CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 1 – Feb 2025

Sub:	Database Management Systems						Sub Code:	MMC103
Date:	06.02.25	Duration:	90 min's	Max Marks:	50	Sem: I	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	RBT
1	Explain the three-schema architecture of a DBMS with a neat diagram. Discuss the importance of each schema layer in managing database systems.	[5+5]	CO1	L1
	OR			
2	Define the concept of keys in a relational database. Explain the differences between candidate keys, primary keys, and foreign keys with examples.	[4+6]	CO1	L2
	PART II			
3	A library database has the following tables: Books(BookID, Title, Author, Genre), Members(MemberID, Name, Age) and BorrowedBooks(MemberID, BookID). Apply relational algebra and find the following: i. Retrieve all books in the "Education" genre. ii. Find all unique members who have either borrowed a book or are registered as library members. iii. Retrieve the titles of books that have not been borrowed.	[3+4+3]	CO2	L3
	OR			
4	Draw an ER diagram for a university system with the following specifications: • Entities: Employees, Departments, Projects, Equipments. • Relationships: Employees work in Departments, Departments manage Projects, Employees are assigned to Projects, Projects use Equipment.	[10]	CO3	L3
	PART III			
5	Explain aggregate functions with examples. Consider the following schema: Student (usn, name, DOB, branch, mark1, mark2, mark3, total, GPA). Write SQL query to find the maximum GPA score of the student branch-wise.	[5+5]	CO3	L3
	OR			
6	What is a trigger? Explain with the syntax of a trigger. Write a trigger named LogSalaryUpdate that logs every salary update in the table AuditLog (ActionID, ActionType, EmployeeID, ActionTime).	[4+6]	CO3	L3
	PART IV			
7	You have an Employee table with the following columns: EmployeeID, Name, Dept_Name, and Salary. Create a view named HighEarningEmployees that includes only employees with a salary greater than 70,000. Write a query to display all data from the HighEarningEmployees view.	[10]	CO3	L3
	OR			
8	You have the following tables: Student(StudentID, Name, Age, Major), Course(CourseID, CourseName, Credits) and Enrollment(StudentID, CourseID, Grade). Create and test a procedure named CalculateAverageGrade that calculates the average grade for a specific student. The procedure should accept the StudentID as input and print the average grade.	[10]	CO3	L3
	PART V			
9	What is functional dependency? Explain partial dependency and transitive dependency using examples.	[2+4+4]	CO1	L2
	OR			
10	Explain 3NF with an example. How does it differ from BCNF? Why is 3NF preferred in some cases?	[4+3+3]	CO2	L2

1 Explain the three-schema architecture of a DBMS with a neat diagram. Discuss the importance of each schema layer in managing database systems.

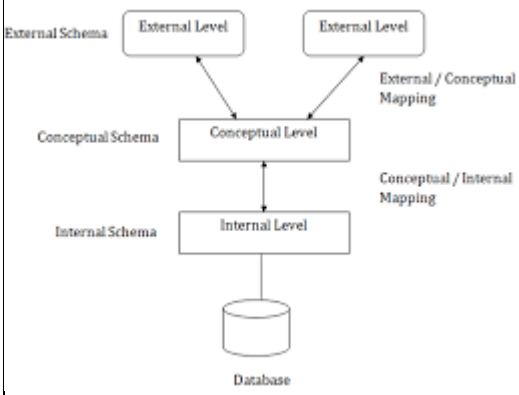
The three-schema architecture of a Database Management System (DBMS) is a framework designed to separate the user's view of the database from the physical storage. It helps in managing data abstraction and independence, making databases more flexible and easier to maintain. The architecture consists of three layers: Internal Schema, Conceptual Schema, and External Schema.

1. Internal Schema (Physical Level):
This is the lowest level, dealing with how data is physically stored in the database. It includes storage details, indexing, data compression, and access paths. The internal schema ensures efficient data storage and retrieval while managing physical data structures.

2. Conceptual Schema (Logical Level):
The middle layer provides a logical view of the entire database, focusing on data structure, relationships, constraints, and security without considering how the data is stored. It represents the core of the database design and defines what data is stored and how it's interrelated.

3. External Schema (View Level):
The top layer defines how different users interact with the database through customized views. It allows multiple user-specific views without altering the conceptual schema, ensuring data security and simplifying user interaction.

Importance of Each Schema Layer:
Internal Schema: Optimizes data storage, performance, and access efficiency.
Conceptual Schema: Maintains data consistency, integrity, and logical structure.
External Schema: Provides data abstraction, security, and tailored user experiences.



This architecture promotes data independence, allowing changes in one layer without affecting others, thus enhancing database management flexibility.

2 Define the concept of keys in a relational database. Explain the differences between candidate keys, primary keys, and foreign keys with examples.

Concept of Keys in a Relational Database
In a relational database, keys are crucial constraints used to uniquely identify rows (or tuples) in a table. They ensure data integrity and help maintain relationships between different tables. Keys can be classified into various types, with each serving a specific purpose in database design.

1. Candidate Keys
A candidate key is a minimal set of attributes that can uniquely identify each record in a table. A table can have multiple candidate keys, but none of them should have redundant attributes.

Example:
Consider a Student table:

StudentID	Email	Name	PhoneNumber
101	john@email.com	John Doe	9876543210
102	alice@email.com	Alice Smith	8765432109

In this table: StudentID, Email, and PhoneNumber can all uniquely identify a student, making them candidate keys.

2. Primary Key
A primary key is a specific candidate key chosen to uniquely identify records in a table. It must contain unique values and cannot have NULL values.

Example:
Using the same Student table, if we choose StudentID as the unique identifier, it becomes the primary key:
PRIMARY KEY (StudentID)
This ensures no two students can have the same StudentID, and it cannot be NULL.

3. Foreign Key
A foreign key is an attribute in one table that refers to the primary key in another table. It establishes a relationship between two tables, ensuring referential integrity.

Example:
Consider a CourseEnrollment table:

EnrollmentID	StudentID	CourseName
1	101	Database Systems
2	102	Operating Systems

Here:

StudentID in CourseEnrollment is a foreign key that references StudentID (the primary key) in the Student table.
FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
This ensures that only existing students can enroll in courses.

✅ Key Differences

Aspect	Candidate Key	Primary Key	Foreign Key
Definition	Set of attributes that can uniquely identify records.	Chosen candidate key to uniquely identify records.	Attribute linking one table to another.
Uniqueness	Must be unique for each record.	Must be unique and not `NULL`.	Can have duplicate values (refers to primary key).
NULL Values	Cannot contain `NULL` values.	Cannot contain `NULL` values.	Can contain `NULL` values if no relationship exists.
Usage	To identify potential unique identifiers.	To uniquely identify each record in a table.	To maintain referential integrity between tables.
Example	`StudentID`, `Email`, `PhoneNumber`	`StudentID` (chosen as primary key)	`StudentID` in `CourseEnrollment` (refers to `Student` table).

🎯 Conclusion
Candidate Keys: Potential unique identifiers.
Primary Key: The chosen candidate key to enforce uniqueness.
Foreign Key: Maintains relationships between tables.
Together, these keys play a vital role in ensuring data consistency, integrity, and efficient querying in relational databases.

3 A library database has the following tables: Books(BookID, Title, Author, Genre), Members(MemberID, Name, Age) and BorrowedBooks(MemberID, BookID).
Apply relational algebra and find the following:

i. Retrieve all books in the "Education" genre.
ii. Find all unique members who have either borrowed a book or are registered as library members.
iii. Retrieve the titles of books that have not been borrowed.

1. Retrieve all books in the "Education" genre

Relational Algebra Expression:

$$\sigma_{Genre='Education'}(Books)$$

Explanation:

- The **σ (selection)** operator filters rows where the `Genre` attribute is **"Education"** from the `Books` table.

2. Find all unique members who have either borrowed a book or are registered as library members

Relational Algebra Expression:

$$\pi_{MemberID,Name,Age}(Members) \cup \pi_{MemberID}(BorrowedBooks) \bowtie Members$$

Explanation:

- The **π (projection)** operator is used to retrieve member details from both `Members` and `BorrowedBooks`.
- The **∪ (union)** operator combines:
  - All members from the `Members` table.
  - Members who have borrowed books by joining `BorrowedBooks` with `Members` using the common `MemberID`.
- This ensures we get **unique members** (registered or borrowers).

3. Retrieve the titles of books that have not been borrowed

Relational Algebra Expression:

$$\pi_{Title}(Books) - \pi_{Title}(Books \bowtie BorrowedBooks)$$

Explanation:

- The **− (set difference)** operator finds the difference between:
  - All book titles in the `Books` table.
  - Book titles that are present in both `Books` and `BorrowedBooks` (joined using the `BookID`).
- This returns **only the titles of books that have never been borrowed.**

4 Draw an ER diagram for a university system with the following specifications:

- Entities: Employees, Departments, Projects, Equipments.
- Relationships: Employees work in Departments, Departments manage Projects, Employees are assigned to Projects, Projects use Equipment.

5 Explain aggregate functions with examples. Consider the following schema: Student (usn, name, DOB, branch, mark1, mark2, mark3, total, GPA). Write SQL query to find the maximum GPA score of the student branch-wise.

Aggregate Functions in SQL:
Aggregate functions perform calculations on a set of values and return a single summarized result. They are often used with the GROUP BY clause to group rows that share the same values in specified columns. Common aggregate functions include:

1. COUNT() - Returns the number of rows.
2. SUM() - Calculates the total sum of a numeric column.
3. AVG() - Computes the average of numeric values.
4. MAX() - Finds the maximum value in a column.
5. MIN() - Retrieves the minimum value in a column.

Example Using the Student Schema
Table: Student

Attributes: usn, name, DOB, branch, mark1, mark2, mark3, total, GPA

SQL Query to Find the Maximum GPA Score Branch-wise: SELECT branch, MAX(GPA) AS Max_GPA FROM Student GROUP BY branch;
Explanation:
SELECT branch, MAX(GPA): Retrieves the branch name and the maximum GPA for each branch.
AS Max_GPA: Assigns an alias to the output column for better readability.
FROM Student: Specifies the table to query.
GROUP BY branch: Groups the data based on the branch column so that MAX(GPA) is calculated for each branch separately.

6 What is a trigger? Explain with the syntax of a trigger. Write a trigger named LogSalaryUpdate that logs every salary update in the table AuditLog (ActionID, ActionType, EmployeeID, ActionTime).

Trigger in SQL
A trigger is a special type of stored procedure that automatically executes (or fires) in response to specific events on a table or view. These events include INSERT, UPDATE, or DELETE operations. Triggers are used to maintain data integrity, enforce business rules, and log audit information.

Syntax of a Trigger

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
  -- Trigger logic here
END;
```

CREATE TRIGGER: Defines a new trigger.
BEFORE / AFTER / INSTEAD OF: Specifies when the trigger should fire:
BEFORE: Executes before the triggering operation.
AFTER: Executes after the triggering operation.
INSTEAD OF: Replaces the triggering operation (used with views).
INSERT / UPDATE / DELETE: Defines the triggering event.
FOR EACH ROW: Indicates the trigger will execute once for each affected row.
Trigger Logic: The SQL statements to be executed when the trigger fires.

Example: Trigger to Log Salary Updates
Objective: Create a trigger named LogSalaryUpdate that logs every salary update in the AuditLog table.

Table Definitions:

```
-- Employee Table
CREATE TABLE Employee (
  EmployeeID INT PRIMARY KEY,
  Name VARCHAR(100),
  Salary DECIMAL(10, 2)
);

-- AuditLog Table
CREATE TABLE AuditLog (
  ActionID INT PRIMARY KEY AUTO_INCREMENT,
  ActionType VARCHAR(50),
  EmployeeID INT,
  ActionTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Trigger:
CREATE TRIGGER LogSalaryUpdate AFTER UPDATE ON Employee FOR EACH ROW BEGIN -- Check if the Salary has changed IF OLD.Salary <> NEW.Salary THEN INSERT INTO AuditLog (ActionType, EmployeeID, ActionTime) VALUES ('Salary Update', NEW.EmployeeID, NOW()); END IF; END;

7 You have an Employee table with the following columns: EmployeeID, Name, Dept_Name, and Salary. Create a view named HighEarningEmployees that includes only employees with a salary greater than 70,000. Write a query to display all data from the HighEarningEmployees view.

Creating a View: HighEarningEmployees
A view in SQL is a virtual table based on the result of a SELECT query. It helps simplify complex queries, improve security, and present data in a specific format without duplicating it.

```
CREATE VIEW HighEarningEmployees AS SELECT EmployeeID, Name, Dept_Name, Salary FROM Employee WHERE Salary > 70000;

SELECT * FROM HighEarningEmployees;
```

This view filters the Employee table to include only those employees whose salary is greater than 70,000. The SELECT * FROM HighEarningEmployees; query then retrieves all records from this view.

8 You have the following tables: Student(StudentID, Name, Age, Major), Course(CourseID, CourseName, Credits) and Enrollment(StudentID, CourseID, Grade). Create and test a procedure named CalculateAverageGrade that calculates the average grade for a specific student. The procedure should accept the StudentID as input and print the average grade.

```
CREATE PROCEDURE CalculateAverageGrade(IN student_id INT)
BEGIN
   DECLARE avg_grade DECIMAL(5,2);
   -- Calculate the average grade for the given student
   SELECT AVG(Grade) INTO avg_grade FROM Enrollment WHERE StudentID = student_id;
   -- Print the result
   IF avg_grade IS NOT NULL THEN
      SELECT CONCAT('The average grade for Student ID ', student_id, ' is: ', avg_grade) AS Result;
   ELSE
      SELECT 'No grades found for the given Student ID' AS Result;
   END IF;
END

CALL CalculateAverageGrade(101);
```

Explanation:

1. The procedure accepts a StudentID as an input parameter.
2. It calculates the average grade from the Enrollment table for the given StudentID.
3. If the student has grades recorded, it prints the average grade.
4. If no grades exist for that student, it prints a message stating that no grades were found.

9 What is functional dependency? Explain partial dependency and transitive dependency using examples.

Functional dependency is a relationship between attributes in a relational database. It states that if two tuples (rows) have the same values for a set of attributes XXX, then they must also have the same values for another set of attributes YYY. This is represented as:
X→YX \rightarrow YX→Y
Where:

- XXX is called the determinant.
- YYY is called the dependent.
- This means that the value of XXX uniquely determines the value of YYY.

Example of Functional Dependency
Consider a table Student:

StudentID	Name	Department	Dept_HOD
101	Alice	CS	Dr. Smith
102	Bob	IT	Dr. Johnson
103	Charlie	CS	Dr. Smith

Here, Department → Dept_HOD is a functional dependency because each department has exactly one head of department.

Partial Dependency

A partial dependency occurs when a non-prime attribute (an attribute that is not part of the candidate key) is functionally dependent on only a part of the primary key rather than the whole primary key. This usually happens in tables with composite primary keys and is a violation of Second Normal Form (2NF).
Example of Partial Dependency
Consider a table Student_Course:

StudentID	CourseID	CourseName	StudentName
1	C101	DBMS	Alice
2	C102	OS	Bob
3	C101	DBMS	Charlie

- Primary Key: (StudentID, CourseID)
- Dependencies:
  - {StudentID, CourseID} → CourseName ✅ (Valid, since CourseName depends on CourseID)
  - StudentID → StudentName ❌ (Partial dependency, as StudentName depends only on StudentID and not on the whole primary key)

This violates 2NF because StudentName depends only on StudentID and not on the full primary key {StudentID, CourseID}.
✅ Solution: Split into two tables:

1. Student Table (StudentID, StudentName)
2. Course Table (CourseID, CourseName)

Transitive Dependency
A transitive dependency occurs when a non-prime attribute depends on another non-prime attribute instead of depending directly on the primary key. This is a violation of Third Normal Form (3NF).
Example of Transitive Dependency
Consider a table Employee:

EmpID	EmpName	DeptID	DeptName
1	Alice	D01	HR
2	Bob	D02	IT
3	Charlie	D01	HR

- Primary Key: EmpID
- Dependencies:
  - EmpID → DeptID ✅ (Valid, as each employee belongs to one department)
  - DeptID → DeptName ✅ (Valid, as each department has one name)
  - EmpID → DeptName ❌ (Transitive dependency, since DeptName is dependent on DeptID, which in turn depends on EmpID)

This violates 3NF because DeptName is indirectly dependent on EmpID through DeptID.
✅ Solution: Split into two tables:

1. Employee Table (EmpID, EmpName, DeptID)
2. Department Table (DeptID, DeptName)

Key Differences Between Partial and Transitive Dependency

Feature	Partial Dependency	Transitive Dependency
Dependency Type	Only on part of a composite primary key	On a non-key attribute via another non-key attribute
Normalization Violated	2NF	3NF
Example	StudentID → StudentName (ignoring composite key)	EmpID → DeptName (indirect dependency)
Solution	Break into separate tables to remove partial dependency	Break into separate tables to remove transitive dependency

10 Explain 3NF with an example. How does it differ from BCNF? Why is 3NF preferred in some cases?

Third Normal Form (3NF) is a normalization technique used to reduce redundancy in relational databases. A table is said to be in 3NF if:

It is in Second Normal Form (2NF).
It has no transitive dependencies—i.e., all non-key attributes must depend only on the primary key.
Example of 3NF

Consider a table Employee:

EmpID	EmpName	DeptID	DeptName
1	Alice	D01	HR
2	Bob	D02	IT
3	Charlie	D01	HR

Primary Key: EmpID
Functional Dependencies:
EmpID → DeptID ✅ (Valid, as each employee belongs to one department)
DeptID → DeptName ✅ (Valid, as each department has one name)
EmpID → DeptName ❌ (Transitive dependency, since DeptName is dependent on DeptID, which in turn depends on EmpID)
Since DeptName is indirectly dependent on EmpID, it violates 3NF.

Solution: Convert to 3NF
To remove transitive dependency, we split the table into two:

Employee Table

EmpID	EmpName	DeptID
1	Alice	D01
2	Bob	D02
3	Charlie	D01

Department Table

DeptID	DeptName
D01	HR
D02	IT

Now, all non-key attributes depend only on the primary key.

Difference Between 3NF and BCNF
Boyce-Codd Normal Form (BCNF) is a stricter version of 3NF.

Feature	Third Normal Form (3NF)	Boyce-Codd Normal Form (BCNF)
Dependency Rule	No transitive dependency	No partial or transitive dependency
Key Dependency	Non-key attributes must depend only on primary key	Every determinant must be a candidate key
Example of Violation	If a non-key attribute depends on another non-key attribute	If a candidate key depends on another non-prime attribute
Practical Use	Used when a table has a single candidate key	Required when a table has multiple candidate keys

Consider a table Course_Instructor:

CourseID	Instructor	Department
C101	Prof. A	CS
C102	Prof. B	IT
C103	Prof. C	CS

Candidate Keys: {CourseID, Instructor}
Functional Dependencies:
CourseID → Department (One course belongs to one department)
Instructor → Department (Each instructor belongs to one department)

Since Instructor is not a primary key but determines Department, it violates BCNF.

✅ To fix this, we split it into two tables:

Course Table (CourseID, Instructor)
Instructor Table (Instructor, Department)

Why is 3NF Preferred Over BCNF in Some Cases?
While BCNF removes all redundancy, it sometimes results in more table joins, which can reduce query performance.
3NF is preferred when:

Some redundancy is acceptable for faster query execution.
There are functional dependencies that cannot be easily split without losing information.
The table has a single candidate key, making BCNF unnecessary.