Internal Assessment Test 2 – May 2025

Sub:	Database Management System					Sub Code:	BCS403	Branch:	AIML /CSE(AIML)
Date:	26-05-25	Duration:	90 minutes	Max Marks:	50	Sem/Sec:	IV		OBE

		Questions Scheme and Solutions	MARKS	CO	RBT
1	a	Explain 1NF,2NF and 3NF.Consider the relation schema R (A, B, C, D, E, F, G, H, I, J) and the functional dependencies {AB->C, A->DE, B->F, F->GH, D->IJ}. Determine the candidate key and the highest normal form for the above relation. *1NF (First Normal Form)* A relation is in First Normal Form if: • All the values in the relation are atomic (indivisible). • Each column contains unique values. • There are no repeating groups or arrays. **Example:** If a table contains a list of phone numbers for a person in a single column, it is not in 1NF. To convert it to 1NF, each phone number should be in a separate row or a separate column. *2NF (Second Normal Form)* A relation is in Second Normal Form if: • It is in 1NF. • All non-prime attributes are fully functionally dependent on the entire primary key (no partial dependency). **Example:** Consider a table with attributes {StudentID, CourseID, StudentName}. If the primary key is {StudentID, CourseID}, and StudentName is dependent only on StudentID, it is a partial dependency and violates 2NF. To achieve 2NF, StudentName should be moved to a separate table where StudentID is the primary key. *3NF (Third Normal Form)* A relation is in Third Normal Form if: • It is in 2NF. • There are no transitive dependencies, meaning non-prime attributes are not dependent on other non-prime attributes.	10	CO3	L3

Example:

Consider a table with attributes {StudentID, CourseID, ProfessorID, ProfessorName}. If the primary key is {StudentID, CourseID}, and ProfessorName is dependent on ProfessorID (a non-prime attribute), it violates 3NF. To achieve 3NF, ProfessorName should be moved to a separate table where ProfessorID is the primary key.

2	a	Differentiate between nested query and correlated query with suitable examples.	6	CO4	L2

Nested Subqueries
A subquery is nested when you are having a subquery

in the where or having clause of another subquery.

Get the result of all the students who are enrolled in

the same course as the student with ROLLNO 12.

```
Select * from result where rollno in (select rollno  from student
                                      Where courseid =( select courseid
                                       From student where rollno=12));
```

The innermost subquery will be executed first and then based on its result the next subquery will be executed and based on that result the outer query will be executed. The levels to which you can do the nesting is implementation-dependent.

Correlated Subquery

A **Correlated Subquery** is one that is executed after the outer query is executed. So correlated subqueries take an approach opposite to that of normal subqueries. The correlated subquery execution is as follows:

-The outer query receives a row.
-For each candidate row of the outer query, the subquery (the correlated subquery) is executed once.
-The results of the correlated subquery are used to determine whether the candidate row should be part of the result set.
-The process is repeated for all rows.

Correlated Subqueries differ from the normal subqueries in that the nested SELECT statement referes back to the table in the first SELECT statement.

To find out the names of all the students who appeared in more than three papers of their opted course, the SQL will be

		Select name from student A where 3 < (select count(*) from result b where a. rollno = b.rollno);			
	b	What is a view in SQL? How to create a view in SQL? In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL statements and functions to a view and present the data as if the data were coming from one single table. A view is created with the CREATE VIEW statement. CREATE VIEW Syntax CREATE VIEW *view_name* AS SELECT *column1*, *column2*, ...FROM *table_name* WHERE *condition*; SQL Updating a View A view can be updated with the CREATE OR REPLACE VIEW statement. SQL CREATE OR REPLACE VIEW Syntax CREATE OR REPLACE VIEW *view_name* AS SELECT *column1*, *column2*, ... FROM *table_name* WHERE *condition*; SQL dropping VIEW VIEW is deleted with DROP VIEW statement DROP VIEW syntax DROP VIEW view_name	4	CO4	L1
3	a	Discuss the ACID properties of database transaction. **Answer:** There are 4 properties (1 mark) 1. Atomicity, 2. Consistency, 3. Isolation and 4. Durability	5	CO4	L2

		To ensure consistency, completeness of the database in scenario of concurrent access, system failure , the following ACID properties can be enforced on to database.
		(4x1=4)
		Atomicity:
		  This property states that all of the instructions within a transaction must be executed or none of them should be executed.
		  This property states that all transactions execution must be atomic i.e. all actions should be carried out or none of the actions should be executed.
		Consistency:
		  The database must remain in consistence state even after performing any kind of
		transaction ensuring correctness of the database.
		  If we execute a particular transaction in isolation (or) together with other transaction in
		multiprogramming environment , the transaction should give same result in any case.
		Isolation:
		  When executing multiple transactions concurrently & trying to access shared resources the system should create an order such that the only one transaction can access the shared
		resource at the same time & release it after completion of it's execution for other transaction.
		  This property ensures that multiple transactions can occur concurrently without leading to inconsistency of database state. Transactions occur independently without interference.
		Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed.
		Durability:
		  This property states that once after the transaction is completed the changes that made should
		be permanent & should be recoverable even after system crash/power failure.
		  This property ensures that once the transaction has completed execution, the updates and
		modifications to the database are stored in and written to disk and they persist even is system
		failure occurs. These updates now become permanent and are stored in a non-volatile
		memory.			

	b	Define Schedule? Illustrate with an example.
		Schedule (3Marks)
		  It refers to the list of actions to be executed by transaction.
		  A schedule is a process of grouping the transactions into one and executing them in a
		predefined order.
		Example (2 Marks)
			5	CO4	L2

4	a	Demonstrate the Two-phase locking protocol used for concurrency control and how it can lead to deadlocks. Two Phases-(7 Marks+ Example) (a) Locking (Growing) (b) Unlocking (Shrinking). Locking (Growing) Phase:	10	CO5	L2

		□ A transaction applies locks (read or write) on desired data items one at a time. Unlocking (Shrinking) Phase: □ A transaction unlocks its locked data items one at a time. Requirement: □ For a transaction these two phases must be mutually exclusively, that is, during locking phase unlocking phase must not start and during unlocking phase locking phase must not begin Two phase locking could lead to deadlock+ Example (3 Marks)			
5	a	Explain CAP theorem. Two Phases-(7 Marks+ Example) (a) Locking (Growing) (b) Unlocking (Shrinking). Locking (Growing) Phase: □ A transaction applies locks (read or write) on desired data items one at a time. Unlocking (Shrinking) Phase: □ A transaction unlocks its locked data items one at a time. Requirement: □ For a transaction these two phases must be mutually exclusively, that is, during locking phase unlocking phase must not start and during unlocking phase locking phase must not begin Two phase locking could lead to deadlock+ Example (3 Marks)	5	CO5	L2
	b	What is NOSQL Graph database? Explain Neo4j. Answer: • Graph databases • Data represented as a graph • Collection of vertices (nodes) and edges • Possible to store data associated with both individual nodes and individual edges Neo4j • Open source system • Uses concepts of nodes and relationships Nodes can have labels • Zero, one, or several • Both nodes and relationships can have properties • Each relationship has a start node, end node, and a relationship type • Properties specified using a map pattern • Somewhat similar to ER/EER concepts	5	CO5	L2

6	a	Consider the following relation schema and write SQL queries, employee (person-name, street, city) works (person-name, company-name, salary) company (company-name, city) i. Find the names, street address, and cities of residence for all employees who work for 'First Bank Corporation' and earn more than Rs.10,000. ii. Find the names of all employees in the database who do not work for 'First Bank Corporation'. iii. Find the names of all employees in the database who earn more than every employee of 'Small Bank Corporation'. iv. Update the salary of people working in 'Infosys' by 15%. v. Display company wise, average salary and maximum salary paid to the employees. Ans. 1. SELECT e.person_name, e.street, e.city FROM employee e JOIN works w ON e.person_name = w.person_name WHERE w.company_name = 'First Bank Corporation' AND w.salary > 10000; 2. SELECT DISTINCT e.person_name FROM employee e WHERE e.person_name NOT IN (SELECT w.person_name FROM works w WHERE w.company_name = 'First Bank Corporation'); 3. SELECT e.person_name FROM works e WHERE e.salary > (SELECT MAX(w.salary)FROM works w WHERE w.company_name = 'Small Bank Corporation'); 4. UPDATE works SET salary = salary * 1.15 WHERE company_name = 'Infosys'; 5. SELECT company_name, AVG(salary) AS average_salary, MAX(salary) AS max_salary FROM works GROUP BY company_name;	10	4	L3