PVP 23

Code: 23CS3402, 23IT3402, 23AM3402, 23DS3402

II B.Tech - II Semester – Regular / Supplementary Examinations APRIL 2026

DATABASE MANAGEMENT SYSTEMS
(Common for CSE, IT, AIML, DS)

Duration: 3 hours Max. Marks: 70

Note: 1. This question paper contains two Parts A and B.
2. Part-A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question carries 10 marks.
4. All parts of Question paper must be answered in one place.

BL – Blooms Level CO – Course Outcome

PART – A

		BL	CO
1.a)	Give one example of a database application used in banking.	L2	CO1
1.b)	Identify the component responsible for data security in DBMS**.**	L2	CO1
1.c)	Predict the attributes to describe an Employee entity.	L2	CO2
1.d)	Give an example of a relationship type with degree greater than two**.**	L2	CO2
1.e)	Write an SQL statement to create a table named Employee with attributes EmpID, Name and Salary.	L2	CO2
1.f)	Distinguish between DELETE and UPDATE commands in SQL.	L2	CO2
1.g)	Describe Partial Dependency.	L2	CO3

1.h)	Discuss the need for normalization.	L2	CO3
1.i)	Distinguish Commit and Rollback operations.	L2	CO1
1.j)	Identify the type of lock required for read and write operations in a database.	L2	CO1

PART – B

			BL	CO	Max. Marks
UNIT-I					
2	a)	Compare Database Systems and File Systems in terms of data redundancy, security and data sharing.	L2	CO1	5 M
	b)	Illustrate how three-tier architecture provides data independence**.**	L2	CO1	5 M
OR					
3	a)	Explain the difference between Controlled and Un-controlled Redundancy.	L2	CO1	3 M
	b)	Explain the working of different types of client-server architectures with neat diagrams.	L2	CO1	7 M
UNIT-II					
4	a)	Apply ER-to-relational mapping to convert a simple ER diagram into relational tables.	L3	CO2	5 M
	b)	Analyze the differences between strong entities and weak entities with examples.	L4	CO4	5 M
OR					
5	a)	Develop an ER diagram for a hospital management database with appropriate keys.	L3	CO2	5 M
	b)	Evaluate the importance of structural constraints in maintaining data integrity**.**	L4	CO4	5 M

		UNIT-III			
6	a)	Apply the relational model concepts to explain domain, attribute, tuple and relation with examples.	L3	CO2	5 M
	b)	Consider the following tables and write queries in SQL. Customer (Cust_ID, Name, Email, Address) Product (Prod_ID, PName, Category, Unitprice) Order (Ord_ID, Cust_ID, Prod_ID, ODate) i) List all the customers staying in 'London' or 'Africa' or 'Dallas'. ii) Find all the orders placed between "01-12-2025" to "31-03-2026". iii) Change the PName of 'P105' product to "Jeans". iv) Count all the orders placed by the Customer "John". v) List the Products with unit price greater than the average unit price of all the products.	L3	CO2	5 M
		OR			
7	a)	Demonstrate the implementation of different types of joins (Inner Join, Left Join, Right Join)**.**	L3	CO2	5 M
	b)	Apply SQL queries to create and use views in a relational database**.**	L4	CO4	5 M
		UNIT-IV			
8	a)	Apply normalization techniques to convert a relation from unnormalized form to 3NF.	L3	CO3	6 M

	b)	Analyze the importance of Fourth Normal Form in eliminating redundancy.	L4	CO4	4 M
OR					
9	a)	Determine the candidate keys of the following relation R(A,B,C,D,E) with given dependencies and decompose the relation into suitable normal form. AB→C, CD→E, DE→B.	L3	CO3	5 M
	b)	Apply normalization rules to convert a relation containing join dependencies into 5NF.	L3	CO3	5 M
UNIT-V					
10	a)	Illustrate how concurrency control protocols ensure database consistency.	L3	CO1	5 M
	b)	Apply ACID properties to explain how a database transaction maintains reliability in real-world applications such as banking systems.	L3	CO1	5 M
OR					
11	a)	Demonstrate the role of conflict serializable schedules in concurrent execution of transactions.	L3	CO1	5 M
	b)	Interpret how immediate update recovery differs from deferred update recovery.	L3	CO1	5 M

P.V.P.SIDDHARTHA INSTITUTE OF TECHNOLOGY

II B.Tech – II Semester – Regular Examinations – April, 2026

DATABASE MANAGEMENT SYSTEMS (23CS3402)

(COMPUTER SCIENCE & ENGINEERING)

Short Scheme of Valuation

PART – A

1. a) One example of a database application used in banking. 2M

   b) Component responsible for data security in DBMS. 2M

   c) Attributes to describe an Employee entity 2M

   d) Example of a relationship type with degree greater than two 2M

   e) SQL statement to create Employee table 2M

   f) Any two differences between DELETE and UPDATE 2M

   g) Partial Dependency 2M

   h) Need for Normalization 2M

   i) Any two differences between Commit and Rollba 2M

   j) Types of lock required for read and write operations 2M

   - Read operation: 1M Write operation: 1M

PART – B

UNIT - I

2(a) Compare Database Systems and File Systems: Any 5 differences 5M

2(b) Three-Tier Architecture and Data Independence :
Explanation : 3M Diagram : 2M 5M

3(a) Controlled vs Uncontrolled Redundancy 3M
Redundancy Definintion :1M Differences :2M

3(b) Types of Client-Server Architectures 7M
Types : 2M Explanation :3M Diagrams: 2M

UNIT - II

4(a) ER-to-Relational Mapping **5M**
Mapping Steps: 3M Example :2M

4(b) Strong vs Weak Entity **5M**
Differences :4M Example:1M

5 a) ER Diagram for Hospital Management System **5M**
ER diagram: 5M

5(b) Importance of Structural Constraints in maintaining data integrity **5M**
Types: 1M Explanation: 4M

UNIT – III

6(a) Relational Model Concepts **5M**
Domain: 1M Attribute:1M Tuple ;1M Relation:2M

6(b) SQL Queries **5M**
Each Query:1M

7(a) Types of Joins **5M**
Definitions: 2M Examples:3M

7(b) Views in SQL **5M**
Definition:1M Examples:4M

UNIT–IV

8 a) Apply normalization techniques to convert a relation from unnormalized form to 3NF. **6M**
Third Normal Form (3NF) Definition: 2M Example:4M

8 b) Importance of fourth normal form in eliminating redundancy **4M**
Fourth Normal Form (4NF) Definition: 2M Importance:2M

.9 a) Candidate Keys and Normalization **5M**
Finding Candidate Keys: 3M Normalization:2M

9(b) Fifth Normal Form (5NF) **5M**
5NF Definintion: 3M Explanation:2M

UNIT - V

10(a) Concurrency Control Protocols **5M**
Explanation of 2PL: 5M

10(b) ACID Properties in Banking **5M**
ACID properties : 3M Banking Example: 2M

11(a) Conflict Serializable Schedules **5M**
Conflict Serializability definintion:2M Explanation:3M

11(b) Immediate vs Deferred Update Recovery **5M**
Definintions: 2M Differences:3M

P.V.P.SIDDHARTHA INSTITUTE OF TECHNOLOGY

II B.Tech – II Semester – Regular Examinations – April, 2026

DATABASE MANAGEMENT SYSTEMS (23CS3402)

(COMPUTER SCIENCE & ENGINEERING)

Scheme of Valuation

PART – A

1. a) One example of a database application used in banking. 2M

A common database application used in banking is to manage customer accounts, transactions, loans, and deposits.

b) Component responsible for data security in DBMS. 2M

The **Authorization and Authentication subsystem** (Security Manager) is responsible for data security in a DBMS. It controls user access through permissions, roles, and privileges.

c) Attributes to describe an Employee entity 2M

EmpID , Name, Age, Gender, Address , Salary, DOB etc.,

d) Example of a relationship type with degree greater than two 2M

A ternary relationship (degree = 3) such as: Supplier supplies Part to Project

e) SQL statement to create Employee table 2M

```
CREATE TABLE Employee (EmpID INT PRIMARY KEY,
                       Name VARCHAR(50), Salary DECIMAL(10,2));
```

f) Difference between DELETE and UPDATE 2M

DELETE:
- Removes rows from a table
- Can delete all or selected rows
- Uses `WHERE` to filter rows
- Example: `DELETE FROM Employee WHERE EmpID=1;`

UPDATE:
- Modifies existing data
- Changes specific column values
- Uses `SET` with `WHERE`
- Example: `UPDATE Employee SET Salary=50000 WHERE EmpID=1;`

g) Partial Dependency **2M**

Partial dependency occurs when a non-prime attribute depends on only part of a composite primary key. It violates Second Normal Form (2NF).

h) Need for Normalization **2M**

Normalization is required to:

- Reduce data redundancy (eliminate duplicate data)
- Avoid anomalies (insertion, deletion, update anomalies)
- Ensure data consistency
- Improve data integrity by organizing data into well-structured relations

i) Difference between Commit and Rollback **2M**

Commit: Permanently saves all changes made in a transaction
Makes changes visible to other users
Cannot be undone after execution
Used when transaction is successful

Rollback: Undoes changes made in a transaction
Restores database to previous consistent state
Can be used before commit

j) Type of lock required for read and write operations in a database **2M**

- Read operation:
  Uses a Shared Lock (S-lock)
  Allows multiple transactions to read simultaneously
- Write operation:
  Uses an Exclusive Lock (X-lock)
  Prevents other transactions from reading or writing the data

PART – B

UNIT - I

2(a) Compare Database Systems and File Systems **5M**

Aspect	Database System (DBMS)	File System
Data Redundancy	Controlled and minimized using normalization	High redundancy due to separate files
Data Consistency	Maintained through constraints and transactions	Difficult to maintain consistency
Data Sharing	Supports multi-user access with Concurrency control	Limited sharing, often leads to conflicts

Security	Strong security	Basic or limited Security

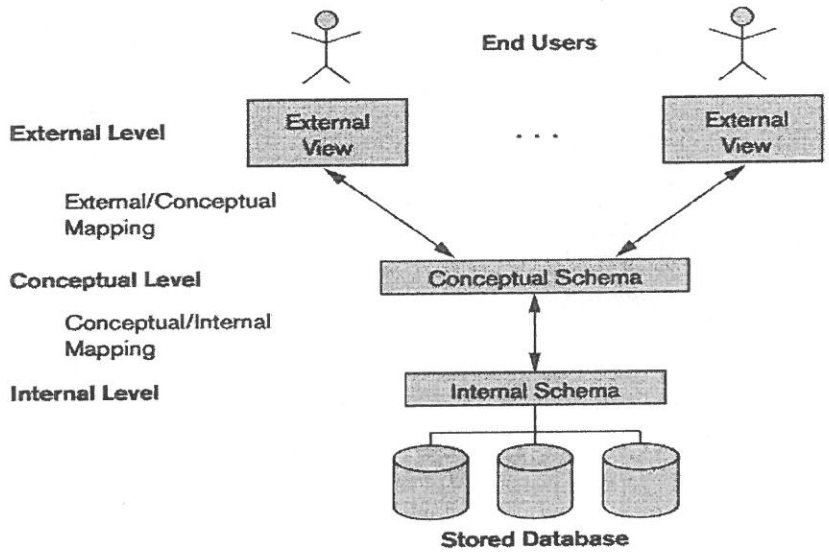
2(b) Three-Tier Architecture and Data Independence **5M**

In this architecture, schemas can be defined at the following three levels:

The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.

The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.Each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.



The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. Separation of layers ensures that modifications in one layer do not impact others, achieving **data independence**.

Two types of data independence:
Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs.
Physical data independence is the capacity to change the internal schema without having to change the conceptual schema.

3(a) Controlled vs Uncontrolled Redundancy **3M**

Controlled Redundancy:

- duplication of data is manageable

- Improves performance (e.g., indexing)
- Managed by DBMS

Uncontrolled Redundancy:

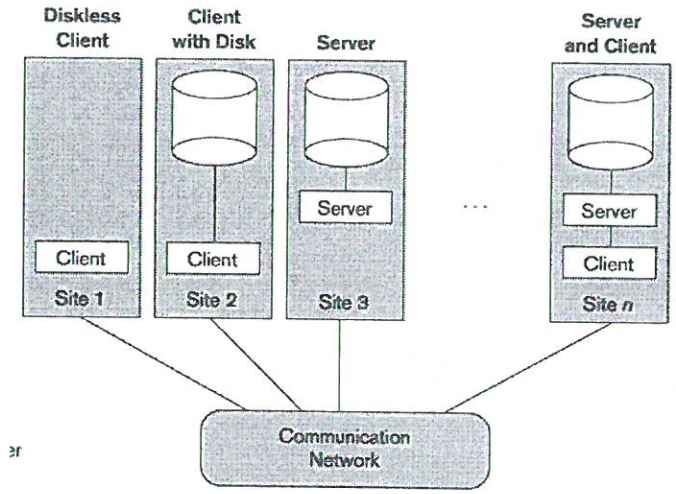
- Unnecessary duplication
- Causes inconsistency
- Not managed properly
- Same data stored in multiple files

3(b) Types of Client-Server Architectures **7M**

. 1. Two-Tier Architecture

- Client directly communicates with database server
- Client handles User Interface and application programs
- The query and transaction functionality related to SQL processing remained on the server side.

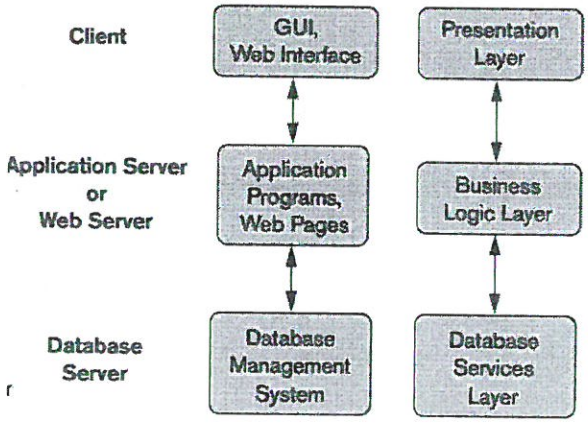
The user interface programs and application programs can run on the client side. When DBMS access is required, the program establishes a connection to the DBMS (which is on the server side); once the connection is created, the client program can communicate with the DBMS. A standard called Open Database (ODBC) provides an application programming interface (API), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed.



2. Three-Tier Architecture:

Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server,

- Separates User Interface, Application Programs and database



This intermediate layer or middle tier is called the application server or the Web server, depending on the application. This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server. It can also improve database security by checking a client's credentials before forwarding a request to the database server. Clients contain GUI interfaces and some additional application-specific business rules. The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server.

3. N-Tier Architecture:

This Architecture has Multiple layers (web server, app server, etc.). Highly scalable and Suitable for large systems.

Client → Web Server → App Server → DB Server

UNIT - II

4(a) ER-to-Relational Mapping **5M**

Step 1: Mapping Strong (Regular) Entities

- Create a separate relation (table) for each strong entity.
- Include all **simple attributes** of the entity.
- Choose the **primary key (PK)**

Step 2: Mapping Weak Entities

Create a relation for the weak entity.

Include:

- All attributes of the weak entity
- **Primary key of the owner entity** (as a foreign key)
- The **primary key** will be a **composite key**: Owner's PK + Partial key

Step 3: Mapping Simple and Composite Attributes

- **Simple attributes** → Directly become columns
- **Composite attributes** → Break into components

Step 4: Mapping Multivalued Attributes

- Create a **separate relation** for each multivalued attribute.
- Include:
  - Attribute value
  - Owner entity's primary key as FK
- Composite PK = (Owner PK + Attribute)

Step 5: Mapping Binary Relationships

(a) One-to-One (1:1)

- Add a **foreign key** to either relation (preferably with total participation).

(b) One-to-Many (1:M)

- Add **foreign key** on the **many side**.

(c) Many-to-Many (M:N)

- Create a **new relation** for the relationship.
- Include:
  - PKs of both entities as FKs
  - Any relationship attributes
- Composite PK = (both FKs)

Step 6: Mapping N-ary Relationships (Degree > 2)

- Create a separate relation.
- Include PKs of all participating entities.
- Add relationship attributes.
- Composite PK includes all FKs.

4(b) Strong vs Weak Entity **5M**

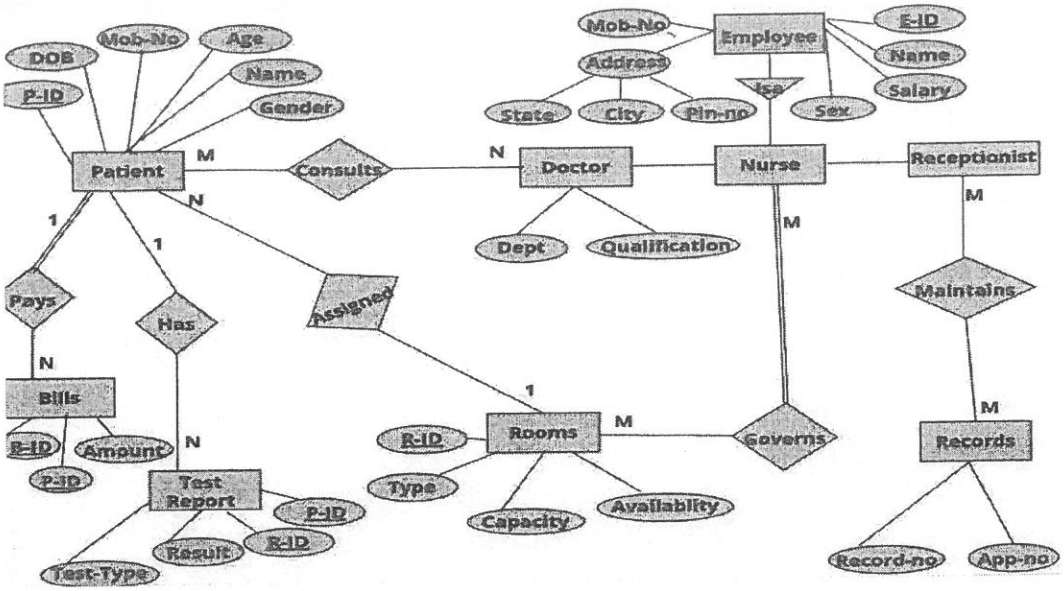
Strong Entity:

- Has its own primary key
- Exists independently
- Represented by single rectangle
- Key attribute underlined

Weak Entity:

- No primary key
- Depends on strong entity
- Represented by Double rectangle
- Partial key defined

5 a) ER Diagram for Hospital Management System **5M**



5(b) Importance of Structural Constraints in maintaining data integrity **5M**

Types:

1. **Cardinality Ratio :** Specifies the maximum number of relationship instances that an entity can participate in. (1:1, 1:M, M:N)

2.**Participation Constraints** : Total or partial participation

Importance:

- Ensures **data integrity**
- Prevents invalid relationships
- Maintains **consistency and accuracy**
- Helps in **proper database design**
- Avoids redundancy and anomalies

UNIT – III

6(a) Relational Model Concepts **5M**

1. Domain

- A **domain** is the set of permissible values for an attribute.
- It defines the **data type and constraints**.

Example: Domain of Age → Integer values (18–60)

2. Attribute

- An **attribute** is a column in a table representing a property of an entity.

Example: Student(**SID**, Name, Age) → Name, Age are attributes

3. Tuple

- A **tuple** is a row (record) in a relation.

Example: (101, "Ravi", 20) is a tuple of Student

4. Relation

- A **relation** is a table consisting of rows (tuples) and columns (attributes).

SID	Name	Age
101	Ravi	20
102	Sita	21

6(b) SQL Queries **5M**

Given Tables: Customer(Cust_ID, Name, Email, Address)
Product(Prod_ID, PName, Category, Unitprice)
Orders(Ord_ID, Cust_ID, Prod_ID, ODate)

i) SELECT * FROM Customer WHERE Address IN ('London', 'Africa', 'Dallas');
ii) SELECT * FROM Orders WHERE ODate BETWEEN '2025-12-01' AND '2026-03-31';
iii) UPDATE Product SET PName = 'Jeans' WHERE Prod_ID = 'P105';
iv) SELECT COUNT(*) FROM Orders JOIN Customer ON Ooder.Cust_ID = Customer.Cust_ID
where Customer.Name = 'John';
v) SELECT * FROM Product WHERE Unitprice > (SELECT AVG(Unitprice) FROM Product);

7(a) Types of Joins **5M**

1. Inner Join: Returns only matching records.

2. Left Join: Returns all records from left table + matching from right.

3. Right Join: Returns all records from right table + matching from left.

Note: Necessary examples must be given

7(b) Views in SQL **5M**

A view **is a** virtual table **derived from one or more base tables.**

- It **does not store data physically** (except materialized views).
- It stores only the **query definition**.
- Data is retrieved dynamically when the view is accessed.

Purpose of Views

- Provide **data abstraction**
- Enhance **security** (restrict access to certain columns/rows)
- Simplify **complex queries**
- Present **customized data** to users

Syntax:

```
CREATE VIEW view_name AS SELECT column1, column2, ... FROM table_name
                            WHERE condition;
```

Note: Necessary example to create and use a view must be given

UNIT–IV

8 a) Apply normalization techniques to convert a relation from unnormalized form to 3NF. 6M

Normalization is the process of organizing data in a database to:

- Reduce **redundancy**
- Eliminate **anomalies** (insertion, deletion, update)
- Improve **data integrity**

Third Normal Form (3NF):

- Must be in 2NF
- Remove **transitive dependency**
- Non-key attributes must not depend on other non-key attributes

Advantages of 3NF

- Eliminates redundancy
- Removes anomalies
- Ensures data consistency
- Improves database design

Necessary example must be given

8 b) Importance of fourth normal form in eliminating redundancy 4M

A relation is in **Fourth Normal Form (4NF)** if:

- It is in **BCNF**, and
- It has **no non-trivial multivalued dependencies (MVDs)**

A **multivalued dependency** occurs when one attribute determines **multiple independent values** of another attribute. The presence of multivalued dependencies in a relation leads to several problems. It results in **data duplication**, where the same information is stored repeatedly across multiple tuples, increasing storage requirements and inconsistency risk. It also causes **insertion anomalies**, as adding a new value (such as a course or hobby) may require repeating existing data unnecessarily. **Update anomalies** arise because a single change must be applied in multiple rows, increasing the chance of errors and inconsistencies. Additionally,

deletion anomalies can occur, where removing a tuple may unintentionally lead to the loss of other important information.

Fourth Normal Form (4NF) plays a crucial role in improving database design by eliminating redundancy that arises from independent multivalued attributes. By decomposing such relations, it removes unnecessary data repetition and thereby helps prevent insertion, deletion, and update anomalies. This decomposition also ensures better data independence by separating unrelated data into different tables, making the structure more logical and easier to maintain. Furthermore, 4NF enhances data integrity by maintaining consistency and correctness across the database. As a result of reduced duplication, it also leads to more efficient storage utilization.

Necessary example can be considered.

9 a) Candidate Keys and Normalization **5M**

Given: Relation: R(A, B, C, D, E)
Functional Dependencies:

- AB → C
- CD → E
- DE → B

Finding Candidate Keys:

We compute attribute closures.

(A, B)+ - AB → C ⇒ {A, B, C} CD → E (need D, not available)
✗ Not a key

(A, D)+ - Start: {A, D} Need B, C, E

Try adding dependencies: DE → B (need E first ✗) CD → E (need C ✗)
✗ Not a key

(A, C, D)+ - Start: {A, C, D} CD → E ⇒ {A, C, D, E} DE → B ⇒ {A, B, C, D, E}

Covers all attributes. Candidate Key = (A, C, D)

(A, B, D)+ - Start: {A, B, D}

Apply the given functional dependencies:

1. AB → C
   Since A and B are present → add C
   ⇒ {A, B, C, D}
2. CD → E
   C and D are present → add E
   ⇒ {A, B, C, D, E}
3. DE → B
   Already included, no change

$(A,B,D)^+ = \{A, B, C, D, E\}$ ABD is a superkey for the relation.

Keys for the relation : **(A,B,C) or (A,B,D)**

Check Normal Form:

- The relation is **not in 2NF** due to partial dependency (**CD → E**)
- The relation is **not in 3NF** due to transitive/non-key dependency (**CD → E**)

Final Decomposed Relations :

- R1(A, B, C)
- R2(C, D, E)
- R3(D, E, B)

9(b) Fifth Normal Form (5NF) **5M**

A relation is in **5NF (Project-Join Normal Form)** if:

- It has **no non-trivial join dependencies**
- It cannot be decomposed further **without loss of information**

Steps to Convert to 5NF

1. Identify **join dependencies (JD)**
2. Check if JD is implied by candidate keys
3. If not, decompose into smaller relations
4. Ensure **lossless join**

Necessary example can be considered

UNIT - V

10(a) Concurrency Control Protocols **5M**

Concurrency control ensures that **multiple transactions execute safely** without violating consistency.

Key Problems Without Control: *Lost Update , Dirty Read , Unrepeatable Read*

Two-Phase Locking (2PL)

- Growing phase → acquire locks
- Shrinking phase → release locks

Ensures **conflict serializability.** The protocols ensure: Isolation, Consistency, Correct concurrent execution

Necessary Example can be considered

10(b) ACID Properties in Banking **5M**

ACID properties ensure the reliability and correctness of database transactions, especially in real-world

applications such as banking systems.

Atomicity guarantees that a transaction is treated as a single unit, meaning either all operations are completed successfully or none are applied; for example, during a fund transfer, both debit and credit operations must occur together.
Consistency ensures that the database moves from one valid state to another, maintaining all defined rules and constraints, such as preserving the total balance across accounts.
Isolation ensures that concurrent transactions do not interfere with each other, so intermediate results of one transaction are not visible to others, thereby preventing inconsistencies.
Durability guarantees that once a transaction is committed, its changes are permanently stored in the database, even in the event of system failures. Together, these properties maintain data integrity, accuracy, and reliability in critical systems like banking.

Example (Bank Transfer):

Transfer ₹100 from A → B: Debit A Credit B

If failure occurs Rollback ensures consistency. ACID properties ensure **reliability and correctness** in real-world systems.

11(a) Conflict Serializable Schedules **5M**

A schedule is **conflict serializable** if it can be transformed into a **serial schedule** by swapping non-conflicting operations.

Conflicting Operations: *Read–Write , Write–Read , Write–Write*

To determine whether a schedule is conflict serializable, a **precedence (serialization) graph** is constructed where each node represents a transaction and directed edges indicate conflict dependencies between them. If the graph contains **no cycles**, the schedule is conflict serializable and guarantees correctness and consistency similar to serial execution. If a cycle exists, the schedule is not conflict serializable and may lead to inconsistent results. Thus, conflict serializability plays a crucial role in ensuring safe and reliable concurrent transaction execution in database systems.

Necessary Example can be considered

11(b) Immediate vs Deferred Update Recovery **5M**

Immediate Update Recovery

In the **immediate update** approach, changes made by a transaction are **written to the database before the transaction commits**. This means the database may contain **uncommitted data** (also called dirty data).

- A **Write-Ahead Logging (WAL)** protocol is used:
  - Changes are first recorded in a log before being applied to the database.
- If a failure occurs:
  - **UNDO** is required for uncommitted transactions (to remove partial effects)
  - **REDO** is required for committed transactions (to ensure durability)

Characteristics:

- Updates are applied immediately
- Both **UNDO and REDO** operations are needed
- Faster execution since data is written early

- More complex recovery process

Deferred Update Recovery

In the **deferred update** approach, changes made by a transaction are **not written to the database until the transaction commits**. All updates are stored in a **log buffer** during execution .Only after a successful commit are changes applied to the database

Characteristics:

- No uncommitted data is written to the database
- Only **REDO** is required (no need for UNDO)
- Simpler recovery process
- Slightly slower commit phase