CELEBRATING 25 YEARS

CMRIT

CMR INSTITUTE OF TECHNOLOGY, BENGALURU.

ACCREDITED WITH A++ GRADE BY NAAC

Internal Question Paper Scheme & Solution

USN | | | | | | | | | |

Internal Assessment Test I – April 2026

Sub:	DATABASE MANAGEMENT SYSTEM					Sub Code:	BCS403	Branch:	ISE	
Date:	17-04-2026	Duration:	90 min's	Max Marks:	50	Sem/Sec:	IV- A, B, C		OBE	
	Answer any FIVE FULL Questions							MARKS	CO	RBT
1	List and explain the advantages of using the DBMS approach. Explain the different categories of data model.							6 4	**CO2**	**L2**
2	Explain the following terms. 1. Weak Entity 2. Recursive Relationship, 3.Cardinality Ratio, 4.Participation 5.Degree of Relationship What is an attribute? Explain the different types of attributes with examples.							5 5	**CO2**	**L2**
3	What is Functional dependency? Explain the inference rules for functional dependency with proof. Consider two sets of functional dependency. F= {A → C, AC→D, E→AD, E→H} E= {A→CD, E→AH}. Are they Equivalent?							10	**CO4**	**L3**
4	Define Normalization. Explain 1NF, 2NF, and 3NF with examples.							2 8	**CO4**	**L2**
5	Illustrate insert, delete, update, and drop commands in SQL with examples. Explain Aggregate functions in SQL							5 5	**CO3**	**L2**
6	What are database triggers? Explain with syntax and examples and how do they differ from stored procedures?							10	**CO3**	**L2**

Faculty Signature CCI Signature HOD Signature

USN | | | | | | | | | |

Internal Assessment Test I – April 2026

Sub:	DATABASE MANAGEMENT SYSTEM					Sub Code:	BCS403	Branch:	ISE	
Date:	17-04-2026	Duration:	90 min's	Max Marks:	50	Sem/Sec:	IV- A,B,C		OBE	
	Answer any FIVE FULL Questions							MARKS	CO	RBT
1	List and explain the advantages of using the DBMS approach. Explain the different categories of data model.							6 4	**CO2**	**L2**
2	Explain the following terms. 1. Weak Entity 2.Recursive Relationship, 3.Cardinality Ratio, 4.Participation 5.Degree of Relationship What is an attribute? Explain the different types of attributes with examples.							5 5	**CO2**	**L2**
3	What is Functional dependency? Explain the inference rules for functional dependency with proof. Consider two sets of functional dependency. F= {A → C, AC→D, E→AD, E→H} E= {A→CD, E→AH}. Are they Equivalent?							10	**CO4**	**L3**
4	Define Normalization. Explain 1NF, 2NF, and 3NF with examples.							2 8	**CO4**	**L2**
5	Illustrate insert, delete, update, and drop commands in SQL with examples. Explain Aggregate functions in SQL							5 5	**CO3**	**L2**
6	What are database triggers? Explain with syntax and examples and how do they differ from stored procedures?							10	**CO3**	**L2**

Faculty Signature CCI Signature HOD Signature

1. List and explain the advantages of using the DBMS approach. (6M) Explain the different categories of data model. (4M)

Advantages 6m + Different categories of data model 4m

Advantages of Using the DBMS Approach

1. **Controlling Redundancy**

Data redundancy such as tends to occur in the "file processing" approach leads to wasted storage space, duplication of effort and a higher likelihood of the introduction of inconsistency. In the database approach, the views of different user groups are integrated during database design. This is known as data normalization, and it ensures consistency and saves storage Space. However, it is sometimes necessary to use controlled redundancy to improve the performance of queries

A DBMS should provide the capability to automatically enforce the rule that no inconsistencies are introduced when data is updated.

2. Restricting Unauthorized Access

When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. For example, financial data is often considered confidential and only authorized persons are allowed to access such data. In addition, some users may only be permitted to retrieve data, whereas others are allowed to retrieve and update. Hence, the type of access operation retrieval or update must also be controlled. A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts, to specify account restrictions and enforce these restrictions automatically.

3. Providing Persistent Storage for Program Objects

The values of program variables or objects are discarded once a program terminates, unless the programmer explicitly stores them in permanent files, which often involves converting these complex structures into a format suitable for file storage. Object-oriented database systems make it easier for complex runtime objects to be saved in secondary storage so as to survive beyond program termination and to be retrievable at a later time.

Object-oriented database systems are compatible with programming languages such as C++ and Java, and the DBMS software automatically performs any necessary conversions.

4. Providing Storage Structures and Search Techniques for Efficient Query Processing

DBMS maintains indexes that are utilized to improve the execution time of queries and updates. DBMS has a buffering or caching module that maintains parts of the database in main memory buffers.The query processing and optimization module is responsible for choosing an efficient query execution plan for each query submitted to the system.

5. Providing Backup and Recovery

The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing. Disk backup is also necessary in case of a catastrophic disk failure.

6. Providing Multiple User Interfaces

Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include

Query languages for casual users

Programming language interfaces for application programmers

Forms and command codes for parametric users

Menu-driven interfaces and natural language interfaces for standalone users.

7. Representing Complex Relationships among Data

A database may include numerous varieties of data that are interrelated in many ways.

For example each section record is related to one course record and to a number of GRADE_REPORT records one for each student who completed that section. A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

8. Enforcing Integrity Constraints

Most database applications are such that the semantics of the data require that it satisfy certain restrictions in order to make sense. The simplest type of integrity constraint involves specifying a data type for each data item. Another type of constraint specifies uniqueness on data item values, such as every course record must have a unique value for Course_number. This is known as a key or uniqueness constraint. It is the responsibility of the database designers to identify integrity constraints during database design.

9. Permitting Inferencing and Actions Using Rules

In a deductive database system, one may specify declarative rules that allow the database to infer new data. For example, figure out which students are on academic probation. Such capabilities would take the place of application programs that would be used to ascertain such information otherwise.

Active database systems go one step further by allowing "active rules" that can be used to initiate actions automatically. triggers with tables.

Explain the different categories of data model. (4M)

Data models can be categorized according to the types of concepts they use to describe the database structure.

1. High-level or conceptual data models: provide concepts that are close to the way many users perceive data. Conceptual data models use concepts such as entities, attributes, and relationships.

2. Representational or implementation data models: provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly. Representational or implementation data models are the models used most frequently

In traditional commercial DBMSs. These include the widely used relational data model, as well as the so-called legacy data models the network and hierarchical models. Representational data models represent data by using record structures and hence are sometimes called record-based data models.

3. Low-level or physical data models: provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks. Physical data models describe how data is stored as files in the computer by representing information such as record formats, record orderings, and access paths.

2. Explain the following terms. 1. Weak Entity 2. Recursive Relationship, 3. Cardinality Ratio, 4. Participation 5. Degree of Relationship (5M) Definition 5m each

What is an attribute? Explain the different types of attributes with examples. (5M)

Definition 1m + Different types 4m

Weak Entity

A weak entity is an entity that does not have a primary key of its own and is identified using a partial key along with the primary key of a related strong entity.

Recursive Relationship

A recursive relationship in a DBMS occurs when an entity type has a relationship with itself, rather than with a different entity. It represents hierarchical structures (like employee-manager) where the same table entity plays different roles (e.g., manager and subordinate).

Cardinality Ratio

The cardinality ratio specifies the maximum number of entity instances that can participate in a relationship. Common cardinalities are one-to-one (1:1), one-to-many (1:N), and many-to-many (M:N).

Participation
Participation Constraints are rules that govern the minimum and maximum number of entities or relationships that must or may participate in a particular relationship. Within the database structure, these restrictions uphold business standards and guarantee data integrity. Comprehending participation limits is essential to creating successful and efficient databases that faithfully replicate real-world situations.

Degree of a Relationship
The degree of a relationship indicates the number of entity types participating in a relationship. For example, unary (1), binary (2), and ternary (3) relationships.

What is an attribute? Explain the different types of attributes with examples. (5M)

Definition 1m + Different types 4m

An attribute in a database is a property or characteristic that describes an entity (a table or object). They are represented as columns in a table or ovals in Entity-Relationship (ER) diagrams, defining the structure of the data stored.

Attribute Type	Description	Example
Simple (Atomic)	Cannot be divided further	Age, Name
Composite	Can be divided into subparts	Name → First, Last
Derived	Computed from other attributes	Age (from DOB)
Multi-valued	Multiple values for a single entity	Phone_Numbers, Emails
Single-valued	Only one value for each entity	Roll_Number
Stored	Physically stored in DB	Date_of_Birth
Key	Uniquely identifies each entity	Student_ID, Emp_ID

3. What is Functional dependency? Explain the inference rules for functional dependency with proof. (5M)

Definition Functional Dependency 1m + Interference Rules 4m

Consider two sets of functional dependency. F= {A → C, AC→D, E→AD, E→H}
E= {A→CD, E→AH}. Are they Equivalent? **(5M)**

Functional Dependency

A **functional dependency (FD)** is a constraint between attributes of a relation.

For a relation **R**, **X → Y** holds if:

For any two tuples, if they agree on X, they must also agree on Y.

X = determinant

Y = dependent

Example: RollNo → Name

Inference Rules (Armstrong's Axioms)

1. Reflexivity

If Y ⊆ X, then X → Y
Subset of X must have same values if X is same.

2. Augmentation

If X → Y, then XZ → YZ
Adding same attributes (Z) to both sides preserves dependency.

3. Transitivity

If X → Y and Y → Z, then X → Z
Dependency flows through Y.

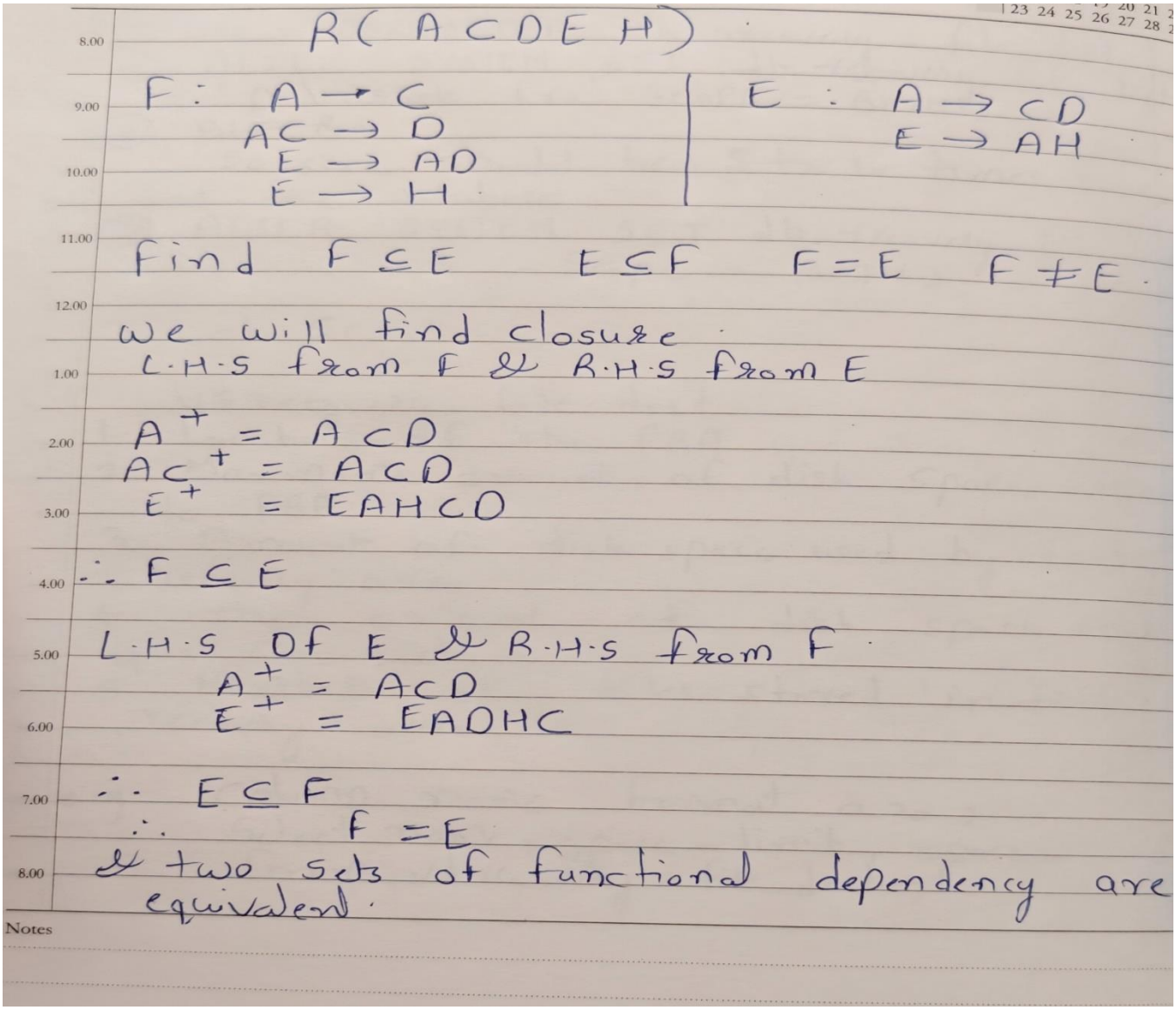
Derived Rules

Union: If X → Y and X → Z, then X → YZ

Decomposition: If X → YZ, then X → Y and X → Z

Pseudotransitivity: If X → Y and WY → Z, then WX → Z

Consider two sets of functional dependency. F= {A → C, AC→D, E→AD, E→H}
E= {A→CD, E→AH}. Are they Equivalent?



4. Define Normalization. Explain 1NF, 2NF, and 3NF with examples. (10 M)

Definition:

Normalization is the process of organizing the attributes and relations of a database to reduce redundancy and avoid anomalies (insertion, deletion, update). It divides large tables into smaller, well-structured tables while preserving data dependencies.

1. Eliminate redundant data.

2. Ensure data consistency.

3. Simplify data maintenance.

1NF

- A relation is in 1NF if all attributes are atomic (no repeating groups or arrays).

- Each column contains only single values.

StudentID	StudentName	Subjects
101	Alice	Math, Physics
102	Bob	Chemistry

Convert to 1NF:

StudentID	StudentName	Subject
101	Alice	Math
101	Alice	Physics
102	Bob	Chemistry

2NF

A relation is in 2NF if:

1. It is in 1NF, and

2. Every non-prime attribute is fully functionally dependent on the whole primary key

(no partial dependency).

Key Concept:

● Partial dependency occurs when a non-key attribute depends only on part of a

composite key.

Example (1NF Table with Composite Key):

Example (1NF Table with Composite Key):

StudentID	CourseID	StudentName	CourseName
101	C1	Alice	Math
101	C2	Alice	Physics
102	C1	Bob	Math

Composite Key: (StudentID, CourseID)

Partial Dependency: StudentName depends only on StudentID.

Solution: Split into two tables:

Student Table (2NF):

StudentID	StudentName
101	Alice
102	Bob

Course Table (2NF):

StudentID	CourseID	CourseName
101	C1	Math
101	C2	Physics
102	C1	Math

Third Normal Form (3NF)

Definition:
A relation is in 3NF if:

1. It is in **2NF**, and
2. There is **no transitive dependency** (non-key attribute depends on another non-key attribute).

Key Concept:

- Transitive dependency occurs when **A → B → C** (B is non-key).

Example (2NF Table):

EmpID	EmpName	DeptID	DeptName
E1	John	D1	Sales
E2	Alice	D2	HR

- **Transitive Dependency:** EmpID → DeptID → DeptName
- **Solution:** Split into two tables:

Employee Table (3NF):

EmpID	EmpName	DeptID
E1	John	D1
E2	Alice	D2

Department Table (3NF):

DeptID	DeptName
D1	Sales
D2	HR

5. Illustrate insert, delete, update, and drop commands in SQL with examples. (5M)

Explain Aggregate functions in SQL (5M)

In SQL, commands are categorized based on their function. **INSERT**, **UPDATE**, and **DELETE** are **Data Manipulation Language (DML)** commands used to manage the data within tables (5marks)

INSERT Statement

Adds new rows (tuples) into a table.

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

INSERT INTO EMPLOYEE (SSN, Fname, Lname, Dno, Salary)
VALUES ('E101', 'Alice', 'Smith', 5, 50000);

DROP Statement

DROP TABLE: Removes an entire table including its structure, data, and constraints.

Syntax: DROP TABLE table_name;

DROP DATABASE: Deletes a database and all objects inside it, such as tables and views.

Syntax: DROP DATABASE database_name;

UPDATE Statement-Modifies existing data in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
UPDATE EMPLOYEE
SET Salary = 55000
WHERE SSN = 'E102';

UPDATE EMPLOYEE
SET Salary = Salary * 1.1
WHERE Dno = 5;
```

Aggregate Functions in SQL **5 marks**

1. COUNT(): Returns the number of rows that match a specified criterion.
2. COUNT(*): Counts every row in the table, including those with NULL values.
3. COUNT(column_name): Counts only the non-NULL values in that specific column.
4. SUM(): Calculates the total sum of a numeric column. It ignores NULL values during the calculation.
5. AVG(): Computes the average (mean) value of a numeric column. Like SUM, it ignores NULL values.
6. MIN(): Identifies the smallest value in a column. It can be used on numeric, string, or date data types.
7. MAX(): Identifies the largest value in a column. It also works with various data types.

6. What are database triggers? Explain with syntax and examples and how do they differ from stored procedures? (10M)

Database Triggers

A **database trigger** is a special type of stored program that is **automatically executed** (fired) in response to certain events on a table or view.

Triggers are commonly used for:

- Enforcing business rules
- Maintaining audit logs
- Ensuring data integrity

Types of Triggers

1. **BEFORE Trigger** – Executes before the event (INSERT, UPDATE, DELETE)

2. **AFTER Trigger** – Executes after the event

3. **INSTEAD OF Trigger** – Executes instead of the actual operation (used with views)

Syntax (General Form)

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
  -- trigger body (SQL statements)
END;
```

Example 1: BEFORE INSERT Trigger

```
CREATE TRIGGER check_salary
BEFORE INSERT ON Employee
FOR EACH ROW
BEGIN
  IF NEW.salary < 0 THEN
    SET NEW.salary = 0;
  END IF;
END;
```

This trigger ensures that salary cannot be negative before inserting a record.

Example 2: AFTER DELETE Trigger (Audit)

```
CREATE TRIGGER delete_log
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
  INSERT INTO Employee_Audit(emp_id, action)
  VALUES (OLD.emp_id, 'DELETED');
END;
```

Stores deleted records in an audit table.

Explanation:
This trigger ensures that salary cannot be negative before inserting a record.

Example: AFTER UPDATE Trigger

```
CREATE TRIGGER after_update_salary
AFTER UPDATE ON Employee
FOR EACH ROW
BEGIN
  INSERT INTO Employee_Log(emp_id, old_salary, new_salary)
  VALUES (OLD.emp_id, OLD.salary, NEW.salary);
END;
```

Explanation

- When a record in Employee table is updated:
  - OLD.salary → value before update
  - NEW.salary → value after update
- The trigger stores both values in a log table for tracking changes

Stored Procedures
A stored procedure is a precompiled collection of SQL statements stored in the database and executed explicitly by the user.

Difference Between Triggers and Stored Procedures

Feature	Trigger	Stored Procedure
Execution	Automatically executed	Manually called
Event	Fired by INSERT/UPDATE/DELETE	Called using CALL/EXEC
Parameters	No parameters	Can accept parameters
Usage	Data integrity, auditing	Business logic, operations
Invocation	Implicit	Explicit