1. Introduction to DBMS

1.1 Data and Database

- **Data**: Raw facts and figures without context.
  Example: 12345, Alice, CSE.
- **Database**: Organized collection of data.
  Example: A university database storing students, faculty, courses, and results.

1.2 What is a DBMS?

- **Database Management System (DBMS)** = software that **stores, retrieves, and manages data**.
- Provides **structured access** to data using **query languages (SQL)**.
- Example: **Oracle, MySQL, PostgreSQL, MongoDB**.

1.3 Characteristics of DBMS

1. **Data Abstraction** – Hides physical details, provides conceptual/ logical view.
2. **Data Independence** – Application remains unaffected by data storage changes.
3. **Data Integrity** – Ensures correctness and validity.
4. **Security** – Only authorized users access/modify data.
5. **Concurrency Control** – Supports multiple users simultaneously.
6. **Backup and Recovery** – Restores data after failures.

1.4 Advantages of DBMS

- Eliminates redundancy.
- Provides data sharing among multiple users.
- Querying with SQL is easier.
- Ensures **ACID** properties.
- Scales better than file systems.

1.5 Applications of DBMS

- **Banking** – Transactions, balances.
- **Airline Systems** – Reservations, ticketing.
- **E-commerce** – Customer orders, payments.
- **Healthcare** – Patient records.
- **Education** – Student results, courses.

---

2. Entity-Relationship (ER) Model

2.1 Entities

- **Entity** = real-world object.
- Example: *Student, Teacher, Book*.

2.2 Attributes

- Properties of entities.
- Types:
    - **Simple Attribute**: RollNo, Name.
    - **Composite Attribute**: FullName = {FirstName, LastName}.
    - **Derived Attribute**: Age (calculated from DOB).
    - **Multivalued Attribute**: PhoneNumbers.

2.3 Relationships

- Association among entities.
- **Degree of Relationship**:
    - Unary (Self relationship).
    - Binary (Most common, e.g., Student–Course).
    - Ternary (Involving three entities).

2.4 Cardinality

- **1:1** → One employee has one ID card.
- **1:N** → A department has many students.
- **M:N** → Students enroll in many courses, courses have many students.

2.5 Example: University ER Diagram (Textual)

- Entities: Student, Course, Instructor.
- Attributes: Student(Name, RollNo), Course(Code, Title), Instructor(ID, Dept).
- Relationships:
    - Student *enrolls* in Course.
    - Instructor *teaches* Course.

---

3. Relational Model & Relational Algebra

3.1 Relational Model

- Represents data as **tables** (relations).
- **Tuple** = Row, **Attribute** = Column.
- Example:

Student Table

RollNo	Name	Dept
1	Alice	CSE
2	Bob	ECE

3.2 Relational Algebra Operations

1. **Selection (σ)**
   Select tuples satisfying a condition.
   Example: σDept='CSE'(Student).
2. **Projection (π)**
   Select attributes (columns).
   Example: πName, Dept(Student).

3. **Union (∪)**
   Combines two relations.
   Example: Student1 ∪ Student2.
4. **Intersection (∩)**
   Common tuples in two relations.
5. **Difference (-)**
   Tuples in one relation but not in the other.
6. **Cartesian Product (×)**
   Combines all tuples of two relations.
7. **Join (⋈)**
   - **Equi-Join** – Join on equality condition.
   - **Natural Join** – Automatically joins common attributes.

3.3 Example Query

Find all students enrolled in CSE courses.

- Step 1: σDept='CSE'(Course).
- Step 2: Student ⋈ Course.

---

4. Structured Query Language (SQL)

4.1 SQL Categories

1. **DDL** (CREATE, DROP, ALTER).
2. **DML** (INSERT, UPDATE, DELETE).
3. **DQL** (SELECT).
4. **DCL** (GRANT, REVOKE).
5. **TCL** (COMMIT, ROLLBACK, SAVEPOINT).

4.2 Example Queries

-- Create Table

```
CREATE TABLE Student (
  RollNo INT PRIMARY KEY,
  Name VARCHAR(50),
  Dept VARCHAR(20)
);

-- Insert Data
INSERT INTO Student VALUES (1, 'Alice', 'CSE');
INSERT INTO Student VALUES (2, 'Bob', 'ECE');

-- Query
SELECT Name FROM Student WHERE Dept = 'CSE';

-- Update
UPDATE Student SET Dept = 'IT' WHERE RollNo = 2;

-- Delete
DELETE FROM Student WHERE RollNo = 1;
```

4.3 Advanced SQL Queries

- **Joins**

```
SELECT Student.Name, Course.Title
FROM Student
JOIN Course ON Student.Dept = Course.Dept;
```

- **Aggregate Functions**

```
SELECT Dept, COUNT(*) AS TotalStudents
FROM Student
GROUP BY Dept;
```

- **Subqueries**

SELECT Name FROM Student

WHERE Dept = (SELECT Dept FROM Instructor WHERE ID = 101);

- **Views**

CREATE VIEW CSE_Students AS

SELECT Name FROM Student WHERE Dept = 'CSE';

---

5. Normalization

5.1 Purpose

- Reduce redundancy.
- Avoid anomalies.

5.2 Normal Forms

- **1NF** – No multivalued attributes.
- **2NF** – No partial dependency.
- **3NF** – No transitive dependency.
- **BCNF** – Stronger 3NF.

5.3 Example

Unnormalized Table:
Student(RollNo, Name, Course1, Course2).

1NF: Student(RollNo, Name, Course).

---

6. Transaction Management

6.1 Transactions

- A sequence of operations treated as one unit.

6.2 ACID Properties

1. **Atomicity** – All or none.
2. **Consistency** – Preserves validity.
3. **Isolation** – Transactions don't interfere.
4. **Durability** – Results survive failures.

6.3 Concurrency Control

- **Locks** (Shared, Exclusive).
- **Deadlocks** and prevention.

6.4 Recovery

- **Logs** → before and after values.
- **Checkpointing** → Save database state periodically.

---

7. Advanced Topics in DBMS

7.1 Distributed Databases

- Data stored across multiple servers.
- Example: Banking – customer can withdraw anywhere.

7.2 Data Warehousing & OLAP

- **Data Warehouse** – Historical data for analysis.
- **OLAP** – Multidimensional queries.

7.3 NoSQL Databases

- Non-relational, schema-less.
- Types: Key-Value, Document (MongoDB), Column (Cassandra).

7.4 Big Data and DBMS

- Integration with **Hadoop, Spark** for large-scale analytics.

7.5 Cloud Databases

- Managed databases on cloud platforms.
- Example: AWS RDS, Google BigQuery.

---

8. Case Study – Online Shopping Database

Entities

- Customer, Order, Product, Payment.

Relationships

- Customer places Order.
- Order contains Products.
- Payment made for Order.

ER Diagram (Textual)

Customer → Order → Product → Payment.

---

9. Summary

- DBMS provides **efficient, secure, scalable** data storage.
- ER models → used for design.
- Relational model → uses **tables, relational algebra**.
- SQL → powerful query language.
- Normalization → removes redundancy.
- Transactions → ensure ACID properties.
- Advanced DBMS → distributed, NoSQL, cloud databases.