Unit 1

2. Advantages of Database Systems over File Systems (10 Marks)

1. Reduced Data Redundancy:

In file systems, the same data may be stored in multiple files.

Example: Student address stored in library file and exam file separately.

In DBMS, data is centralized → no duplication.

2. Data Consistency:

Because redundancy is controlled, updates happen in one place, ensuring consistent data.

Example: A student's phone number updated once and reflected everywhere.

3. Improved Data Sharing:

Multiple users/applications can access the same data concurrently.

Example: Library, accounts, exam departments accessing the same student database.

4. Enhanced Security:

DBMS offers user authentication and role-based access.

Example: Student data can be accessible only to admin; marks only visible to exam cell.

5. Backup and Recovery:

DBMS provides automatic backup and crash recovery.

Example: If system fails during fee update, DBMS restores the last consistent state.

6. Data Integrity Enforcement:

DBMS provides constraints like primary key, foreign key, unique, etc.

Example: Cannot insert duplicate roll number in student table.

7. Concurrent Access:

DBMS handles multiple users reading/writing using concurrency control.

Example: Many students applying online simultaneously.

---

3. Three-Tier Architecture of DBMS & Data Independence

Three-tier DBMS contains:

1. External Level (View Level)

Shows only a part of the database to users.

Different users see different views.

Example: Students see marks; admin sees all records.

2. Conceptual Level (Logical Level)

Represents the entire logical structure of the database.

Example: Tables, attributes, constraints.

3. Internal Level (Physical Level)

Deals with physical storage of data on hard disk.

Example: Indexes, file organization.

How it Supports Data Independence

Logical Data Independence:

Changes in conceptual schema do not affect external views.

Example: Adding a new column Email does not affect user applications.

Physical Data Independence:

Changes in physical storage do not affect conceptual schema.

Example: Changing from heap file to B+ tree storage doesn't affect tables.

---

4. Centralized vs Client-Server DBMS Architecture

Feature Centralized DBMS Client–Server DBMS

Location of DB	Single central server	Distributed between clients and server
Processing	Done at central server	Server handles queries; clients handle UI
Performance	Slower for large users	Faster and scalable
Reliability	Single point of failure	More reliable (distributed clients)
Cost	Low cost	Higher cost
Example	Small office database	Online banking, college portal

---

5. Types of Data Models

1. Hierarchical Model

Organizes data in a tree structure (parent–child).

Example: Department → Employees.

2. Network Model

Data organized as a graph: a child can have multiple parents.

Example: Student enrolled in multiple courses.

3. Relational Model

Data stored in tables (relations).

Example: STUDENT (RollNo, Name, Dept).

4. Entity–Relationship (ER) Model

Represents data using entity, attributes, and relationships.

Example: Student—Enrolls—Course.

5. Object-Oriented Data Model

Stores data as objects with classes, inheritance, methods.

Example: STUDENT object containing functions and attributes.

---

6. ER Diagram for College Management System

Entities:

Student (ID, Name, Dept, Phone)

Course (CourseID, CourseName, Credits)

Faculty (FacultyID, Name, Dept)

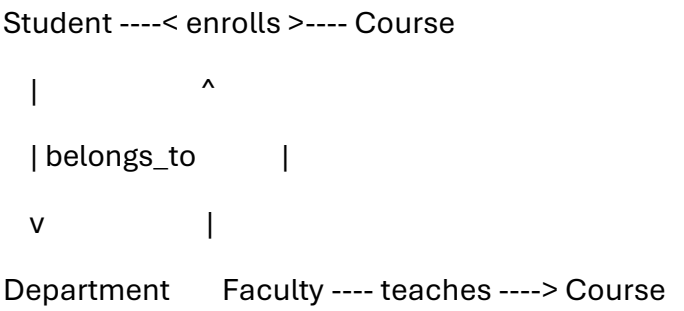
Department (DeptID, DeptName)

Relationships:

Student enrolls in Course (Many-to-Many).

Faculty teaches Course (1-to-Many).

Student belongs to Department (Many-to-1).



---

7. Define Entity, Attribute, Relationship + ER Diagram

Entity: Object having independent existence.

Example: Student, Employee.

Attribute: Properties of an entity.

Example: Student(Name, Age, RollNo).

Relationship: Association between entities.

Example: Student enrolls in Course.

Simple ER Diagram:

Student ---- enrolls ---- Course

(ID, Name) (CID, CName)

---

8. Generalization, Specialization, and Inheritance

Generalization

Combining lower-level entities into a higher-level entity.

Example: CAR, TRUCK → VEHICLE.

Specialization

Dividing higher-level entity into lower-level entities.

Example: EMPLOYEE → TEACHER, CLERK.

Inheritance

Lower-level entity inherits attributes of higher-level entity.

Example: TEACHER inherits attributes of EMPLOYEE.

---

9. Schema vs Instance

Schema

Overall design/structure of the database.

Fixed and rarely changes.

Example: Student(SID, Name, Dept).

Instance

Snapshot of the database at a particular time.

Example: Current rows stored in STUDENT table.

10(a). Data Independence

Ability to change schema at one level without affecting other levels.

Types:

Logical Data Independence:

Change in conceptual schema without affecting external views.

Physical Data Independence:

Change in internal storage without affecting logical schema.

10(b). Centralized vs Client-Server Architecture

(Same as Q4; you can repeat.)

Unit 2

Below are clear, exam-oriented answers for all questions.

Each answer is structured for 5M / 10M depending on requirement.

---

1. Discuss different types of constraints in relational databases with examples.

(CO2, L2, 10 Marks)

Constraints ensure correctness, validity, and consistency of data in a relational database.

---

1. Domain Constraint

Specifies that the value of an attribute must lie within a valid range or type.

Example:

Age INT CHECK (Age >= 18);

Age must be ≥ 18.

---

2. Entity Integrity Constraint (Primary Key Constraint)

Ensures that the primary key value is unique and not NULL.

Example:

RollNo INT PRIMARY KEY;

No two students can have the same RollNo.

---

3. Referential Integrity Constraint (Foreign Key Constraint)

Ensures relationship consistency between tables.

Foreign key value must match an existing primary key or be NULL.

Example:

DeptID INT REFERENCES Department(DeptID);

---

4. Key Constraints

Every table must have a key that uniquely identifies a tuple.

Types: Primary Key, Candidate Key, Super Key.

Example:

RollNo is a candidate key in STUDENT table.

---

5. Unique Constraint

Ensures that all values in a column are unique.

Example:

Email VARCHAR(50) UNIQUE;

---

6. Check Constraint

Used to limit the range of values.

Example:

Salary CHECK (Salary > 5000);

---

7. Not Null Constraint

Prevents a column from accepting NULL values.

Example:

Name VARCHAR(30) NOT NULL;

---

8. Default Constraint

Assigns a default value if the user does not provide one.

Example:

Status VARCHAR(10) DEFAULT 'Active';

---

2. Define and explain fundamental concepts of relational model

(CO2, L1, 5 Marks)

a) Relation

A relation is a table with rows and columns.

Example: STUDENT table.

b) Attribute

A column in a table.

Example: RollNo, Name, Dept.

c) Tuple

A row in a table.

Example: (101, 'Ravi', 'CSE').

d) Domain

Set of valid values an attribute can take.

Example: Age domain = INTEGER 18–60.

e) Relation Schema

Structure of a relation including its attributes.

Example:

STUDENT(SID, Name, Age, Dept).

---

3. Basic concepts of the relational model

(CO2, L1, 5 Marks)

1. Relation – Table consisting of rows & columns.

2. Tuple – A row in the relation.

3. Attribute – A column of the relation.

4. Domain – Allowed values of attributes.

5. Primary Key – Uniquely identifies a tuple.

6. Foreign Key – Refers to primary key of another table.

7. Candidate Key – Attribute(s) capable of becoming primary key.

8. Degree – Number of attributes in a relation.

9. Cardinality – Number of tuples in a relation.

---

4. Relational Algebra Operations

(CO2, L3, 10 Marks)

---

1. Selection (σ)

Selects rows based on condition.

Example:

σ Dept='CSE' (STUDENT)

---

2. Projection (π)

Selects specific columns.

Example:

π Name, Dept (STUDENT)

---

3. Join (⋈)

Combines related tuples from two relations.

Example:

STUDENT ⋈ STUDENT.DeptID = DEPT.DeptID DEPARTMENT

---

4. Union (∪)

Combines tuples from two relations removing duplicates.

Example:

A ∪ B

---

5. Set Difference (−)

Returns tuples present in one relation but not in another.

Example:

A − B

---

5. SQL Examples: Domain, Key, Referential Integrity Constraints

(CO2, L3, 10 Marks)

CREATE TABLE Department (

```
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(30)
);

CREATE TABLE Student (
    RollNo INT PRIMARY KEY,              -- Key constraint
    Name VARCHAR(30) NOT NULL,           -- Domain + Not Null
    Age INT CHECK (Age >= 18),           -- Domain Constraint
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID)   -- Referential Integrity
);
```

---

6. STUDENT Database Schema + SQL Table Creation

(CO2, L3, 10 Marks)

Schema

STUDENT (RollNo, Name, Age, DeptID)

DEPARTMENT (DeptID, DeptName)

SQL Code

```
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(50)
);

CREATE TABLE Student (
   RollNo INT PRIMARY KEY,
   Name VARCHAR(50),
   Age INT,
   DeptID INT,
   FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);
```

---

7. Basic SQL Queries: Insert, Update, Delete

(CO2, L3, 10 Marks)

Insert

```
INSERT INTO Student VALUES (101, 'Ravi', 20, 1);
```

Update

UPDATE Student SET Age = 21 WHERE RollNo = 101;

Delete

DELETE FROM Student WHERE RollNo = 101;

---

8. SELECT Queries with WHERE, ORDER BY, GROUP BY

(CO2, L3, 10 Marks)

WHERE

SELECT * FROM Student WHERE DeptID = 1;

ORDER BY

SELECT * FROM Student ORDER BY Name ASC;

GROUP BY

SELECT DeptID, COUNT(*) FROM Student GROUP BY DeptID;

---

9. Defining PRIMARY & FOREIGN KEY in SQL

(CO2, L3, 10 Marks)

```
CREATE TABLE Course (
   CourseID INT PRIMARY KEY,
   CourseName VARCHAR(50)
);

CREATE TABLE Enroll (
   RollNo INT,
   CourseID INT,
   PRIMARY KEY (RollNo, CourseID),
   FOREIGN KEY (RollNo) REFERENCES Student(RollNo),
   FOREIGN KEY (CourseID) REFERENCES Course(CourseID)
);
```

---

10. ALTER and DROP with Syntax & Example

(CO2, L3, 10 Marks)

ALTER TABLE

Add a column

ALTER TABLE Student ADD Email VARCHAR(50);

Modify a column

ALTER TABLE Student MODIFY Age INT;

Drop a column

ALTER TABLE Student DROP COLUMN Email;

DROP TABLE

DROP TABLE Student;

If you want, I can combine everything into a single PDF, 2-mark answers, or 15-mark extended notes.

Unit 3

Below are clear, exam-ready answers for all questions. All answers are written for 5M / 10M format with examples.

1. Explain different SQL functions with examples: numeric, date/time, and string conversion.

(CO3, L2, 10 Marks)

SQL provides built-in functions to perform calculations, manipulate dates, and convert data types.

A. Numeric Functions

1. ABS() – Absolute value

SELECT ABS(-25);

Output → 25

2. POWER() – Calculates power

SELECT POWER(3, 2);

Output → 9

3. ROUND() – Rounds a number

SELECT ROUND(12.678, 2);

Output → 12.68

4. CEIL() and FLOOR()

```
SELECT CEIL(12.3), FLOOR(12.9);
```

Output → 13, 12

B. Date/Time Functions

1. CURRENT_DATE / NOW()

```
SELECT CURRENT_DATE;
```

2. DATEADD() – Adds interval to a date

```
SELECT DATEADD(day, 7, '2025-01-10');
```

3. DATEDIFF() – Difference between dates

```
SELECT DATEDIFF(day, '2025-01-01', '2025-01-10');
```

Output → 9

4. EXTRACT() – Extracts year/month/day

```
SELECT EXTRACT(YEAR FROM '2025-02-12');
```

C. String Conversion Functions

1. CAST()

```
SELECT CAST(123 AS CHAR);
```

2. CONVERT()

```
SELECT CONVERT(VARCHAR(10), 2025);
```

3. TO_CHAR() – converts date to string

```
SELECT TO_CHAR(CURRENT_DATE, 'DD-MM-YYYY');
```

2. Describe how integrity and key constraints are implemented in SQL.

(CO3, L2, 10 Marks)

SQL supports several constraints to maintain accuracy and consistency of data.

1. PRIMARY KEY Constraint

Ensures uniqueness + not NULL.

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

2. FOREIGN KEY Constraint

Maintains referential integrity.

CREATE TABLE Enroll (RollNo INT, CourseID INT, FOREIGN KEY (RollNo) REFERENCES Student(RollNo));

3. UNIQUE Constraint

Ensures no duplicate values.

Email VARCHAR(50) UNIQUE

4. NOT NULL Constraint

Prevents missing values.

Name VARCHAR(50) NOT NULL

5. CHECK Constraint

Defines a valid range or expression.

Age INT CHECK (Age >= 18)

6. DEFAULT Constraint

Assigns default value.

Status VARCHAR(10) DEFAULT 'Active'

3. Discuss different types of SQL joins with examples.

(CO3, L4, 10 Marks)

Assume tables: Student(RollNo, Name, DeptID) Department(DeptID, DeptName)

1. INNER JOIN

Returns matching rows from both tables.

SELECT Name, DeptName FROM Student INNER JOIN Department ON Student.DeptID = Department.DeptID;

2. LEFT JOIN

Returns all rows from left table + matched rows from right.

SELECT Name, DeptName FROM Student LEFT JOIN Department ON Student.DeptID = Department.DeptID;

3. RIGHT JOIN

Returns all rows from right table + matched rows from left.

SELECT Name, DeptName FROM Student RIGHT JOIN Department ON Student.DeptID = Department.DeptID;

4. FULL OUTER JOIN

Returns all rows from both tables.

SELECT Name, DeptName FROM Student FULL OUTER JOIN Department ON Student.DeptID = Department.DeptID;

4. Define a View and Characteristics of Updatable & Non-Updatable Views.

(CO3, L1, 5 Marks)

Definition of View

A view is a virtual table created using a query. It does not store data physically; it displays data from underlying tables.

CREATE VIEW CSE_Students AS SELECT RollNo, Name FROM Student WHERE DeptID = 1;

Characteristics of Updatable Views

Based on a single table.

Contains primary key of the base table.

Does not use GROUP BY, DISTINCT, or aggregate functions.

INSERT, UPDATE, DELETE are allowed.

Characteristics of Non-Updatable Views

Based on multiple tables (joins).

Contains aggregate functions.

Contains DISTINCT or GROUP BY.

Cannot be used to modify data.

5. Write SQL queries to demonstrate subquery & nested query.

(CO3, L3, 5 Marks)

A. Subquery

Used inside WHERE, SELECT, or FROM.

SELECT Name FROM Student WHERE DeptID = (SELECT DeptID FROM Department WHERE DeptName = 'CSE');

B. Nested Query

Query inside another query level-by-level.

SELECT Name FROM Student WHERE RollNo IN (SELECT RollNo FROM Enroll WHERE CourseID IN (SELECT CourseID FROM Course WHERE Credits > 3));

6. Write SQL queries using arithmetic and logical operations in WHERE clause.

(CO3, L3, 10 Marks)

Arithmetic Operators

SELECT Name, Age + 1 AS NextYearAge FROM Student WHERE Age + 5 > 25;

Logical Operators (AND, OR, NOT)

SELECT * FROM Student WHERE Age > 18 AND DeptID = 1;

SELECT * FROM Student WHERE DeptID = 1 OR DeptID = 2;

SELECT * FROM Student WHERE NOT (Age < 18);

7. Create tables with relationships and apply keys.

(CO3, L3, 10 Marks)

CREATE TABLE Department (DeptID INT PRIMARY KEY, DeptName VARCHAR(50));

CREATE TABLE Student (RollNo INT PRIMARY KEY, Name VARCHAR(50), Age INT, DeptID INT, FOREIGN KEY (DeptID) REFERENCES Department(DeptID));

8. SQL Queries Using GROUP BY, HAVING, ORDER BY

(CO3, L3, 10 Marks)

SELECT DeptID, COUNT*() AS TotalStudents FROM Student GROUP BY DeptID HAVING COUNT()* > 10 ORDER BY TotalStudents DESC;

9. Implement all JOINs with examples.

(CO3, L3, 10 Marks)

Using Student and Department tables:

INNER JOIN

SELECT Student.Name, Department.DeptName FROM Student INNER JOIN Department ON Student.DeptID = Department.DeptID;

LEFT JOIN

SELECT Student.Name, Department.DeptName FROM Student LEFT JOIN Department ON Student.DeptID = Department.DeptID;

RIGHT JOIN

SELECT Student.Name, Department.DeptName FROM Student RIGHT JOIN Department ON Student.DeptID = Department.DeptID;

FULL OUTER JOIN

SELECT Student.Name, Department.DeptName FROM Student FULL OUTER JOIN Department ON Student.DeptID = Department.DeptID;

Unit 4

Below are clear, exam-oriented, full-mark answers for all questions.

All answers follow CO4 – Normalization Concepts, with examples and stepwise explanation.

---

2. Define BCNF. Differentiate between 3NF and BCNF with examples.

(CO4, L4, 10 Marks)

BCNF (Boyce–Codd Normal Form) – Definition

A relation R is in BCNF if:

For every non-trivial functional dependency X → Y,

X must be a superkey.

This is a stronger version of 3NF.

Difference Between 3NF and BCNF

Feature	3NF	BCNF
Condition	For every FD X→Y, either X is a superkey OR Y is a prime attribute	For every FD X→Y, X must be a superkey
Strength	Weaker	Stronger
Handles anomalies?	Some anomalies may remain	Removes all anomalies
Use case	Practical databases (dependency preserving)	Highly normalized systems (minimal redundancy)

Example Showing Difference

Relation: R(A, B, C)

Functional dependencies:

1. A → B

2. B → A

Candidate keys = A, B

Check 3NF

For A → B → A is a superkey → OK

For B → A → B is key → OK

Relation is in 3NF

Check BCNF

A → B (A is key) OK

B → A (B is key) OK

So this example satisfies both.

Example Where 3NF Holds but BCNF Fails

Relation: R(Student, Course, Teacher)

FDs:

1. Student, Course → Teacher

2. Teacher → Course

Candidate key = (Student, Course)

Check BCNF:

Teacher → Course violates BCNF (Teacher not a key)

But allowed in 3NF because Course is prime attribute

Thus:

3NF but not BCNF

---

3. Explain Lossless Join and Dependency Preserving Decomposition.

(CO4, L4, 10 Marks)

A. Lossless Join Decomposition

A decomposition of R into R1 and R2 is lossless if:

(R1 ∩ R2) → R1 OR (R1 ∩ R2) → R2

Meaning no information is lost after splitting.

Example

R(A, B, C)

FD: A → B

Decompose into:

R1(A, B)

R2(A, C)

Check:

R1 ∩ R2 = {A}

Does A → B hold? YES

Therefore lossless.

B. Dependency Preserving Decomposition

All original functional dependencies must be preserved in decomposed relations without needing a join.

Example

FDs: A → B, B → C

Decompose R(A, B, C) into:

R1(A, B)

R2(B, C)

All FDs appear in individual tables → Dependency preserved

Non-preserving Example

R(A, B, C)

FD: A → B, B → C

Decompose into:

R1(A, B)

R2(A, C)

Now B → C lost → Not dependency preserving.

4(a). Explain Functional Dependency with examples.

(CO4, L1, 5 Marks)

Functional Dependency (FD) X→Y means:

If two tuples match on X, they must match on Y.

Example

In STUDENT(RollNo, Name, Dept):

RollNo → Name

RollNo → Dept

Because RollNo uniquely determines Name & Dept.

4(b). Characteristics of a Good Database Schema

(CO4, L1, 5 Marks)

A good schema must have:

1. Minimal redundancy → avoid anomalies

2. Clear structure → easy to understand tables

3. Provides normalization → 3NF or BCNF

4. Supports constraints → Keys, FKs, checks

5. Ensures data integrity

6. Efficient performance → minimal joins

7. Extensible → easy to modify

---

5. Multivalued Dependency, 4NF, 5NF with examples

(CO4, L2, 5 Marks)

Multivalued Dependency (MVD)

X →→ Y means:

For each value of X, there exists a set of independent values of Y.

Example

R(Student, Hobby, Skill)

Student →→ Hobby

Student →→ Skill

Hobby and Skill are independent.

4NF

A relation is in 4NF if:

For every MVD X↠Y, X must be a key.

Example

R(Student, Hobby)

Student ↠ Hobby

Student is key → Already in 4NF

5NF (Project–Join Normal Form)

A relation is in 5NF if it cannot be decomposed into smaller tables without losing information.

Used when all dependencies come from join dependencies.

6(a). Difference Between Lossless Join and Dependency Preservation

(CO4, L4, 5 Marks)

Lossless Join	Dependency Preserving
Ensures no data loss	Ensures no FD is lost
Mandatory	Not always possible
Based on common attributes	Based on preserving FDs
Ensures correctness	Ensures constraints remain usable

6(b). Explain 1NF, 2NF, 3NF with one example each

(CO4, L1, 5 Marks)

1NF

No repeating groups; all values atomic.

Example:

STUDENT(Name, Subjects) → Not allowed

Convert to:

(Student, Subject)

2NF

No partial dependency (for composite keys).

Example:

R(RollNo, Course, Instructor)

FD: (RollNo, Course) → Instructor

OK; if RollNo → Instructor (partial) → violates 2NF

3NF

No transitive dependencies.

Example:

FDs: A → B, B → C

Remove transitive dependency B → C.

---

7. Normalize a Relation to 3NF (Step-by-Step)

(CO4, L3, 10 Marks)

Example relation:

R(SID, SName, Dept, HOD, HOD_Phone)

FDs:

SID → SName, Dept

Dept → HOD, HOD_Phone

Step 1: Identify Keys

SID determines all attributes → SID is key.

Step 2: Identify Violations

Dept → HOD, HOD_Phone → transitive dependency → violates 3NF.

Step 3: Decompose

Create R1 (Student Info)

R1(SID, SName, Dept)

Create R2 (Department Info)

R2(Dept, HOD, HOD_Phone)

Both are now in 3NF.

---

8. Identify candidate keys and perform BCNF decomposition

(CO4, L3, 10 Marks)

Given:

R(A, B, C)

FDs: A → B, B → C

Step 1: Find candidate keys

A → B → C

Thus A is key

B → C, but B cannot determine A → Not key

C cannot determine any → Not key

Candidate key = {A}

BCNF Check

1. A → B OK (A is key)

2. B → C violates BCNF (B not key)

Step 2: Decompose using violating FD B → C

Create R1 for dependency:

R1(B, C)

Create R2 with remaining attributes:

R2(A, B)

Both relations are now in BCNF

---

9. Lossless Join & Dependency Preservation Check

(CO4, L3, 10 Marks)

Decomposition:

R(A, B, C) → R1(A, B) & R2(B, C)

Lossless Check

Common attribute = B

Does B → A or B → C hold?

Given B → C

→ Yes

Decomposition is lossless.

Dependency Preservation

FDs: A → B, B → C

A → B in R1

B → C in R2

All preserved → Dependency preserving

---

10. Decompose a relation with MVDs into 4NF

(CO4, L3, 10 Marks)

Given:

R(Student, Hobby, Skill)

MVDs:

Student ↠ Hobby

Student ↠ Skill

Step 1: Check 4NF

Student is key?

No, key is (Student, Hobby, Skill)

Thus violates 4NF.

Step 2: Decompose

Since Student ↠ Hobby:

R1(Student, Hobby)

Since Student ↠ Skill:

R2(Student, Skill)

Step 3: Final 4NF Relations

R1(Student, Hobby)

R2(Student, Skill)

These are now in 4NF because Student is key in each.

Unit 4

Below are clear, exam-oriented, full-mark answers for all questions.

All answers follow CO4 – Normalization Concepts, with examples and stepwise explanation.

---

2. Define BCNF. Differentiate between 3NF and BCNF with examples.

(CO4, L4, 10 Marks)

BCNF (Boyce–Codd Normal Form) – Definition

A relation R is in BCNF if:

For every non-trivial functional dependency X → Y,

X must be a superkey.

This is a stronger version of 3NF.

Difference Between 3NF and BCNF

Feature	3NF	BCNF
Condition	For every FD X→Y, either X is a superkey OR Y is a prime attribute	For every FD X→Y, X must be a superkey
Strength	Weaker	Stronger
Handles anomalies?	Some anomalies may remain	Removes all anomalies
Use case	Practical databases (dependency preserving)	Highly normalized systems (minimal redundancy)

Example Showing Difference

Relation: R(A, B, C)

Functional dependencies:

1. A → B

2. B → A

Candidate keys = A, B

Check 3NF

For A → B → A is a superkey → OK

For B → A → B is key → OK

Relation is in 3NF

Check BCNF

A → B (A is key) OK

B → A (B is key) OK

So this example satisfies both.

Example Where 3NF Holds but BCNF Fails

Relation: R(Student, Course, Teacher)

FDs:

1. Student, Course → Teacher

2. Teacher → Course

Candidate key = (Student, Course)

Check BCNF:

Teacher → Course violates BCNF (Teacher not a key)

But allowed in 3NF because Course is prime attribute

Thus:

3NF but not BCNF

---

3. Explain Lossless Join and Dependency Preserving Decomposition.

(CO4, L4, 10 Marks)

A. Lossless Join Decomposition

A decomposition of R into R1 and R2 is lossless if:

(R1 ∩ R2) → R1 OR (R1 ∩ R2) → R2

Meaning no information is lost after splitting.

Example

R(A, B, C)

FD: A → B

Decompose into:

R1(A, B)

R2(A, C)

Check:

R1 ∩ R2 = {A}

Does A → B hold? YES

Therefore lossless.

B. Dependency Preserving Decomposition

All original functional dependencies must be preserved in decomposed relations without needing a join.

Example

FDs: A → B, B → C

Decompose R(A, B, C) into:

R1(A, B)

R2(B, C)

All FDs appear in individual tables → Dependency preserved

Non-preserving Example

R(A, B, C)

FD: A → B, B → C

Decompose into:

R1(A, B)

R2(A, C)

Now B → C lost → Not dependency preserving.

4(a). Explain Functional Dependency with examples.

(CO4, L1, 5 Marks)

Functional Dependency (FD) X→Y means:

If two tuples match on X, they must match on Y.

Example

In STUDENT(RollNo, Name, Dept):

RollNo → Name

RollNo → Dept

Because RollNo uniquely determines Name & Dept.

4(b). Characteristics of a Good Database Schema

(CO4, L1, 5 Marks)

A good schema must have:

1. Minimal redundancy → avoid anomalies

2. Clear structure → easy to understand tables

3. Provides normalization → 3NF or BCNF

4. Supports constraints → Keys, FKs, checks

5. Ensures data integrity

6. Efficient performance → minimal joins

7. Extensible → easy to modify

---

5. Multivalued Dependency, 4NF, 5NF with examples

(CO4, L2, 5 Marks)

Multivalued Dependency (MVD)

X →→ Y means:

For each value of X, there exists a set of independent values of Y.

Example

R(Student, Hobby, Skill)

Student →→ Hobby

Student →→ Skill

Hobby and Skill are independent.

4NF

A relation is in 4NF if:

For every MVD X↠Y, X must be a key.

Example

R(Student, Hobby)

Student ↠ Hobby

Student is key → Already in 4NF

5NF (Project–Join Normal Form)

A relation is in 5NF if it cannot be decomposed into smaller tables without losing information.

Used when all dependencies come from join dependencies.

6(a). Difference Between Lossless Join and Dependency Preservation

(CO4, L4, 5 Marks)

Lossless Join	Dependency Preserving
Ensures no data loss	Ensures no FD is lost
Mandatory	Not always possible
Based on common attributes	Based on preserving FDs
Ensures correctness	Ensures constraints remain usable

6(b). Explain 1NF, 2NF, 3NF with one example each

(CO4, L1, 5 Marks)

1NF

No repeating groups; all values atomic.

Example:

STUDENT(Name, Subjects) → Not allowed

Convert to:

(Student, Subject)

2NF

No partial dependency (for composite keys).

Example:

R(RollNo, Course, Instructor)

FD: (RollNo, Course) → Instructor

OK; if RollNo → Instructor (partial) → violates 2NF

3NF

No transitive dependencies.

Example:

FDs: A → B, B → C

Remove transitive dependency B → C.

---

7. Normalize a Relation to 3NF (Step-by-Step)

(CO4, L3, 10 Marks)

Example relation:

R(SID, SName, Dept, HOD, HOD_Phone)

FDs:

SID → SName, Dept

Dept → HOD, HOD_Phone

Step 1: Identify Keys

SID determines all attributes → SID is key.

Step 2: Identify Violations

Dept → HOD, HOD_Phone → transitive dependency → violates 3NF.

Step 3: Decompose

Create R1 (Student Info)

R1(SID, SName, Dept)

Create R2 (Department Info)

R2(Dept, HOD, HOD_Phone)

Both are now in 3NF.

---

8. Identify candidate keys and perform BCNF decomposition

(CO4, L3, 10 Marks)

Given:

R(A, B, C)

FDs: A → B, B → C

Step 1: Find candidate keys

A → B → C

Thus A is key

B → C, but B cannot determine A → Not key

C cannot determine any → Not key

Candidate key = {A}

BCNF Check

1. A → B OK (A is key)

2. B → C violates BCNF (B not key)

Step 2: Decompose using violating FD B → C

Create R1 for dependency:

R1(B, C)

Create R2 with remaining attributes:

R2(A, B)

Both relations are now in BCNF

---

9. Lossless Join & Dependency Preservation Check

(CO4, L3, 10 Marks)

Decomposition:

R(A, B, C) → R1(A, B) & R2(B, C)

Lossless Check

Common attribute = B

Does B → A or B → C hold?

Given B → C

→ Yes

Decomposition is lossless.

Dependency Preservation

FDs: A → B, B → C

A → B in R1

B → C in R2

All preserved → Dependency preserving

---

10. Decompose a relation with MVDs into 4NF

(CO4, L3, 10 Marks)

Given:

R(Student, Hobby, Skill)

MVDs:

Student ↠ Hobby

Student ↠ Skill

Step 1: Check 4NF

Student is key?

No, key is (Student, Hobby, Skill)

Thus violates 4NF.

Step 2: Decompose

Since Student ↠ Hobby:

R1(Student, Hobby)

Since Student ↠ Skill:

R2(Student, Skill)

Step 3: Final 4NF Relations

R1(Student, Hobby)

R2(Student, Skill)

These are now in 4NF because Student is key in each.

Unit 5

Below is a clear, exam-ready 10-mark answer set for all questions you listed (CO5 – Transactions, Concurrency Control, Recovery, B+ Trees). Each answer is complete, structured, and suitable for a 10-mark response.

**2. Explain all ACID properties of a transaction with suitable examples.

CO5 – L2 – 10 Marks**

ACID properties ensure reliable and consistent execution of database transactions.

1. Atomicity

Ensures all operations of a transaction are executed completely, or none at all.

If a failure occurs, the transaction is rolled back.

Example:

T1: Transfer ₹500 from A to B

1. A = A – 500
2. B = B + 500

If step 2 fails, DB rolls back step 1 → ensuring no partial update.

2. Consistency

Ensures the database moves from one valid state to another valid state.

All integrity constraints must be satisfied.

Example:

If total balance = A + B = ₹10,000 After transfer, total must still be ₹10,000.

3. Isolation

Concurrent transactions must behave as if they execute one after another (serially).

Example:

Two users booking the last movie ticket must not get the same seat.

4. Durability

Once a transaction commits, its effects are permanent, even after a crash.

Example:

After T1 commits a bank transfer, the new balance remains stored even if power fails.

**3. List various concurrency control protocols used to manage concurrent transactions.

CO5 – L2 – 10 Marks**

1. Lock-Based Protocols

Binary Locks (0/1 locks)

Shared (S) and Exclusive (X) locks

Two-Phase Locking (2PL)

Growing phase: acquire locks

Shrinking phase: release locks

Strict 2PL – release locks only after commit.

2. Timestamp-Based Protocols

Each transaction gets a timestamp.

Uses:

Read Timestamp (RTS)

Write Timestamp (WTS)

Ensures serial order based on timestamps.

3. Optimistic Concurrency Control (Validation)

Assumes conflicts are rare.

Three phases:

1. Read phase
2. Validation phase
3. Write phase

4. Multi-Version Concurrency Control (MVCC)

Maintains multiple versions of data items.

Readers never block writers; widely used in PostgreSQL, Oracle.

5. Graph-Based Protocols

Precedence graph avoids deadlocks by locking items in a particular order.

6. Deadlock Prevention/Detection Protocols

Wait-Die and Wound-Wait

Timeout-based protocols

**4. Discuss deadlock detection and prevention in database systems.

CO5 – L4 – 10 Marks**

Deadlock Definition

A deadlock occurs when two or more transactions wait indefinitely for each other's locked resources.

A. Deadlock Prevention

Ensures deadlock never occurs.

1. Wait-Die Protocol

Older transaction waits.

Younger transaction aborts.

2. Wound-Wait Protocol

Older transaction preempts (wounds) younger one.

Younger one rolls back.

3. Resource Ordering

All transactions lock resources in a predefined order.

4. Timeout Mechanism

If waiting > threshold → abort transaction.

B. Deadlock Detection

Allows deadlocks but identifies and resolves them.

1. Wait-For Graph (WFG)

Nodes = transactions

Edge Ti → Tj means Ti waits for Tj

A cycle in WFG = deadlock

2. Deadlock Resolution

Choose a victim transaction to abort based on:

least cost

minimum progress

number of data items used

**5. Given a schedule, test whether it is conflict serializable. Show reasoning.

CO5 – L3 – 10 Marks**

Steps to Test Conflict Serializability

1. Identify conflicting operations

Same data item

At least one is a write

Belong to different transactions

2. Construct Precedence Graph

Nodes → Transactions (T1, T2, ...)

Edges Ti → Tj if Ti must come before Tj

3. Check for cycles

No cycle → conflict serializable

Cycle → not serializable

Example Schedule:

T1: R(A) W(A) T2: R(A) W(A)

Conflicts

T1 W(A) → T2 R(A) = edge T1 → T2

T2 W(A) → T1 R(A) = edge T2 → T1

Precedence Graph

T1 → T2

T2 → T1

Cycle exists → Not conflict serializable.

**6. Demonstrate how B+ Tree operations (insertion and search) work with example.

CO5 – L3 – 10 Marks**

Consider order m = 3 B+ tree (each node can have 2–3 keys).

A. Insertion example

Insert keys: 10, 20, 5, 6, 12

Step 1: Insert 10, 20

Leaf: [10, 20]

Step 2: Insert 5

Leaf becomes [5, 10, 20] (overflow, split)

Split into:

Left: [5, 10]

Right: [20] Parent: [10]

Step 3: Insert 6

Insert in correct leaf → [5, 6, 10] Split again:

Left: [5, 6]

Right: [10] Parent becomes: [6, 10]

Step 4: Insert 12

Goes to right leaf → [10, 12] (no split)

Final B+ Tree

[6, 10] / |

[5,6] [10] [12,20]

B. Search Operation

To search key 12:

1. Start at root → compare with 6, 10
2. Go to rightmost pointer
3. Leaf found → contains 12

Search complexity = O(log n)

**7. Write and explain a recovery algorithm using Deferred Update.

CO5 – L3 – 10 Marks**

Deferred Update (NO-UNDO/REDO)

Updates are not written to the database until transaction commits.

Only written to log first.

Algorithm Steps

1. Start Transaction

Log

2. Perform Write Operations

Changes made in memory (cache).

Log <T, X, NewValue> but DB not updated.

3. Commit

Log

Apply all writes to DB

Log

Recovery after Crash

Case 1: Committed Transaction

Log shows

REDO transaction changes using log records.

Case 2: Uncommitted Transaction

No found

NO-UNDO needed because DB not updated before commit.

Example

Transaction T1 updates:

A = 200 → 500 B = 100 → 300

Log:

Recovery scans the log:

Finds commit → Redo T1

**8 (a). Explain serializability and its importance.

CO5 – L2 – 5 Marks**

Serializability

A concurrent schedule is serializable if its result is equivalent to some serial schedule.

Types

1. Conflict Serializability
2. View Serializability

Importance

Prevents anomalies:

Lost updates

Dirty reads

Unrepeatable reads

Ensures correctness while allowing concurrency.

**8 (b). Differentiate between Lock-Based and Timestamp-Based Protocols.

CO5 – L4 – 5 Marks**

Feature Lock-Based Timestamp-Based

Control method Locks (S, X) Timestamps (TS) Conflicts Block (wait) Abort/restart Deadlock Possible Impossible Performance Good in low conflict Good in high conflict Examples 2PL, Strict 2PL TO (Thomas' rule)

**9. Implement Timestamp Ordering Protocol with example.

CO5 – L3 – 10 Marks**

Each data item X has:

RTS(X) – latest read

WTS(X) – latest write

Rules

1. Read Rule

If TS(T) < WTS(X) → abort T Else → allow read and update RTS(X)

2. Write Rule

If:

TS(T) < RTS(X) → abort T

TS(T) < WTS(X) → ignore write (Thomas' rule) Else → allow write and update WTS(X)

Example

Assume:

TS(T1) = 5
TS(T2) = 10

Operations:

T1: R(X) T2: W(X)

Step 1: T1 reads X

→ RTS(X) = 5

Step 2: T2 writes X

TS(T2)=10 > RTS(X)=5 → allowed → WTS(X)=10

No conflict, schedule is serializable.

**10. Explain recovery and atomicity. Describe how DB ensures recovery using logs.

CO5 – L4 – 5 Marks**

Atomicity

Ensures transaction is all-or-nothing.

Recovery

Performed after:

Crash

System failure

Transaction abort

Log-Based Recovery

Log Records

<T, X, old, new>

Phases of Recovery

1. Undo Phase

For uncommitted transactions

Use old values from logs

2. Redo Phase

For committed transactions

Use new values from logs

Techniques

1. Deferred Update

Only redo required

2. Immediate Update

Both redo and undo required

.