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

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

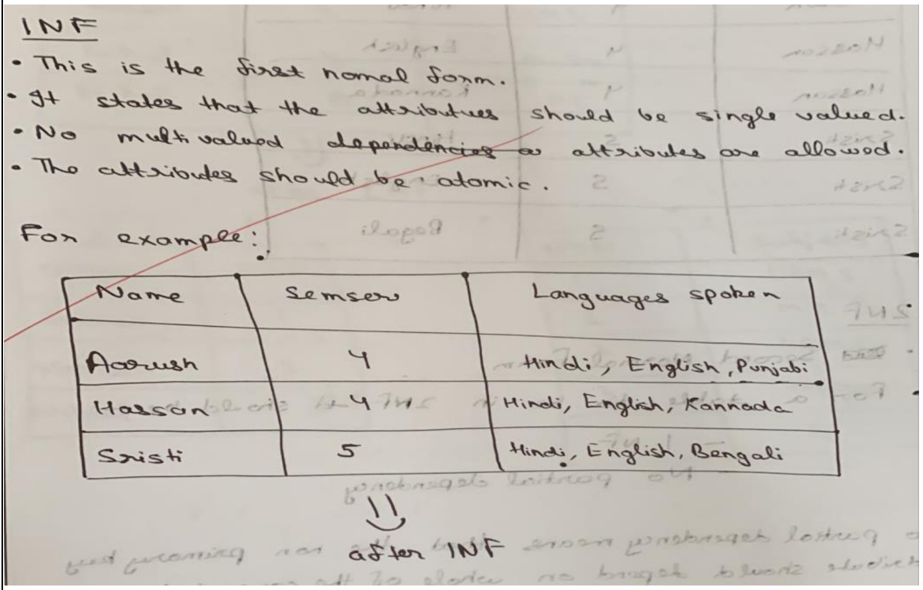
Internal Assessment Test 2 – May 2025

Sub:	Database Management System			Sub Code:	BCS 403	Branch:	AINDS / CS (DS)		
Date:	24/05/2025	Duration:	90 minutes	Max Marks:	50	Sem	IV	OBE	

Answer any FIVE Questions

			Marks	CO	RBT
1		What is the need for Normalization? Explain INF, 2NF, 3NF and BCNF with examples?	10	CO4	L2
2	a	Write an algorithm to find the closure of functional dependency 'F'.			
	b	R(ABCDEF) Check the highest Normal Form using closure Algorithm ? **FD { AB->C,C->DE,E->F,F->A}.**	10	CO4	L3
3	a	Explain the basic data types available for attributes in sql?	5	CO3	L1
	b	What is Cursor? Explain Implicit, Explicit cursor with suitable examples and syntax?	5	CO3	L1
4	a	Discuss the ACID properties of a database transaction	4	CO5	L2
	b	Why is Concurrency control needed? Explain 2 phase locking techniques with suitable example	6	CO5	L2
5	a	With a neat diagram, Explain the various states of a transaction execution.	5	CO5	L2
	b	ck whether the below schedule is conflict serializable or not {b2, r2(X), b1, r1(X), w1(X), r1(Y), w1(Y), w2(X), e1, c1, e2, c2}	5	CO6	L3
6		What is the CAP Theorem? Which of the three properties (consistency, availability, partition tolerance) are most important in NOSQL systems?	10	CO6	L3

1		**What is Normalization and explain different types** **Normalization:** Database Design Theory – Introduction to Normalization using Functional and Multivalued Dependencies: Informal design guidelines for relation schema, Functional Dependencies, Normal Forms based on Primary Keys, Second and Third Normal Forms, Boyce-Codd Normal Form, Multivalued Dependency and Fourth Normal Form, Join Dependencies and Fifth Normal Form.	10	CO 1	L1



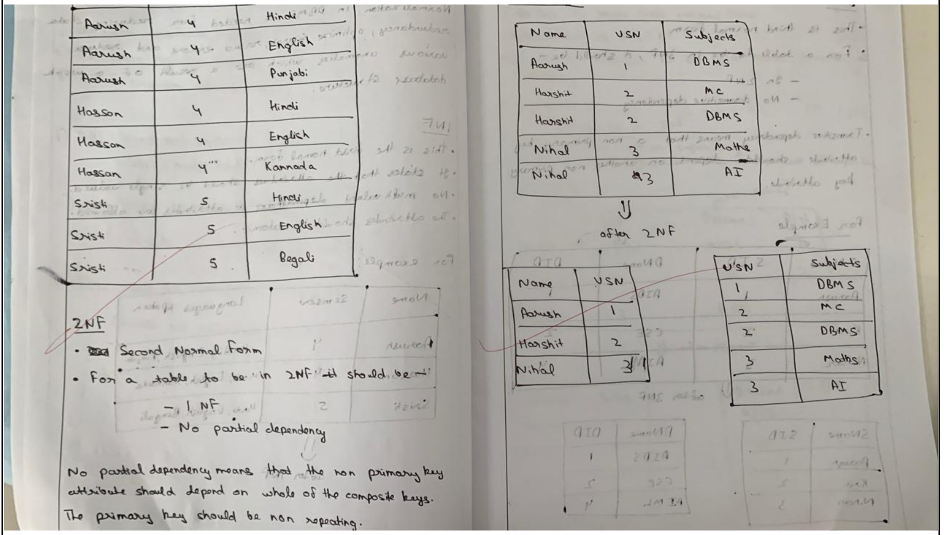
1NF

- This is the first normal form.
- It states that the attributes should be single valued.
- No multi valued ~~dependencies~~ or attributes are allowed.
- The attributes should be atomic.

For example:

Name	Semser	Languages spoken
Aarush	4	Hindi, English, Punjabi
Hasson	4	Hindi, English, Kannada
Sristi	5	Hindi, English, Bengali

⇓ after 1NF



Aarush	4	Hindi
Aarush	4	English
Aarush	4	Punjabi
Hasson	4	Hindi
Hasson	4	English
Hassan	4	Kannada
Sristi	5	Hindi
Sristi	5	English
Sristi	5	Begali

2NF

- ~~Sec~~ Second Normal Form
- For a table to be in 2NF it should be –
  - 1 NF
  - No partial dependency

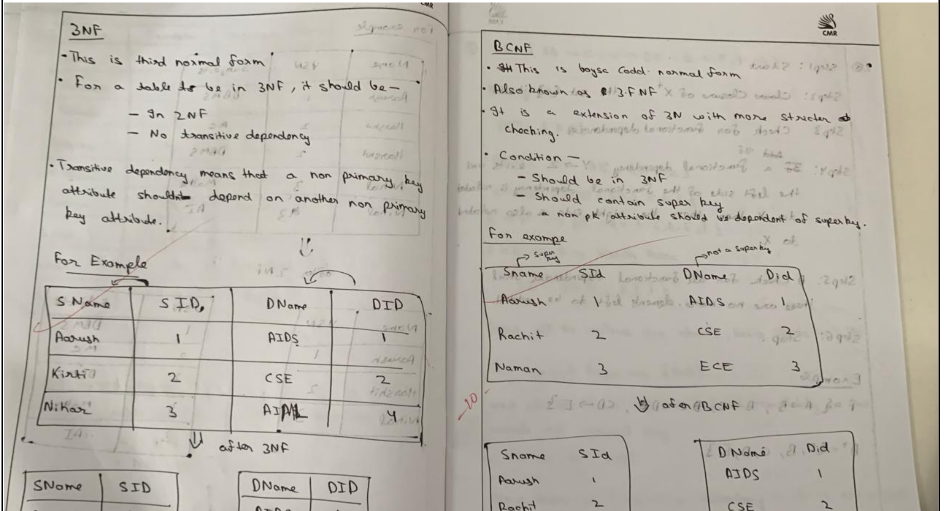
No partial dependency means that the non primary key attribute should depend on whole of the composite keys. The primary key should be non repeating.

Name	USN	Subjects
Aarush	1	DBMS
Harshit	2	MC
Harshit	2	DBMS
Nihal	3	Maths
Nihal	3	AI

⇓ after 2NF

Name	USN
Aarush	1
Harshit	2
Nihal	3

USN	Subjects
1	DBMS
2	MC
2	DBMS
3	Maths
3	AI



3NF

- This is third normal form
- For a table to be in 3NF, it should be –
  - In 2NF
  - No transitive dependency
- Transitive dependency means that a non primary key attribute ~~should~~ depend on another non primary key attribute.

For Example

S Name	S ID	DName	DID
Aarush	1	AIDS	1
Kirti	2	CSE	2
Nikar	3	AIML	4

⇓ after 3NF

SName	SID

DName	DID

BCNF

- This is boyse Codd normal form
- Also known as 3.FNF
- It is a extension of 3N with more stricter checking.
- Condition –
  - Should be in 3NF
  - Should contain super key
  - a non pk attribute should be dependent of super key.

For exampe

super key · not a super key

Sname	SId	DName	Did
Aarush	1	AIDS	1
Rachit	2	CSE	2
Naman	3	ECE	3

10

⇓ after BCNF

Sname	SId
Aarush	1
Rachit	2

D Name	Did
AIDS	1
CSE	2

2	a	Write an algorithm to find the closure of functional dependency 'F' ? To find the closure of a set of functional dependencies F, we need to determine the set of all attributes that can be functionally determined by a given set of attributes. This is typically denoted as X+ for a set of attributes X. ● Algorithm to Find the Closure of Functional Dependency F ● Input: A set of attributes X A set of functional dependencies F ● Output: The closure of X (denoted as X+) Steps: ● Initialize the closure: ○ Start with the closure X+ containing all attributes in X. ● Iterate through functional dependencies: ○ For each functional dependency A → B in F, if A is a subset of X+, add all attributes in B to X+. ● Repeat until no new attributes are added: ○ Continue the process until X+ no longer changes. ● Return the closure: ○ The final set X+ is the closure of X.	6	CO 2	L1

b

R(ABCDEF) Check the highest Normal Form using closure Algorithm ?

FD { AB->C,C->DE,E->F,F->A}.

Step 1: Candidate Key

First, we need to determine the candidate key(s) for the relation. We can do this by finding the closure of different attribute sets to see which ones determine all attributes in the relation.

Find Closure of AB:

- ○ Start with AB: {A,B}
- ○ AB→C:{A,B,C}
- ○ C→DE:{A,B,C,D,E}
- ○ E→F:{A,B,C,D,E,F}
- ○ Closure of AB is {A,B,C,D,E,F}. Hence, AB is a candidate key.

- Since AB is a candidate key and it alone determines all attributes, no other subsets are candidate keys.
- C.K□{AB

AB+=ABCDEF

FB

FB+=FBACDE

EB

EB+=EBFACD

CB}

CB+=CBDEFA

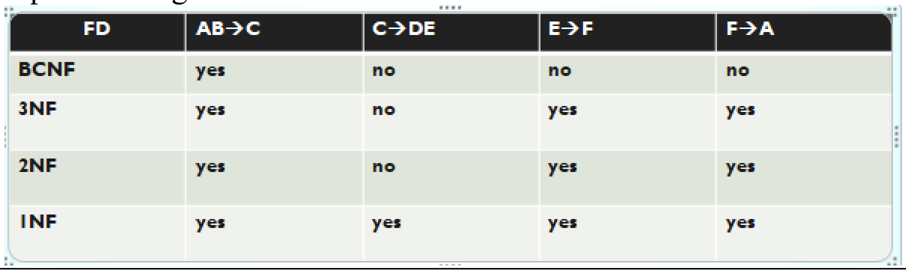
- Step 2: Write All Prime Attributes

  {A,B,C,E,F}

- Step 3: Write All Non-Prime Attributes

  { D }

Step 4:-Finding FD's



FD	AB→C	C→DE	E→F	F→A
BCNF	yes	no	no	no
3NF	yes	no	yes	yes
2NF	yes	no	yes	yes
1NF	yes	yes	yes	yes

CO 4

L3

3	a	Explain the basic data types available for attributes in SQL? . Numeric Data Types Used to store numbers (both integers and floating-point numbers). ● INT / INTEGER: Whole numbers (e.g., 1, 100, -5). ● SMALLINT: Smaller range of integers, consumes less storage. ● BIGINT: Larger range of integers, for very large numbers. ● DECIMAL(p, s) / NUMERIC(p, s): Fixed-point numbers with precision p (total digits) and scale s (digits after the decimal). ● FLOAT / REAL / DOUBLE PRECISION: Approximate, floating-point numbers. Used for scientific or imprecise calculations. --- 2. Character/String Data Types Used to store text. ● CHAR(n): Fixed-length character string. Always stores exactly n characters (padded with spaces if shorter). ● VARCHAR(n): Variable-length character string with a maximum of n characters. More flexible than CHAR. ● TEXT (or CLOB): Large blocks of text. Used when the size of the string may exceed VARCHAR lim 3. Date and Time Data Types Used to store temporal data. ● DATE: Stores date values (year, month, day). ● TIME: Stores time of day (hour, minute, second). ● TIMESTAMP: Stores both date and time. ● INTERVAL: Represents a duration (e.g., 5 days, 3 hours). 4. Boolean Data Type Used to store truth values. ● BOOLEAN: Stores TRUE, FALSE, or NULL.	5	CO 3	L

B

In SQL and PL/SQL, a cursor is a pointer that allows you to retrieve, manipulate, and traverse through rows returned by a query one at a time. Cursors are essential when you need to process individual rows of a query result, especially in procedural operations.

Types of Cursors in PL/SQL

1. Implicit Cursor

- Automatically created by Oracle/SQL engine for single SQL statements such as SELECT INTO, INSERT, UPDATE, DELETE.
- You don’t need to declare or open/close it.
- Managed internally.

□ Syntax & Example (Implicit Cursor):

```
DECLARE
   emp_name employees.first_name%TYPE;
   emp_id   employees.employee_id%TYPE := 101;
BEGIN
   SELECT first_name INTO emp_name
   FROM employees
   WHERE employee_id = emp_id;

   DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
END;
```

- This uses an implicit cursor for the SELECT INTO statement.

□ Cursor Attributes (for implicit cursors):

- %FOUND: Returns TRUE if DML affected one or more rows.
- %NOTFOUND: Returns TRUE if DML affected no rows.
- %ROWCOUNT: Returns the number of rows affected.
- %ISOPEN: Always FALSE for implicit cursors.

2. Explicit Cursor

- Declared by the programmer when a query returns multiple rows.
- Allows row-by-row processing using OPEN, FETCH, CLOSE.

CO 3

L1

4	a	Discuss the ACID properties of a database transaction? The **ACID properties** are a set of four key principles that ensure reliable processing of database transactions. The acronym **ACID** stands for: **1. Atomicity** • **Definition:** A transaction is treated as a single, indivisible unit, which either **completes in full** or **does not happen at all**. • **Example:** If you transfer money from Account A to Account B, the debit from A and credit to B must both succeed. If one fails, the entire transaction is rolled back. **2. Consistency** • **Definition:** A transaction must bring the database from one **valid state** to another, maintaining all **integrity constraints**. • **Example:** If a database enforces that account balances must never be negative, a transaction violating this rule will be rejected. --- **3. Isolation** • **Definition:** Transactions should **not interfere** with each other. Each transaction should behave as if it were the **only** one running. • **Example:** If two users update the same account balance at the same time, their transactions should not conflict or result in incorrect data. --- **4. Durability** • **Definition:** Once a transaction is **committed**, its effects are **permanent**, even in the case of a system crash. • **Example:** If a transfer is confirmed, the updated account balances must be preserved after a power failure	4	CO 1	L2

Why is Concurrency control needed? Explain 2 phase locking techniques with suitable example

Why is Concurrency Control Needed?

Concurrency control is required in database systems to ensure **correctness and consistency** when **multiple transactions** execute **simultaneously**. Without proper control, concurrent transactions can lead to problems such as:

✓Common Issues Without Concurrency Control:

1. **Lost Update** – Two transactions overwrite each other's updates.
2. **Dirty Read** – One transaction reads uncommitted changes of another.
3. **Non-repeatable Read** – A row is changed by another transaction between reads.
4. **Phantom Read** – New rows added by one transaction appear in another's result set.

Concurrency control ensures:

- Isolation (I in ACID)
- Serializability (transactions appear as if executed one after another)

□ Two-Phase Locking (2PL) Protocol

Two-Phase Locking (2PL) is a concurrency control technique that ensures **serializability** by dividing the transaction's lock operations into **two distinct phases**:

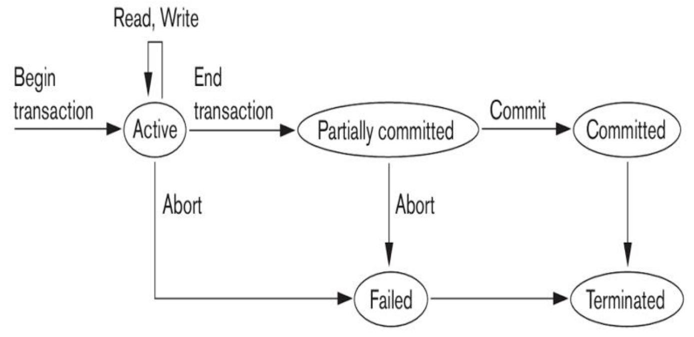
□ Phase 1: Growing Phase

- The transaction **acquires all the locks** it needs (shared for read, exclusive for write).
- It **cannot release** any lock during this phase.

□ Phase 2: Shrinking Phase

- Once the transaction **releases its first lock**, it **cannot acquire any new locks**.

Once a lock is released, no new locks can be acquired.

5		With a neat diagram, Explain the various states of a transaction execution.  Read, Write Begin transaction Active End transaction Partially committed Commit Committed Abort Abort Failed Terminated **Figure 20.4** State transition diagram illustrating the states for transaction execution. ● BEGIN_TRANSACTION:-This marks the beginning of transaction execution. ● READ or WRITE.:-These specify read or write operations on the database items that are executed as part of a transaction. ● END_TRANSACTION. This specifies that READ and WRITE transaction operations have ended and marks the end of transaction execution. ● However, at this point it may be necessary to check whether the changes introduced by the transaction can be permanently applied to the database (committed) or whether the transaction has to be aborted because it violates Serializability or for some other reason. ● COMMIT_TRANSACTION. This signals a successful end of the transaction so that any changes (updates) executed by the transaction can be safely committed to the database and will not be undone. ● ROLLBACK (or ABORT). This signals that the transaction has ended unsuccessfully, so that any changes or effects that the transaction may have applied to the database must be undone. ● Active State: ● Description: Transaction begins execution, able to perform READ and WRITE operations on the database. ● Transition: Moves to the next state after starting execution. Partially Committed State: ● Description: Transaction prepares to commit changes. Additional checks may be performed by concurrency control protocols. ● Commit Point: Ensures changes can be permanently recorded, often by logging them in the system log. ● Transition: Moves to committed state if checks are successful. Committed State: ● Description: Transaction successfully completes execution. ● Outcome: All changes made by the transaction are permanently recorded in the database, even in case of system failure. Failed State: ● Description: Transaction fails due to unsuccessful checks or			

B

eCheck whether the below schedule is conflict serializable or not
{b2, r2(X), b1, r1(X), w1(X), r1(Y), w1(Y), w2(X), e1, c1, e2, c2}
To determine whether the given schedule is conflict serializable, we need to:

1. Understand the operations and transactions involved.
2. Build a precedence graph (also known as a serializability graph).
3. Check for cycles — if the graph has no cycles, the schedule is conflict serializable.

Step 1: Parse the schedule

Schedule:
{b2, r2(X), b1, r1(X), w1(X), r1(Y), w1(Y), w2(X), e1, c1, e2, c2}

Let's break it down by transaction:

- T1: b1, r1(X), w1(X), r1(Y), w1(Y), e1, c1
- T2: b2, r2(X), w2(X), e2, c2

Step 2: Identify conflicting operations

Conflicts occur when:

1. Two operations are from different transactions.
2. They access the same data item.
3. At least one operation is a write.

We analyze all conflicting operations between T1 and T2:

- r2(X) (T2) vs w1(X) (T1): Read-Write conflict → T2 → T1
- r1(X) (T1) vs w2(X) (T2): Read-Write conflict → T1 → T2
- w1(X) (T1) vs w2(X) (T2): Write-Write conflict → T1 → T2

Step 3: Build the precedence graph

- From r2(X) → w1(X) ⇒ edge T2 → T1
- From r1(X) → w2(X) ⇒ edge T1 → T2
- From w1(X) → w2(X) ⇒ another edge T1 → T2

Now we have:

- T1 → T2

6

The CAP Theorem, proposed by Eric Brewer, states that in a distributed database system, it is impossible to simultaneously guarantee all three of the following properties:

1. Consistency (C):
   Every read receives the most recent write or an error.
   ➤*Same data across all nodes at any time.*

2. Availability (A):
   Every request (read/write) gets a response (success or failure), even if some nodes are down.
   ➤*System remains responsive.*

3. Partition Tolerance (P):
   The system continues to operate even if there is a communication breakdown between nodes in the network.
   ➤*Handles network failures gracefully.*

```
 Consistency
        /\
       /  \
      /    \
Availability ---- Partition Tolerance
```

Combinations Allowed by CAP:

Combination	Description
CP (Consistency + Partition Tolerance)	System remains consistent and tolerates network failures, but may become unavailable during partition.
AP (Availability + Partition Tolerance)	System is always available and partition-tolerant, but may not always show the most recent data (eventual consistency).
CA (Consistency + Availability)	System is consistent and available only if there is no network partition, which is unrealistic in distributed systems.

Most Important in NoSQL

10

CO 3

L3

CI	CCI	HOD