (data about data).
- Embedded DML Pre-compiler →
It processes DML statements embedded in an applica-tions program into procedural calls.
- Query Optimizer →
The query optimizer executes Instructions generated by the DML compiler and improves query execution efficiency by choosing the best query plan considering factors such as indexing, join order & available system resour-ces for instance, if a query involves joining two large tables, the optimizer will select the best join order to minimize query execution time.

2) Storage Manager →
Storage manager is an interface between the data stored in the database and the queries received it is also known as Database control system. It maintains the consistency and integrity of the database by applying of the database by applying the constraints & executing the DCL statements. It is responsible for updating, storing, deleting and retrieving data in the database. It contains the following components.
- Authorization manager →
It ensures role-based access control i.e. checks whether the particular person is privileged to perform the requested operation or not
- Integrity manager →
It checks the integrity constraints when the database is modified.
- Transaction manager →
It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before, & after the execution of a transaction.

- File manager →
It manages the file space and the data structure used to represent information in the database.
- Buffer manager →
It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

3) Disk storage →
It contains the following essential components →
- Data files →
It stores the actual data in the database.
- Data Dictionary →
It contains the information about the structure of data -base objects such as tables, constraints and relation -ships. It is the repository of information that governs the metadata.
- Indices →
Provides faster data retrieval by allowing the DBMS to find records quickly, improving query performance.

Architecture →

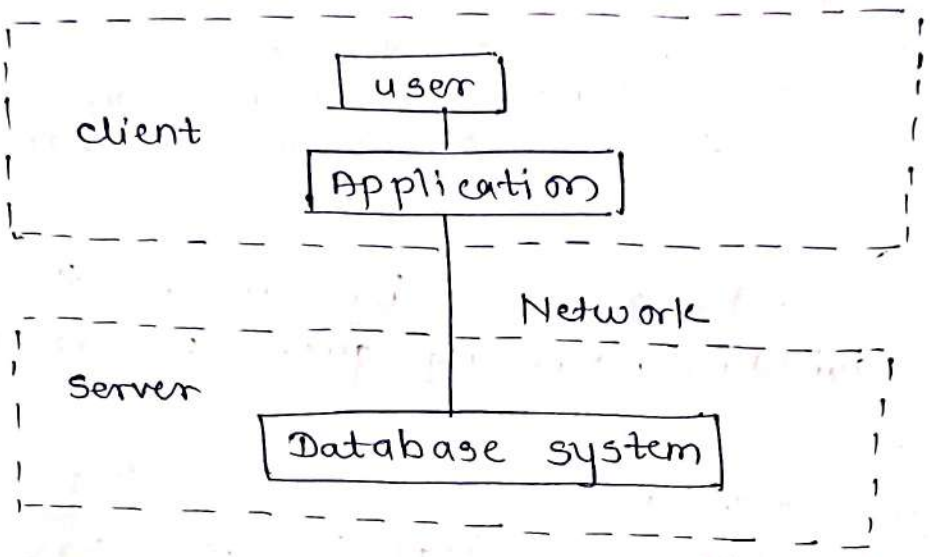
Two-tier Architecture →
The 2-tier Architecture is similar to client-server model. The application at the client end directly communicates with the database on the server side. APIs like ODBC & JDBC are used for this interaction. The serve side is responsible for providing query processing and transaction management functionalities in the client side, the user interfaces and application programs are run. The application on the client side establish a connection with server side to communicate with the DBMS - for ex- a library management system used in schools or small organizations is a classic example of 2-tier architecture.

1) Client layer (Tier 1) →
This is the user interface that library staff or users interact with. for ex - they might use a desktop applica- -tion to search for books, issue them, or check due dates -

2) Database layer (tier 2) →

The database server stores all the library records such as book details, user information and transaction logs. The client layer sends a request (like searching for a book) to the database layer which processes it and sends back the result. This separation allows the client to focus on the user interface, while the server handles data storage and retrieval.

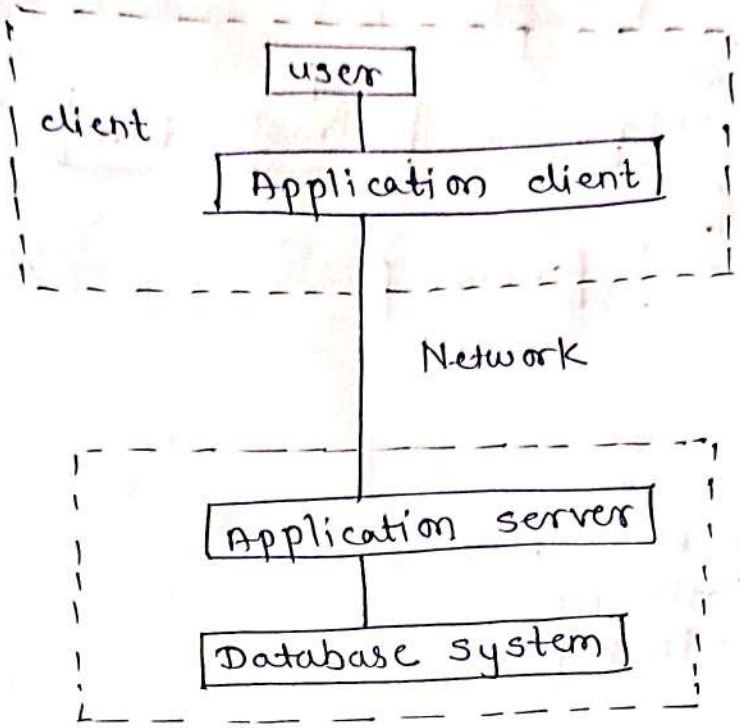
Two-Tier Architecture



Three-tier Architecture →

In 3-tier Architecture, there is another layer betn the client & the server, the client does not directly comm-unicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transac-tion management takes place, this intermediate layer acts as a medium for the exchange of partially processed data betn the server & the client this type of architecture is used in the case of the large web applications for ex. E-commerce store.

Three-Tier Architecture



Data Models →

Data Model gives us an ideaf that how the final system will look like after its complete implementation.

(or)

It defines the data elements and the relationships between the data elements.

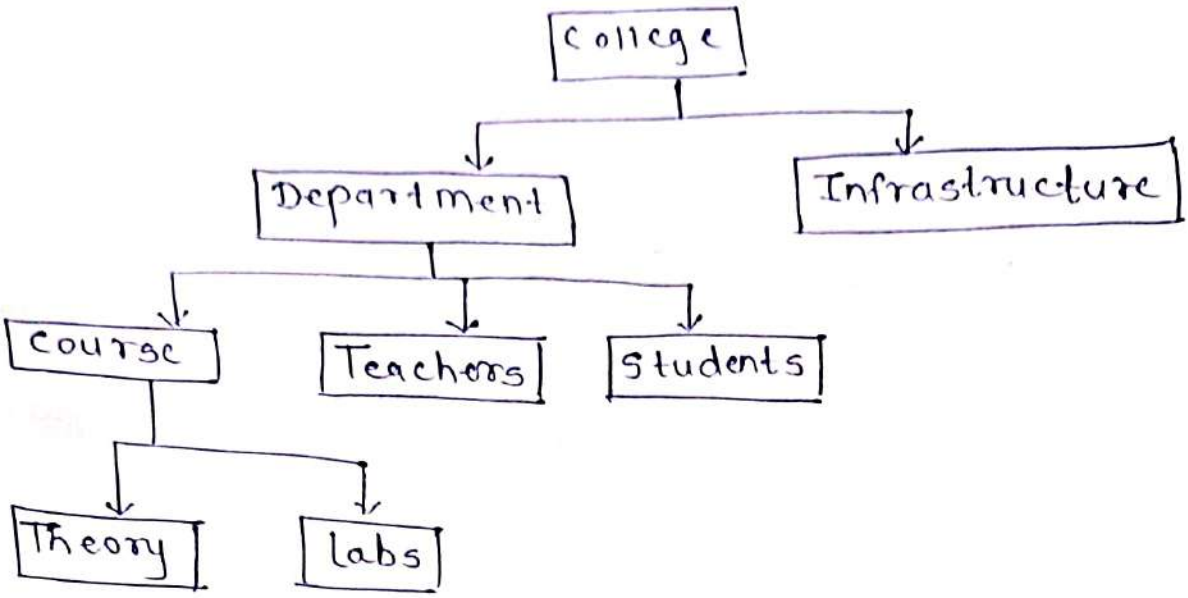
(or)

Data models are used to show how data is stored, connected, accessed and updated in the database management system.

1) Hierarchical Model →

- Hierarchical model was the first DBMS model.
- This model organizes the data in the hierarchical tree structure, with a single root, to which all the other data is linked.
- The hierarchy starts from the root which has root data and then it expand in the form of a tree adding child node to the parent node.
- This model easily represents some of the real-world relationships thi like food recipes, index of a book, etc.

Ex→



Features of Hierarchical Model →

1) One-to-many relationship.
2) Parent-child Relationship
3) Deletion Problem
4) Pointers

Advantanges →

- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

Disadvantages →

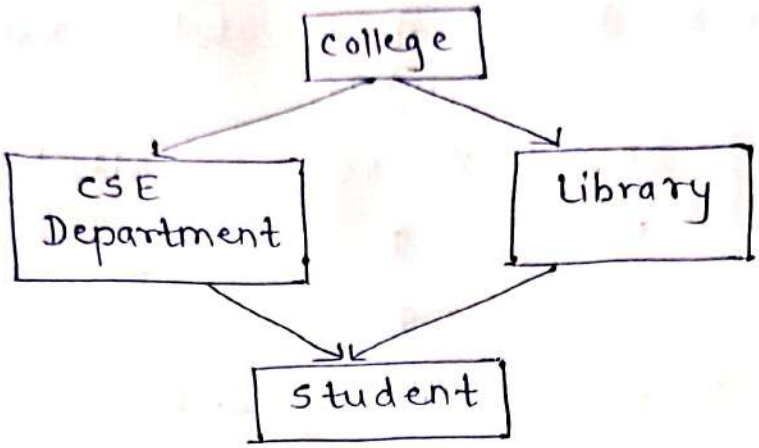
- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent node then that can't be represented using this model.
- If a parent node is deleted then the child node is automatically deleted.

2) Network Model →

- Network model is an extension of the hierarchical model.
- It was the most popular model before the relational model.
- This model is the same as the hierarchical model, the only difference in this record can have more than one parent. It replaces the hierarchical tree like with a graph.
- In this database model data is more related as more relationships

are established in this database model. Also, as the data is more related, hence accessing the data is also easier & fast.

Ex →



Features of a Network model →

1) Ability to merge more Relationships.
2) Many paths.
3) Circular linked list.

Advantages →

- The data can be accessed faster as compared to the hierarchical model. The data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is parent-child relationship so data integrity is present. Any data in parent record is reflected in the child record.

Disadvantages →

- As more and more relationships need to be handled the system might get complex. so, a user must be having detailed knowledge of the model to work with the model.
- Any change like updation, deletion, insertion is very complex.

3) Relational Model →

- Relational Model is the most widely used model.
- In this model, the data is maintained in the form of a two-dimensional table.
- All the information is stored in the form of row and columns.

- The basic structure of a relational model is tables.
- So, the tables are also called relations in the relational model.
- Ex- In this ex., we are discuss about student table.

column ↓

SID	SName	SAge	SClass
1101	Alex	14	9
1102	Maria	15	9
1103	Maya	14	10

→ attributes

→ tuple

table. (relation)

Features of Relational Model →

1) Tuples / Rows / Records.
2) Fields / Columns / Attributes.

Advantages →

- Simple - This model is more simple as compared to the network and hierarchical model.
- Scalable - This model can be easily scaled as we can add as many rows and columns we want.
- Structural Independence → We can make changes in database structure without changing the way to access the data. When we can make changes to the database structure without affecting the capability to DBMS to access the data we can say that structural independence has been achieved.

Disadvantages →

- Hardware Overheads - For hiding the complexities and making things for the user this model requires more powerful hardware computers and storage devices.
- Bad Design - As the relational model is very easy to design & use. The users don't need to know how the data is stored in order to access it. The design can lead to the development of a poor database which would down. if the database grows.

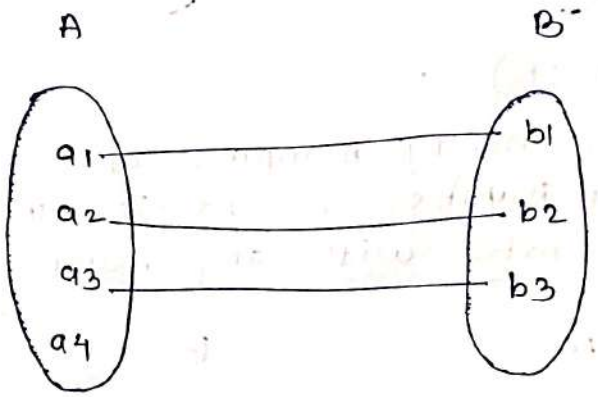
Cardinality →

Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship.

Types of Cardinality →

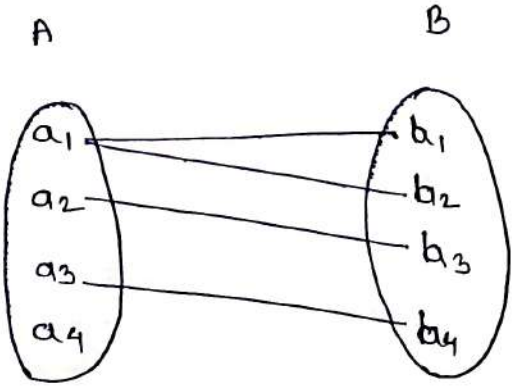
1) One-to-one (1:1) →

In this type of cardinality mapping, an entity in A is connected to at most one entity in B. or we can say that a unit or item in B is connected to at most one unit or item in A.



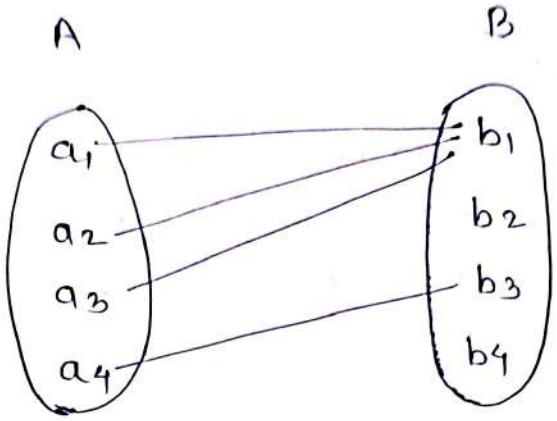
2) One-to-many (1:N) →

In this type of cardinality mapping, an entity in A is associated with any number of entities in B. Or we can say that one unit or item in B can be connected to at most one unit or items in A.



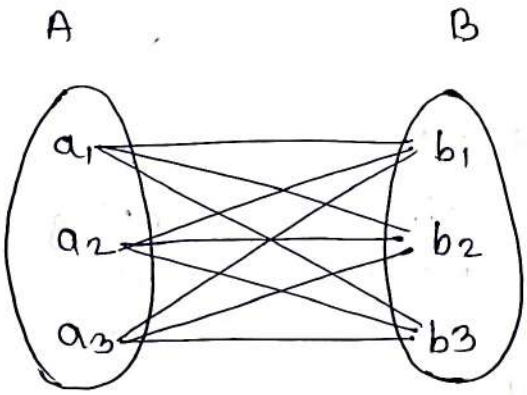
3) Many-to-One → (N:1)

In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say a unit or item in B can be associated with any ~~zero~~ number of entities or items in A.



4) Many-to-Many (N:N) →

In this type of cardinality mapping, an entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.



20/5/26