ADITYA DHIMAN

QUESTION BANK SOLUTION ADBMS

UNIT 1

Short Answers (2 Marks Questions)

1. **Define the terms data and information?**
    - **Data**: Raw, unorganized facts that need to be processed. Example: numbers, names, dates.
    - **Information**: Data that has been processed and organized in a meaningful way. Example: a report summarizing sales data.
2. **Define (i) Database (ii) DBMS**
    - **Database**: A structured collection of data stored and accessed electronically. Example: Customer records in a retail system.
    - **DBMS (Database Management System)**: Software that manages databases, providing an interface for users to interact with data. Example: MySQL, Oracle.
3. **What are the disadvantages of the file processing system?**
    - Data redundancy and inconsistency
    - Difficulty in accessing data
    - Data isolation
    - Integrity problems
    - Concurrent access anomalies
    - Security problems
4. **Define instances and schemas of the database?**
    - **Instance**: The collection of information stored in the database at a particular moment. Example: A snapshot of the database.
    - **Schema**: The overall design of the database. Example: The structure of tables and relationships.

5. **What is a data model? List the types of data models?**
   - **Data Model**: A conceptual framework for organizing and defining the data, its relationships, and constraints. Types: Hierarchical, Network, Relational, Object-oriented.
6. **What is data abstraction?**
   - Data abstraction is the process of hiding the complex details of the database and showing only the essential features to the users. It simplifies database design and use by separating the logical aspects from the physical aspects.
7. **Who is DBA? What are the responsibilities of DBA?**
   - **DBA (Database Administrator)**: A person responsible for managing the database environment. Responsibilities include database design, implementation, maintenance, security, and ensuring data integrity and performance tuning.
8. **Discuss Data Independence?**
   - Data independence refers to the capability to change the schema at one level without affecting the schema at the next higher level. Types: Logical data independence and Physical data independence.
9. **Explain the Application of DBMS?**
   - DBMS applications include banking systems, airline reservations, universities, credit card transactions, telecommunication systems, and e-commerce platforms.
10. **What are the advantages of DBMS?**
    - Reduced data redundancy
    - Improved data integrity
    - Enhanced data security
    - Easier data access and manipulation
    - Data consistency and concurrency control
11. **Define the terms: 1. Physical schema 2. Logical schema**
    - **Physical Schema**: Describes how data is stored physically in the database (storage details).

- **Logical Schema**: Describes the logical structure of the database (tables, relationships).

12. **Give the levels of data abstraction?**
    - **Physical Level**: How data is stored.
    - **Logical Level**: What data is stored and the relationships.
    - **View Level**: How users see the data.
13. **What is the purpose of the storage manager?**
    - The storage manager is responsible for the efficient storage, retrieval, and update of data. It handles data storage structure, access methods, and managing space.
14. **Differentiate between the database system and the file system?**
    - **Database System**: Organized collection of data managed by DBMS, supports multi-user environment, and provides data integrity and security.
    - **File System**: Simple storage of files without DBMS capabilities, prone to data redundancy and inconsistency.
15. **Define Database Recovery.**
    - Database recovery involves restoring the database to a correct state in the event of a failure, ensuring data integrity and consistency. It uses mechanisms like backup and transaction logs.
16. **Define Two Tier and Three Tier Architecture?**
    - **Two-Tier Architecture**: Client directly communicates with the database server.
    - **Three-Tier Architecture**: Client communicates with an application server, which then interacts with the database server, enhancing scalability and security.
17. **Discuss Transaction management?**
    - Transaction management ensures that all database transactions are processed reliably and adhere to ACID properties (Atomicity, Consistency, Isolation, Durability).
18. **Define Relationship and Relationship set?**
    - **Relationship**: An association among entities. Example: Student enrolls in a Course.

- **Relationship Set**: A set of relationships of the same type. Example: All enrollments of students in courses.

19. **Explain the architecture of DBMS?**
    - The DBMS architecture typically includes layers such as the storage manager, query processor, and transaction manager, with interfaces for users and applications.
20. **What is the difference between database and data warehouse? Explain with example.**
    - **Database**: Designed for real-time operations, handles day-to-day transactions. Example: Online banking system.
    - **Data Warehouse**: Designed for analysis and reporting, consolidates data from multiple databases. Example: Business intelligence platform for sales analysis.

Medium Answers (5 Marks Questions)

Q21: Define Data Abstraction and Discuss Levels of Abstraction

Data Abstraction Data abstraction simplifies database interaction by hiding complexities and showing only essential details.

Levels of Data Abstraction

1. **Physical Level**
    - Describes how data is physically stored.
    - Example: File organization and storage formats.
2. **Logical Level**
    - Describes what data is stored and the relationships among them.
    - Example: Tables, columns, and relationships in a database.
3. **View Level**
    - Describes specific parts of the database for different users.
    - Example: User-specific views like a sales report view for managers.

Q22: Discuss Different Types of Data Models

Data Models Data models define the structure, relationships, and constraints of the data in a database.

1. **Hierarchical Model**
   - Organizes data in a tree-like structure with parent-child relationships.
   - Example: Organizational charts.
2. **Network Model**
   - Uses a graph structure to represent many-to-many relationships.
   - Example: Airline reservation systems.
3. **Relational Model**
   - Represents data in tables with rows (tuples) and columns (attributes).
   - Example: Customer and Order tables in an e-commerce database.
4. **Object-Oriented Model**
   - Represents data as objects, similar to object-oriented programming.
   - Example: Multimedia databases.
5. **Entity-Relationship Model**
   - Uses entities, attributes, and relationships to design databases.
   - Example: ER diagrams for a university database.

Q23: Describe the Architecture of DBMS

Architecture of DBMS The DBMS architecture consists of three primary levels: internal, conceptual, and external.

1. **Internal Level**
   - Describes physical storage details.
   - Involves file storage, indexing, and data blocks.
2. **Conceptual Level**
   - Describes the logical structure of the entire database.
   - Focuses on tables, columns, and relationships.
3. **External Level**
   - Provides user-specific views of the database.

- Different users have different views tailored to their needs.

Components

- **Storage Manager**: Manages data storage and retrieval.
- **Query Processor**: Translates and executes database queries.
- **Transaction Manager**: Ensures reliable transaction processing with ACID properties.
- **Database Engine**: Core service for storing, processing, and securing data.

Q24: Compare and Contrast File Systems with Database Systems

File Systems

- **Data Redundancy**: Higher due to lack of central control.
- **Data Isolation**: Hard to integrate data spread across multiple files.
- **Integrity Problems**: Difficult to enforce consistent data integrity.
- **Concurrent Access**: Challenging to manage simultaneous data access.
- **Security**: Basic, often insufficient for complex needs.

Database Systems

- **Data Redundancy**: Reduced through normalization.
- **Data Integration**: Centralized control, easier integration.
- **Integrity Enforcement**: Strong support for data integrity through constraints.
- **Concurrent Access**: Advanced mechanisms for safe concurrent access.
- **Security**: Robust security features like authentication and access control.

Q25: Discuss Additional Features of the ER-Model

Enhanced ER-Model (EER) The EER model extends the original ER model with additional features to handle complex data structures.

1. **Specialization and Generalization**
    - Specialization: Creating sub-entities from a parent entity.
    - Generalization: Combining similar entities into a parent entity.
    - Example: Employee entity specialized into full-time and part-time employees.

2. **Aggregation**
   - Treats relationships as higher-level entities.
   - Example: Project and department relationship aggregated to show additional relationships.
3. **Inheritance**
   - Sub-entities inherit attributes from parent entities.
   - Example: A student entity inheriting attributes from a person entity.
4. **Categorization**
   - Divides entities into categories based on criteria.
   - Example: Accounts categorized into savings and checking.
5. **Multiple Inheritance**
   - Entities inherit attributes from multiple parent entities.
   - Example: Vehicle entity inheriting from land vehicle and water vehicle.

Q26: Discuss About Logical Database Design

Logical Database Design Logical database design involves creating a blueprint for the database structure that captures the logical view of the data.

Steps in Logical Database Design

1. **Requirement Analysis**
   - Identify data to be stored and user requirements.
2. **Conceptual Design**
   - Develop an ER model with entities, attributes, and relationships.
3. **Normalization**
   - Organize data into tables to eliminate redundancy and ensure integrity.
4. **Logical Schema Definition**
   - Translate the ER model into a relational schema with tables and constraints.
5. **Validation**
   - Ensure the schema supports required operations and queries.

Advantages

- Ensures consistency and integrity.
- Facilitates efficient query processing.
- Provides a clear structure for physical database design.

Q27: What is Physical, Logical, and View Level Data Abstraction?

Physical Level

- Describes how data is stored on storage devices.
- Involves storage allocation and indexing.
- Example: Data blocks on disk.

Logical Level

- Describes the logical structure and relationships of data.
- Example: Tables, columns, and relationships.

View Level

- Provides specific views of the database for different users.
- Example: Customized view showing customer details for a sales team.

Conclusion Data abstraction levels manage complexity by separating storage, structure, and user interaction, enhancing usability and security.

Q28: What is a Constraint in Database? Explain Types of Constraints with Suitable Examples

Constraints Constraints are rules enforced on data columns to ensure data integrity and reliability.

Types of Constraints

1. **Primary Key Constraint**
    - Ensures each row is unique and not null.
    - Example: id column in the students table.
2. **Foreign Key Constraint**
    - Ensures referential integrity by linking columns in different tables.
    - Example: student_id in the enrollments table referring to id in the students table.

3. **Unique Constraint**
   - Ensures all values in a column are unique.
   - Example: email column in the users table.
4. **Not Null Constraint**
   - Ensures a column cannot have null values.
   - Example: name column in the employees table.
5. **Check Constraint**
   - Ensures values in a column meet a specific condition.
5. **Check Constraint**
   - Ensures values in a column meet a specific condition.
   - Example: age column in the persons table with a check constraint that ensures values are greater than 18.

Conclusion Constraints maintain the integrity, accuracy, and reliability of the data in a database by enforcing specific rules on the data columns.

Q29: Describe the Relational Model

Relational Model The relational model is a widely used database model that organizes data into tables (relations) composed of rows (tuples) and columns (attributes).

Key Concepts

1. **Tables (Relations)**
   - Represents entities and their attributes.
   - Example: A Customers table with columns CustomerID, Name, Address.
2. **Rows (Tuples)**
   - Represents individual records within a table.
   - Example: A specific customer's details in the Customers table.
3. **Columns (Attributes)**
   - Represents the properties of entities.
   - Example: Name and Address in the Customers table.

4. **Keys**
    - **Primary Key**: Unique identifier for rows in a table.
    - **Foreign Key**: Links rows in one table to rows in another, ensuring referential integrity.
    - Example: CustomerID as a primary key in Customers and as a foreign key in Orders.

Advantages

- Simplifies data organization and retrieval.
- Supports powerful querying through SQL.
- Ensures data integrity through constraints.

Q30: Difference Between 2-Tier and 3-Tier Architecture

2-Tier Architecture

1. **Client-Server Model**
    - The client directly communicates with the database server.
    - Example: A desktop application interacting with a database.
2. **Components**
    - **Client**: User interface and application logic.
    - **Server**: Database management.

Advantages

- Simplicity and straightforward implementation.
- Fast development and deployment.

Disadvantages

- Limited scalability.
- Security risks due to direct database access.

3-Tier Architecture

1. **Three Layers**
    - **Presentation Layer**: User interface.

   - **Application Layer**: Business logic and data processing.
   - **Database Layer**: Data storage and management.
2. **Components**
   - **Client**: User interface.
   - **Application Server**: Business logic.
   - **Database Server**: Data management.

Advantages

- Enhanced scalability and security.
- Better separation of concerns, making maintenance easier.

Disadvantages

- More complex to implement.
- Requires more resources.

Conclusion 2-tier architecture is simpler and quicker to implement but lacks scalability and security. 3-tier architecture, though more complex, offers greater flexibility, scalability, and security for enterprise applications.

Answers (10 Marks Questions)

Q31: Explain the Purpose of the Database System. Explain Different Database Users. What Are the Responsibilities of a DBA?

Purpose of Database System A database system efficiently stores, manages, and retrieves large volumes of data while ensuring integrity, security, and consistency. It facilitates data sharing and enhances accessibility for multiple users and applications.

Different Database Users

1. **End Users**: Regular users interacting with the database through applications.
2. **Application Programmers**: Developers creating applications that interact with the database.
3. **Database Administrators (DBA)**: Specialists responsible for database maintenance, security, and performance.

4. **System Analysts**: Professionals designing database schemas and analyzing data requirements.

Responsibilities of DBA DBAs are responsible for database design, performance tuning, security management, backup and recovery, and providing user support.

Q32: Define Database Management System. What is the Role of Database Administrator?

Database Management System (DBMS) A DBMS is software that facilitates the creation, management, and manipulation of databases. It provides an interface for users and applications to interact with the database efficiently and securely.

Role of Database Administrator (DBA) DBAs are responsible for database schema design, performance monitoring, security implementation, backup and recovery, data integrity maintenance, and user support.

Q33: Draw and Explain System Structure of Database Management System

System Structure of DBMS A DBMS typically consists of three main layers: internal, conceptual, and external. The internal layer manages physical storage, the conceptual layer defines the logical structure, and the external layer provides user-specific views.

Components of DBMS Architecture

- **Internal Layer**: Manages physical storage.
- **Conceptual Layer**: Defines logical structure.
- **External Layer**: Provides user views.

34. Disadvantages of File-Processing Systems

Data Redundancy and Inconsistency

Data redundancy and inconsistency are common due to decentralized data storage, leading to synchronization challenges and accuracy issues.

Data Isolation

Scattered data storage makes accessing related data difficult, causing inefficiencies and complexity in data retrieval.

Integrity Problems

File-processing systems lack centralized control, making it challenging to enforce data integrity constraints, increasing the risk of corruption and inconsistency.

Concurrent Access Issues

Managing concurrent access by multiple users is problematic without proper concurrency control, risking data corruption and reliability.

Security Problems

Basic security measures are inadequate, leaving sensitive data vulnerable to unauthorized access, manipulation, and theft.

Limited Data Sharing

Decentralized data storage restricts data sharing, requiring manual processes and increasing the likelihood of errors and inefficiencies.

In summary, file-processing systems suffer from redundancy, isolation, integrity, concurrency, security, and sharing limitations, underscoring the need for more advanced data management solutions.

Q35: Define and Explain the Following Terms: Levels of Data Abstraction, Instances, Schema, Physical Data Independence, Logical Data Independence

Levels of Data Abstraction: Physical, logical, and view levels hide complexities and provide different perspectives of the database.

Instances: Actual content of the database at a particular moment.

Schema: Structure of the database, including tables, columns, and relationships.

Physical Data Independence: Ability to change physical schema without affecting logical schema.

Logical Data Independence: Ability to change logical schema without impacting external schema.

Q36: What is DBMS? List and Explain Various Database Users

Database Management System (DBMS) A DBMS is software that facilitates database creation, management, and manipulation, providing an interface for users and applications to interact with the database efficiently and securely.

Various Database Users

1. **End Users**: Regular users interacting with applications.
2. **Application Programmers**: Developers creating applications.
3. **Database Administrators (DBA)**: Specialists maintaining databases.
4. **System Analysts**: Professionals designing database schemas.
5. **Data Scientists and Analysts**: Experts analyzing database data.

UNIT 2

Short Answers (2 Marks Questions)

Q1: Define (i) Entity (ii) Attribute

Entity

- Represents a real-world object or concept, distinguishable from other objects.
- Example: Student, Employee.

Attribute

- Describes properties or characteristics of an entity.
- Example: Name, Age.

Q2: Define Relationship and Relationship Set

Relationship

- Describes an association between entities.
- Example: Works_For relationship between Employee and Department.

Relationship Set

- Collection of similar relationships.
- Example: Set of all Works_For relationships.

Q3: Explain about Querying Relational Data

Querying relational data involves retrieving specific information from a relational database using SQL commands like SELECT, INSERT, UPDATE, and DELETE. Queries specify criteria to filter and manipulate data, allowing users to extract meaningful insights.

Q4: Explain Different Types of Entities

Entities can be classified into:

1. **Strong Entities**: Exist independently and have a primary key.
2. **Weak Entities**: Depend on another entity for existence and have a partial key.
3. **Associative Entities**: Represent relationships between other entities and typically have attributes of their own.

Q5: Explain Different Types of Attributes

Attributes can be categorized as:

1. **Simple Attributes**: Cannot be divided further.
2. **Composite Attributes**: Composed of multiple simple attributes.
3. **Single-Valued Attributes**: Hold a single value.
4. **Multi-Valued Attributes**: Can hold multiple values.

Q6: What are the Basic Constructs of ER Model?

Basic constructs of the ER model include:

1. **Entities**: Represent real-world objects.
2. **Attributes**: Describe properties of entities.
3. **Relationships**: Describe associations between entities.

Q7: Write a Short Note on Entity-Relationship Model

The Entity-Relationship (ER) model is a conceptual data model used to represent the structure of databases in terms of entities, attributes, and relationships. It provides a graphical representation of the database schema, facilitating communication between stakeholders during the design phase.

Q8: Define Weak Entity Set

A weak entity set is an entity set that does not have sufficient attributes to form a primary key. It relies on a related entity set (strong entity set) for its existence and is identified by a partial key.

Q9: Explain Referential Integrity

Referential integrity ensures the consistency and accuracy of data relationships in a database. It requires that every foreign key value must match a primary key value in another relation or be null.

Q10: Define Entity, Entity Set, and Extensions of Entity Set

- **Entity**: A distinct object or concept in the real world.
- **Entity Set**: A collection of similar entities.
- **Extensions of Entity Set**: The current instances of an entity set at a particular time.

Q11: Define and Give Examples to Illustrate the Four Types of Attributes in Database

- **Simple Attributes**: Cannot be divided further (e.g., Age).
- **Composite Attributes**: Composed of multiple simple attributes (e.g., Address).
- **Single-Valued Attributes**: Hold a single value (e.g., DateOfBirth).
- **Multi-Valued Attributes**: Can hold multiple values (e.g., PhoneNumbers).

Q12: Define Relationship and Participation

- **Relationship**: Describes an association between entities.
- **Participation**: Indicates whether each entity in an entity set must participate in a relationship.

Q13: Define Mapping Cardinality or Cardinality Ratio

Mapping cardinality specifies the number of entities to which another entity can be associated through a relationship. It describes the relationship between entity sets in terms of one-to-one, one-to-many, or many-to-many mappings.

Q14: Define Single-Valued and Multi-Valued Attributes

- **Single-Valued Attributes**: Hold only one value per entity instance (e.g., Height).
- **Multi-Valued Attributes**: Can hold multiple values for each entity instance (e.g., Skills).

Q15: Define the Terms i) Key Attribute ii) Value Set

- **Key Attribute**: An attribute that uniquely identifies an entity within an entity set.
- **Value Set**: The range of values that an attribute can take.

Q16: Explain Candidate Key, Primary Key, and Foreign Key

- **Candidate Key**: A minimal set of attributes that uniquely identifies each tuple in a relation.
- **Primary Key**: A candidate key chosen as the main identifier for tuples in a relation.
- **Foreign Key**: An attribute or set of attributes in one relation that refers to the primary key in another relation.

Q17: Define Database State or Snapshot

A database state or snapshot refers to the contents of a database at a specific moment in time, including all data and schema information.

Q18: How are Complex Attributes Represented?

Complex attributes are represented using composite attributes, which are composed of multiple simple attributes.

Q19: Define Value Sets of Attributes

Value sets of attributes define the range of allowable values for each attribute. They specify the domain from which attribute values must be chosen.

Q20: Define Degree of Relationship Type

The degree of a relationship type indicates the number of entity sets that participate in the relationship. It can be unary (one entity set), binary (two entity sets), ternary (three entity sets), etc.

Intermediate Questions (5 Marks Each)

Q21: Data Models and E-R Model

Data Models: Data models are frameworks used to organize and structure data. They provide a way to represent the structure of a database.

E-R Model: The Entity-Relationship (E-R) model is a graphical representation used to describe the data and its relationships in a database. It uses entities to represent real-world objects and relationships to represent connections between them.

Q22: Mapping Cardinalities

Mapping Cardinalities: Mapping cardinalities define the relationship between entities in a database. They specify how many instances of one entity are associated with instances of another entity.

Examples:

- One-to-One: Each instance of one entity is associated with only one instance of another entity.
- One-to-Many: Each instance of one entity is associated with multiple instances of another entity.
- Many-to-One: Multiple instances of one entity are associated with one instance of another entity.
- Many-to-Many: Multiple instances of one entity are associated with multiple instances of another entity.

Q23: Specialization in ER Diagram

Specialization: Specialization is the process of creating subclasses based on a superclass in an Entity-Relationship (ER) diagram. It allows for the creation of more specific entity types that inherit attributes and relationships from a broader category.

Example: Consider the superclass "Vehicle" being specialized into subclasses "Car" and "Bike". Each subclass inherits attributes such as "Color" and "Manufacturer" from the superclass but may have additional attributes specific to its type.

Q24: Generalization in E-R Diagram

Generalization: Generalization is the reverse process of specialization, where a superclass is created based on common attributes and relationships of its subclasses. It represents a higher-level view that encompasses multiple entity types.

Illustration: Building on the previous example, if "Car" and "Bike" share common attributes like "Wheels" and "Engine Type", they can be generalized into a superclass "Vehicle".

Q25: Aggregation Operation in ER Diagram

Aggregation Operation: Aggregation is a relationship type in an ER diagram where one entity can be composed of multiple entities. It represents a "whole-part" relationship.

Application Example: In a hospital database, the "Department" entity may aggregate multiple "Doctor" entities, where each doctor is part of a department.

Difficult Questions (10 Marks Each)

Q31: Specialization and Generalization in ER Diagram

Specialization: Specialization involves creating subclasses based on a superclass, allowing for more specific entity types.

Generalization: Generalization is the process of creating a superclass based on common attributes and relationships of its subclasses, representing a higher-level view.

Q32: Functional Dependency in Database Design

Functional Dependency: Functional dependency occurs when the value of one attribute uniquely determines the value of another attribute in a relation. It plays a crucial role in database normalization and design.

Types:

- Full Functional Dependency: A functionally dependent attribute is fully determined by another attribute.
- Partial Functional Dependency: A functionally dependent attribute is only partially determined by another attribute.
- Transitive Dependency: Dependency exists between non-key attributes via other non-key attributes.

Q33: Detailed Explanation of Entity Set and Attributes

Entity Set: An entity set is a collection of similar entities in a database, representing real-world objects or concepts.

Attributes Types:

- **Simple Attribute**: Represents indivisible data components.
- **Composite Attribute**: Composed of multiple simple attributes.
- **Single-Valued Attribute**: Holds a single value for each entity.
- **Multi-Valued Attribute**: Can hold multiple values for each entity.
- **Null Attribute**: Represents missing or unknown data.
- **Derived Attribute**: Derived from other attributes, calculated at runtime.

Solution for Q34: E-R Diagram for a Hospital

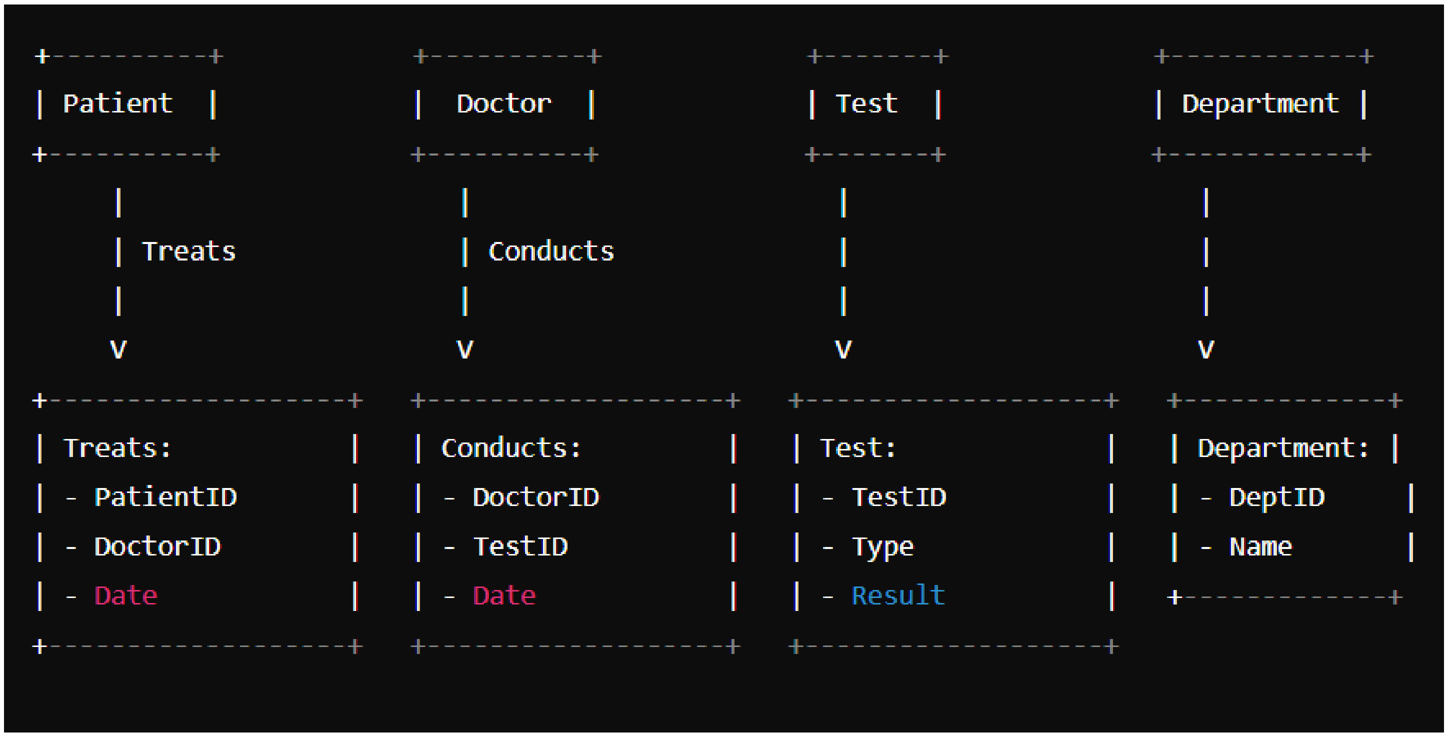
E-R Diagram Overview: The E-R diagram for a hospital depicts entities like "Patient," "Doctor," "Test," and "Department," with relationships like "Treats" and "Conducts."

Entities and Relationships:

- **Patient**: Information about admitted patients.
- **Doctor**: Medical staff responsible for treatments.
- **Test**: Details of medical tests conducted.
- **Department**: Various hospital departments.

Relationships:

- **Treats**: Treatment provided by doctors to patients.
- **Conducts**: Tests conducted by doctors on patients.



```
+----------+         +----------+        +-------+            +------------+
| Patient  |         |  Doctor  |        | Test  |            | Department |
+----------+         +----------+        +-------+            +------------+
     |                    |                   |                    |
     | Treats             | Conducts          |                    |
     |                    |                   |                    |
     v                    v                   v                    v
+------------------+ +------------------+ +------------------+ +------------+
| Treats:          | | Conducts:        | | Test:            | | Department: |
| - PatientID      | | - DoctorID       | | - TestID         | | - DeptID     |
| - DoctorID       | | - TestID         | | - Type           | | - Name       |
| - Date           | | - Date           | | - Result         | +------------+
+------------------+ +------------------+ +------------------+
```

Solution for Q35: E-R Diagram for a Library Management System

E-R Diagram Overview: The E-R diagram includes entities like "Book," "Author," "Member," and "Loan," with relationships like "WrittenBy," "Borrow," and "Return."

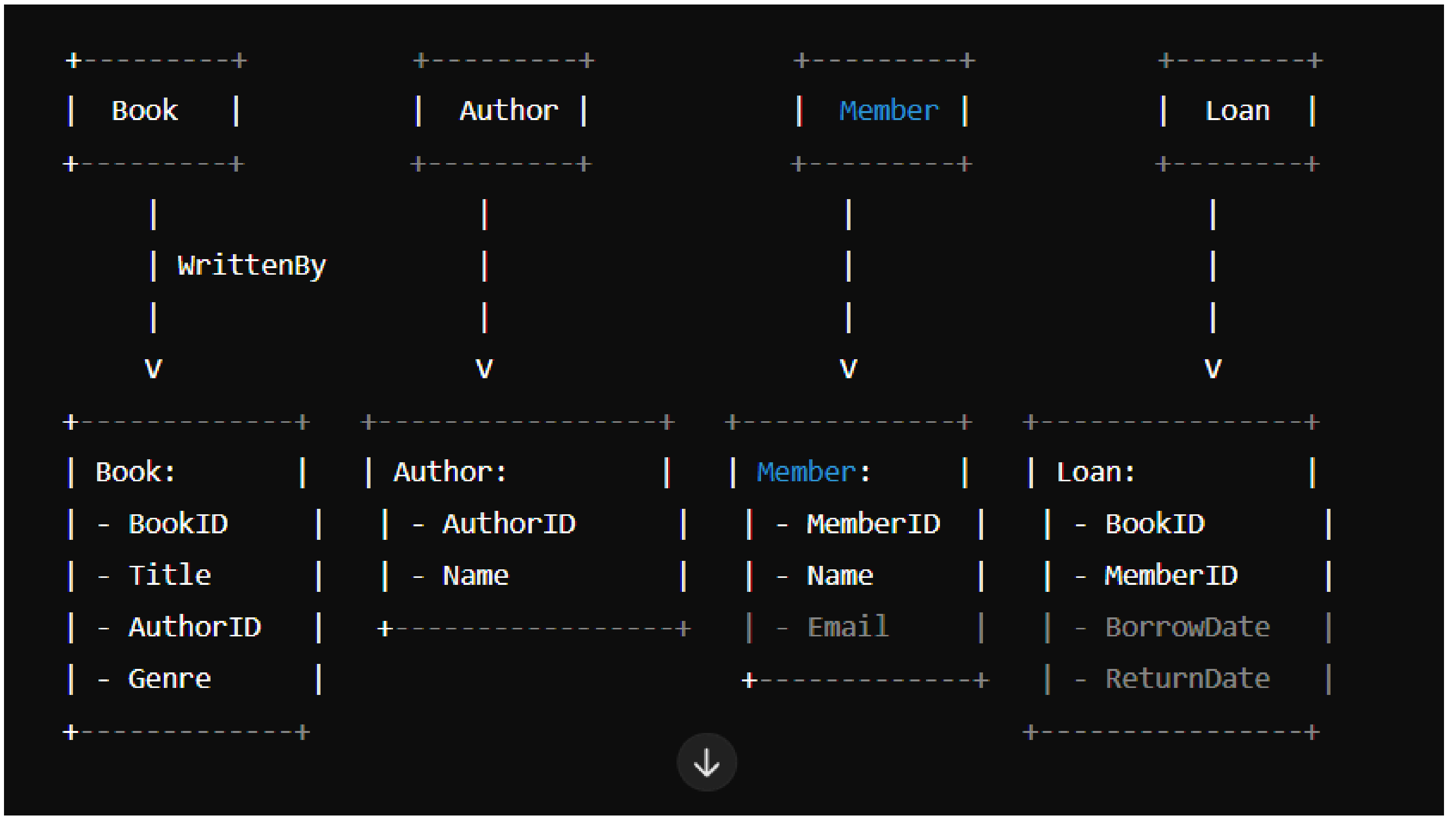
Entities and Relationships:

- **Book**: Information about library books.
- **Author**: Writers of library books.

- **Member**: Individuals who borrow books.
- **Loan**: Records of borrowed books.

Relationships:

- **WrittenBy**: Authors associated with books.
- **Borrow**: Books borrowed by members.
- **Return**: Returned books by members.



```
+---------+        +---------+          +---------+         +--------+
|  Book   |        |  Author |          |  Member |         |  Loan  |
+---------+        +---------+          +---------+         +--------+
     |                  |                    |                   |
     | WrittenBy        |                    |                   |
     |                  |                    |                   |
     V                  V                    V                   V
+------------+ +----------------+ +------------+ +---------------+
| Book:      | | Author:        | | Member:    | | Loan:         |
| - BookID     |  | - AuthorID       |  | - MemberID  |  | - BookID        |
| - Title      |  | - Name           |  | - Name      |  | - MemberID      |
| - AuthorID   |  +-----------------+  | - Email     |  | - BorrowDate    |
| - Genre      |                       +-------------+  | - ReturnDate    |
+------------+                                        +---------------+
```

Solution for Q36: E-R Diagram for a School Management System

E-R Diagram Overview: The E-R diagram depicts entities like "Student," "Teacher," "Class," and "Subject," with relationships like "Enroll," "Teaches," and "Attend."

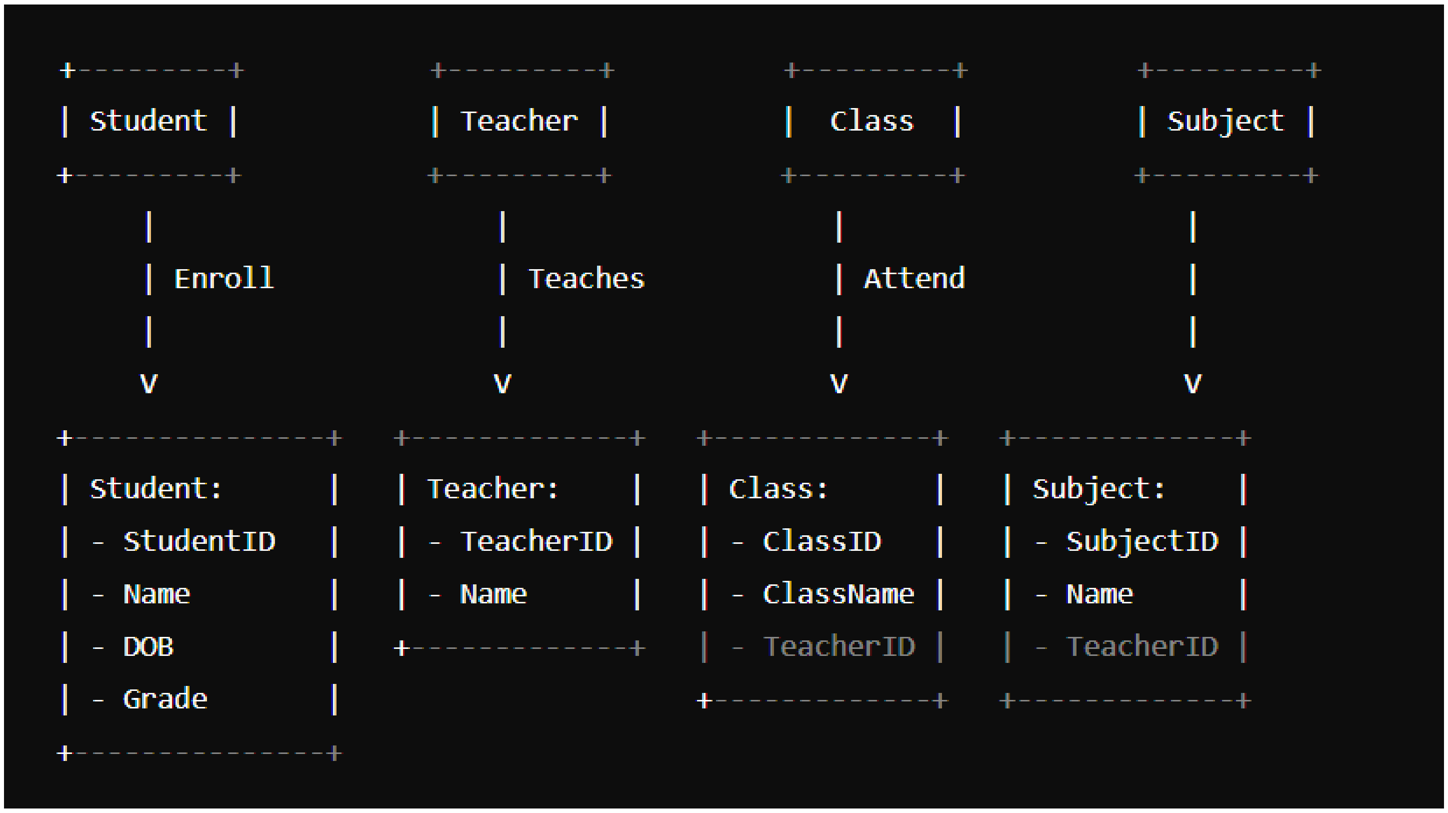
Entities and Relationships:

- **Student**: Enrolled students' information.
- **Teacher**: Teaching staff details.

- **Class**: Class information.
- **Subject**: Subjects taught in classes.

Relationships:

- **Enroll**: Students enrolled in classes.
- **Teaches**: Subjects taught by teachers.
- **Attend**: Student attendance in classes.



```
+---------+          +---------+          +---------+          +---------+
| Student |          | Teacher |          |  Class  |          | Subject |
+---------+          +---------+          +---------+          +---------+
     |                    |                    |                    |
     | Enroll             | Teaches            | Attend             |
     |                    |                    |                    |
     v                    v                    v                    v
+---------------+  +-------------+  +-------------+  +-------------+
| Student:      |  | Teacher:    |  | Class:      |  | Subject:    |
| - StudentID   |  | - TeacherID |  | - ClassID   |  | - SubjectID |
| - Name        |  | - Name      |  | - ClassName |  | - Name      |
| - DOB         |  +-------------+  | - TeacherID |  | - TeacherID |
| - Grade       |                   +-------------+  +-------------+
+---------------+
```

UNIT 3

Level A. Easy Questions (2 marks each)

Q1: State about SELECT operation in Relational algebra?

CO1

The SELECT operation in relational algebra retrieves rows from a relation (table) that satisfy a given condition. It acts as a filter, selecting rows based on specified criteria.

Q2: State about PROJECT operation in Relational algebra?

CO1

The PROJECT operation in relational algebra selects specific columns (attributes) from a relation (table) while discarding the others. It removes duplicate rows and retains only unique combinations of values.

Q3: Define Aggregate Functions?

CO1

Aggregate functions in SQL perform calculations on a set of values and return a single value as output. Common aggregate functions include SUM, AVG, COUNT, MIN, and MAX.

Q4: Discuss the basic form of SQL query?

CO1

The basic form of an SQL query consists of the SELECT statement followed by the columns to retrieve from a table, the FROM clause specifying the table to query, and optional clauses such as WHERE for conditions, GROUP BY for grouping, HAVING for filtering groups, and ORDER BY for sorting results.

Q5: Define Null Values

CO1

Null values in SQL represent missing or unknown data. They indicate the absence of a value in a column and can be assigned to columns that allow nulls.

Q6: List the aggregate functions supported by SQL?

CO1

SQL supports several aggregate functions, including:

- SUM
- AVG
- COUNT
- MIN

- MAX

Q7: List the table modification commands in SQL?

CO2

Table modification commands in SQL include:

- INSERT: Adds new rows to a table.
- UPDATE: Modifies existing rows in a table.
- DELETE: Removes rows from a table.

Q8: What is the use of group by clause?

CO1

The GROUP BY clause in SQL is used to group rows that have the same values into summary rows, typically for use with aggregate functions like SUM or AVG.

Q9: Discuss about the operators renaming, joins, division?

CO1

Renaming operators in SQL allow renaming of attributes or relations using the AS keyword. Joins combine rows from two or more tables based on a related column between them. Division is an operation that retrieves rows from one table that are associated with all rows from another table.

Q10: What is domain integrity? Give example

CO2

Domain integrity ensures that all data entered into a database meets specified domain constraints, such as data type, range, and format. For example, a domain constraint on a "Age" column may enforce that the age values must be between 18 and 100.

Q11: Define Joins?

CO1

Joins in SQL are used to combine rows from two or more tables based on a related column between them. There are different types of joins, including INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.

Q12: What is Like patterns in Sql?

CO3

The LIKE pattern in SQL is used to search for a specified pattern in a column. It allows the use of wildcards such as "%" (matches zero or more characters) and "_" (matches a single character).

Q13: Perform ascending and descending order in table?

CO1

To perform ascending order in SQL, use the ORDER BY clause followed by the column name. For descending order, add the keyword DESC after the column name.

Q14: Discuss about trigger?

CO1

A trigger in SQL is a set of SQL statements that automatically "fires" (executes) when a specified event occurs in the database. Triggers can be used to enforce business rules, maintain data integrity, or perform logging actions.

Q15: Demonstrate how to add a NOT NULL column to a table?

CO1

To add a NOT NULL column to a table, use the ALTER TABLE statement followed by the ADD COLUMN clause, specifying the column name and data type, and adding the NOT NULL constraint.

Q16: Discuss different types of Identity functions in SQL?

CO1

In SQL, identity columns are used to generate unique values automatically for a specified column. Common identity functions include IDENTITY, AUTO_INCREMENT, and SERIAL, depending on the database system used.

Q17: Difference between Having and Where Clause?

CO1

The WHERE clause in SQL filters rows before data is grouped or aggregated, while the HAVING clause filters grouped rows after data is grouped or aggregated.

Q18: Implementation of foreign key on database.

CO1

Foreign keys in SQL establish a relationship between two tables by referencing the primary key of one table as a foreign key in another. They ensure referential integrity, maintaining consistency between related tables.

Q19: What is the use of return and output parameters in stored procedure?

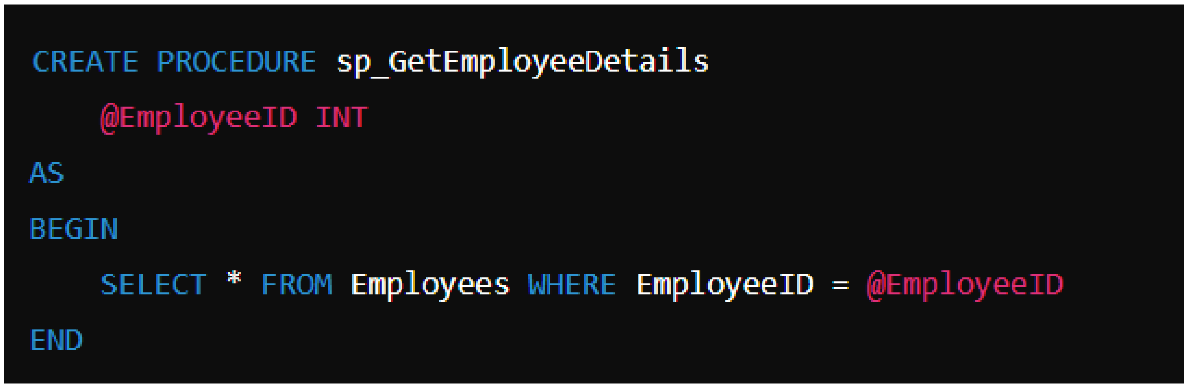
CO1

Return and output parameters in stored procedures allow passing values back to the calling program or client application. Return parameters are used to return a single value, while output parameters can return multiple values.

Q20: Create Stored procedure with Input Parameters?

CO2

Stored procedures in SQL are reusable sets of SQL statements stored in the database. They can accept input parameters for dynamic behavior. Below is an example of creating a stored procedure with input parameters:



```sql
CREATE PROCEDURE sp_GetEmployeeDetails
    @EmployeeID INT
AS
BEGIN
    SELECT * FROM Employees WHERE EmployeeID = @EmployeeID
END
```

Level B. Intermediate Questions (5 marks each)

Q21: Operations in Relational Algebra

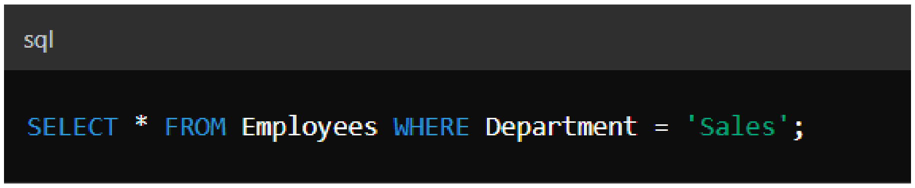
CO2

Overview: Relational algebra comprises fundamental operations to manipulate relational data.

Operations:

1. **SELECT:** Retrieves rows from a relation based on specified conditions.
2. **PROJECT:** Selects specific columns from a relation while discarding others.
3. **UNION:** Combines rows from two relations, removing duplicates.
4. **SET DIFFERENCE:** Retrieves rows from one relation not found in another.
5. **CARTESIAN PRODUCT:** Generates a combination of all rows from two relations.
6. **JOIN:** Combines rows from two or more relations based on related columns.

Example:



sql

```sql
SELECT * FROM Employees WHERE Department = 'Sales';
```

Q22: Joins in SQL

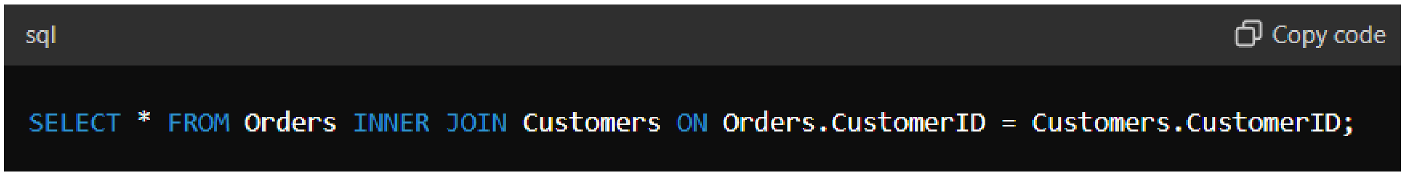
CO1

Definition: A join in SQL combines rows from two or more tables based on related columns.

Types of Joins:

1. **INNER JOIN:** Returns rows when there is at least one match in both tables.
2. **LEFT JOIN:** Returns all rows from the left table and matched rows from the right table.
3. **RIGHT JOIN:** Returns all rows from the right table and matched rows from the left table.
4. **FULL JOIN:** Returns rows when there is a match in one of the tables.

Explanation:



sql Copy code

```sql
SELECT * FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Q23: Trigger in SQL

CO1

Definition: A trigger in SQL is a set of SQL statements that automatically executes in response to specified events.

Parts of a Trigger:

1. **Triggering Event:** Specifies the event that triggers the execution.
2. **Trigger Condition:** Determines whether the trigger should be executed.
3. **Trigger Action:** Specifies the SQL statements to execute when the trigger fires.

Differentiation:

- **Row-Level Triggers:** Execute once for each affected row.
- **Statement-Level Triggers:** Execute once for each triggering event.

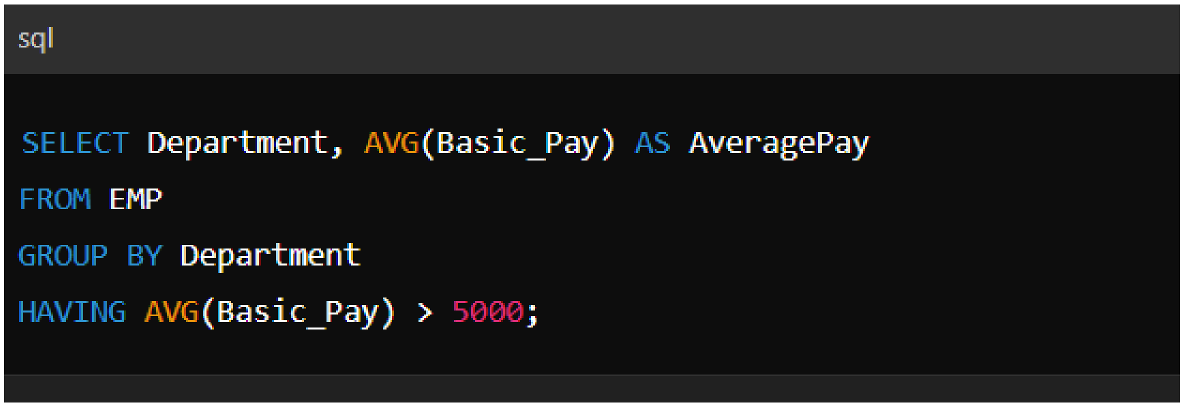
Q24: Group By and Having Clauses

CO1

Explanation:

- **GROUP BY Clause:** Groups rows sharing common values into summary rows.
- **HAVING Clause:** Filters groups based on specified conditions.

Example:



```sql
SELECT Department, AVG(Basic_Pay) AS AveragePay
FROM EMP
GROUP BY Department
HAVING AVG(Basic_Pay) > 5000;
```

Q25: Complex Integrity Constraints

CO1

Definition: Complex integrity constraints in SQL enforce intricate data consistency requirements across multiple rows or tables.

Examples:

- **Multi-Table Foreign Key Constraints**
- **CHECK Constraints involving Subqueries**

Q26: Null Values in SQL

CO2

Definition: Null values in SQL represent missing or unknown data.

Effect:

- Nulls can affect calculations involving arithmetic operations, comparisons, and aggregations.
- They signify the absence of a value in a column.

Q27: Aggregate Operators in SQL

CO2

Overview: Aggregate operators in SQL perform calculations on a set of values and return a single value.

Types:

1. **SUM:** Calculates the sum of values in a column.
2. **AVG:** Calculates the average of values in a column.
3. **COUNT:** Counts the number of rows or non-null values in a column.
4. **MIN:** Returns the minimum value in a column.
5. **MAX:** Returns the maximum value in a column.

Q28: Average Basic Pay by Department

CO2

sql

```sql
SELECT Department, AVG(Basic_Pay) AS AveragePay
FROM EMP
GROUP BY Department
HAVING AVG(Basic_Pay) > 5000;
```

This SQL query retrieves the average basic pay for each department from the EMP table. Here's a breakdown of each part of the query:

- **SELECT Department, AVG(Basic_Pay) AS AveragePay:** Specifies the columns to be selected in the result set. It calculates the average basic pay for each department and aliases it as AveragePay.
- **FROM EMP:** Specifies the table from which data is retrieved. In this case, it's the EMP table.
- **GROUP BY Department:** Groups the rows of the result set by the Department column. This means that the average basic pay will be calculated separately for each department.
- **HAVING AVG(Basic_Pay) > 5000:** Filters the groups created by the GROUP BY clause. It selects only those groups where the average basic pay (AVG(Basic_Pay)) is greater than 5000.

Q29: DDL and DML

CO1

Overview:

- **DDL (Data Definition Language):** Defines and manages database structures.
- **DML (Data Manipulation Language):** Manipulates data within the database.

Q30: Delete, Drop, and Truncate

CO1

Explanation:

- **DELETE:** Removes rows from a table based on specified conditions.
- **DROP:** Removes objects like tables or indexes entirely from the database.
- **TRUNCATE:** Removes all rows from a table while preserving the table structure.

LONG ANSWERS

Q33: Procedure Creation for Employee Table

CO2

To create a procedure for the Employee table and return a string value:

```
CREATE PROCEDURE GetEmployeeString
AS
BEGIN
    SELECT empname + ', ' + street + ', ' + city AS EmployeeDetails
    FROM Employee;
END;
```

This simple procedure selects and concatenates the empname, street, and city columns from the Employee table to form a string representing each employee's details.

Q35: DDL and DML

CO2

DDL (Data Definition Language):

DDL statements are used to define and manage the structure of database objects such as tables, indexes, and constraints. These statements include:

1. **CREATE:** Used to create new database objects like tables, indexes, or views.
2. **ALTER:** Modifies the structure of existing database objects.
3. **DROP:** Deletes database objects like tables, indexes, or views.
4. **TRUNCATE:** Removes all rows from a table, but keeps the table structure intact.
5. **COMMENT:** Adds comments to database objects to provide descriptive information.

DML (Data Manipulation Language):

DML statements are used to manipulate data stored in the database. These statements include:

1. **INSERT:** Adds new rows of data into a table.
2. **UPDATE:** Modifies existing data in a table based on specified conditions.

3. **DELETE:** Removes one or more rows from a table based on specified conditions.
4. **MERGE:** Combines insert, update, and delete operations into a single statement based on specified conditions.
5. **SELECT:** Retrieves data from one or more tables based on specified criteria.

In summary, DDL is used to define and modify the structure of database objects, while DML is used to manipulate the data within those objects.

Q36: Difference Between Delete, Drop, and Truncate

CO

Delete:

- DELETE is a DML (Data Manipulation Language) statement used to remove one or more rows from a table based on specified conditions.
- It maintains the table structure and any associated triggers.
- The operation can be rolled back using a transaction rollback.
- Example: DELETE FROM Employees WHERE Salary < 50000;

Drop:

- DROP is a DDL (Data Definition Language) statement used to remove database objects like tables, indexes, or views entirely from the database.
- It permanently deletes the object along with its data and structure.
- The operation cannot be rolled back.
- Example: DROP TABLE Employees;

Truncate:

- TRUNCATE is a DDL statement used to remove all rows from a table.
- It resets identity columns, deallocates storage space, and cannot be rolled back.
- TRUNCATE is faster than DELETE as it doesn't log individual row deletions.
- Example: TRUNCATE TABLE Employees;

ADITYA DHIMAN

UNIT 4

Level A. Easy Questions (2 marks each)

Q1: Define Redundancy

CO1 Redundancy refers to the repetition of data in a database system, where the same piece of information is stored multiple times. It can lead to inconsistencies, wasted storage space, and update anomalies.

Q2: Define Functional Dependency

CO1 Functional dependency is a constraint between two sets of attributes in a relation from a database. It implies that the values of one set of attributes determine the values of another set. Some functional dependencies are trivial because they hold for all possible relations.

Q3: Discuss Normalization

CO1 Normalization is the process of organizing data in a database efficiently. It involves decomposing a table into smaller tables and defining relationships between them. The main aim of normalization is to minimize redundancy and dependency.

Q4: Illustrate Functional Dependency with Example

CO1 In a relation R, attribute B is functionally dependent on attribute A if for every value of A, there is exactly one corresponding value of B. For example, in a table of employees, the employee ID uniquely determines the employee's name.

Q5: Illustrate Fully Functional Dependency with Example

CO1 A fully functional dependency occurs when an attribute is functionally dependent on all the attributes of a composite primary key, not just part of it. For example, in a table of orders where (Order_ID, Product_ID) is the primary key, the Product_Name is fully functionally dependent on both Order_ID and Product_ID.

Q6: Demonstrate Transitive Dependency

CO1 Transitive dependency occurs when an attribute is functionally dependent on another through a third attribute. For example, in a table where A determines B and B determines C, but A does not directly determine C, there's a transitive dependency. In a table of employees, if Department determines Manager, and Manager determines Salary, then Department indirectly determines Salary.

Q7: Discuss Domain-Key Normal Form

CO2 Domain-Key Normal Form (DKNF) is a normal form used in database normalization. It ensures that all constraints on the relation are expressed by way of key and domain constraints. It guarantees that all facts in the table are directly about the attributes of the key.

Q8: Define Armstrong Axioms for FD's

CO1 Armstrong's axioms are a set of inference rules used to infer all the functional dependencies on a database schema. The axioms include reflexivity, augmentation, transitivity, and decomposition.

Q9: Define First Normal Form

CO1 First Normal Form (1NF) is a database normalization form that requires all entries in a table to be atomic. It means that each column should contain only a single value, and each row should be unique.

Q10: Define Second Normal Form

CO2 Second Normal Form (2NF) is a database normalization form that requires a table to be in 1NF and ensures that all non-key attributes are fully functionally dependent on the entire primary key, not on just a part of it.

Q11: Define Third Normal Form

CO1 Third Normal Form (3NF) is a database normalization form that requires a table to be in 2NF and ensures that there are no transitive dependencies. It means that all non-key attributes are dependent only on the primary key, not on other non-key attributes.

Q12: Define Fourth Normal Form

CO3 Fourth Normal Form (4NF) is a database normalization form that extends the concept of 3NF and further eliminates multi-valued dependencies. It ensures that no non-trivial multi-valued dependencies exist in the table.

Q13: List out the Problems Related to Decompositions

CO1 Decomposition in database design can lead to problems such as loss of information, insertion anomalies, deletion anomalies, and update anomalies. These problems occur when data is divided into multiple tables without considering dependencies between attributes.

Q14: Explain about Lossless-Join Dependency

CO1 Lossless-Join Dependency ensures that the decomposition of a relation into smaller relations preserves the ability to join the smaller relations to reconstruct the original relation without losing any information.

Q15: Explain about BCNF

CO1 Boyce-Codd Normal Form (BCNF) is a database normalization form that ensures that there are no non-trivial functional dependencies of attributes on anything other than a superkey. It eliminates all redundant dependencies between attributes.

Q16: Explain about Multi-valued Dependencies

CO1 Multi-valued dependencies occur when a set of attributes functionally determines another set of attributes, and the functional dependency cannot be satisfied by decomposing the relation into separate relations. It is a form of dependency where a key determines multiple values, which are independent of each other.

Q17: Define Join Dependency and Fifth Normal Form

CO1 Join Dependency occurs when a relation can be reconstructed by joining multiple smaller relations. Fifth Normal Form (5NF) is a normalization form that ensures that all join dependencies are represented explicitly in the database schema.

Q18: Explain the Concept Scheme Refinement in Database Design

CO1 Scheme refinement in database design involves refining the conceptual schema into internal and external schemas. It ensures that the database design is flexible, efficient, and meets the requirements of various users and applications.

Q19: Define Dependency Preserving Decomposition

CO1 Dependency Preserving Decomposition is a database normalization technique that ensures that all functional dependencies in the original relation are preserved in the decomposed relations. It avoids losing any dependency information during the decomposition process.

Q20: Explain about Inclusion Dependency

CO2 Inclusion Dependency is a type of constraint that specifies that the values appearing in a set of attributes (foreign key) must also appear in another set of attributes (primary key) in another relation.

Level B. Intermediate Questions (5 marks each)

Q21: Illustrate Redundancy and Its Problems

CO2 Redundancy refers to the repetition of data in a database, which can lead to various problems such as:

- Wastage of storage space.
- Update anomalies where inconsistencies occur due to redundant data.
- Insertion anomalies when adding new data.
- Deletion anomalies when removing data.

Q22: Define Decomposition and **Addressing Redundancy**

CO1 Decomposition is the process of breaking down a relation into smaller, more manageable relations. It addresses redundancy by eliminating duplicate data and reducing the likelihood of anomalies. However, decomposition can also introduce problems such as:

- Loss of information.
- Insertion, deletion, and update anomalies if dependencies are not carefully considered.

Q23: Define Functional Dependencies and Relationship with Primary Keys

CO1 Functional dependencies are constraints between two sets of attributes in a relation. They imply that the values of one set determine the values of another set. Primary keys play a crucial role in functional dependencies as they uniquely identify each tuple in a relation and are often used to enforce key constraints.

Q24: Define Normalization and Explain 1NF, 2NF, 3NF Normal Forms

CO1 Normalization is the process of organizing data in a database to reduce redundancy and dependency.

- **1NF (First Normal Form):** Requires that each attribute in a table has atomic values, and there are no repeating groups.
- **2NF (Second Normal Form):** Requires the removal of partial dependencies, ensuring that non-key attributes are fully functionally dependent on the primary key.
- **3NF (Third Normal Form):** Requires the removal of transitive dependencies, ensuring that non-key attributes are not dependent on other non-key attributes.

Q25: Compare and Contrast BCNF with 3NF

CO1

- Both BCNF (Boyce-Codd Normal Form) and 3NF (Third Normal Form) are normalization forms aimed at reducing redundancy and dependency.
- BCNF is stricter than 3NF and eliminates all non-trivial functional dependencies on candidate keys, while 3NF allows transitive dependencies.
- BCNF may result in more relations than 3NF, but it ensures that all functional dependencies are directly on the primary key.

Q26: Properties of Decompositions

CO2 Decompositions in database design have several properties:

- **Lossless Join:** The decomposed relations can be joined to reconstruct the original relation without loss of information.
- **Dependency Preservation:** The functional dependencies present in the original relation are preserved in the decomposed relations.
- **Minimality:** The decomposed relations are minimal, meaning that they cannot be further decomposed without losing the above properties.

Q27: Schema Refinement in Database Design

CO2 Schema refinement involves refining the conceptual schema into internal and external schemas to meet the requirements of various users and applications. It ensures that the database design is flexible, efficient, and scalable. Refinement also involves optimizing the database schema for performance and ensuring data integrity and security.

Q28: Illustrate Multivalued Dependencies and Fourth Normal Form

CO2 Multivalued dependencies occur when a set of attributes functionally determines another set of attributes, and the dependency cannot be expressed by decomposing the relation into separate relations. Fourth Normal Form (4NF) is a normalization form that eliminates multivalued dependencies, ensuring that each non-key attribute is fully functionally dependent on the primary key.

Q29: Discuss Join Dependencies and Fifth Normal Form

CO1 Join dependencies occur when a relation can be reconstructed by joining multiple smaller relations. Fifth Normal Form (5NF) is a normalization form that ensures that all join dependencies are represented explicitly in the database schema. It guarantees that the database schema is free from redundancy and anomalies related to join operations.

Q30: Illustrate Inclusion Dependencies with Example

CO1 Inclusion dependencies specify that the values appearing in one set of attributes (foreign key) must also appear in another set of attributes (primary key) in another relation. For example, consider two relations: Employees and Departments. An inclusion dependency would ensure that the values in the Department_ID column of the Employees relation must exist in the Department_ID column of the Departments relation.

Level C. Difficult Questions (10 marks each)

Q32: Proving Relation in 4NF is in BCNF

CO3

To prove that a relation in 4NF must be in BCNF, we need to understand the properties of both normal forms and their implications. In 4NF, a relation is free from multivalued dependencies, while in BCNF, it eliminates all non-trivial functional dependencies on candidate keys.

Given a relation in 4NF, it means that it is free from multivalued dependencies and satisfies the properties of 4NF. Since 4NF implies that the relation has no non-trivial multivalued dependencies, all the remaining dependencies are functional dependencies.

Now, for a relation to be in BCNF, it must satisfy the condition that for every non-trivial functional dependency X → Y, X must be a superkey. Since a relation in 4NF already eliminates multivalued dependencies, all functional dependencies remaining are single-valued.

Therefore, in a relation that is in 4NF, all functional dependencies are single-valued and trivially satisfy the condition for BCNF. Hence, a relation in 4NF must also be in BCNF.

Q33: Definition of BCNF and Comparison with 3NF

CO2

Definition of BCNF (Boyce-Codd Normal Form): BCNF is a database normalization form that ensures that there are no non-trivial functional dependencies of attributes on anything other than a superkey. It eliminates all redundant dependencies between attributes.

Difference from 3NF:

- 3NF eliminates transitive dependencies, while BCNF eliminates all non-trivial functional dependencies on candidate keys.
- BCNF is stricter than 3NF and may result in more relations than 3NF, but it ensures that all functional dependencies are directly on the primary key.

Example:
Consider a relation R(A, B, C) with functional dependencies {A → B, B → C}:

- In 3NF, it would be:
  - {A → B}
  - {B → C}
- In BCNF, it would be:
  - {A → B}
  - {B → C}

Q34: Explanation of Schema Refinement in Database Design

CO2

Schema refinement in database design involves refining the conceptual schema into internal and external schemas to meet the requirements of various users and applications. It ensures that the database design is flexible, efficient, and scalable. Refinement also involves optimizing the database schema for performance and ensuring data integrity and security.

Q35: Proof that Any Relation Schema with Two Attributes is BCNF

CO3

A relation schema with two attributes is trivially in BCNF if it has only one candidate key, as any non-trivial functional dependency must be on the entire set of attributes. In such a case, the relation would satisfy the conditions of BCNF automatically.

Q36: Determining Closure and Candidate Keys

CO3

Functional Dependencies:

- AB → C
- BD → EF
- AD → G
- A → H

Candidate Keys of R:

- Candidate keys are the minimal set of attributes that can uniquely identify each tuple in the relation.
- To find candidate keys, we compute the closure of each attribute or attribute set to determine if it includes all attributes of the relation.
- The closure of a set of attributes under a set of functional dependencies is the set of all attributes that are functionally determined by that set.
- From the given functional dependencies, we compute the closures of all possible attribute sets to find the candidate keys.

Example of Closure Calculation:

- Closure of {AB} = {ABCH} (using functional dependency AB → C and A → H)
- Closure of {BD} = {BDEF} (using functional dependency BD → EF)

By computing the closure of all possible attribute sets, we identify the candidate keys of the relation R(A,B,C,D,E,F,G,H).

UNIT 5

Level A. Easy Questions (2 marks each)

Q1: Definition of a Transaction and its Properties

CO1 A transaction is a logical unit of work performed within a database management system (DBMS) that must be executed as a whole. It follows the ACID properties, which include Atomicity, Consistency, Isolation, and Durability.

Q2: Different Phases of a Transaction

CO1 The transaction model typically consists of four main phases:

1. **Initialization:** The transaction is initiated.
2. **Execution:** The actual work or operations specified by the transaction are carried out.

3. **Commit:** The transaction is either committed, meaning that its changes are made permanent, or rolled back, meaning that its changes are undone.
4. **Cleanup:** Any resources or locks held by the transaction are released.

Q3: Recoverable Schedules

CO1 Recoverable schedules are transaction schedules in which if a transaction T1 writes a data item that is later read by another transaction T2, then T1's commit must occur before T2's commit. This ensures that if T1 aborts, T2 can still read a consistent value from the data item.

Q4: Cascadeless Schedules

CO1 Cascadeless schedules are transaction schedules in which the effect of a rollback of a transaction T1 does not cascade to other transactions. This means that if T1 rolls back, any changes made by T1 should not affect the committed status of other transactions that have already completed.

Q5: Definition of Two-Phase Commit Protocol

CO1 The Two-Phase Commit (2PC) protocol is a distributed algorithm used to ensure the atomicity of transactions across multiple databases. It ensures that all databases involved in a distributed transaction either commit or abort the transaction together, preventing inconsistencies.

Q6: Implementation of Isolation

CO1 Isolation in database transactions refers to the property that the execution of transactions is isolated from each other, meaning that the intermediate states of transactions are not visible to other transactions until they are committed. This is typically achieved through concurrency control mechanisms such as locks or timestamps.

Q7: Procedure to Test Serializability

CO2 Testing for serializability involves analyzing the transaction schedule to determine if it is equivalent to a serial execution of transactions. One common approach is to construct a precedence graph, where each node represents a transaction and edges represent conflicts between transactions. If the graph is acyclic, the schedule is serializable.

Q8: Explanation of Different Types of Locks

CO1 Different types of locks used in database systems include:

- **Shared Locks:** Allow multiple transactions to read a resource simultaneously but prevent any transaction from writing to it.

- **Exclusive Locks:** Prevent other transactions from both reading and writing to a resource while it is locked.
- **Intent Locks:** Indicate the intention of a transaction to acquire a certain type of lock on a resource.

Q9: Discussion on Failure Classification

CO1 Failures in database systems can be classified into various categories, such as:

- **Transaction Failures:** Failures that occur due to errors in the execution of transactions.
- **System Failures:** Failures that occur due to hardware or software malfunctions.
- **Media Failures:** Failures that occur due to errors in storage media.
- **Site Failures:** Failures that occur at individual sites in a distributed system.

Q10: Definition of a Checkpoint

CO2 A checkpoint is a point in the transaction log at which all database changes made by transactions up to that point are guaranteed to have been written to stable storage. It is used for recovery purposes, allowing the system to restart from a consistent state after a failure.

Q11: Discussion on Failures with Loss of Non-volatile Storage

CO1 Failures involving the loss of non-volatile storage, such as disk failures or power outages, can lead to data loss or corruption. In such cases, recovery mechanisms like backups, redundancy, or journaling are essential to restore the system to a consistent state.

Q12: Demonstration of Conflict Serializability

CO3 Conflict serializability is a property of transaction schedules in which the execution order of transactions does not affect the final outcome. It can be demonstrated using a precedence graph, where conflicting operations between transactions are represented as edges. If the graph is acyclic, the schedule is conflict serializable.

Q13: Discussion on View Serializability

CO1 View serializability is a weaker form of serializability that ensures that the execution of concurrent transactions produces the same result as if they were executed one at a time. It considers only the final states of transactions and does not require considering intermediate states.

Q14: Explanation of Transition States

CO1 Transition states in database transactions refer to the intermediate states of a transaction as it progresses through its execution phases. These states include the initial state, execution state, commit state, and abort state. Proper management of transition states ensures the consistency and integrity of the database.

Q15: Explanation of ACID Properties

CO1 ACID properties are the key characteristics of a reliable database transaction:

- **Atomicity:** Transactions are atomic, meaning they either complete fully or have no effect at all.
- **Consistency:** Transactions maintain the consistency of the database by transforming it from one consistent state to another.
- **Isolation:** Transactions are isolated from each other, ensuring that the execution of one transaction does not interfere with others.
- **Durability:** Once a transaction is committed, its effects are permanent and survive system failures.

Q16: Explanation of Locking Protocols

CO1 Locking protocols are mechanisms used to manage concurrent access to data in a database system. They include:

- **Two-Phase Locking (2PL):** Transactions acquire locks in two phases (growing phase and shrinking phase) and release them only after they have committed or aborted.
- **Strict Two-Phase Locking (S2PL):** Similar to 2PL but requires that all locks be held until the transaction commits or aborts.
- **Deadlock Prevention:** Techniques to prevent deadlock situations, such as timeouts or wait-die and wound-wait algorithms.

Q17: Definition of Timestamp-Based Protocol

CO1 The timestamp-based protocol is a concurrency control technique that assigns a unique timestamp to each transaction based on its start time. It uses timestamps to determine the order of transactions and resolve conflicts between conflicting operations by allowing the transaction with the earlier timestamp to proceed.

Q18: Explanation of Multiple Granularity

CO1 Multiple granularity refers to the ability of a database management system to support different levels of locking granularity, ranging from coarse-grained locks at the level of entire tables to fine-grained locks at the level of individual data items. It allows for better concurrency control and resource utilization.

Q19: Explanation of Storage Structure

CO1 The storage structure of a database refers to the organization of data on physical storage devices such as disks. It includes:

- **Heap Files:** Unordered collections of records stored sequentially.
- **Sorted Files:** Records stored in sorted order based on a key field.
- **Hashed Files:** Records stored in a hash table for quick access based on a hash function.
- **Index Structures:** Data structures like B-trees or hash tables used to speed

Level B. Intermediate Questions (5 marks each)

Q21: Explanation of ACID Properties with Examples

CO2

ACID Properties:

- **Atomicity:** Ensures that a transaction is either completed in its entirety or not at all.
- **Consistency:** Guarantees that the database remains in a consistent state before and after the transaction.
- **Isolation:** Ensures that the execution of one transaction is isolated from others until it is complete.
- **Durability:** Ensures that the effects of committed transactions persist even in the event of system failures.

Illustration with Examples:

- *Atomicity:* Consider a bank transfer where money is debited from one account and credited to another. If the debit succeeds but the credit fails, the transaction should be rolled back to maintain atomicity.
- *Consistency:* In a library database, if a book is borrowed by a user, the database should ensure that the book's availability status is updated consistently to reflect the transaction.

- *Isolation:* In an online shopping scenario, if two users simultaneously attempt to purchase the last item in stock, the database should handle each transaction independently to prevent interference.
- *Durability:* After a successful online payment transaction, the confirmation email sent to the user should remain accessible even if the server crashes immediately after.

Q22: Implementation of Atomicity and Durability

CO1

Atomicity Implementation:

- Ensure that all operations within a transaction are completed successfully or none at all.
- Use techniques such as transaction logging and rollback mechanisms to revert changes in case of failure.

Durability Implementation:

- Write transaction updates to stable storage (e.g., disk) before committing to ensure persistence.
- Employ techniques like write-ahead logging to record changes before they are made, ensuring that they can be recovered in the event of a failure.

Q23: Illustration of Concurrent Transaction Execution

CO1

Example: Consider two transactions T1 and T2:

- T1: Transfer $100 from account A to account B.
- T2: Withdraw $50 from account A.

Concurrent Execution:

- If T1 reads the balance of account A before T2 completes, it may transfer more money than available, violating consistency.
- Proper concurrency control mechanisms like locking or timestamp ordering must be employed to ensure consistent outcomes.

Q24: Discussion on Serializability

CO1

Serializability:

- Ensures that the execution of concurrent transactions produces the same result as if they were executed sequentially, preserving consistency.
- Achieved by ensuring that transactions are executed in a serializable order, meaning there is no interference between transactions.

Techniques for Ensuring Serializability:

- Precedence Graph: Construct a graph where each node represents a transaction, and edges represent conflicts. If the graph is acyclic, the schedule is serializable.
- Conflict Serializable Schedules: Schedules in which the conflicting operations of transactions can be swapped without changing the final outcome.

Q25: Discussion on Two-Phase Locking (2PL) and Strict Two-Phase Locking (S2PL) Protocols

CO1

Two-Phase Locking (2PL):

- Transactions acquire locks in two phases: growing phase (acquire locks) and shrinking phase (release locks).
- Guarantees conflict serializability but may lead to deadlocks if not managed properly.

Strict Two-Phase Locking (S2PL):

- Similar to 2PL but holds all locks until the transaction commits or aborts.
- Eliminates the possibility of deadlock by preventing transactions from releasing locks prematurely.

Q26: Description of Timestamp-Based Locking Protocols

CO2

Timestamp-Based Locking:

- Assigns a unique timestamp to each transaction based on its start time.
- Determines the order of transactions using timestamps and resolves conflicts by allowing the transaction with the earlier timestamp to proceed.
- Helps in maintaining serializability and avoids deadlocks.

Q27: Description of Validation-Based Locking Protocols

CO2

Validation-Based Locking:

- Transactions execute without acquiring locks initially.
- Before committing, a transaction validates its operations against other concurrent transactions to ensure consistency.
- If validation succeeds, the transaction commits; otherwise, it aborts and restarts.

Q28: Discussion on Multiple Granularity

CO2

Multiple Granularity:

- Allows different levels of locks to be applied at various levels of data granularity.
- Provides flexibility in managing concurrency by allowing locks at the level of entire tables, pages, or individual records.
- Ensures efficient resource utilization and reduces contention.

Q29: Explanation of Storage Structure

CO1

Storage Structure:

- Refers to how data is organized and stored on physical storage devices like disks.
- Includes heap files, sorted files, hashed files, and index structures.
- Determines how data is accessed and manipulated by the database management system.

Q30: Discussion on Deferred and Immediate Database Modification

CO2

Deferred Database Modification:

- Delayed application of transaction updates until the transaction commits.
- Improves concurrency by allowing multiple transactions to execute concurrently without locking conflicts.

Immediate Database Modification:

- Applies transaction updates immediately upon their execution.
- Ensures that changes are visible to other transactions immediately but may lead to increased contention and locking overhead.

Level C. Difficult Questions (10 marks each)

Q31: Recovery from Concurrent Transactions

CO2

Recovery from Concurrent Transactions:

- **Transaction Rollback:** If a transaction encounters a failure, it needs to be rolled back to its previous consistent state. Concurrent transactions may interfere with each other, causing cascading rollbacks. To recover, a system needs to identify and undo the effects of failed transactions.

Concurrency Control Techniques:

- **Locking:** Ensures that transactions access shared resources in a controlled manner to prevent conflicts.
- **Timestamp Ordering:** Assigns timestamps to transactions and ensures that conflicting operations are executed in a predetermined order.

Q32: Explanation of Buffer Management

CO1

Buffer Management:

- **Purpose:** Optimizes data transfer between disk and memory to improve database performance.
- **Buffer Pool:** Reserved portion of memory for storing database pages temporarily.
- **Buffer Replacement Policies:** Algorithms to decide which pages to keep in memory and which to replace when new pages are needed.

Q33: Different Types of Advanced Recovery Techniques

CO2

Advanced Recovery Techniques:

1. **Shadow Paging:** Maintains a shadow copy of the database, allowing recovery to a consistent state by reverting to the shadow copy.
2. **Log-based Recovery:** Uses transaction logs to recover from failures by replaying logged operations to restore the database to a consistent state.
3. **Checkpointing:** Periodic saving of the database state to reduce the amount of log to be replayed during recovery, improving efficiency.

Q34: Detailed Explanation of Remote Backup Systems

CO4

Remote Backup Systems:

- **Purpose:** Ensures data availability and disaster recovery by storing backups of critical data at off-site locations.
- **Types:**
    - **Physical Off-Site Backup:** Physical storage devices containing backups stored at a remote location.
    - **Cloud Backup:** Data backups stored on remote servers maintained by cloud service providers.
- **Benefits:** Provides protection against data loss due to disasters, theft, or system failures.
- **Challenges:** Ensuring data security, maintaining data consistency, and managing bandwidth for backup operations.

Q35: Discussion on Strict Two-Phase Locking Protocol and Timestamp-Based Protocol

CO3

Strict Two-Phase Locking Protocol:

- **Characteristics:** Requires transactions to acquire all locks before any are released. Prevents cascading aborts and ensures serializability.
- **Drawbacks:** May lead to lock contention and deadlocks, impacting concurrency.

Timestamp-Based Protocol:

- **Characteristics:** Assigns unique timestamps to transactions and orders conflicting operations based on timestamps. Ensures serializability without acquiring locks.

- **Advantages:** Reduces lock contention and deadlocks, improving concurrency and performance.

Q36: Explanation of Transaction and ACID Properties

CO4

Transaction:

- **Definition:** A unit of work performed on a database, typically consisting of multiple read and write operations.
- **ACID Properties:**
  - **Atomicity:** Ensures that transactions are either completed in their entirety or not at all.
  - **Consistency:** Maintains data consistency before and after transaction execution.
  - **Isolation:** Ensures that the execution of one transaction is isolated from others until it is complete.
  - **Durability:** Guarantees that committed transaction changes are permanently saved and recoverable, even in the event of system failures.