Chapter-1 (4 Marks)

Explain the advantages of using Relational Database Management Systems. What are the levels of data abstraction in Database Management system? [4+2]

Define database and database management system. List advantages of DBMS over file management. [1+3]

Define the terms Data Abstraction and Data Independence. Why they are important in DBMS? [2+3]

Explain the layers of abstract in database design. Why physical data independence is important in data modeling? [2+2]

Differentiate between schema and instances. What are the disadvantages of conventional file system? [4]

Briefly describe the significant differences between a file processing system to a DBMS. [4]

. What do you mean by scheme and instances? Mention the different levels of data abstraction and explain. [2+2]

Mention the advantages of the DBMS over the file processing system and explain briefly. [4]

What do you mean by data abstraction? List the various level of data abstraction and briefly explain. [1+3]

Why is data independence important in data modeling? Differentiate between schema and instances. [4]

with suitable example and draw an ER

What are the drawbacks of file system to store data? [4]

Why data independence is importance in data modeling? Differentiate between physical and logical data independence. [4]

1. Distinguish between a database and a DBMS. What is the advantage of separating the logical level and physical level in database design? [2+2]

1. What difficulties would you face if you used file system directly to implement a database application? What is physical data independence? [3+1]

✓ *Assume suitable data if necessary.*

1. Explain the difference between DDL, DML and DCL along with examples. [4]

A database is an organized collection of data that can be easily accessed, managed, and updated. It stores data in a structured format, making it easier to retrieve, manipulate, and analyze.

Examples of databases:

1. Relational Databases: MySQL, PostgreSQL, SQLite
2. NoSQL Databases: MongoDB, Firebase Firestore

A Database Management System (DBMS) is software that allows users to create, manage, and interact with a database. It provides tools for storing, retrieving, and modifying data efficiently.

Applications of DBMS

1. Banking & Finance – Storing customer details, transactions
2. E-Commerce – Managing products, orders, and customer information
3. Education – University databases for student records
4. Healthcare – Storing patient records, appointments, and prescriptions

Objectives of DBMS

1. Data Organization – Ensuring structured storage of data for easy retrieval.
2. Data Security – Protecting data from unauthorized access and corruption.
3. Data Consistency & Integrity – Preventing duplication and maintaining accuracy.
4. Concurrent Access – Allowing multiple users to access data simultaneously without conflicts.
5. Backup & Recovery – Providing data recovery in case of failures.
6. Scalability – Supporting the growth of data efficiently.
7. Data Independence – Separating data from application logic to avoid direct dependency.

Characteristics of DBMS/Advantages

1. Data Abstraction – Hides the complexities of data storage from users.
2. Data Redundancy Control – Reduces duplicate data storage.
3. ACID Properties – Ensures Atomicity, Consistency, Isolation, and Durability in transactions.
4. Multi-User Support – Allows concurrent access to data.
5. Query Language – Provides an interface (SQL, NoSQL) to interact with the database.
6. Data Integrity – Ensures data accuracy and consistency.
7. Data Security – Implements authentication and authorization mechanisms.
8. Backup & Recovery Mechanism – Provides data protection against crashes.

Disadvantages of Conventional File Systems

1. Redundancy: In a file system, the same piece of data might be stored in multiple files, leading to wasted storage space.
2. Inconsistency: Updates to one file may not be automatically reflected in others, causing inconsistent data across the system.
3. Lack of Data Integrity: Ensuring data integrity is challenging, as there are no built-in mechanisms to enforce consistency or validate data.
4. Poor Data Security: Limited security features, usually relying on basic file permissions, making it difficult to protect sensitive information.
5. Limited Querying Capabilities: Lacks the ability to perform complex queries.
6. Manual Data Management: Requires manual intervention for data organization, backup, recovery, and integrity checks, increasing the risk of errors.
7. Scalability Issues: Not suitable for large datasets or multi-user environments
8. Concurrency Issues: Poor handling of concurrent access, leading to potential data conflicts or corruption when multiple users try to access or modify the same data.

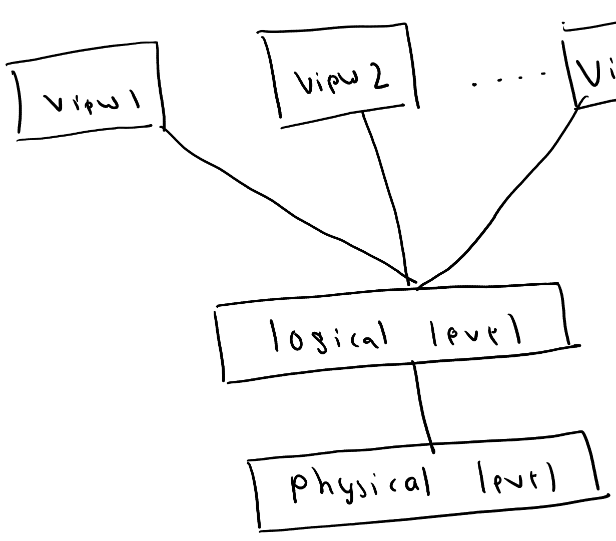
DBMS (Database Management System) and Flat File Management System:

DBMS	Flat File Management System
1. Structured with tables, rows, and columns	1. Stored in simple text or binary files
2. Minimizes redundancy through normalization	2. High redundancy as data is repeated across files
3. Allows complex queries using SQL	3. Requires manual searching
4. Supports multiple users accessing data simultaneously	4. no concurrency support
5. Provides authentication, authorization, and encryption	5. Basic security, mostly file permissions
6. Scalable for large datasets	6. Not suitabe for large data
7. Automated backup and recovery mechanisms	7. Backup must be handled manually
8. Supports relationships (e.g., primary and foreign keys)	8. No built-in relationship handling
9. MySQL, PostgreSQL, MongoDB, Oracle	9. CSV files, JSON files, text logs

Data Abstraction

Data Abstraction is a process in DBMS that hides complex details and only exposes essential features to the user. It helps simplify system design and provides a clear interface while concealing implementation details.

The diagram illustrates the three levels of data abstraction in DBMS, which are:



1. Physical Level (Lowest Level)

-> Describes how data is actually stored in memory (e.g., files, indexes, data structures).

-> Focuses on data storage format, compression, indexing, etc.

-> Example: Data is stored in binary files or B-trees for efficient retrieval.

2. Logical Level (Middle Level)

-> Defines what data is stored and the relationships between them.

-> Deals with tables, schemas, attributes, and constraints.

-> Example: A database schema with tables like Students (id, name, age) without concern for how it's stored.

3. View Level (Highest Level)

-> The user interacts with this level, seeing only the necessary data.

-> Hides complexity by providing different views for different users.

-> Example: A teacher's view may include Students(name, age), while an admin sees additional data like fee details.

These levels ensure data independence, meaning changes at one level do not necessarily affect other levels.

Data Independence

Data independence is the ability to modify a database schema at one level without affecting the schema at the next higher level.

Types of Data Independence

1. Physical Data Independence

-> The ability to change the physical storage of data without affecting the logical structure.
-> Changes in file organization, indexing, or storage format do not impact the database schema or application.
-> Example: Moving from HDD to SSD storage or changing from a B-Tree index to a Hash index without altering table structures.

2. Logical Data Independence

-> The ability to change the logical structure (schema) of the database without affecting the applications using it.
-> Changes in table structure (e.g., adding a new column) should not require changes in application code.
-> Example: Adding an email field to a Students table should not break existing queries that fetch name and age

Key Difference:

-> Physical Data Independence protects the logical schema from changes in physical storage.
-> Logical Data Independence protects application programs from changes in the logical schema.

Schema (Structure of the Database)

-> A schema is the overall design or blueprint of the database.

-> It defines the structure, tables, attributes, and relationships.

-> It remains constant unless modified by the database administrator.

Example:

```
CREATE TABLE Students (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

-> Here, the schema defines that the table Students has three attributes: id, name, and age.

Instance (Actual Data in the Database)

-> An instance is the actual data stored in the database at a particular moment.

-> It changes frequently as users insert, delete, or update records.

-> Example: After inserting some data, an instance of the Students table might look like this:

id	name	age
1	John	20
2	Alice	22
3	Bob	19

Schema	Instance
1. The overall structure/design of the database.	The actual data present in the database at a given moment.
2. Static, does not change frequently.	Dynamic, changes frequently as data is updated.
3. Defined once and typically remains the same throughout the database's life.	Varies over time as data is added, modified, or deleted.
4. Covers the entire structure of the database.	Covers the data stored within the structure at a specific point in time
5. Visible when designing or modifying the database.	Visible when querying or accessing the database.
6. Example: Table structure (columns, data types)	Example: Data inside the table rows

SQL (Structured Query Language

SQL is a standardized programming language used to interact with relational databases. It allows you to define, manipulate, and query data. SQL is used for creating, modifying, and querying databases and tables.

SQL Categories:

SQL is divided into different sublanguages based on the type of operation they perform:

1. DDL (Data Definition Language)

-> Deals with defining and managing database structure.

-> These commands are used to create, alter, and delete database objects (tables, schemas, etc)

-> Common DDL Commands:

- -> CREATE: Creates a new database object (e.g., table).
- -> ALTER: Modifies an existing database object (e.g., adding a column).
- -> DROP: Deletes an existing database object.
- -> TRUNCATE: Removes all data from a table but keeps the structure.

Example:

```
CREATE TABLE Students (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

2. DQL (Data Query Language)

-> Deals with querying the data in the database.

-> These commands are used to retrieve data.

-> SELECT: Fetches data from a database.

Example:

```
SELECT name, age FROM Students;
```

3. DML (Data Manipulation Language)

-> Deals with manipulating data in the database.

-> These commands are used to insert, update, delete, or manipulate the data within database tables.

-> Common DML Commands:

-> INSERT: Adds new data to a table.

-> UPDATE: Modifies existing data in a table.

-> DELETE: Removes data from a table.

-> MERGE: Combines data from two tables

Example:

```
INSERT INTO Students (id, name, age) VALUES (1, 'John', 20);
```

4. DCL (Data Control Language)

-> Deals with controlling access to the data in the database.

-> These commands are used to grant or revoke permissions and control user access

-> Common DCL Commands:

-> GRANT: Gives privileges to a user.

-> REVOKE: Removes privileges from a user.

Example:

```
GRANT SELECT, INSERT ON Students TO user1;
```