SEM-3

DBMS QUESTION BANK

SOLUTION

2301030700076

2301030700077

SOLVED BY – LAKSHYA BHATT

AND

RAJVEER SINGH BHADORIYA

UNIT - 1

1. **Compare traditional file processing systems and database management systems. 3 MARKS**

- ANS : **Data Redundancy:** Traditional file systems have high redundancy with duplicate data, while DBMS minimizes redundancy by centralizing data.
- **Data Consistency:** File systems may lead to inconsistencies, but DBMS ensures consistency across the database.
- **Data Security and Integrity:** File systems have limited security and integrity controls, whereas DBMS provides better data security, integrity, and access control.

2. What is DBMS? Lis out applications of DBMS. 4 marks

ANS : DBMS (Database Management System): A DBMS is software that helps store, manage, and retrieve data efficiently. It provides tools for data organization, access, security, and manipulation.

Applications of DBMS:

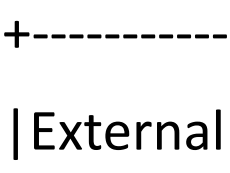
1. **Banking:** Managing customer accounts, transactions, and loans.
2. **E-commerce:** Handling orders, payments, and customer data.
3. **Education:** Student records, course management, and grading.
4. **Healthcare:** Patient records, billing, and scheduling.

3. Draw and explain the three-level architecture of DBMS. 6 marks

ANS : **Three-Level Architecture of DBMS**

1. **Internal Level**:
    - **Description**: Deals with the physical storage of data.
    - **Details**: Describes how data is stored in the database, including data structures and file organization.
2. **Conceptual Level**:
    - **Description**: Provides a logical view of the entire database.
    - **Details**: Defines what data is stored and the relationships among those data, hiding physical storage details from users.
3. **External Level**:
    - **Description**: Provides a user-specific view of the database.
    - **Details**: Different users can have different views of the same database, tailored to their needs.

Diagram



+-----------------+

|Conceptual Level|

+-----------------+

| Internal Level|

+-----------------+

4. State how an Entity-Relationship (ER) model represents real-world entities. 4 marks

Ans : An **Entity-Relationship (ER) model** represents real-world entities using:

1. **Entities:** Objects or concepts (e.g., Student, Employee) that have a distinct existence in the real world.
2. **Attributes:** Properties or characteristics of entities (e.g., Name, Age) that describe them.
3. **Relationships:** Associations between entities (e.g., Student *enrolls in* Course) showing how they interact.

ER diagrams use rectangles for entities, ovals for attributes, and diamonds for relationships, making data modeling intuitive and visual.

5. Define Data Abstraction and explain different levels of data abstraction. 4 marks

Ans : Data Abstraction in DBMS is the process of hiding the complexities of data storage and presenting users with a simpler view of the database.

Levels of Data Abstraction:

1. Physical Level: Describes how data is stored physically in the database, including file structures and storage details.

2. Logical Level: Describes what data is stored and the relationships among the data; focuses on the database's structure without physical details.

3. View Level: Presents data to users through different customized views; hides the complexity and provides only relevant information.

This abstraction helps in data independence and simplifies database management.

6. Explain Data Independence. 4 marks

Ans : **Data Independence** refers to the ability to modify a database schema at one level without affecting the schema at the next higher level. It is a key concept in database management systems (DBMS) and is categorized into two types:

1. **Logical Data Independence**: The capacity to change the conceptual schema without altering the external schema or application programs. For example, adding a new field to a table should not affect the user views or application programs.
2. **Physical Data Independence**: The ability to change the physical storage of data without

affecting the conceptual schema. For example, reorganizing the file structure or using different storage devices does not impact how data is accessed logically.

Key Point: Data independence ensures flexibility and ease of maintenance in a database system.

7. Explain the DBMS languages with examples: DDL, DML, and DCL.

Ans : **DBMS (Database Management System) Languages** are used to define, manipulate, and control data in a database. They are mainly categorized into three types: DDL, DML, and DCL.

1. DDL (Data Definition Language)

- **Definition**: DDL commands are used to define and manage the database structure.
- **Examples**:
    -

CREATE: Creates a new table or database.

CREATE TABLE Students (ID INT, Name VARCHAR(50));

ALTER: Modifies an existing database structure.

ALTER TABLE Students ADD Age INT;

DROP: Deletes a table or database

DROP TABLE Students;

2. DML (Data Manipulation Language)

- **Definition**: DML commands are used to manipulate data stored in the database.
- **Examples**:

INSERT: Adds new data to a table

INSERT INTO Students (ID, Name, Age) VALUES (1, 'Alice', 20);

UPDATE: Modifies existing data in a table

UPDATE Students SET Age = 21 WHERE ID = 1;

DELETE: Removes data from a table

DELETE FROM Students WHERE ID = 1;

SELECT: Retrieves data from a table

SELECT * FROM Students;

3. DCL (Data Control Language)

- **Definition**: DCL commands are used to control access to the data in the database.
- **Examples**:

GRANT: Gives permissions to users

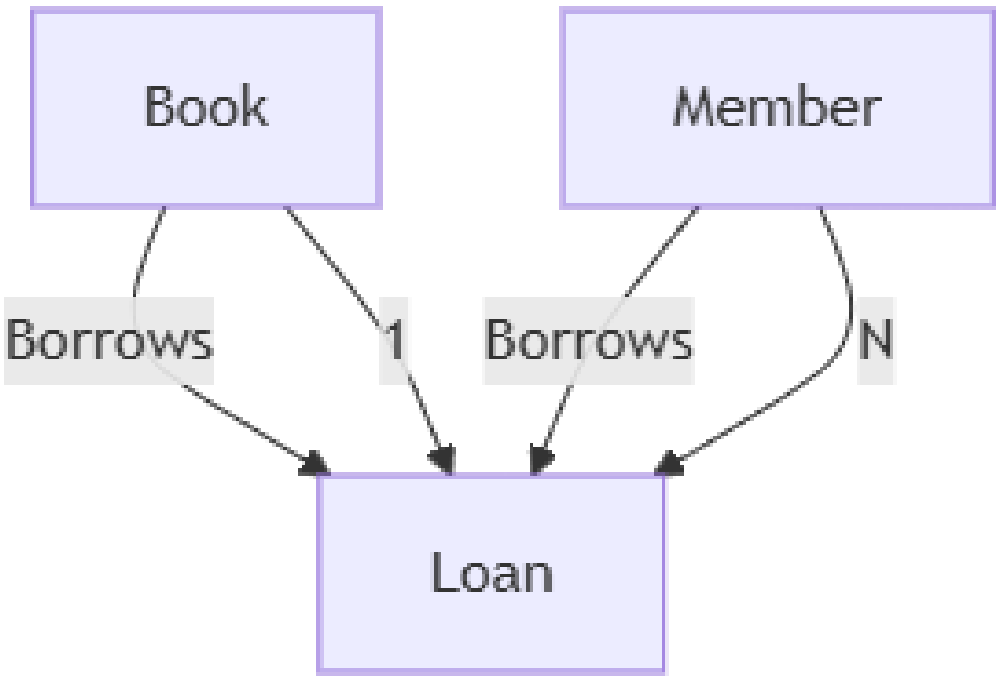
GRANT SELECT ON Students TO User1;

REVOKE: Removes permissions from users

REVOKE SELECT ON Students FROM User1;

8.Draw an E-R diagram for the Library Management System. 6 marks

Ans :



9. Define E-R Diagram. Discuss generalization in the E-R Diagram. 6 marks

Ans : Definition of E-R Diagram:

An E-R (Entity-Relationship) Diagram is a graphical representation of entities, their attributes, and the relationships between them in a database system. It is used to design and model the structure of a database, showing how data is organized and how entities are connected.

Generalization in E-R Diagram:

Generalization is a process in E-R diagrams where two or more entities with similar attributes are combined into a higher-level entity called a superclass. This helps in reducing redundancy and organizing data more efficiently.

- Example: Consider two entities, Car and Bike, which have common attributes like Vehicle_ID, Manufacturer, and Price. These can be generalized into a higher-level entity called Vehicle.
- Hierarchy: Car and Bike become subclasses of the generalized Vehicle entity. The superclass Vehicle represents all common attributes, while subclasses represent specific attributes.

10. Differentiate strong entity set and weak entity set. Demonstrate the concept using both real-time examples and an E-R diagram. 6 marks

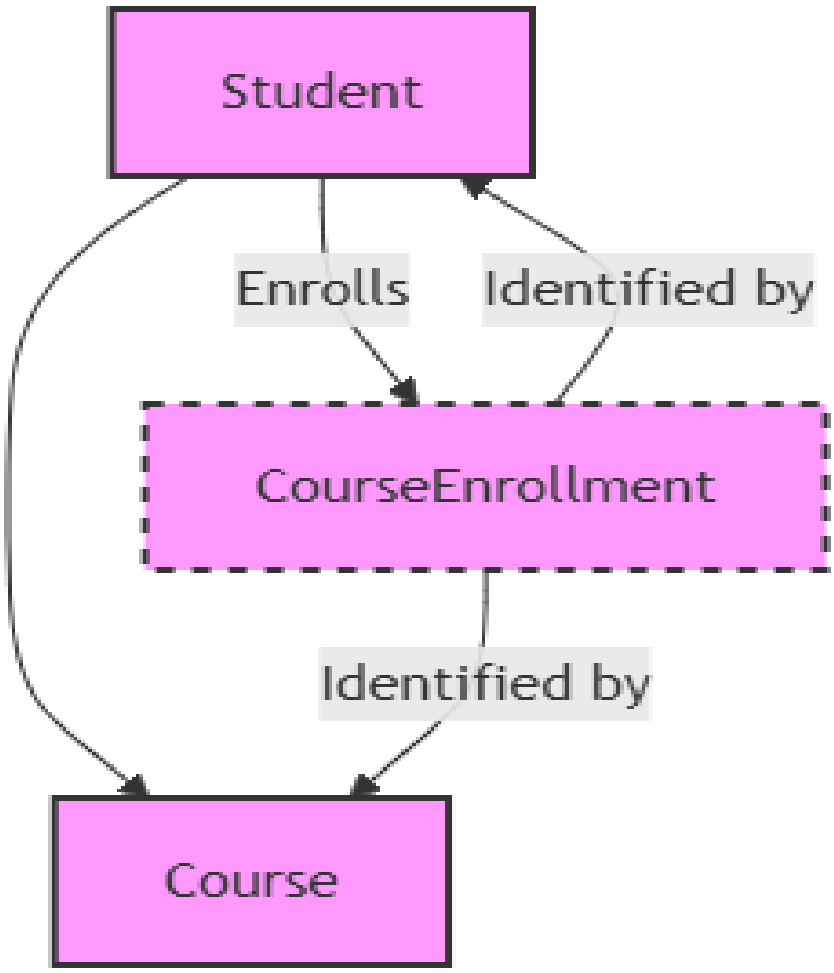
Ans : **Difference Between Strong Entity Set and Weak Entity Set:**

1. **Strong Entity Set**:
    - **Definition**: A strong entity set has a primary key that uniquely identifies each entity in the set. It does not depend on any other entity.
    - **Characteristics**:
        - Has a primary key.
        - Represented by a single rectangle in an E-R diagram.
    - **Example**: Student entity with attributes like Student_ID (primary key), Name, Age.

1. **Weak Entity Set**:
    - **Definition**: A weak entity set does not have a primary key of its own and relies on a **strong entity** for its identification.
    - **Characteristics**:
        - Has a partial key (known as a discriminator) and depends on a foreign key from a strong entity.

        - Represented by a double rectangle in an E-R diagram.
    - **Example**: Dependent entity in an insurance system, where Dependent (e.g., children or spouse) relies on the Employee entity for identification. Attributes could include Dependent_Name, Relationship with Employee.

E-R Diagram



11. What are the constraints in DBMS? explain with a proper example. 6 marks

Ans : In a Database Management System (DBMS), constraints are rules applied to data to ensure its accuracy and integrity. Here are the main types of constraints with simple examples:

1. Primary Key Constraint: Ensures each record in a table is unique and not null.
    - Example: In a Users table, UserID might be a primary key, ensuring each user has a unique ID.
2. Foreign Key Constraint: Ensures that a value in one table matches a value in another table, maintaining referential integrity.
    - Example: In an Orders table, a CustomerID foreign key might reference CustomerID in the Customers table to ensure each order is linked to a valid customer.
3. Unique Constraint: Ensures all values in a column are distinct.
    - Example: In a Products table, the ProductCode column might have a unique constraint to ensure no two products have the same code.
4. Not Null Constraint: Ensures a column cannot have a null value.

    - Example: In a Books table, the Title column might have a not null constraint to ensure every book has a title.
5. Check Constraint: Ensures that values in a column satisfy a specific condition.
    - Example: In an Employees table, a check constraint might ensure that the Age column has values greater than or equal to 18.

12. Define Primary Key, Foreign Key, NOT NULL constraints, and referential integrity (Foreign Key) constraint. 6 marks

1. Primary Key: A column or a set of columns in a table that uniquely identifies each row. It must have unique values and cannot contain NULL values.

- Example: StudentID in a Students table is a primary key because each student has a unique ID.
-

2. Foreign Key: A column or set of columns in one table that refers to the primary key in another table. It creates a link between two tables to ensure referential integrity.

- Example: StudentID in a Enrollments table is a foreign key that references the StudentID primary key in the Students table
- .

3. NOT NULL Constraint: Ensures that a column cannot have a NULL value, meaning the column must always contain a value.

- Example: Email column in a Users table might have a NOT NULL constraint to ensure every user has an email.
-

4. Referential Integrity (Foreign Key) Constraint: Ensures that a foreign key value always matches a primary key value in the related table, maintaining consistency between related tables.

- Example: If a CourseID in the Enrollments table refers to CourseID in a Courses table, there must be a corresponding CourseID in

the Courses table for every CourseID in Enrollments.

13. Explain the Network model and Relational model in brief. 4 marks

Ans :

1. Network Model:
    - A database model where data is represented using a graph structure with nodes (records) and edges (links).
    - Allows many-to-many relationships between entities, meaning a child can have multiple parents.
    - Uses pointers to navigate between records, making data retrieval fast but complex to manage.
    - Example: An employee can belong to multiple departments, and each department can have multiple employees.

2. Relational Model:
    - A database model that represents data in tables (relations) consisting of rows (tuples) and columns (attributes).
    - Supports one-to-one, one-to-many, and many-to-many relationships using primary and foreign keys.
    - It is easier to use and more flexible, with powerful query capabilities using SQL (Structured Query Language).
    - Example: A Students table linked to a Courses table via a StudentID foreign key.

The Network Model is hierarchical and complex, while the Relational Model is simpler and widely used in modern databases.

UNIT – 2

14. Describe the difference between Tuple Relational Calculus and Domain Relational Calculus. 6 marks

Ans :

tuple Relational Calculus (TRC) and Domain Relational Calculus (DRC) are both types of relational calculus used to query relational databases, but they differ in their approach:

1. Tuple Relational Calculus (TRC):
    - TRC uses tuples (rows) as variables.
    - Queries are expressed by specifying the desired properties of the tuples.
    - Example: {t | t ∈ Student AND t.Age > 18} retrieves all tuples t from the Student relation where the Age is greater than 18.
2. Domain Relational Calculus (DRC):
    - DRC uses domain variables that represent individual fields (attributes).
    - Queries are expressed by specifying the conditions on these attributes.

- Example: {<Name, Age> | ∃ (Student.Name = Name ∧ Student.Age = Age ∧ Age > 18)} retrieves the Name and Age of students where Age is greater than 18.

15. Explain the working of the Cartesian Product operation with an appropriate table. 6 marks

Ans :

The **Cartesian Product** operation in relational databases combines every row of one table with every row of another table. If Table A has mmm rows and Table B has nnn rows, the Cartesian Product (denoted as A×BA \times BA×B) results in m×nm \times nm×n rows. Each row in the result is a combination of one row from Table A and one row from Table B.

Example

Consider two tables:

Table A

ID	Name
1	Alice
2	Bob

Table B

Color
Red
Blue

Cartesian Product (A × B):

Combines each row of Table A with each row of Table B.

ID	Name	Color
1	Alice	Red
1	Alice	Blue
2	Bob	Red
2	Bob	Blue

The result contains all possible combinations of rows from both tables.

16. List the relational algebra operators. Discuss any one such algebra operator with suitable examples. 6 marks

Ans :

Relational Algebra Operators:

1. Selection (σ)
2. Projection (π)
3. Union (∪)
4. Set Difference (-)
5. Cartesian Product (×)
6. Rename (ρ)
7. Intersection (∩)
8. Join (⋈)
9. Division (÷)

Discussion: Selection (σ)

The Selection operator (σ) is used to select rows from a relation (table) that satisfy a given condition.

Example

Consider a table **Student**:

ID	Name	Age
1	Alice	20
2	Bob	18
3	Carol	22

To select students with `Age > 18`, the selection operation is:

σ(Age > 18) (Student)

Result:

ID	Name	Age
1	Alice	20
3	Carol	22

This operation filters rows based on the condition, returning only those that match.

17. Explain all types of join in detail with an example. 3 marks

Ans :

1. **Inner Join**: Returns only matching rows from both tables.

   Example:

   - **Table A:**

ID	Name
1	Alice
2	Bob

   - **Table B:**

ID	Course
1	Math
3	Science

   - **Inner Join Result:**

ID	Name	Course
1	Alice	Math

2. **Left Join (Left Outer Join)**: Returns all rows from the left table and matching rows from the right table. Non-matching rows from the right are filled with `NULL`.

   - **Left Join Result:**

ID	Name	Course
1	Alice	Math
2	Bob	NULL

3. **Right Join (Right Outer Join)**: Returns all rows from the right table and matching rows from the left table. Non-matching rows from the left are filled with `NULL`.

   - **Right Join Result:**

ID	Name	Course
1	Alice	Math
3	NULL	Science

4. **Full Join (Full Outer Join)**: Returns all rows when there is a match in either table. Non-matching rows are filled with `NULL`.

   - **Full Join Result:**

ID	Name	Course
1	Alice	Math
2	Bob	NULL
3	NULL	Science

5. **Cross Join**: Returns the Cartesian product of both tables, combining each row of the first table with every row of the second table.

UNIT-3

18. Explain Armstrong's axioms. 3 MARKS

ANS :

Armstrong's Axioms are a set of rules used to infer all functional dependencies in a relational database. They are fundamental for normalization and help in finding the closure of a set of attributes.

Armstrong's Axioms:

1. **Reflexivity (R):**
   If Y⊆XY \subseteq XY⊆X, then X→YX \rightarrow YX→Y.
   (Any set of attributes determines itself or its subsets.)
2. **Augmentation (A):**
   If X→YX \rightarrow YX→Y, then XZ→YZXZ \rightarrow YZXZ→YZ.
   (Adding the same set of attributes to both sides of a dependency keeps it valid.)
3. **Transitivity (T):**
   If X→YX \rightarrow YX→Y and Y→ZY \rightarrow ZY→Z, then X→ZX \rightarrow ZX→Z.
   (If one set of attributes determines a second set, and the second set determines a third, then the first set determines the third.)

19. Define functional dependency with a simple example. 3 marks.

Ans :

A **Functional Dependency (FD)** is a constraint between two sets of attributes in a relational database. It states that if two tuples (rows) have the same values for a set of attributes X, they must also have the same values for another set of attributes Y. This is denoted as $X \rightarrow Y$ (X determines Y).

Example:

Consider a table **Student**:

StudentID	Name	Age
1	Alice	20
2	Bob	22
3	Alice	20

Functional Dependency:

`StudentID → Name, Age`

This means the `StudentID` uniquely determines the `Name` and `Age` of a student. No two rows with the same `StudentID` can have different `Name` or `Age` values.

20. Compute closure of following set F of functional dependencies for relation schema r (A, B, C, D, E). A → BC , CD → E , B → D , E → A List the candidate keys for R. 6 marks

Ans :

To find the **closure** of a set of attributes and **candidate keys**, we need to compute the attribute closure and determine which attributes can

uniquely identify all attributes in the relation schema R(A,B,C,D,E)R(A, B, C, D, E)R(A,B,C,D,E).

Given Functional Dependencies (F):

1. A→BCA \rightarrow BCA→BC
2. CD→ECD \rightarrow ECD→E
3. B→DB \rightarrow DB→D
4. E→AE \rightarrow AE→A

Step 1: Compute Attribute Closures

1. **Closure of A (A+^++):**
    - Start: AAA
    - A→BCA \rightarrow BCA→BC → A+={A,B,C}A^+ = \{A, B, C\}A+={A,B,C}
    - B→DB \rightarrow DB→D → A+={A,B,C,D}A^+ = \{A, B, C, D\}A+={A,B,C,D}
    - CD→ECD \rightarrow ECD→E → A+={A,B,C,D,E}A^+ = \{A, B, C, D, E\}A+={A,B,C,D,E}

A+={A,B,C,D,E}A^+ = \{A, B, C, D, E\}A+={A,B,C,D,E} (all attributes)

2. **Closure of B (B+^++):**
    - Start: BBB
    - B→DB \rightarrow DB→D → B+={B,D}B^+ = \{B, D\}B+={B,D}
    - DDD does not determine any new attributes.

B+={B,D}B^+ = \{B, D\}B+={B,D}

3. **Closure of C (C+^++):**
    - Start: CCC
    - CCC does not determine any new attributes on its own.

C+={C}C^+ = \{C\}C+={C}

4. **Closure of CD (CD+^++):**
    - Start: CDCDCD
    - CD→ECD \rightarrow ECD→E → CD+={C,D,E}CD^+ = \{C, D, E\}CD+={C,D,E}
    - E→AE \rightarrow AE→A → CD+={C,D,E,A}CD^+ = \{C, D, E, A\}CD+={C,D,E,A}

- A→BCA \rightarrow BCA→BC → CD+={A,B,C,D,E}CD^+ = \{A, B, C, D, E\}CD+={A,B,C,D,E}

CD+={A,B,C,D,E}CD^+ = \{A, B, C, D, E\}CD+={A,B,C,D,E} (all attributes)

Step 2: Determine Candidate Keys

A **candidate key** is an attribute (or a set of attributes) whose closure contains all attributes of the relation RRR.

- From the closures, A+={A,B,C,D,E}A^+ = \{A, B, C, D, E\}A+={A,B,C,D,E} and CD+={A,B,C,D,E}CD^+ = \{A, B, C, D, E\}CD+={A,B,C,D,E}.

Candidate Keys for R:

- **A**
- **CD**

These are the minimal sets of attributes that can uniquely identify all attributes in the relation schema RRR.

21. What is meant by normalization? Write its need. List and discuss various normalization forms. 6 marks

Ans :

Need for Normalization:

1. **Reduce Data Redundancy**: Avoid duplication of data.
2. **Ensure Data Integrity**: Maintain data consistency across the database.
3. **Optimize Queries**: Improve performance and reduce storage space.
4. **Simplify Maintenance**: Easier updates, deletions, and insertions without anomalies.

Normal Forms:

Normalization involves several stages called **Normal Forms (NFs)**:

1. **First Normal Form (1NF):**
    - A table is in 1NF if all columns have atomic (indivisible) values.

- **Example**: A column should not have multiple values like {1, 2, 3}.

2. **Second Normal Form (2NF):**
   - A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the **entire** primary key.
   - **Example**: No partial dependency of any column on a part of the primary key.
3. **Third Normal Form (3NF):**
   - A table is in 3NF if it is in 2NF and has no transitive dependency (non-key attribute depending on another non-key attribute).
   - **Example**: If A → B and B → C, then C should not depend on B.
4. **Boyce-Codd Normal Form (BCNF):**
   - A stronger version of 3NF. A table is in BCNF if every determinant (attribute that determines another attribute) is a candidate key.
   - **Example**: Handles cases not covered by 3NF where non-trivial dependencies exist.

5. **Fourth Normal Form (4NF):**
    - A table is in 4NF if it is in BCNF and has no multi-valued dependencies.
    - **Example**: A student may have multiple skills and multiple projects; skills and projects should not create multi-valued dependencies.
6. **Fifth Normal Form (5NF):**
    - A table is in 5NF if it is in 4NF and has no join dependency (where information can be reconstructed without loss).
    - **Example**: Decomposing tables without losing information or creating redundancy.

Normalization is crucial for creating efficient, scalable, and maintainable databases by organizing data logically..

22. Discuss the importance of achieving a lossless design in relational database design. Is it always possible? 6 marks

Ans :

Importance of Lossless Design:

1. **Data Integrity**: Ensures no data is lost during decomposition, maintaining accuracy.
2. **Consistency**: Prevents inconsistency in data after decomposition.
3. **Avoid Redundancy**: Keeps the data clean without unnecessary duplicates.
4. **Efficient Queries**: Improves query performance by reducing the size and complexity of tables.
5. **Simplifies Maintenance**: Easier to update and manage smaller, well-structured tables without data loss.

Is Lossless Design Always Possible?

- Yes, it is possible to achieve a lossless design, but it requires careful decomposition. Decomposition should follow certain rules, such as ensuring that the common attribute between decomposed tables is a **superkey** in at least one of the resulting tables.

- **Example of Lossless Decomposition**: If a relation R(A,B,C)R(A, B, C)R(A,B,C) has a functional dependency A→BA \rightarrow BA→B, it can be decomposed into R1(A,B)R1(A, B)R1(A,B) and R2(A,C)R2(A, C)R2(A,C). The join of R1R1R1 and R2R2R2 will reconstruct RRR without loss.

23. State true or false: Any relation schema that satisfies BCNF also satisfies 3NF. 6 marks

Ans :

True.

A relation schema that satisfies **BCNF (Boyce-Codd Normal Form)** also satisfies **3NF (Third Normal Form)** because BCNF is a stricter form of 3NF. BCNF requires every determinant to be a candidate key, which inherently satisfies all the conditions of 3NF.

24. Construct a relational schema for a database and implement normalization up to the 3rd

Normal Form (3NF). 6 marks

Consider a **Student Database** that contains information about students, their courses, and instructors.

Initial Schema (Unnormalized Form - UNF):

Student

StudentID	Name	CourseID	CourseName	InstructorID	InstructorName
1	Alice	C101	Math	I01	Dr. Smith
2	Bob	C102	Science	I02	Dr. John
3	Alice	C102	Science	I02	Dr. John

Step 1: First Normal Form (1NF)

Remove **repeating groups** and ensure **atomic** values:

Student

StudentID	Name	CourseID	InstructorID
1	Alice	C101	I01
2	Bob	C102	I02
3	Alice	C102	I02

Course

CourseID	CourseName
C101	Math
C102	Science

Instructor

InstructorID	InstructorName
I01	Dr. Smith
I02	Dr. John

Step 2: Second Normal Form (2NF)

Remove **partial dependencies** (dependencies on part of a composite key):

- **StudentCourse**: Combines **StudentID** and **CourseID**.
- **Instructor** table is already in 2NF.

Student

StudentID	Name
1	Alice
2	Bob

Course

CourseID	CourseName	InstructorID
C101	Math	I01
C102	Science	I02

Instructor

InstructorID	InstructorName
I01	Dr. Smith
I02	Dr. John

StudentCourse

StudentID	CourseID
1	C101
2	C102
1	C102

Step 3: Third Normal Form (3NF)

Remove **transitive dependencies** (non-key attribute depending on another non-key attribute):

- The **Course** table is dependent on the **InstructorID**.

Student

StudentID	Name
1	Alice
2	Bob

Course

CourseID	CourseName
C101	Math
C102	Science

Instructor

InstructorID	InstructorName
I01	Dr. Smith
I02	Dr. John

CourseInstructor

CourseID	InstructorID
C101	I01
C102	I02

StudentCourse

StudentID	CourseID
1	C101
2	C102
1	C102

Final Schema in 3NF:

The schema is now in **3NF** with all tables having no partial or transitive dependencies, ensuring data integrity and minimizing redundancy.

25. Explain how trivial and non-trivial dependencies affect the design of a database schema. 6 marks

Ans :

Trivial Dependency:

- A **trivial dependency** is a functional dependency where an attribute (or set of attributes) is dependent on itself or its subset.
    - Example: A→AA \rightarrow AA→A or A,B→AA, B \rightarrow AA,B→A.
- **Effect on Design**:
    - Trivial dependencies do not affect schema design since they are always true and do not imply any redundancy or anomalies.

Non-Trivial Dependency:

- A **non-trivial dependency** is a functional dependency where an attribute (or set of attributes) is dependent on another attribute (or set) that is not a subset of itself.

- Example: $A \rightarrow BA$ \rightarrow $BA \rightarrow B$ where $B \not\subseteq AB$ \not\subseteq $AB \subseteq A$.

- **Effect on Design**:
  - Non-trivial dependencies need careful consideration in schema design to avoid **data redundancy**, **update anomalies**, and **inconsistencies**.
  - Schemas are normalized (e.g., to 3NF or BCNF) to remove undesirable non-trivial dependencies, ensuring each attribute depends only on a candidate key.

Unit – 4

Relational database design

26 Explain the structure of a B-tree.

Ans:-

B-tree Structure in DBMS (Simplified):

1. **Definition:**
    - **A B-tree is a type of balanced tree that keeps data sorted.**
    - **It helps in fast searching, adding, and deleting of data.**
2. **Order (M):**
    - **The order (M) decides how many keys and children each node can have:**

- **A node can have up to M-1 keys and M child nodes.**

3. **Node Structure:**
    - **Each node contains:**
        - **Keys(numbers or data) arranged in order.**
        - **Pointers (links) to child nodes.**
4. **Balanced Property:**
    - **All leaf nodes (last nodes) are at the same level.**
    - **The root node must have at least 2 children, while other nodes must have at least half the maximum number of children.**
5. **Operations:**
    - **Search: Start from the root and follow the keys to find your target.**
    - **Insert/Delete: If a node gets too full or empty, it is split or merged to keep the tree balanced.**
6. **Benefits:**
    - **B-trees are great for databases because they allow fast access to large amounts of data.**

27.Explain B-tree and Hashing.

Ans:-

B-tree in DBMS:

1. Balanced Tree: It keeps data sorted and balanced, so all parts of the tree are at the same level.

2. Multiple Keys: Each node can hold several keys, not just one.

3. Order: The order (M) defines how many keys and child nodes a B-tree can have.

4. Efficient Search: Searching, inserting, and deleting data are fast (logarithmic time).

5. Used For: Commonly used for database indexing.

Hashing in DBMS:

1. Direct Access: Data is stored in a table, and each data item is given a unique key or "hash."

2. Hash Function: Converts a key into an address to find data quickly.

3. Fast Search: Allows quick search (constant time), making it faster than trees for lookups.

4. Used For: Searching in databases and efficient data retrieval.

28. Define Static and Dynamic Hashing.

Ans:-

Static Hashing:

- In static hashing, the size of the hash table is fixed.

- When data is inserted, it is stored in a predefined number of buckets.

- If the table becomes full, overflow occurs, and handling techniques like chaining are used.

- It does not adjust the table size automatically.

Dynamic Hashing:

- In dynamic hashing, the hash table size can grow or shrink as needed.

- When more data is added, the table expands by adding more buckets.

- It handles data more efficiently, especially when the amount of data increases or decreases over time.

29. Explain Indices in DBMS.

Ans:-

Indices in DBMS (Simple Explanation):

1. Definition:

- An index is a data structure that helps speed up the retrieval of data in a database.

2. Purpose:

- It works like a book index, helping you quickly find specific rows (records) in a large table.

3. Types of Indices:

- Primary Index: Based on the primary key; unique for each record.

- Secondary Index: Created for non-primary key columns to improve search speed.

- Clustered Index: Data is physically arranged in the order of the index.

- Non-clustered Index: Data is not physically sorted, but the index has pointers to the actual data.

4. Benefits:

- Makes searching, sorting, and filtering faster.

5. Downside:

- Takes up extra space and can slow down insert, update, and delete operations.

This helps databases handle large amounts of data more efficiently.

31. Describe different join strategies for a query and explain how they affect performance.

1. Nested Loop Join

- **How it works**: For each row in one table, it checks all rows in the other table to find matches.
- **When to use**: Best for small tables or when there is an index on the join column.
- **Advantages**: Simple to implement.

- **Disadvantages**: Because of repeated comparisons becomes slow with large tables.

2. Hash Join

- **How it works**: A hash table is built for one table, and the other table is scanned to find matches using the hash.
- **When to use**: Efficient for large, unsorted tables.
- **Advantages**: Fast for large datasets.
- **Disadvantages**: Requires more memory for the hash table and is slower if memory is limited.

3. Merge Join

- **How it works**: Both tables are sorted, and matching rows are found by comparing them in order.
- **When to use**: Ideal when both tables are already sorted.
- **Advantages**: Very fast if the data is pre-sorted.
- **Disadvantages**: Sorting takes time if tables are not already sorted.

4. Sort-Merge Join

- **How it works**: It sorts the tables and then merges them.
- **When to use**: Best when tables are unsorted and large.
- **Advantages**: More efficient than nested loops for larger datasets.
- **Disadvantages**: Sorting adds extra overhead.

Performance Comparison:

- **Nested loop** is simplest but slowest for large data.
- **Hash join** is fast for large datasets but needs memory.
- **Merge join** is fast with sorted data.
- **Sort-merge join** works well for unsorted large tables but takes time for sorting.

---

32. Describe the differences between primary and secondary indices in databases.

Ans:-

Differences between Primary and Secondary Indices (DBMS)

1. **Definition:**

- **Primary Index:** Created on the primary key or unique key of a table. It ensures data is stored in a sorted order.
- **Secondary Index:** Created on non-primary key attributes to allow faster searches based on those attributes.

2. **Uniqueness:**
   - **Primary Index:** Unique and cannot have duplicate values.
   - **Secondary Index:** May have duplicate values, depending on the data.

3. **Data Structure:**
   - **Primary Index:** Directly points to the data stored on disk, typically in sorted order.
   - **Secondary Index:** Points to the primary index or directly to the data but is not in any specific order.

33. Use a B-tree to illustrate how data is inserted and searched in a database.

Ans:-

A B-tree is a balanced tree data structure used in databases to keep data sorted and allow quick insertions, deletions, and searches.

1. **Insertion:**
   - **Data is inserted in sorted order. If a node (like a page in the database) gets full, it splits, and the middle value goes up to the parent node.**
   - **Example: If we insert values 10, 20, 5 into a B-tree, they will be arranged in nodes so that 10 is the root, with 5 and 20 as children.**
2. **Searching:**
   - **We compare the value to be searched with the root node.**
   - **If the value is smaller, we move to the left child; if bigger, move to the right.**
   - **Continue this until the value is found or we reach a leaf (no more children).**

34. Explain linear search and binary search algorithms for selection operation.

Ans:-

1. Linear Search:

- **Definition: Linear search goes through each element in the list, one by one, to find the target value.**

  ----> How it works:

  1. **Start from the first element.**
  2. **Compare each element with the target value.**
  3. **If it matches, return the position of the element.**
  4. **If not, move to the next element.**
  5. **Continue until the end of the list.**

- **Time Complexity: O(n) (n = number of elements).**
- **Use Case: Works well for small or unsorted lists.**

Example: To search for 5 in the list [3, 1, 5, 9]:

- **Compare 3 → no match.**
- **Compare 1 → no match.**
- **Compare 5 → match! (position 3).**

2. Binary Search:

- **Definition: Binary search works on a sorted list and divides the search space in half each time, making it faster than linear search.**
- **How it works:**

  1. **Check the middle element of the list.**
  2. **If it matches the target, return its position.**
  3. **If the target is smaller, search the left half of the list.**
  4. **If the target is larger, search the right half of the list.**
  5. **Repeat until the element is found or the list is exhausted.**

- **Time Complexity: O(log n) (n = number of elements).**
- **Use Case: Works well for large sorted lists.**

Example: To search for 5 in the sorted list [1, 3, 5, 7, 9]:

- **Middle element is 5 → match! (position 3).**

Summary (for 6 marks):

- **Linear Search checks each element one by one and works on unsorted lists, but it's slower for large lists (O(n)).**

- **Binary Search divides the list in half and is much faster for sorted lists (O(log n)), but the list must be sorted first.**

35. Describe the role of query processing in a DBMS.

Ans:-

Query Processing in DBMS refers to the steps taken by the system to execute a user's query. Here's a simple breakdown of its role:

1. **Input Query: A user writes a query (often in SQL) to request data from the database.**
2. **Query Parsing: DBMS checks the query for errors and creates a "parse tree," which shows the structure of the query.**
3. **Query Optimization: The DBMS analyzes different ways to run the query and picks the best (most efficient) one, minimizing time and resources.**
4. **Query Execution: The DBMS follows the optimized plan to get the data requested in the query.**
5. **Result: The requested data is provided to the user.**

Importance:

- **Efficiency: Ensures the query runs fast and uses fewer resources.**
- **Correctness: Confirms that the query runs without errors and gives accurate results.**

36. Explain the evaluation expression Process in query optimization.

Ans:-

In query optimization, the **evaluation expression process** is about finding the best way to execute a database query, ensuring it runs faster and uses fewer resources.

Breaking Down the Query: The query is first broken into smaller parts or expressions. Each part can be executed in different ways, like scanning the whole table or using an index.

1. **Generating Different Plans**: The system creates different ways (called **execution plans**) to execute the query.
   - ➔ For example, it can join tables using different methods (like **nested-loop** or **hash join**).
2. **Cost Estimation**: For each plan, the system estimates the **cost**. Cost depends on factors like how much data is processed, how much memory or CPU is used, etc.

3. **Selecting the Best Plan**: The plan with the **lowest cost** is selected. This is the most **efficient way** to run the query.
4. **Execution**: The database then runs the query based on the selected plan.

37. Explain the External Sort Merge Algorithm with an example

Ans:-

The **External Sort Merge** algorithm is used to sort large amounts of data that don't fit into memory (RAM). It works by dividing the data into smaller manageable chunks that can fit in memory, sorting those chunks, and then merging them together to get the final sorted result.

Here's an easy breakdown of how it works:

Steps:

1. **Divide Data**: Break the large data into smaller chunks that fit in memory. These are called "runs."
2. **Sort Chunks**: Load each chunk into memory, sort it using a simple sorting algorithm (like Merge Sort or Quick Sort), and then write the sorted chunk back to the disk.
3. **Merge Sorted Chunks**: Once all chunks are sorted, merge them together in multiple passes to get the final sorted data. This is done by loading parts of each sorted chunk, merging them in memory, and writing the merged result back to disk.

Example:

Suppose you have a file of **10 GB**, but your memory can only handle **1 GB** at a time.

- **Step 1: Divide** the 10 GB file into 10 chunks of 1 GB each.
- **Step 2: Sort** each chunk of 1 GB individually in memory. After sorting, write them back to disk.
- **Step 3: Merge** these sorted chunks together in passes. You merge parts of the sorted chunks in memory and write the merged result back to disk, continuing until all the chunks are fully merged into a single sorted file.

Why is this important?

- **External Sort Merge** is useful when data is too large to fit into memory.
- It ensures efficient sorting by breaking down the problem and handling it piece by piece.

38. Explain how hashing contributes to efficient data retrieval.

Ans:-

Hashing and Efficient Data Retrieval

Hashing is a technique used to quickly locate and retrieve data from a database or data structure.

1. **Hash Function: A hash function converts a data item (like a name or ID) into a fixed-size value, called a hash code or hash value. This value typically appears random but is consistent for the same input.**
2. **Hash Table: The hash code is used to determine the index (or position) in a hash table, where the data is stored.**
3. **Direct Access: When you need to retrieve data, you use the hash function to compute the hash code of the key. This tells you the exact position in the hash table where the data is stored, making retrieval very fast.**
4. **Efficiency: By using hashing, you reduce the time it takes to find data. Instead of searching through a list or database, you jump straight to the location where the data should be. This speeds up data retrieval, especially in large datasets.**

Unit – 7

39. Describe how the UNIQUE constraint differs from the PRIMARY KEY constraint.

Ans:-

UNIQUE:

- **You can apply it to multiple columns in a table.**
- **It allows one NULL value per column.**
- **Can be used to enforce uniqueness for specific columns but isn't necessarily used for identifying rows.**
- **Ensures all values in a column are different, but it allows one NULL value. You can have multiple UNIQUE constraints in a table.**

PRIMARY KEY:

- **It's the main identifier for each row in a table.**
- **Can only be used on one column or a set of combined columns (composite key).**
- **Automatically creates a unique index to speed up lookups.**
- **Combines both UNIQUE and NOT NULL, so it ensures all values are unique and no NULL values. A table can only have one PRIMARY KEY.**

40. Apply different SQL aggregate functions with an example.

Ans:-

- **COUNT(): Counts the number of rows.**

- **SUM(): Adds up values.**
- **AVG(): Calculates the average.**
- **MAX(): Finds the maximum value.**
- **MIN(): Finds the minimum value.**

Here's an example using a table named Sales:

ID	Product	Quantity	Price
1	Pen	10	5
2	Pencil	15	3
3	Eraser	8	2
4	Book	5	20
5	Ruler	12	4

SQL Queries using Aggregate Functions:

1. **COUNT():**

SELECT COUNT(*) FROM Sales;

This counts the total number of rows.

2. **SUM():**

SELECT SUM(Quantity) FROM Sales;

This adds the total quantity of products.

3. **AVG():**

SELECT AVG(Price) FROM Sales;

This calculates the average price of products.

4. **MAX():**

SELECT MAX(Price) FROM Sales;

This finds the highest product price.

5. **MIN():**

SELECT MIN(Price) FROM Sales;

This finds the lowest product price.

Summary:-

- **SQL aggregate functions like COUNT(), SUM(), AVG(), MAX(), and MIN() help in summarizing data from a table.**
- **Use COUNT() to find how many rows are there, SUM() to get total values, AVG() to find the average, MAX() for the highest value, and MIN() for the lowest value.**
- **These functions are useful for analyzing data and generating reports.**

41. Same Answer mentioned above..

42. Same Answer mentioned in 39..

43. How does the CHECK constraint help maintain data integrity? Explain with an example

Ans:-

The CHECK constraint helps maintain data integrity by ensuring that only valid data is entered into a table based on specified conditions. It restricts the values that can be inserted into a column.

Simple Example:

Imagine you have a table for storing information about employees, and you want to ensure that the age of an employee is always between 18 and 60. You can use the CHECK constraint to enforce this rule.

Code:-

```
CREATE TABLE Employees (
    EmployeeID INT,
    Name VARCHAR(50),
```

```
    Age INT,
    CHECK (Age BETWEEN 18 AND 60)
);
```

How it works:

- **If someone tries to insert a value for Age outside the range of 18 to 60, the database will reject the entry.**
- **This keeps the data valid and ensures only meaningful age values are stored.**

By using the CHECK constraint, you maintain data integrity by controlling what can or can't be added to the database.

44. What are primary and foreign keys in a database table?

Ans:-

A primary key is a unique identifier for each record (row) in a database table.

It ensures that no two rows have the same value in that column, and it can't be left empty (null).

A foreign key is a column in one table that links to the primary key in another table.

It helps maintain relationships between tables and ensures that the value in the foreign key column must match an existing value in the linked table's primary key.

Example:

-Primary key: Student ID in a "Students" table.

- Foreign key: Student ID in an "Enrollments" table that links to the "Students" table.

This way, the database keeps records connected and organized.

45 Write SQL statements (Query) for the following : 1. Create table Employee (emp_no, emp_name, emp_sal, emp_comm, dept_no). 2. Insert a minimum of 8 values in the table Employee 3. display table Employee.

Ans:-

1.Create Table

- **emp_no: Employee number (integer).**
- **emp_name: Employee name (string up to 50 characters).**
- **emp_sal: Employee salary (decimal number with 2 decimal places).**
- **emp_comm: Employee commission (decimal number with 2 decimal places).**
- **dept_no: Department number (integer).**

Code:-

```
CREATE TABLE Employee (
   emp_no INT,
   emp_name VARCHAR(50),
   emp_sal DECIMAL(10, 2),
   emp_comm DECIMAL(10, 2),
   dept_no INT
);
```

2.Insert Values

Code:-

```
INSERT INTO Employee (emp_no, emp_name, emp_sal, emp_comm, dept_no) VALUES
(1, 'Alice', 50000.00, 5000.00, 101),
(2, 'Bob', 55000.00, 5500.00, 102),
(3, 'Charlie', 60000.00, 6000.00, 103),
(4, 'David', 65000.00, 6500.00, 104),
(5, 'Eva', 70000.00, 7000.00, 105),
(6, 'Frank', 75000.00, 7500.00, 106),
(7, 'Grace', 80000.00, 8000.00, 107),
(8, 'Hank', 85000.00, 8500.00, 108);
```

Insert 8 rows with sample employee data.

3.Display Table

Code:-

SELECT * FROM Employee;

This statement will show all columns and rows from the Employee table.

These statements cover creating a table, inserting data, and displaying the table contents.