Course Code	:	**MCS-023**
Course Title	:	**Introduction to Database Management Systems**
Assignment Number	:	**BCA(III)/023/Assignment/2024-25**
Maximum Marks	:	**100**
Weightage	:	**25%**
Last Date of Submission	:	**31st October,2024(For July Session)** **30th April,2025(For January Session)**

This assignment has eight questions. Answer all questions of total 80 marks. Rest 20 marks are for viva voce. You may use illustrations and diagrams to enhance explanations. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation. Answer to each part of the question should be confined to about 300 words.

Q1. **(2 Marks)**

a) What is SQL? Explain its important features.

b) Consider the following schemas:

BOOK (Book_ID, Title, Publisher_ID, Year_of Pub, Price)

AUTHOR (Author_ID, Book_ID, Author Name)

PUBLISHER (Publisher_ID, Book_ID, Address, Name_of Pub, No._of Copies)

Write a query in SQL for the following:

(i) Find the name of authors whose books are published by "ABC Press".

(ii) Find the name of the author and price of the book, whose Book_ID is '100'.

(iii) Find the title of the books which are published by Publisher_ID '20' and are published in year 2011.

(iv) Find the address of the publisher who has published Book_ID "500".

Make suitable assumptions, if any. **(8 Marks)**

Q2.

a) With the help of a suitable example, discuss the insertion, deletion and updation anomalies that can occur in a database. Briefly discuss the mechanism to remove such anomalies. **(6 Marks)**

b) Write SQL commands for each of the following. Also illustrate the usage of each command through suitable example. **(4 Marks)**

(i) Creation of views

(ii) Creation of sequences

(iii) Outer join

(iv) To give access permission to any user

Q3.

a) What are integrity constraints? Discuss the various types of integrity constraints that can be imposed on database. **(3 Marks)**

b) How are database security and database integrity related? Briefly discuss the different levels of security measures which may be considered to protect the database. **(3 Marks)**

c) Consider the relation R (A, B, C, D, E) and the set of functional dependencies :-

F(A $\rightarrow$ D, {A,B} $\rightarrow$ C, D $\rightarrow$ E)

Assume that the decomposition of R into {R1 (A, B, C) and R2 (A, D, E)}.

Is this decomposition lossless? Justify? **(4 Marks)**

Q4.

a) Explain the Log-based recovery scheme with the help of an example. **(5 Marks)**

b) Compute the closure of the following set F of functional dependencies for relation schema R = (A, B, C, D, E).

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

List the candidate keys for R. **(5 Marks)**

Q5.

a) Give the limitations of file based system. How can they be overcome using DBMS? **(5 Marks)**

b) Discuss the importance of file organisation in databases. Mention the different types of file organisations available. Discuss any one of the mentioned file organisations in detail. **(5 Marks)**

Q6.

a) For what reasons is 2-phase locking protocol required? Explain. Discuss the disadvantages of basic 2-phase locking protocol. List the ways and means that can be used to overcome the disadvantages. **(5 Marks)**

b) List and explain the 4 basic properties of a Transaction with the help of appropriate examples. **(5 Marks)**

Q7.

a) What do you mean by fragmentation of a database? What is the need of fragmentation in DDBMS environment? Explain different types of fragmentation with an example of each. **(5 Marks)**

b) Explain the need of Distributed DBMS over Centralized DBMS. Also give the structure of Distributed DBMS. **(5 Marks)**

Q8. An organization needs to provide Medical facilities to its employees and their dependents. Organization is having a list of Doctors, Hospitals and Test centres for the employees facility. An employee may get Medical facility from the list of Doctors, Hospitals and Test centres provided by the organization to them. Employee does not need to pay anything for the facilities availed. The Doctors, Hospitals and Test centres directly raise their bill to the organization.

Identify the entities, relationships, constraints and cardinality and construct an ER diagram for the above mentioned specifications. List your assumptions and clearly indicate the cardinality mappings as well as any role indicators in your ER diagram. **(10 Marks)**

Q1.

a) What is SQL? Explain its important features.

SQL (Structured Query Language) is a standardized programming language used for managing and manipulating relational databases. It is used to perform tasks such as querying data, updating records, inserting data, and deleting records from databases.

Important Features of SQL:

1. **Data Definition Language (DDL):** SQL allows the creation, modification, and deletion of database structures like tables, indexes, and views. Common DDL commands include `CREATE`, `ALTER`, and `DROP`.
2. **Data Manipulation Language (DML):** It allows manipulation of data, i.e., adding, modifying, and deleting records. Common DML commands include `INSERT`, `UPDATE`, `DELETE`, and `SELECT`.
3. **Data Control Language (DCL):** SQL helps in controlling access to data through `GRANT` and `REVOKE` commands for assigning permissions.
4. **Transaction Control Language (TCL):** SQL allows transaction management with commands like `COMMIT`, `ROLLBACK`, and `SAVEPOINT` to ensure data integrity.
5. **Querying Capability:** SQL has powerful querying capabilities to retrieve specific data from a database using the `SELECT` statement with `WHERE`, `ORDER BY`, `GROUP BY`, etc.
6. **Joins:** SQL supports combining data from multiple tables using `JOIN` (e.g., INNER JOIN, LEFT JOIN, RIGHT JOIN).
7. **Data Integrity:** SQL helps maintain integrity using constraints like `PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, and `NOT NULL`.

b) SQL Queries Based on the Given Schema:

Schema Information:

- BOOK (Book_ID, Title, Publisher_ID, Year_of_Pub, Price)
- AUTHOR (Author_ID, Book_ID, Author_Name)
- PUBLISHER (Publisher_ID, Book_ID, Address, Name_of_Pub, No_of_Copies)

SQL Queries:

i) Find the name of authors whose books are published by "ABC Press".

```sql
SELECT A.Author_Name
FROM AUTHOR A
JOIN BOOK B ON A.Book_ID = B.Book_ID
JOIN PUBLISHER P ON B.Publisher_ID = P.Publisher_ID
WHERE P.Name_of_Pub = 'ABC Press';
```

This query retrieves author names by joining the `AUTHOR`, `BOOK`, and `PUBLISHER` tables and filtering the publisher name to "ABC Press".

ii) Find the name of the author and price of the book, whose Book_ID is '100'.

```sql
SELECT A.Author_Name, B.Price
FROM AUTHOR A
JOIN BOOK B ON A.Book_ID = B.Book_ID
WHERE B.Book_ID = 100;
```

This query fetches the author's name and the price of the book where `Book_ID` is 100 by joining the `AUTHOR` and `BOOK` tables.

iii) Find the title of the books which are published by Publisher_ID '20' and are published in the year 2011.

```sql
SELECT B.Title
FROM BOOK B
WHERE B.Publisher_ID = 20 AND B.Year_of_Pub = 2011;
```

This query retrieves the title of books where `Publisher_ID` is 20, and the publication year is 2011.

iv) Find the address of the publisher who has published Book_ID "500".

```sql
SELECT P.Address
FROM PUBLISHER P
JOIN BOOK B ON P.Publisher_ID = B.Publisher_ID
WHERE B.Book_ID = 500;
```

This query retrieves the address of the publisher by joining the `PUBLISHER` and `BOOK` tables where `Book_ID` is 500.

Q2.

a) With the help of a suitable example, discuss the insertion, deletion, and updation anomalies that can occur in a database. Briefly discuss the mechanism to remove such anomalies.

1. Insertion Anomaly: An insertion anomaly occurs when certain data cannot be inserted into the database without the presence of other data. This happens in poorly designed databases where attributes depend on more than one entity.

Example:

Consider a table storing information about students and their courses:

Student_ID	Student_Name	Course_ID	Course_Name
1	John	101	Mathematics
2	Alice	102	Physics

If we want to add a new course without assigning it to any student, we cannot insert it in this table unless we also have a student to associate with it. This is an insertion anomaly.

Solution:

To remove this anomaly, normalization can be applied, where we break this table into two tables: `STUDENT` and `COURSE`.

- STUDENT (Student_ID, Student_Name)
- COURSE (Course_ID, Course_Name)

2. Deletion Anomaly: A deletion anomaly occurs when deleting a record also removes unintended data.

Example:

In the same table:

Student_ID	Student_Name	Course_ID	Course_Name
1	John	101	Mathematics
2	Alice	102	Physics

If a student drops out of a course and we delete the student's record, we also lose information about the course. If we delete John's record, we also lose the information about the "Mathematics" course.

Solution:

Again, by normalizing the database into two tables (`STUDENT` and `COURSE`), we prevent such issues. Deleting a student does not affect the course information.

3. **Updation Anomaly:** An update anomaly occurs when data is duplicated in multiple rows, and updating data in one place requires updating it in multiple places.

Example:
If the course name "Mathematics" is changed to "Advanced Mathematics" in the above table, we must ensure that the course name is updated in all rows where it appears. If we miss any row, the data becomes inconsistent.

Solution:
To solve this anomaly, normalization is used. By creating separate tables for students and courses, we only need to update the course name in the `COURSE` table, and it reflects everywhere.

Mechanism to Remove Anomalies: The mechanism to remove these anomalies is **Database Normalization**. Normalization involves organizing data into multiple related tables and reducing redundancy. Common normalization forms include:

- **1st Normal Form (1NF):** Ensure that each column contains atomic values and that there are no repeating groups.
- **2nd Normal Form (2NF):** Ensure that all non-key attributes are fully functionally dependent on the primary key.
- **3rd Normal Form (3NF):** Ensure that no transitive dependency exists, meaning non-key attributes are not dependent on other non-key attributes.

b) Write SQL commands for each of the following. Also illustrate the usage of each command through suitable example.

i) Creation of views: A view is a virtual table in SQL, which provides a way to look at data in one or more tables.

sql Copy code

```sql
CREATE VIEW StudentCourseView AS
SELECT Student_Name, Course_Name
FROM STUDENT S
JOIN COURSE C ON S.Student_ID = C.Student_ID;
```

This command creates a view called `StudentCourseView`, which shows student names and the courses they are enrolled in.

ii) **Creation of sequences:** A sequence in SQL is used to generate a series of unique numbers, often used for generating primary keys.

```sql
CREATE SEQUENCE student_seq
START WITH 1
INCREMENT BY 1;
```

This command creates a sequence `student_seq` that starts from 1 and increments by 1 for every next value. It can be used to auto-generate unique student IDs.

iii) **Outer join:** An outer join returns all rows from one table and the matching rows from another table. If there is no match, NULL values are returned.

```sql
SELECT S.Student_Name, C.Course_Name
FROM STUDENT S
LEFT OUTER JOIN COURSE C ON S.Student_ID = C.Student_ID;
```

This query uses a `LEFT OUTER JOIN` to return all students, even if they are not enrolled in any course.

iv) **To give access permission to any user:** The `GRANT` command in SQL is used to give privileges to users.

```sql
GRANT SELECT, INSERT ON STUDENT TO 'user1';
```

This command gives the `SELECT` and `INSERT` privileges on the `STUDENT` table to a user named `user1`.

Question 3

(a) What are integrity constraints? Discuss the various types of integrity constraints that can be imposed on a database.

- Introduction:
  - Integrity constraints are rules enforced on data in a database to maintain its accuracy and consistency. These constraints ensure that the data adheres to the business rules and prevents the entry of invalid data.
- Main Points:
  1. Domain Integrity:
     - Ensures that all values in a column fall within a specified domain.
     - **Example:** A column `Age` might have a constraint that only allows integers between 0 and 150.
  2. Entity Integrity:
     - Ensures that every table has a primary key and that the columns that make up the primary key are unique and not null.
     - **Example:** In a `Student` table, `Student_ID` as a primary key ensures each student is uniquely identified.
  3. Referential Integrity:
     - Ensures that a foreign key value in one table corresponds to a valid primary key in another table.
     - **Example:** A `Course_ID` in an `Enrollment` table must match a `Course_ID` in the `Courses` table.
  4. Unique Constraints:
     - Ensures that all values in a column are unique across the database.
     - **Example:** An email column in a user table must have unique values to prevent duplicate registrations.
  5. Check Constraints:
     - Ensures that all values in a column satisfy a specific condition.
     - **Example:** A `Salary` column may have a check constraint ensuring that the salary is greater than zero.

(b) How are database security and database integrity related? Discuss different levels of security measures that may be considered to protect the database.

- **Introduction:**
  - Database security refers to the mechanisms that protect the database against unauthorized access, misuse, or theft, while database integrity ensures the accuracy and consistency of data over its entire lifecycle.
- **Main Points:**
  1. **Relationship Between Security and Integrity:**
     - Integrity depends on security measures to prevent unauthorized modifications that could compromise the correctness of data.
     - For example, strong access controls prevent unauthorized users from altering critical data, thereby maintaining its integrity.
  2. **Levels of Security Measures:**
     - **Physical Security:** Protection of the hardware and storage media from physical threats such as theft or natural disasters.
     - **Network Security:** Implementation of firewalls, encryption, and secure protocols (e.g., HTTPS, SSH) to protect data in transit.
     - **User Access Controls:** Role-based access control (RBAC) where users are granted permissions based on their role within the organization, ensuring that only authorized users can access certain data.
     - **Database Hardening:** Disabling unnecessary services, applying patches, and configuring the database securely to reduce vulnerabilities.
     - **Encryption:** Protecting sensitive data by converting it into a secure format that can only be decrypted by authorized users.
- **Conclusion:** Implementing robust security measures is crucial for maintaining the integrity of a database. By preventing unauthorized access and ensuring proper controls, an organization can protect its data from corruption or loss.

(c) Lossless Decomposition

- **Introduction:**
  - A decomposition is lossless if the original relation can be perfectly reconstructed from the decomposed relations without any loss of information.
- **Main Points:**
  1. **Given Relation and Functional Dependencies:**
     - Relation: `R(A, B, C, D, E)`
     - Functional Dependencies: `F(A → D, {A,B} → C, D → E)`
  2. **Decomposition:**
     - Decomposed into `R1(A, B, C)` and `R2(A, D, E)`
  3. **Checking Losslessness:**
     - According to the functional dependencies and the decomposition, the decomposition is lossless if one of the following conditions is satisfied:
       - The common attribute is a superkey in either of the decomposed relations.
     - In this case, attribute `A` is common between `R1` and `R2`, and `A` can determine `D` in `R2`. Thus, it satisfies the condition for a lossless decomposition.
- **Conclusion:** The decomposition of relation `R` into `R1(A, B, C)` and `R2(A, D, E)` is lossless, as it preserves all the original information.

Question 4

(a) Explain the Log-based recovery scheme with the help of an example.

- **Introduction:**
  - Log-based recovery is a method used in database management systems to ensure that a database can be restored to a consistent state after a failure by using logs that record the history of transactions.
- **Main Points:**
  1. **Transaction Log:**
     - A log is a sequence of records, with each record containing the details of transactions. It records changes made by transactions in a chronological order.
  2. **Types of Log Records:**
     - **Start Transaction:** Indicates the start of a transaction.
     - **Update:** Records the old value and the new value for a data item when a transaction updates it.
     - **Commit:** Indicates that a transaction has been successfully completed.
     - **Abort:** Indicates that a transaction has been rolled back.
  3. **Recovery Process:**
     - **Redo:** If a transaction was committed, the changes it made are reapplied to the database during recovery.
     - **Undo:** If a transaction was not committed, the changes it made are rolled back during recovery.
  4. **Example:**
     - Suppose a transaction `T1` updates a data item `X` from 10 to 20. The log would record:

```plaintext
[START T1]
[T1, X, 10, 20]
[COMMIT T1]
```

     - In case of a failure, the system will use the log to either redo the changes made by `T1` or undo them, depending on whether `T1` was committed or not.

(b) Compute the closure of the given set of functional dependencies for relation schema R = (A, B, C, D, E). List the candidate keys.

- Introduction:
  - The closure of a set of attributes is the set of attributes that can be functionally determined by that set. It is useful in determining candidate keys.
- Main Points:
  1. Given Functional Dependencies:
     - `A → BC`
     - `CD → E`
     - `B → D`
     - `E → A`
  2. Closure Calculation:
     - Closure of {A}:
       - Start with {A}.
       - Since `A → BC`, add B and C to the closure: {A, B, C}.
       - `B → D` adds D: {A, B, C, D}.
       - `CD → E` adds E: {A, B, C, D, E}.
       - Therefore, closure of {A} = {A, B, C, D, E}.
     - Since the closure of {A} gives all attributes, {A} is a candidate key.
  3. Candidate Keys:
     - Since {A} determines all attributes, it is the candidate key.
- **Conclusion:** The candidate key for the given relation schema is {A}, which can uniquely identify each tuple in the relation.

Question 5

(a) Give the limitations of a file-based system. How can they be overcome using DBMS?

- Introduction:
  - Traditional file-based systems have several limitations that affect their efficiency and reliability in handling large volumes of data.
- Main Points:
  1. Limitations of File-Based Systems:
     - **Data Redundancy and Inconsistency:** Same data might be duplicated in multiple files, leading to inconsistencies.
     - **Difficulty in Accessing Data:** No standard query language exists, making data retrieval complex and time-consuming.
     - **Data Isolation:** Data is stored in multiple files in different formats, making it difficult to combine and retrieve related data.
     - **Integrity Problems:** Enforcement of data integrity rules is challenging, leading to potential data corruption.
     - **Concurrent Access Anomalies:** Simultaneous access to data by multiple users can lead to data conflicts without proper management.
     - **Security Problems:** File-based systems lack robust security mechanisms, exposing data to unauthorized access.
  2. Overcoming Limitations Using DBMS:
     - **Data Integration:** DBMS integrates data into a single database with a standardized structure, reducing redundancy.
     - **Efficient Data Access:** SQL provides a powerful and standardized query language for efficient data retrieval.
     - **Data Consistency and Integrity:** DBMS enforces integrity constraints, ensuring consistent and reliable data.
     - **Concurrent Access Management:** DBMS provides mechanisms like locking and transaction control to manage concurrent access, preventing conflicts.
     - **Enhanced Security:** DBMS supports role-based access control, encryption, and other security features to protect data.

(b) Discuss the importance of file organization in databases. Mention the different types of file organizations available. Discuss any one in detail.

- **Introduction:**
  - File organization refers to the way data is stored in a database. Proper file organization is crucial for efficient data retrieval, storage management, and overall database performance.
- **Main Points:**
  1. **Importance of File Organization:**
     - **Efficient Data Access:** Well-organized files ensure that data can be retrieved quickly and efficiently.
     - **Storage Management:** Proper file organization optimizes the use of storage space, reducing redundancy and wastage.
     - **Data Integrity:** Organized files reduce the risk of data corruption and inconsistencies.
  2. **Types of File Organizations:**
     - **Heap (Unordered) File Organization:** Data is stored in no particular order. Retrieval requires scanning the entire file, making it less efficient for large datasets.
     - **Sequential (Ordered) File Organization:** Data is stored in a specific order, usually based on a key field, which allows for faster retrieval.
     - **Hashed File Organization:** Data is stored based on a hash function applied to a key field, which provides direct access to records.
     - **Clustered File Organization:** Related records from different tables are stored together, which can improve retrieval times for certain queries.
  3. **Detailed Discussion on Sequential File Organization:**
     - **Sequential File Organization:**
       - **How It Works:** Data is stored in a sorted order based on a key field (e.g., ID, name). New records are placed in their correct position to maintain the order.
       - **Advantages:** Efficient for retrieving records in sorted order or for range queries. Useful for applications where records are frequently read in sequence.
       - **Disadvantages:** Insertion and deletion of records can be time-consuming as the file must be reorganized to maintain the order. Not ideal for random access.

Question 6

(a) For what reasons is the 2-phase locking protocol required? Discuss the disadvantages of basic 2-phase locking protocol. List the ways to overcome the disadvantages.

- Introduction:
  - The 2-phase locking (2PL) protocol is a concurrency control method used in databases to ensure that transactions are executed in a serializable manner, preventing conflicts and maintaining data integrity.
- Main Points:
  1. Need for 2-Phase Locking:
     - **Serializable Schedules:** Ensures that the schedule of transactions is serializable, meaning it produces the same result as if the transactions were executed in some sequential order.
     - **Prevention of Conflicts:** Locks are acquired before accessing data, preventing conflicting operations from different transactions on the same data.
     - **Data Consistency:** Maintains consistency by ensuring that no other transaction can modify data while it is being used by another transaction.
  2. Phases of 2-Phase Locking:
     - **Growing Phase:** Transactions may acquire locks but cannot release any locks.
     - **Shrinking Phase:** Transactions may release locks but cannot acquire any new locks.
  3. Disadvantages of 2-Phase Locking:
     - **Deadlock:** Occurs when two or more transactions are waiting indefinitely for locks held by each other.
     - **Reduced Concurrency:** The strict locking protocol can reduce the level of concurrency, leading to performance bottlenecks.
     - **Long Wait Times:** Transactions may have to wait for long periods if another transaction holds a required lock.
  4. Ways to Overcome Disadvantages:
     - **Deadlock Prevention:** Techniques like timeout and deadlock detection can help in managing deadlocks.
     - **Optimistic Concurrency Control:** Transactions are executed without locking, and conflicts are checked at the end, reducing the need for strict locking.
     - **Use of Timestamps:** Assigning timestamps to transactions can help in managing the order of execution and reducing conflicts.

(b) List and explain the 4 basic properties of a Transaction with examples.

- **Introduction:**
  - A transaction in a database is a sequence of operations performed as a single logical unit of work. It must satisfy the ACID properties to ensure data integrity and consistency.
- **Main Points:**
  1. **Atomicity:**
     - Ensures that a transaction is all-or-nothing. If any part of the transaction fails, the entire transaction is rolled back, leaving the database unchanged.
     - **Example:** Transferring money from one bank account to another involves debiting one account and crediting another. If either operation fails, the entire transaction is rolled back.
  2. **Consistency:**
     - Ensures that a transaction brings the database from one valid state to another, maintaining database integrity.
     - **Example:** After a successful transaction, all constraints (e.g., foreign keys, unique constraints) must hold true.
  3. **Isolation:**
     - Ensures that transactions are executed in isolation from each other, so intermediate states of a transaction are invisible to other transactions.
     - **Example:** If two transactions are updating the same record, one must wait until the other completes to avoid conflicts.
  4. **Durability:**
     - Ensures that once a transaction is committed, the changes are permanent, even in the case of a system failure.
     - **Example:** Once a booking is confirmed in an airline reservation system, it remains confirmed, even if the system crashes immediately afterward.
- **Conclusion:** The ACID properties are fundamental to the reliable operation of database transactions, ensuring that even in the face of failures, the database remains consistent and trustworthy.

Question 7

(a) What do you mean by fragmentation of a database? What is the need for fragmentation in a DDBMS environment? Explain different types of fragmentation with examples.

- Introduction:
  - Fragmentation refers to the process of dividing a database into smaller, manageable pieces called fragments, which can be stored at different sites in a Distributed Database Management System (DDBMS).
- Main Points:
  1. Need for Fragmentation in DDBMS:
     - **Improved Performance:** Local access to data is faster as data is stored closer to where it is needed.
     - **Parallel Processing:** Different fragments can be processed in parallel, improving the overall performance of the system.
     - **Increased Availability:** If one fragment is unavailable, others can still be accessed, increasing system reliability.
  2. Types of Fragmentation:
     - Horizontal Fragmentation:
       - **Definition:** Divides a table into rows, where each fragment contains a subset of the rows that satisfy a certain condition.
       - **Example:** A `Customer` table could be horizontally fragmented into `Customer_US` (containing customers from the US) and `Customer_Europe` (containing European customers).
     - Vertical Fragmentation:
       - **Definition:** Divides a table into columns, where each fragment contains a subset of the columns.
       - **Example:** A `Customer` table could be vertically fragmented into `Customer_Contact` (containing contact details) and `Customer_Orders` (containing order details).
     - Mixed (Hybrid) Fragmentation:
       - **Definition:** A combination of horizontal and vertical fragmentation, where a table is first horizontally fragmented and then vertically fragmented (or vice versa).
       - **Example:** A `Customer` table could be first horizontally fragmented by region and then vertically fragmented into contact and order details.

(b) Explain the need for Distributed DBMS over Centralized DBMS. Also, give the structure of Distributed DBMS.

- Introduction:
  - A Distributed Database Management System (DDBMS) manages a database that is distributed across multiple locations, unlike a Centralized DBMS where all data is stored at a single location.
- Main Points:
  1. Need for DDBMS:
     - **Scalability:** DDBMS can handle larger datasets and a higher number of transactions by distributing the load across multiple servers.
     - **Reliability and Availability:** Data is replicated across multiple sites, ensuring that even if one site fails, the data is still available.
     - **Local Autonomy:** Each site in a DDBMS can operate independently, making the system more flexible and easier to manage.
     - **Improved Performance:** Data is stored closer to where it is used, reducing access times and network traffic.
  2. Structure of DDBMS:
     - **Global Schema:** The overall structure of the database that defines what data is available and how it is organized.
     - **Fragmentation Schema:** Defines how the database is divided into fragments, which can be either horizontal, vertical, or hybrid.
     - **Replication Schema:** Specifies how data is replicated across different sites to ensure reliability and availability.
     - **Allocation Schema:** Defines where each fragment or replica is stored in the network.
     - **Local DBMS:** Each site in the network has its own local DBMS, which manages the data stored at that site.
     - **Global Query Processor:** Manages queries that involve data from multiple sites, ensuring that the query is processed efficiently across the distributed network.

Question 8

Construct an ER diagram for an organization providing medical facilities to its employees and their dependents, listing the entities, relationships, constraints, and cardinality.

- **Introduction:**
  - The problem statement requires the design of an ER (Entity-Relationship) diagram to represent an organization's medical facilities management system, including entities like Employees, Doctors, Hospitals, and Test Centres, and their relationships.
- **Entities:**
  1. **Employee:**
     - Attributes: Employee_ID (Primary Key), Name, Department, Contact_Details.
  2. **Dependent:**
     - Attributes: Dependent_ID (Primary Key), Name, Relationship, Employee_ID (Foreign Key).
  3. **Doctor:**
     - Attributes: Doctor_ID (Primary Key), Name, Specialty, Contact_Details.
  4. **Hospital:**
     - Attributes: Hospital_ID (Primary Key), Name, Address, Contact_Details.
  5. **Test_Centre:**
     - Attributes: Test_Centre_ID (Primary Key), Name, Address, Contact_Details.
  6. **Medical_Facility:**
     - Attributes: Facility_ID (Primary Key), Type (e.g., Consultation, Surgery, Test), Date, Employee_ID (Foreign Key), Doctor_ID (Foreign Key), Hospital_ID (Foreign Key), Test_Centre_ID (Foreign Key).

- Relationships:
    1. **Employee - Dependent:**
        - One-to-Many relationship between Employee and Dependent.
    2. **Employee - Medical_Facility:**
        - One-to-Many relationship between Employee and Medical_Facility.
    3. **Doctor - Medical_Facility:**
        - Many-to-One relationship between Medical_Facility and Doctor.
    4. **Hospital - Medical_Facility:**
        - Many-to-One relationship between Medical_Facility and Hospital.
    5. **Test_Centre - Medical_Facility:**
        - Many-to-One relationship between Medical_Facility and Test_Centre.
- Constraints:
    - **Participation Constraint:** Each Medical_Facility must involve exactly one Employee, one Doctor (if applicable), one Hospital (if applicable), or one Test_Centre (if applicable).
    - **Cardinality:**
        - Employee to Dependent: 1
          (One Employee can have multiple Dependents).
        - Medical Facility to Employee: N:1 (Multiple Medical Facilities can be availed by one Employee).
        - Medical Facility to Doctor: N:1 (Multiple Medical Facilities can involve the same Doctor).
        - Medical Facility to Hospital: N:1 (Multiple Medical Facilities can involve the same Hospital).
        - Medical Facility to Test Centre: N:1 (Multiple Medical Facilities can involve the same Test Centre).
- Conclusion:
    - The ER diagram effectively models the organization's medical facilities management system, capturing all relevant entities, relationships, and constraints. This diagram serves as the blueprint for designing the underlying database schema, ensuring data integrity and efficient data management.