Q1. Give the Drawbacks of File Processing System (4/6 marks).

Answer:

The major drawbacks of File Processing System are:

1. **Data Redundancy & Inconsistency** – Same data may be stored in multiple files, causing duplication and mismatch.
2. **Difficulty in Accessing Data** – Retrieving specific data requires writing complex programs.
3. **Lack of Security** – Files do not provide controlled access, so sensitive data is not secure.
4. **Limited Data Sharing** – Data stored separately in files cannot be easily shared among multiple applications.
5. **Integrity Problems** – No constraints are enforced, so invalid or inconsistent data can be stored.
6. **Poor Concurrent Access** – Multiple users accessing the same data may cause conflicts or data loss.

Q2. Define the following terms: DBMS, Database, Query, Tables, Records, Key, Candidate Key, Foreign Key, Primary Key, DDL, DML.

1. **DBMS (Database Management System):**
   Software that manages data in a structured way and provides efficient storage, retrieval, and manipulation of data.
   Example: Oracle, MySQL, SQL Server.
2. **Database:**
   A collection of related data organized in a structured format for easy access, retrieval, and management.
3. **Query:**
   A request for data or information from a database written using query language (like SQL).
4. **Table:**
   A collection of rows and columns used to store data in a structured form inside a database.
5. **Record (or Tuple):**
   A single row in a table that represents one complete set of related data.
6. **Key:**
   An attribute (or set of attributes) used to uniquely identify records in a table.
7. **Candidate Key:**
   All attributes that can uniquely identify a record in a table. One of them is chosen as the **Primary Key**.
8. **Primary Key:**
   A unique attribute that identifies each record in a table. It cannot have duplicate or NULL values.
9. **Foreign Key:**
   An attribute in one table that refers to the **Primary Key** of another table to establish a relationship.
10. **DDL (Data Definition Language):**
    Part of SQL used to define and modify database structure.
    Commands: CREATE, ALTER, DROP.
11. **DML (Data Manipulation Language):**
    Part of SQL used to manipulate and manage data stored in tables.
    Commands: INSERT, UPDATE, DELETE, SELECT.

Q3. Give the advantages of DBMS over File Processing System (4/6 marks).

Answer:

Advantages of DBMS over File Processing System are:

1. **Reduced Data Redundancy** – DBMS stores data centrally, avoiding unnecessary duplication.
2. **Data Consistency** – Centralized control ensures data is uniform across applications.
3. **Improved Data Security** – Access to data is controlled using authorization and authentication.
4. **Better Data Sharing** – Multiple users can access the same data concurrently without conflict.
5. **Integrity Enforcement** – Constraints (Primary Key, Foreign Key, etc.) ensure accuracy of data.
6. **Backup and Recovery** – DBMS provides automatic backup and recovery features in case of system failure.

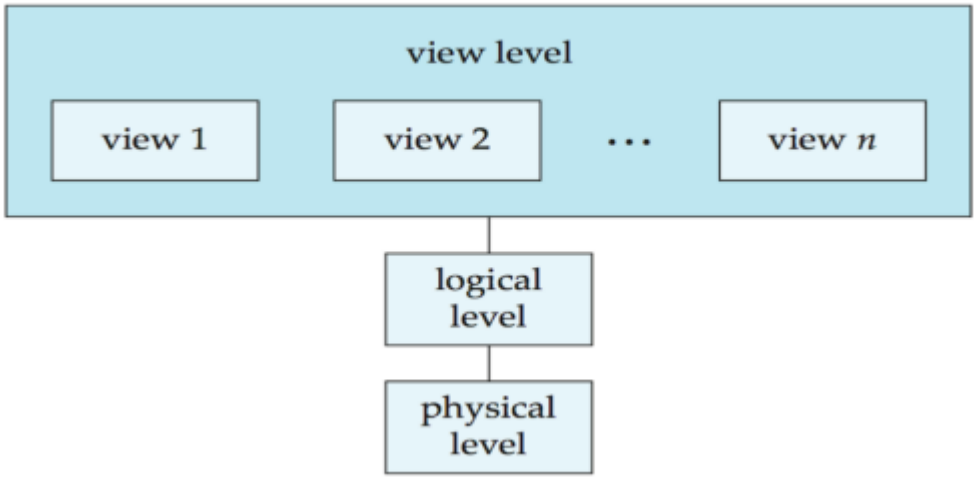
Q4. Describe the three levels of data abstraction (4/6 marks).

Answer:

In DBMS, data is viewed at **three levels of abstraction**:

1. **Physical Level (Lowest Level):**
   - Describes *how data is actually stored* in the database.
   - Deals with storage structures, indexes, and access methods.
   - Example: Data stored on hard disk using B-trees or hashing.
2. **Logical Level (Intermediate Level):**
   - Describes *what data is stored* in the database and the relationships among them.
   - Defines tables, attributes, and constraints without worrying about physical storage.
   - Example: A table "Student" with attributes Roll_No, Name, Class.
3. **View Level (Highest Level):**
   - Describes *only a part of the database* that a particular user needs.
   - Provides multiple views of the same database for different users.
   - Example: A librarian sees only issued books, while an accountant sees only fee records.

An architecture for a database system



Q5. Describe Functional Dependency in DBMS (4/6 marks).

Answer:

- A **Functional Dependency (FD)** exists when the value of one attribute uniquely determines the value of another attribute in a relation.
- It is denoted as **X → Y**, meaning attribute Y is functionally dependent on attribute X.

Example:

In a **Student** table (Stu_Id, Stu_Name, Stu_Age):

```
Stu_Id → Stu_Name
```

If we know Stu_Id, we can uniquely determine Stu_Name.

Types of Functional Dependencies:

1. **Trivial Functional Dependency:**
   - If Y is a subset of X in **X → Y**.
   - Example: {Student_Id, Student_Name} → Student_Id.
2. **Non-Trivial Functional Dependency:**
   - If Y is not a subset of X in **X → Y**.
   - Example: emp_id → emp_name.
3. **Completely Non-Trivial FD:**
   - If X ∩ Y = ∅ in **X → Y**.
   - Example: emp_id → emp_address.
4. **Transitive Dependency:**
   - If X → Y and Y → Z, then X → Z holds.
   - Example: Book → Author, Author → Author_Age ⇒ Book → Author_Age.
5. **Multivalued Dependency:**
   - Exists when one attribute determines multiple independent attributes.
   - Example: bike_model →> manuf_year and bike_model →> color.

Q6. Explain Database Instance & Database Schema with suitable example.

Answer:

1. **Database Schema:**
   - The **logical design/structure** of the database.
   - It defines how data is organized and the relationships among them.
   - Describes tables, attributes, keys, constraints, and relationships.
   - **It is fixed (rarely changes).**

   Example:

```pgsql
Student (Roll_No, Name, Class)
```

   → This definition of table is part of the schema.

---

2. **Database Instance:**
   - The **snapshot of data stored** in the database at a particular moment of time.
   - Represents the actual content (records) in the tables.
   - **It changes frequently** as data is inserted, deleted, or updated.

   Example:

```pgsql
Roll_No   Name      Class
1         Ramesh    TYCO
2         Priya     SYIT
3         Aarav     FYME
```

   → These rows are the instance of the **Student** table at that moment.

Q7. Explain various Data Models in DBMS (6 marks).

Answer:

A **Data Model** is a way of organizing and describing data, its relationships, and constraints in a datab

The main types of data models in DBMS are:

1. **Hierarchical Data Model:**
   - Data is organized in a **tree structure** with parent–child relationships.
   - Each parent can have multiple children, but each child has only one parent.
   - Example: Organization chart (Company → Departments → Employees).
2. **Network Data Model:**
   - Data is represented as **records connected by links (pointers).**
   - Supports many-to-many relationships.
   - Example: Students enrolled in multiple courses.
3. **Relational Data Model (Most popular):**
   - Data is stored in **tables (relations)** consisting of rows (tuples) and columns (attributes).
   - Provides high-level query language (SQL).
   - Example: Student(Roll_No, Name, Class).
4. **Entity-Relationship (E-R) Model:**
   - Represents real-world entities and relationships between them.
   - Uses diagrams with symbols like rectangles (entities), diamonds (relationships), and ovals (attributes).
   - Example: Student –[Enrolled]–> Course.
5. **Object-Oriented Data Model:**
   - Extends relational model with concepts of object-oriented programming (objects, classes, inheritance).
   - Example: Multimedia database (storing images, videos as objects).

Q.8 Explain Database System Architecture.

Answer:

Database system architecture explains how data in a database is stored, organized, and accessed. It provides a layered approach so that users do not need to know the details of physical storage. According to the ANSI/SPARC model, the architecture is divided into three levels:

1. **External Level (View Level):** This is the highest level, closest to the users. It provides different views of the same database according to the requirements of various users. For example, a student may only see his grades, while the teacher sees the results of the whole class.
2. **Conceptual Level (Logical Level):** This level represents the logical structure of the entire database for the community of users. It defines what data is stored, the relationships among them, and constraints, while hiding the physical details of storage.
3. **Internal Level (Physical Level):** This is the lowest level and deals with how the data is physically stored in memory, files, indexes, and access paths.

The main purpose of this architecture is to achieve **data abstraction** and **data independence**, so changes in one level do not affect other levels.

Q.9 Describe the role or functions of Database Administrator (DBA).

Answer:

A Database Administrator (DBA) is responsible for the overall management of the database system. The DBA ensures that the database is available, secure, and performing efficiently. The main functions of a DBA are:

1. **Database Design:** Decides the structure of the database including tables, relationships, and constraints.
2. **Storage Management:** Manages how data is stored on disk and optimizes space utilization.
3. **User Administration:** Creates user accounts, assigns privileges, and ensures data security.
4. **Backup and Recovery:** Maintains regular backups and restores data in case of failures.
5. **Performance Monitoring:** Monitors query execution and optimizes database performance.
6. **Integrity and Security:** Enforces rules, constraints, and access controls to maintain data consistency and security.

Thus, the DBA plays a crucial role in maintaining the reliability, efficiency, and security of a database system.

Q.10 Describe diverse types of database users.

Answer:

Different types of users interact with a database system, each with specific roles and requirements. The main types of database users are:

1. **Database Administrators (DBA):** Responsible for managing the overall database system, including storage, security, backup, recovery, and performance.
2. **Application Programmers:** Write programs and applications that interact with the database using query languages like SQL.
3. **End Users:** Use the database through applications to perform their tasks. They are of three types:
   - **Casual Users:** Occasionally access the database with simple queries.
   - **Naïve or Parametric Users:** Regularly use pre-defined queries and forms, e.g., bank clerks.
   - **Sophisticated Users:** Write complex queries and interact directly using SQL.
4. **System Analysts and Designers:** Design the structure of the database and develop application requirements for users.

Thus, database users can range from technical experts (DBA, programmers) to ordinary end users, each playing an important role in efficient database utilization.

Q.11 Explain several types of constraints in DBMS with example.

Answer:

Constraints in DBMS are rules applied on data in a table to maintain accuracy, consistency, and integrity. The major types of constraints are:

1. **NOT NULL Constraint:** Ensures that a column cannot have NULL values.
   Example: `Name VARCHAR(50) NOT NULL` → Name must always be entered.
2. **UNIQUE Constraint:** Ensures all values in a column are unique (no duplicates).
   Example: `Email VARCHAR(50) UNIQUE` → No two users can have the same email.
3. **PRIMARY KEY Constraint:** Combines **NOT NULL** and **UNIQUE**; uniquely identifies each row in a table.
   Example: `StudentID INT PRIMARY KEY` → Each student has a unique ID.
4. **FOREIGN KEY Constraint:** Creates a relationship between two tables by referencing the primary key of another table.
   Example: `CourseID INT REFERENCES Courses(CourseID)` → Ensures only valid CourseIDs are entered.
5. **CHECK Constraint:** Ensures values in a column satisfy a specific condition.
   Example: `Age INT CHECK (Age >= 18)` → Age must be 18 or above.
6. **DEFAULT Constraint:** Assigns a default value to a column if no value is given.
   Example: `City VARCHAR(30) DEFAULT 'Surat'` → If no city is entered, Surat is stored.

Q.12 Describe various DDL and DML commands.

Answer:

SQL (Structured Query Language) provides two main categories of commands: **DDL (Data Definition Language)** and **DML (Data Manipulation Language)**.

1. Data Definition Language (DDL)

DDL commands are used to define and manage the structure of the database (tables, schema).

- **CREATE:** Creates a new database object (table, view, index).
  Example: `CREATE TABLE Students (ID INT, Name VARCHAR(50));`
- **ALTER:** Modifies an existing database object.
  Example: `ALTER TABLE Students ADD Age INT;`
- **DROP:** Deletes a database object permanently.
  Example: `DROP TABLE Students;`
- **TRUNCATE:** Removes all rows from a table but keeps the structure.
  Example: `TRUNCATE TABLE Students;`

2. Data Manipulation Language (DML)

DML commands are used to manipulate and manage the data stored in the database.

- **INSERT:** Adds new records into a table.
  Example: `INSERT INTO Students VALUES (1, 'Amit', 20);`
- **SELECT:** Retrieves data from the database.
  Example: `SELECT * FROM Students;`
- **UPDATE:** Modifies existing data in a table.
  Example: `UPDATE Students SET Age = 21 WHERE ID = 1;`
- **DELETE:** Removes specific records from a table.
  Example: `DELETE FROM Students WHERE ID = 1;`

Q.13 Give example of following relationships:

a. Many-to-One:

Many entities in one table are related to a single entity in another table.

Example: Many employees work in one department.

```
scss

Employees → Department
(E1, E2, E3 → D1)
```

b. One-to-One:

A single entity in one table is related to exactly one entity in another table.

Example: Each student has one unique ID card.

```
scss

Student ↔ ID_Card
(S1 ↔ C1)
```

c. One-to-Many:

One entity in one table is related to multiple entities in another table.

Example: One teacher teaches many students.

```
scss

Teacher → Students
(T1 → S1, S2, S3)
```

d. Many-to-Many:

Multiple entities in one table are related to multiple entities in another table.

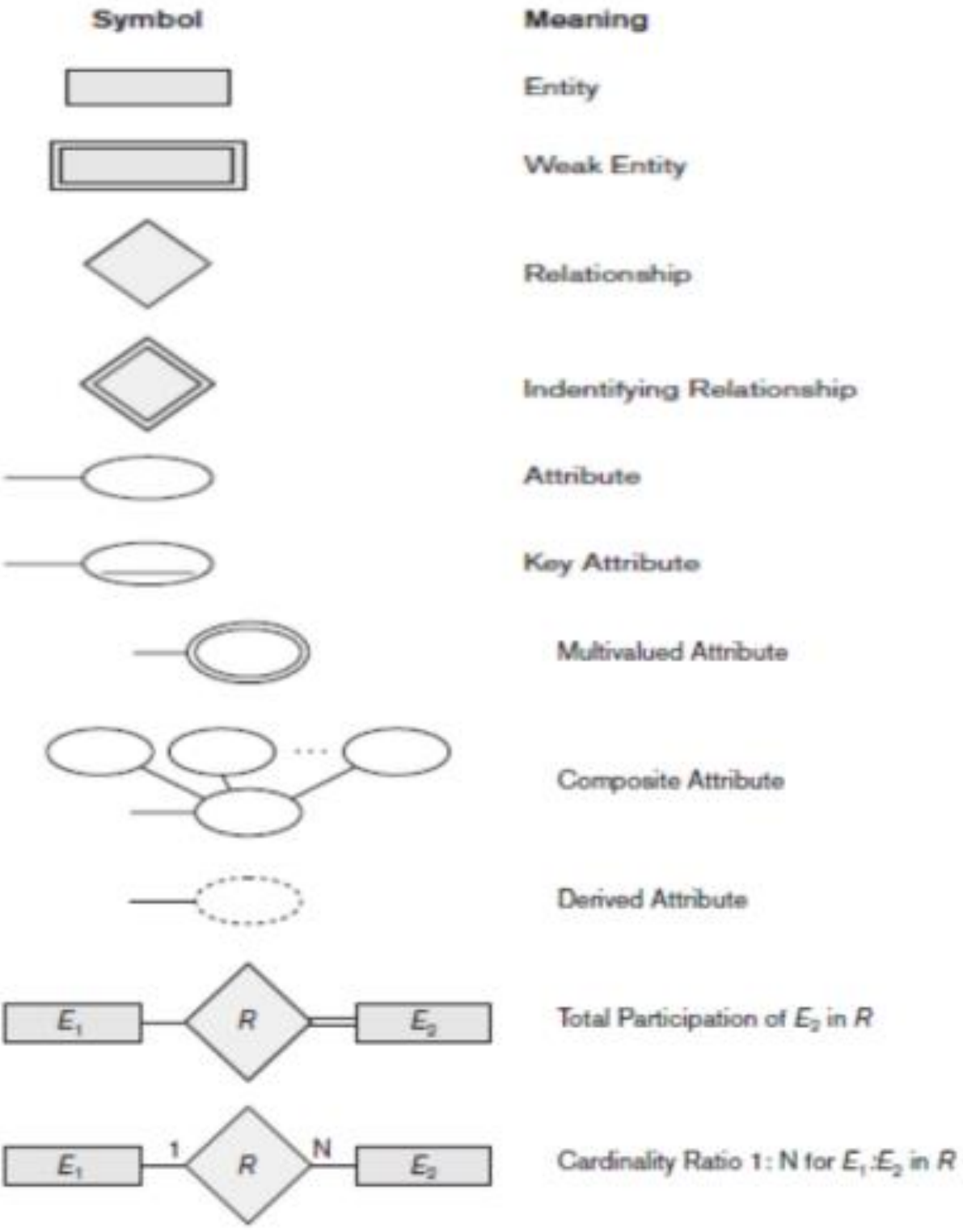
Example: Students enroll in many courses, and each course has many students.

```
scss

Students ↔ Courses
(S1, S2 ↔ C1, C2)
```

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

Ans:



Q.15 Explain Primary key and Foreign key concepts in DBMS with example.

Answer:

1. **Primary Key:**
   A primary key is a column (or set of columns) in a table that uniquely identifies each row.

- It cannot contain NULL values.
- It must have unique values.
- Each table can have only one primary key.

Example:

sql Copy Edit

```sql
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT
);
```

Here, `StudentID` is the primary key that uniquely identifies each student.

---

2. **Foreign Key:**
   A foreign key is a column (or set of columns) in one table that refers to the **primary key** of another table.

- It is used to establish a relationship between two tables.
- It ensures **referential integrity**, i.e., only valid data from the referenced table can be entered.

Example:

sql Copy Edit

```sql
CREATE TABLE Enrollments (
    EnrollID INT PRIMARY KEY,
    StudentID INT,
    CourseID INT,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
);
```

Here, `StudentID` in the **Enrollments** table is a foreign key that refers to `StudentID` in the **Students** table.

Q.16)Draw an ER diagram for the given scenario 6

Q.17 Explain Super Key and Candidate Key concepts in DBMS with example.

Answer:

1. **Super Key:**
   A super key is a set of one or more attributes that can uniquely identify each record in a table.
- It may include extra attributes not required for uniqueness.
- Every candidate key and primary key is a super key, but not every super key is a candidate key.

Example:
In a **Students** table with attributes `(StudentID, RollNo, Name)`:

- `(StudentID)`
- `(RollNo)`
- `(StudentID, Name)`
  All are **super keys** because they uniquely identify a student.

---

2. **Candidate Key:**
   A candidate key is a **minimal super key** – the smallest set of attributes that uniquely identify each record without redundancy.
- A table can have multiple candidate keys.
- From candidate keys, one is chosen as the **primary key**.

Example:
In the same **Students** table:

- `StudentID`
- `RollNo`
  Both are **candidate keys** because they are minimal and unique.

Q.18 Explain given DDL/DML command with suitable example.

Answer:

(A) DDL Commands

DDL is used to define and manage the structure of the database.

1. **CREATE:** Creates a new database object such as table, view, or index.
   Example:

```sql
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT
);
```

2. **ALTER:** Modifies the structure of an existing table.
   Example:

```sql
ALTER TABLE Students ADD Email VARCHAR(50);
```

3. **DROP:** Deletes an existing table or database object permanently.
   Example:

```sql
DROP TABLE Students;
```

4. **TRUNCATE:** Removes all rows from a table but keeps the table structure.
   Example:

```sql
TRUNCATE TABLE Students;
```

(B) DML Commands

DML is used to manipulate and manage the data stored in the database.

1. **INSERT:** Adds new records into a table.
   Example:

```sql
INSERT INTO Students VALUES (1, 'Amit', 20, 'amit@gmail.com');
```

2. **SELECT:** Retrieves data from a table.
   Example:

```sql
SELECT * FROM Students;
```

3. **UPDATE:** Modifies existing data in a table.
   Example:

```sql
UPDATE Students SET Age = 21 WHERE StudentID = 1;
```

4. **DELETE:** Removes specific records from a table.
   Example:

```sql
DELETE FROM Students WHERE StudentID = 1;
```

Q.19 Explain various clauses with suitable example.

Answer:

In SQL, clauses are used with commands to provide conditions and control the retrieval or modification of data. The main clauses are:

1. **WHERE Clause:** Used to filter records based on a condition.

   Example:

   sql Copy Edit

   ```sql
   SELECT * FROM Students WHERE Age > 18;
   ```

   → Displays only students older than 18.

2. **ORDER BY Clause:** Used to sort the result in ascending (ASC) or descending (DESC) order.

   Example:

   sql Copy Edit

   ```sql
   SELECT Name, Age FROM Students ORDER BY Age DESC;
   ```

   → Displays students sorted by age in descending order.

3. **GROUP BY Clause:** Used to group rows that have the same values in a column.

   Example:

   sql Copy Edit

   ```sql
   SELECT CourseID, COUNT(*) FROM Enrollments GROUP BY CourseID;
   ```

   → Shows number of students enrolled in each course.

4. **HAVING Clause:** Used to apply conditions on groups (with GROUP BY).

   Example:

   sql Copy Edit

   ```sql
   SELECT CourseID, COUNT(*)
   FROM Enrollments
   GROUP BY CourseID
   HAVING COUNT(*) > 10;
   ```

   → Displays only courses with more than 10 students.

Q.20) Write SQL query for given database. 4/6

Q.21 Explain different Data types in SQL.

Answer:

SQL provides different data types to define the type of values that can be stored in a table column. The main categories of SQL data types are:

1. **Numeric Data Types:** Used to store numbers.
   - **INT / INTEGER:** Whole numbers (e.g., 10, -20).
   - **DECIMAL(p,s) / NUMERIC:** Fixed precision numbers (e.g., 123.45).
   - **FLOAT / REAL:** Approximate decimal values (e.g., 12.34).
2. **Character/String Data Types:** Used to store text values.
   - **CHAR(n):** Fixed-length character string.
     Example: `CHAR(5)` → 'Amit ' (padded if shorter).
   - **VARCHAR(n):** Variable-length string up to n characters.
     Example: `VARCHAR(20)` → 'Database'.
3. **Date and Time Data Types:** Used to store date and time values.
   - **DATE:** Stores date in format `YYYY-MM-DD`.
     Example: `DATE '2024-06-15'`.
   - **TIME:** Stores time in format `HH:MM:SS`.
   - **DATETIME / TIMESTAMP:** Stores both date and time.
4. **Boolean Data Type:** Stores logical values `TRUE` or `FALSE` (supported in some SQL versions).
5. **Other Data Types (DBMS specific):**
   - **BLOB (Binary Large Object):** Stores large binary data like images, files.
   - **CLOB (Character Large Object):** Stores large text data like documents.

Q.22 Explain DCL and TCL commands.

Answer:

1. Data Control Language (DCL):

DCL commands are used to control access and permissions on the database. They manage security of the database system.

- **GRANT:** Provides privileges to users.
  Example:

```sql
GRANT SELECT, INSERT ON Students TO User1;
```

  → Allows User1 to select and insert records in Students table.

- **REVOKE:** Removes previously given privileges.
  Example:

```sql
REVOKE INSERT ON Students FROM User1;
```

  → Removes the insert permission from User1.

2. Transaction Control Language (TCL):

TCL commands are used to manage transactions in a database to ensure data consistency and integrity.

- **COMMIT:** Saves all changes made in the current transaction permanently.
  Example:

```sql
COMMIT;
```

- **ROLLBACK:** Undoes changes of the current transaction before commit.
  Example:

```sql
ROLLBACK;
```

- **SAVEPOINT:** Creates a point within a transaction to which a rollback can be done.
  Example:

```sql
SAVEPOINT sp1;
ROLLBACK TO sp1;
```

Q.23 Describe aggregate functions with example.

Answer:

Aggregate functions in SQL are used to perform calculations on a group of values and return a single result. They are often used with the **GROUP BY** clause. The main aggregate functions are:

1. **COUNT():** Returns the number of rows.

   Example:

   ```sql
   SELECT COUNT(*) FROM Students;
   ```

   → Returns total number of students.

2. **SUM():** Returns the total sum of a numeric column.

   Example:

   ```sql
   SELECT SUM(Salary) FROM Employees;
   ```

   → Returns total salary of all employees.

3. **AVG():** Returns the average value of a numeric column.

   Example:

   ```sql
   SELECT AVG(Age) FROM Students;
   ```

   → Returns average age of students.

4. **MIN():** Returns the smallest value.

   Example:

   ```sql
   SELECT MIN(Salary) FROM Employees;
   ```

   → Returns lowest salary.

5. **MAX():** Returns the largest value.

   Example:

   ```sql
   SELECT MAX(Salary) FROM Employees;
   ```

   → Returns highest salary.

Q.24 Write the query using relation algebra.

Answer:

Relational Algebra is a procedural query language that uses operators to retrieve data from relations (tables). The common operations are:

1. **Selection (σ):** Selects rows based on a condition.

   Query: Find students with age > 18.

   Relational Algebra:

   ```
   scss

   σ Age > 18 (Students)
   ```

2. **Projection (π):** Selects specific columns.

   Query: Display only names of students.

   Relational Algebra:

   ```
   scss

   π Name (Students)
   ```

3. **Union (∪):** Combines tuples from two relations (removes duplicates).

   Query: Find all course IDs from UG and PG courses.

   Relational Algebra:

   ```
   pgsql

   UG_Courses ∪ PG_Courses
   ```

4. **Set Difference (−):** Finds tuples in one relation but not in another.

   Query: Find students who enrolled but did not pay fees.

   Relational Algebra:

   ```
   nginx

   Enrolled − Paid
   ```

5. **Cartesian Product (×):** Combines all tuples of two relations.

   Query: Combine Students and Courses.

   Relational Algebra:

   ```
   nginx

   Students × Courses
   ```

6. **Join (⋈):** Combines related tuples from two relations based on condition.

   Query: Get student names with their course IDs.

   Relational Algebra:

   ```
   nginx

   Students ⋈ Students.StudentID = Enrollments.StudentID Enrollments
   ```

Q.25 Elaborate Decomposition with diagram.

Answer:

In DBMS, **Decomposition** is the process of breaking a relation (table) into two or more smaller relations. The main purpose of decomposition is to remove **data redundancy**, **anomalies** (insertion, deletion, update problems), and to achieve a higher **normal form**.

Types of Decomposition:

1. **Lossless-Join Decomposition:**
   - After decomposing, when we join the smaller relations, we get back the original relation without losing any data.
   - This is the desired property.
2. **Dependency-Preserving Decomposition:**
   - Ensures that functional dependencies in the original relation are still preserved in the decomposed relations.
   - This helps in enforcing data integrity without costly joins.

Example:

Suppose we have a relation:

```scss
STUDENT (StudentID, Name, CourseID, CourseName)
```

This relation has redundancy because `CourseName` repeats for every student of the same course.

We decompose into:

```scss
STUDENT (StudentID, Name, CourseID)
COURSE (CourseID, CourseName)
```

Now redundancy is removed, and we can join them back using `CourseID`.

Q.26 Describe several types of anomalies in database with example.

Answer:

Anomalies in a database occur when data is stored in an unnormalized table (not properly structured). These anomalies cause redundancy and inconsistency. The main types are:

1. **Insertion Anomaly:**
   - Problem occurs when we cannot insert data into a table without the presence of some other data.
   - *Example:* In a table storing **StudentID, Name, CourseID, CourseName**, if no student has enrolled in a new course, we cannot insert that course information.
2. **Deletion Anomaly:**
   - When deleting one fact accidentally causes the loss of another useful fact.
   - *Example:* If we delete the record of a student enrolled in a course, the information about that course may also be lost.
3. **Update Anomaly:**
   - When the same data is stored in multiple places and updating it in one place but not everywhere causes inconsistency.
   - *Example:* If a course name changes (e.g., "DBMS" → "Database Systems"), we must update it in all rows. If missed in one row, inconsistency occurs.

Q.27 Explain 1NF with example. (4/6 marks)

Answer:

First Normal Form (1NF):

A relation is in 1NF if:

1. Each column contains only **atomic (indivisible) values.**
2. There are **no repeating groups** or arrays.

Example:

Unnormalized table:

```scss
STUDENT (StudentID, Name, Courses)
101, Amit, {DBMS, OS, CN}
```

Here, `Courses` contains multiple values → violates 1NF.

1NF Table:

```scss
STUDENT (StudentID, Name, Course)
101, Amit, DBMS
101, Amit, OS
101, Amit, CN
```

Now each cell has atomic values, so the relation is in **1NF.**

Q.28 Explain 2NF with example. (4/6 marks)

Answer:

Second Normal Form (2NF):

A relation is in 2NF if:

1. It is in **1NF.**
2. There is **no partial dependency** (i.e., non-key attributes must depend on the whole primary key, not part of it).

Example:

Consider relation:

```scss
STUDENT_COURSE (StudentID, CourseID, StudentName, CourseName)
```

- Primary Key = (StudentID, CourseID)
- Problem: `StudentName` depends only on `StudentID`, and `CourseName` depends only on `CourseID` → partial dependency.

2NF Decomposition:

```scss
STUDENT (StudentID, StudentName)
COURSE (CourseID, CourseName)
ENROLLMENT (StudentID, CourseID)
```

Now non-key attributes depend on the **whole key**, so it is in **2NF.**

Q.29 Explain 3NF with example. (4/6 marks)

Answer:

Third Normal Form (3NF):

A relation is in 3NF if:

1. It is in **2NF.**
2. There is **no transitive dependency** (i.e., non-key attributes must not depend on other non-key attributes).

Example:

Consider relation:

```scss
STUDENT (StudentID, StudentName, DeptID, DeptName)
```

- Primary Key = StudentID
- Problem: `DeptName` depends on `DeptID`, which is not a key but another non-key attribute → transitive dependency.

3NF Decomposition:

```scss
STUDENT (StudentID, StudentName, DeptID)
DEPARTMENT (DeptID, DeptName)
```

Now no transitive dependency exists, so the relation is in **3NF.**