DBMS Question Bank PT-1

Q.1) Give the Drawbacks of File Processing System.

Ans: mnemonic: DID CAD

i. **Data redundancy and inconsistency.**
   - Multiple file formats, duplication of information in different files. Data redundancy means the same information is repeated in several files.

ii. **Difficulty accessing data.**
   - Need to write a new program to carry out each new task.

iii. **Data isolation.**
   - Multiple files and formats. The data is scattered in various files with different formats.

iv. **Integrity problems.**
   - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly.
   - Hard to add new constraints or change existing ones

v. **Atomicity of updates.**
   - **Failures may leave the database in an inconsistent state with partial** updates carried out.
   - Example: Transfer of funds from one account to another should either complete or not happen at all.

vi. **Concurrent access by multiple users.**
   - Concurrent access is needed for performance.
   - Uncontrolled concurrent access can lead to inconsistencies.
   - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time.

Q.2) Define the following terms:

DBMS, Database, query, tables, records, key, candidate key, foreign key, primary key, Query, DDL, DML.

Ans:

1) DBMS

DBMS (Database Management System provides an interface for users and application to interact with the data stored in a database. The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users.

2) Database

A structured collection of data, that is organized and stored in a computer system. Databases are designed to mange and manipulate large sets of data efficiency.

3) Query

A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language.

4) Tables

In a relational table database, data is organized onto tables. We use small tables in the next to illustrate concepts. A table is a collection of multiple rows and columns, where each row represents a record, and each column represents a field or attribute which has a unique name.

5) Records

Records are individual entries in a database table, also known as rows or tuples. A record represents a complete set of information about a particular entity.

6) Key

A key is a property of the entire relation, rather than of the individual tuples. The designation of a key represents a constraint in the real-worlds enterprise being modeled.

An alternative is to use some unique combination of other attributes as a key

7) Types of keys –

a. Candidate key

- Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table.
- It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table.
- There can be more than one candidate key for a table.
- A candidate key can never be NULL or empty. And its value should be unique.
- A candidate key can be a combination of more than one columns (attributes).

b. Foreign key

- A foreign key is an attribute value in a table that acts as a primary key.
- Foreign keys are the column of the table which is used to point to the primary key of another table.

c. Primary key

- A column or group of columns in a table that uniquely identify every row in that table.
- The Primary Key can't be a duplicate, meaning the same value can't appear more than once in the table.
- A table cannot have more than one primary key.

8) DDL

Data Definition Language (DDL) is a language for specifying the database schema and as well as other properties of the data. These are the commands used to create the objects like tables, indexes in the database for the first time. In other words, they create structure of the database.

9) DML

Data-manipulation language (DML) is a language that enables users to access or manipulate data. The SQL DML provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.

Q.3) Give the advantages of DBMS over file processing systems.

Ans:

A Database Management System (DBMS) offers significant advantages over traditional file processing systems, primarily by centralizing data and implementing robust control.

Key Benefits of a DBMS:

- **Reduced Data Redundancy and Inconsistency:** Unlike file systems where information can be duplicated across multiple files, a DBMS stores data centrally. This minimizes redundancy and eliminates inconsistencies that often arise from having several copies of the same data.
- **Improved Data Access:** A DBMS provides user-friendly query languages, allowing users to access and query data efficiently without needing to write new programs for each task, as is often required in file systems.
- **Enhanced Data Integration:** Data in file systems is frequently isolated and scattered in various formats. A DBMS, however, offers a unified, logical view of the data, simplifying its management and access across different parts of the database.
- **Greater Data Integrity:** In file systems, integrity constraints (business rules) are embedded in program code, making them difficult to modify. A DBMS allows these constraints to be defined directly on the data, ensuring consistent and centralized data integrity.

- **Ensured Atomicity:** A DBMS guarantees that transactions are atomic. This means a series of operations, such as a fund transfer, is either completed entirely or not at all, preventing inconsistent data states that can occur from partial updates during system failures in file systems.
- **Controlled Concurrent Access:** File systems lack control over concurrent access, potentially leading to data inconsistencies when multiple users access and update the same data simultaneously. A DBMS incorporates concurrency control mechanisms to prevent such interference between simultaneous transactions.
- **Improved Security:** While challenging to manage differential access rights in file systems, a DBMS offers strong security features. Administrators can grant specific access rights to different users for various parts of the database, ensuring data security.

Q.4) Describe the three levels of data abstraction.

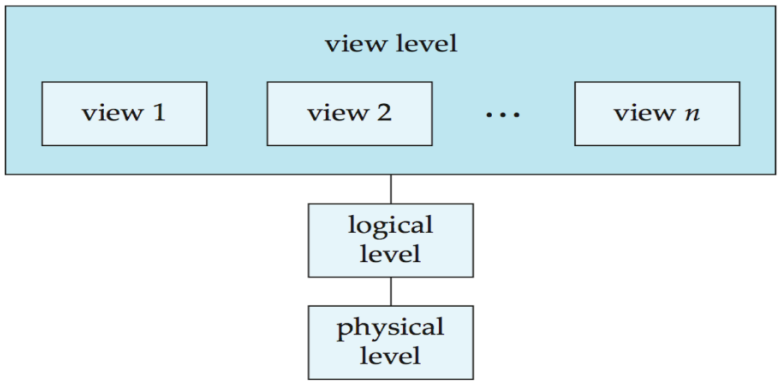
Ans:

i. **Physical Level:** describes how a record (e.g., instructor) is stored.
ii. **Logical Level:** describes data stored in a database, and the relationships among the data.

type *instructor* = **record**
ID: string;
n*ame*: string;
dept_name: string;
salary: integer;
***e*nd;**

iii. **View Level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

An architecture for a database system



Q.5) Describe Functional dependency in DBMS.

Ans:

Q.6) Explain database instance & database schema with suitable examples.

Ans:

Database instance: A **database instance** is a **snapshot** of the data in the database at a specific moment in time. It's the actual data that resides in the database according to the structure defined by the schema.

Database schema: A database schema is the blueprint of a database. It defines the database's structure, including the tables, the columns within each table (along with their data types and constraints), and the relationships between tables.

Example of Database schema:

possible schema for two tables, `Books` and `Borrowers`:

`Books` Table Schema:

- `BookID` (Integer, Primary Key, Auto-increment)
- `Title` (Text, Not Null)
- `Author` (Text, Not Null)
- `PublishedYear` (Integer)
- `ISBN` (Text, Unique)

`Borrowers` Table Schema:

- `BorrowerID` (Integer, Primary Key, Auto-increment)
- `FirstName` (Text, Not Null)
- `LastName` (Text, Not Null)
- `MembershipDate` (Date)

Q.7) Explain various data models in DBMS.

Ans:

1. E-R Model

- The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects
- Entity: is a ‘thing’ or ‘object’ in the real world which is distinguishable from other objects. Examples: person, student, doctor, etc.
- Relationship: is an association among several entities. Example: A depositor relationship associates a customer with each account that he has.

2. Hierarchical Model
   - The hierarchical DBMS is used to model one-to-many relationships, presenting data to users in a treelike structure.
   - Within each record, data elements are organized into pieces of records called segments.
   - Top level segment is called the root. An upper segment is connected logically to a lower segment in a parent–child relationship.
   - A parent segment can have more than one child, but a child can have only one parent.
3. Network Model
   - The network DBMS depicts data logically as many-to-many relationships. That is, parents can have multiple children, and a child can have more than one parent.
   - Example: There are many courses in a university and many students. A student takes many courses, and a course has many students.
   - Banks, insurance companies, etc. Are using this legacy data model.
   - Can handle 1:n as well as n:n relationships.

Q.8)

Q.9) Describe the role or functions of a Database administrator.

Ans:

The person to have central control of both the data and the programs that access those data is called a database administrator (DBA).

The functions of a DBA include:

- Schema definition. The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- Storage structure and access-method definition.
- Schema and physical-organization modification.
- Granting authorization for data access: It can regulate which part of the database various users can access.

- Routine maintenance: database backup, ensuring free disk space, monitoring of jobs running on the database

Q.10) Describe diverse types of database users.

Ans:

1. Database Administrator (DBA)
DBAs define schemas, manage architecture, create user accounts, control access, ensure security, handle breaches, optimize performance, monitor recovery/backup, provide support, and repair damage. They possess DCL privileges (GRANT/REVOKE) and utilize a crucial "superuser" account.

2. Naive/ End Users
These unsophisticated users, lacking DBMS knowledge, frequently use database applications for daily tasks (e.g., railway ticket booking, bank clerks).

3. A System Analyst
System Analysts analyze parametric end-user requirements for satisfaction.

4. Sophisticated Users
Engineers and analysts interact directly with databases via SQL queries, developing custom applications without writing full code.

5. Database Designers
Database Designers create the database structure (tables, views, indexes, triggers, constraints) based on user requirements before data population.

6. Application Programmers

Also known as System Analysts or Software Engineers, these programmers write backend code (C, Visual Basic, COBOL) for "canned transactions" that help naive users interact with the database. They design, test, and maintain these programs.

Q.11)Explain several types of constraints in DBMS with examples.

Ans:

Q.12)Describe various DDL and DML commands.

Ans:

Data Definition Language (DDL)

DDL commands define and manage database structure and objects (tables, schemas, indexes). DBAs typically use DDL for initial schema creation, dealing with metadata and data storage definitions.

Common DDL commands:

- **CREATE**: Creates new database objects.
    - **CREATE DATABASE**: New database.
    - **CREATE TABLE**: New table.
    - **CREATE INDEX**: Index for query performance.
    - **CREATE VIEW**: Virtual table from SQL statement.
- **ALTER**: Modifies existing database object structure.
    - **ALTER TABLE**: Adds, deletes, or modifies table columns.
- **DROP**: Deletes existing database objects.
    - **DROP TABLE**: Deletes table and data.
    - **DROP INDEX**: Deletes index.
- **TRUNCATE**: Removes all table records quickly, structure remains.
- **RENAME**: Renames database objects.

Data Manipulation Language (DML)

DML commands manage and manipulate data within database objects (inserting, updating, deleting, retrieving).

Common DML commands:

- **SELECT**: Retrieves data (most common).
- **INSERT**: Adds new rows.
- **UPDATE**: Modifies existing records.
- **DELETE**: Removes existing records.

Q.13)Provide examples for each of the following relationship types:

a. Many-to-One

b. One-to-One

c. One-to-Many

d. Many-to-Many

Ans:

Here are examples for each of the specified database relationship types:

As a text-based AI, I can't generate diagrams directly, but I can describe how each relationship type would be represented using standard notation:**a. Many-to-One**

- **Tables:** "Students" (`student_id`, `name`, `major_id`) and "Majors" (`major_id`, `major_name`).
- **Relationship:** `major_id` links "Students" to "Majors".
- **Cardinality:** "Students" side has "many" (crow's foot) to "Majors" side's "one" (single line), meaning one major has many students.

b. One-to-One

- **Tables:** "Persons" (`person_id`, `name`) and "Passports" (`passport_id`, `person_id`, `expiration_date`).
- **Relationship:** `person_id` links "Persons" to "Passports".
- **Cardinality:** Both sides have "one" (single line), meaning one person has one passport.

c. One-to-Many

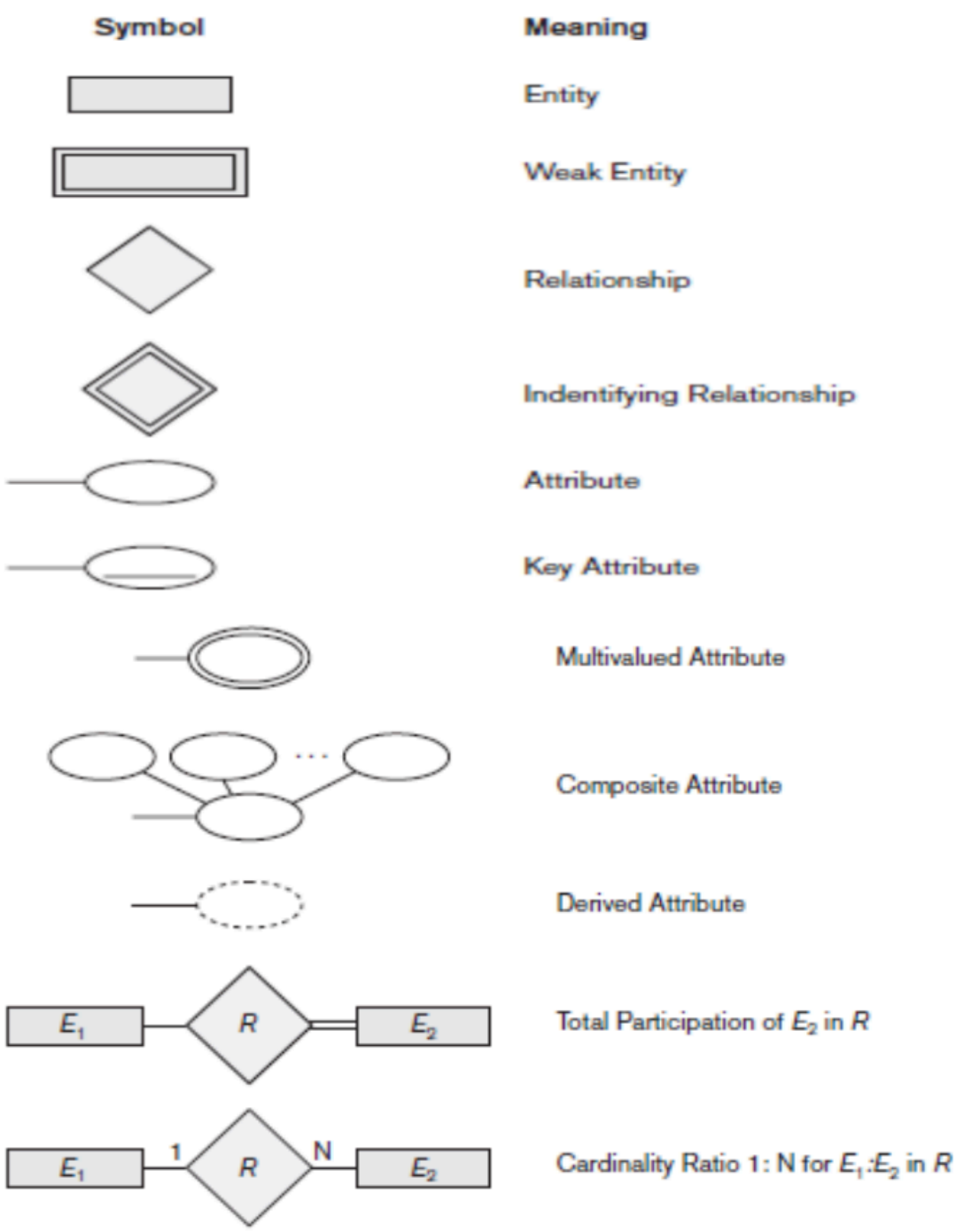
- **Tables:** "Customers" (`customer_id`, `name`) and "Orders" (`order_id`, `customer_id`, `order_date`).
- **Relationship:** `customer_id` links "Customers" to "Orders".
- **Cardinality:** "Customers" side has "one" (single line) to "Orders" side's "many" (crow's foot), meaning one customer can place many orders.

d. Many-to-Many

- **Tables:** "Students" (`student_id`, `name`), "Courses" (`course_id`, `course_name`), and "Enrollments" (join table: `student_id`, `course_id`).
- **Relationship:** `student_id` links "Students" to "Enrollments"; `course_id` links "Courses" to "Enrollments".
- **Cardinality:**
    - "Students" to "Enrollments": "one" on "Students" side, "many" on "Enrollments" side.
    - "Courses" to "Enrollments": "one" on "Courses" side, "many" on "Enrollments" side.
    - This setup allows one student to enroll in many courses, and one course to have many students.

Q.14)Draw and label all symbols of ER diagram.

Ans:



Q.15)Explain Primary key and foreign key concepts in DBMS with examples.

Ans:

Primary Key

PRIMARY KEY is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.
Rules for defining Primary key:

- Two rows can't have the same primary key value.
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Examples:

- In a `StudentTable`, `StudentID` is an ideal primary key because it uniquely identifies each student.
- In a `Results` table where a single student can have multiple subject entries, `StudentID` alone is not unique. A **composite primary key**, combining `StudentID` and `Subject`, is used to uniquely identify each row.

Foreign Key

- A foreign key is an attribute value in a table that acts as the primary key in another table. Hence, the foreign key is useful in linking together two tables. Data should be entered in the foreign key column with great care, as wrongly entered data can invalidate the relationship between the two tables.
- Foreign keys are the column of the table which is used to point to the primary key of another table.
- In an institute, every teacher works in a specific department, and teacher and department are two different entities. So we can't store the information of the department in the teacher table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the Department table, DeptCode as a new attribute in the Teacher table.

Examples:

After normalizing initial data, consider the following tables: `StudentTable`, `HouseTable`, `SubjectsTable`, and `GradesTable`.

- The `HouseName` column in `StudentTable` acts as a **foreign key**, linking to the `HouseName` **primary key** in `HouseTable`. This establishes a one-to-many relationship where multiple students can belong to a single house.
- Similarly, in the `GradesTable`, `StudentID` is a foreign key linking to the `StudentTable`, and `Subject` is a foreign key linking to the `SubjectsTable`.

PRIMARY KEY FOREIGN KEY

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

Q.16)Draw an ER diagram for the given scenario.

Ans:

Q.17)Explain Super key and Candidate key concepts in DBMS with examples.

Ans:

I. Super key:
Super key is defined as a set of attributes within a table that can uniquely identify each record in a table.

II. Candidate key:
Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.

- A candidate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns (attributes).

Q.19)Explain given DDL/DML command with suitable example.
Ans:

Q.20)Explain various clauses with examples.
Ans:

1. `SELECT` Clause

- **Purpose:** Specifies the columns you want to retrieve in the result set. You can use `*` to select all columns.
- **Example**

```
SELECT FirstName, LastName, Salary
FROM Employees;
```

- **Result:** This query will return only the `FirstName`, `LastName`, and `Salary` columns for all employees.

2. `FROM` Clause

- **Purpose:** Specifies the table from which to retrieve the data. Every `SELECT` statement must have a `FROM` clause.
- **Example:** (Same as above, `FROM` is essential)

```
SELECT *

FROM Employees;
```

- **Result:** This query selects all columns (`*`) from the `Employees` table.

3. `WHERE` Clause

- **Purpose:** Filters records from a table based on a specified condition. It is used to extract only those records that fulfill a specific criterion.
- **Example:**

```
SELECT FirstName, Salary

FROM Employees

WHERE DepartmentID = 2;
```

- **Result:** This will return the names and salaries of employees who work in Department 2 (Sales).

4. `GROUP BY` Clause

- **Purpose:** Groups rows that have the same values in specified columns into summary rows. It is often used with aggregate functions like `COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()` to perform calculations on each group.
- **Example:**

```
SELECT

    DepartmentID,

    COUNT(EmployeeID) AS NumberOfEmployees,

    AVG(Salary) AS AverageSalary

FROM Employees

GROUP BY DepartmentID;
```

- **Result:** This query counts the number of employees and calculates the average salary for each department.

5. `HAVING` Clause

- **Purpose:** Filters the results of a `GROUP BY` clause. While `WHERE` filters rows *before* they are grouped, `HAVING` filters the groups *after* they have been created.
- **Example:**

```
SELECT
    DepartmentID,
    AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY DepartmentID
HAVING AVG(Salary) > 70000;
```

- **Result:** This query first calculates the average salary per department and then shows only those departments where the average salary is greater than $70,000.

6. `ORDER BY` Clause

- **Purpose:** Sorts the result set in ascending (`ASC`) or descending (`DESC`) order based on one or more columns. The default is `ASC`.
- **Example:**

```
SELECT FirstName, LastName, Salary
FROM Employees
ORDER BY Salary DESC;
```

- **Result:** This lists all employees, starting with the highest-paid and ending with the lowest-paid.

Q.21)Write SQL query for given database.

Ans:

Q.22)Explain various datatypes in SQL.

Ans:

DATA TYPE	DESCRIPTION
NUMBER	40 Digits with decimal and minus sign.
NUMBER(W)	Number of digits specified with decimal and minus sign.
NUMBER(W,D)	Number of digits, number of decimal digits specified with minus sign.
CHAR(W)	Fixed length character data. Maximum size is 2000 (bytes) and default is 1 (byte).
VARCHAR(W)	This variable is pre-reserved and maybe used by Oracle in later future, we should not use it now.
VARCHAR2(W)	String data of specified width. Maximum width is 4000 (bytes).
DATE	Date with default format 26-JAN-14.
LONG	String data of length 0-2 gigabytes. Only one LONG column per table. Can't be used with function, expression or WHERE clause.
RAW(W)	Binary data of specified length. Maximum width is 2000 (bytes).
LONG ROW	Binary data of length 0-2 gigabytes. Only one LONG ROW column per table.
CLOB	A character large object. Maximum size is 4 gigabytes.
BLOB	A binary large object. Maximum size is 4 gigabytes.
BFILE	Contains a locator to a large binary file stored outside the database. Maximum size is 4 gigabytes.

Q.23)Explain DCL and TCL commands.

Ans:

I. DCL (Data Control Language)

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

GRANT: This command gives users access privileges to the database.

Syntax –

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER,ANOTHER_USER;
```

REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

Syntax –

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

II. TCL (Transaction Control Language)

Transactions combine tasks into a single unit. A transaction succeeds if all tasks complete, otherwise it fails. Thus, transactions have two outcomes: success or failure. TCL commands manage transaction execution.

COMMIT: Commits a Transaction.

Syntax –

```
COMMIT;
```

ROLLBACK: Rollbacks a transaction in case of any error occurs.

Syntax –

```
ROLLBACK;
```

ROLLBACK TO: Rolls back only to a specific savepoint.

Syntax —
`ROLLBACK TO sp1;`
SAVEPOINT: Sets a save point within a transaction.
Syntax –
`SAVEPOINT SAVEPOINT_NAME;`

Q.23)Write the query using relational algebra.
Ans:
Q.24)Elaborate decomposition with diagram
Ans:

In database design (especially in **Normalization**), **Decomposition** means breaking a complex relation (table) into two or more smaller relations **without losing data or meaning**.

The goal is to:

- Remove **data redundancy** (repeated data).
- Eliminate **anomalies** (insertion, update, deletion problems).
- Maintain **data integrity**.

Types of Decomposition

1. **Lossless (Non-loss) Decomposition** ✅
    - No information is lost after decomposing and joining back.
    - **Join** of decomposed tables should give the original table.
    - Achieved when **common attributes** are a key in at least one table.
2. **Lossy Decomposition** ❌
    - Some information is lost after decomposing.
    - Joining decomposed tables may produce extra or missing tuples.

Example

We have a table:

Student(RollNo, Name, Dept, DeptLocation)

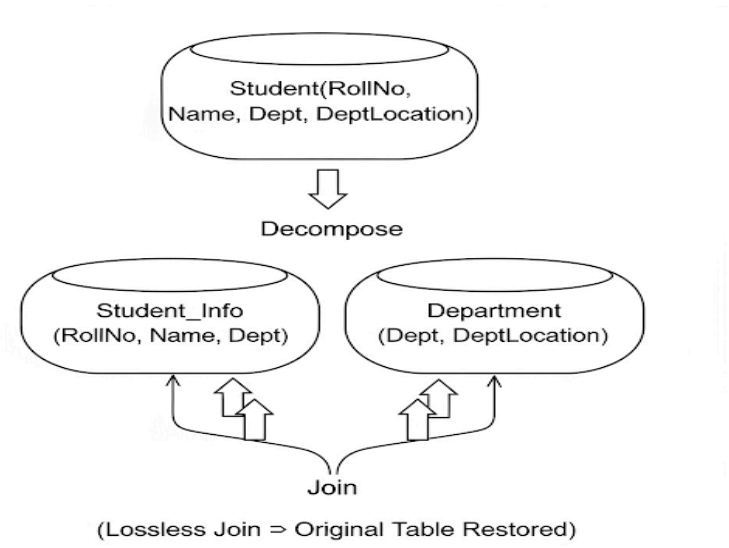
Problems:

- Dept info is repeated for every student in the same department.
- If we delete all students from a department, dept location is lost.

Decomposition Process

Student_Info(RollNo, Name, Dept)
Department(Dept, DeptLocation)

Diagram:



Q.25)Explain various anomalies in the database with example.
Ans:
Database anomalies occur when a database is not normalized (typically in 1NF) and contains redundant data. These anomalies can lead to data inconsistencies, unnecessary data storage, or the loss of important information during database operations.

The three primary types of anomalies are:
1. Insertion Anomaly
Meaning: This anomaly prevents the insertion of new data into a table without the presence of unrelated data.

Example: Consider a table with `StudentID`, `StudentName`, `Course`, and `Instructor` columns. If you wish to add a new course ("Java") but no student has yet enrolled in it, you cannot add the course without creating a fictitious student record.

2. Update Anomaly
Meaning: This anomaly arises when the same data is stored in multiple locations, and updating one instance without updating all others results in data inconsistency.

Example: In a table containing `StudentID`, `StudentName`, `Course`, and `Instructor`, imagine "Prof. A" teaches "DBMS" to both John and Mike. If Prof. A's name changes to "Prof. B," every row where "Prof. A" appears must be updated. Forgetting to update even one row will cause data inconsistency.
3. Delet ion Anomaly

Meaning: This anomaly occurs when deleting a record inadvertently removes valuable information that should have been retained.

Example: In a table with `StudentID`, `StudentName`, `Course`, and `Instructor`, if the last student enrolled in "Java" (taught by Prof. C) drops out and their record is deleted, the information that "Java" was taught by Prof. C is also lost.Summary of Anomalies

Anomaly	Problem	Example Consequence
Insertion	Inability to add data without unrelated data	Cannot add a new course without a student
Update	Data inconsistency if not updated everywhere	Instructor name mismatches
Deletion	Loss of important information upon data deletion	Loss of course details if the last student is deleted