Reg. No.														

SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
Academic Year: 2022-23 (EVEN)

Batch 2

Test: CT-3 — **Date: 02-05-2023**
Course Code &Title: 18CSC303J Database Management Systems — **Duration:10.30 am to 12.10 pm**
Year & Sem: III Year / VISem — **Max. Marks:50**
Instruction: MCQs to be collected within first 15 minutes

Course Articulation Matrix:

S.No.	Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
1	CO5	H	H	L	M	L	-	-	-	M	M	M	L
2	CO6	H	L	L	L	L	-	-	-	H	L	L	L

Part – A MCQ(10x 1 =10 Marks)Instructions: Answer all

Q. No	Question	Marks	BL	CO	PO	PI Code
1	Which of the following is a binary operation in relational algebra? a) Select b) Project **c) Join** d) Divide	1	1	5	1	1.6.1
2	Which of the following statements is true about Tuple Relational Calculus? a) It uses a procedural approach to specify queries. b) **It is a declarative language that specifies what data to retrieve without specifying how to retrieve it.** c) It uses operators such as select, project, and join. d) It is used to specify constraints on the data stored in a database.	1	2	5	1	1.6.1
3	Which of the following is a true statement about cursors in SQL? a) Cursors are used to execute a single SQL statement at a time. b) **Cursors are used to retrieve a subset of rows from a table based on a specific condition.** c) Cursors are used to update multiple rows in a table at once. d) Cursors are used to perform calculations on the data stored in a table.	1	1	5	1	1.6.1

4	Which of the following set of dependencies is suitable for making a relation R(ABCD) to be in 3 NF but not in BCNF.
	a) **{AB->CD,A->C,D->B}** b) {AB->CD, C->DA}
	c {A->BCD, C->A, D->B} d) {AB->CD, C->A, D>}
		1	2	5	1	1.6.1

5	Choose the incorrect statement: a) A relation in 2NF must also be in 1NF b) A relation in 3NF must also be in 2NF **c) A relation in 3NF must also be in BCNF** d) A relation in BCNF must also be in 3NF	1	2	5	1	1.6.1

6	Which of the following is a true statement about serializability in database systems? a) Serializability is a property that ensures the correctness of concurrent transactions. b) Serializable schedules always result in a deadlock situation. c) Conflict serializability ensures that transactions can be executed in parallel without any conflict. d) **A schedule is serializable if and only if it is a conflict serializable schedule.**	1	2	6	2	2.7.2
7	This occurs when more than one database transaction attempts to read or write the same database item simultaneously (i.e., concurrent execution), causing the values of the item to become incorrect, resulting in a _____database. A. Consistent **B. Inconsistent** C. Concurrent D. Not-concurrent	1	1	6	1	1.6.1
8	When transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has a timestamp smaller than that of Tj (that is, Ti is older than Tj). Otherwise, Ti is rolled back (dies). This is a) **Wait-die** b) Wait-wound c) Wound-wait d) Wait	1	1	6	2	2.6.1
9	It is possible to detect deadlock situations using methods such as "___", but this method is suitable for smaller databases only. A. Build for graph B. Design for graph C. Take for graph **D. Wait for graph**	1	2	6	2	2.6.1
10	All logs are created and stored in ___ storage when the deferred database method is used, and data is synced to the database when a transaction commits. **A. Stable** B. Unstable C. System D. Recovery	1	1	6	2	2.6.1

Reg. No.															

of Science and

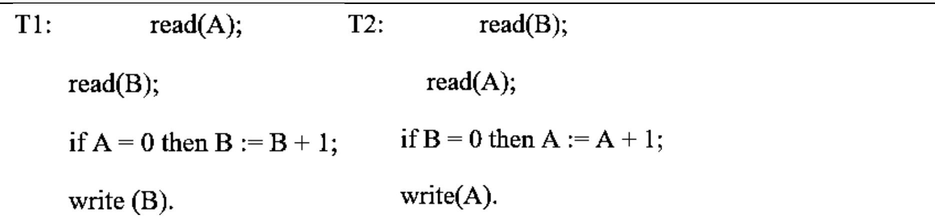
Technology
College of Engineering and Technology
School of Computing
Academic Year: 2022-23 (EVEN)

Batch - 2

Test: CT-3 **Date: 02-05-2023**
Course Code &Title: 18CSC303J Database Management Systems **Duration:10.30 am to 12.10 pm**
Year & Sem: III Year / VISem **Max. Marks:50**
Instruction: MCQs to be collected within first 15 minutes

Part – B(4 x4= 16 Marks)Instructions: Answer any 4

		Marks	BL	CO	PO	PI Code
11.	Consider the following relation. R (A, B, C, D). Assume the set of functional dependency F= {B ->C, D ->A}. R is decomposed in to R1(B, C) and R2(A, D) a. List down the Armstrong's axioms. (2 Marks) b. Find the candidate key(s) for R. (2 marks) Axiom of reflexivity, Axiom of augmentation, Axiom of transitivity Additional axioms. Union, composition, decomposition, pseudo transitivity. (BD)+ = BCDA. It includes all attributes of R, hence BD is the candidate key.	4	4	5	2	2.7.2
12.	Suppose you have a relation R representing a database of employees, with attributes Employee_ID, Name, Department, and Salary. Write the syntax of tuple relational calculus expression and a query to find all employees who work in the "Marketing" department and have a salary greater than $50,000. In tuple relational calculus, expressions are written in the form , {t | P(t)}, where t is a tuple variable and P(t) is a predicate that defines a condition that the tuples in the resulting relation must satisfy. We can write the tuple relational calculus expression for the given scenario as: {t | t ∈ R ∧ t.Department = "Marketing" ∧ t.Salary > 50000}	4	3	5	1	1.6.1
13.	Provide an example and brief about join dependency? A relation is said to have join dependency **if it can be recreated by joining multiple sub relations** and each of these sub-relations has a subset of the attributes of the original relation. **Condition for join dependency** Consider a relation R(P,Q,S) . where R1 and R2 are the decompositions such that R1 (P, Q) and R2 (Q, S) Then R1 and R2 are a lossless decomposition of R if R can be obtained by joining R1 and R2. $\begin{tikzcd} & R \arrow[dl] \arrow[dr] & \\ R1 \arrow[dr] & & R2\text{ (Decompose)} \arrow[dl] \\ & R & \end{tikzcd}$	4	3	5	2	2.6.1
14.	Consider the following two transactions:	4	4	6	2	2.6.1

	
	T1: read(A); T2: read(B);
	read(B); read(A);
	if A = 0 then B := B + 1; if B = 0 then A := A + 1;
	write (B). write(A).
	Add lock and unlock instructions to transactions T1 and T2, so that they observe the two-phase locking protocol.
	Ans:
	Lock and unlock instructions:
	T1: lock-S(A)
	read(A)
	lock-X(B)
	read(B)
	if A = 0 then B := B + 1
	write(B)
	unlock(A)
	unlock(B)
	
	T2:
	
	lock-S(B)
	read(B)
	lock-X(A)
	read(A)
	if B = 0 then A := A + 1
	write(A)
	unlock(B)
	unlock(A)					

15.	Describe Intent Locking mechanism with example? Intention locks allow a higher level node to be locked in S or X mode without having to check all descendent nodes. ***intention-shared*** (IS): indicates explicit locking at a lower level of the tree but only with shared locks. ***intention-exclusive*** (IX): indicates explicit locking at a lower level with exclusive or shared locks ***shared and intention-exclusive*** (SIX): the sub-tree rooted by that node is locked explicitly in shared mode and explicit locking is being done at a lower level with exclusive-mode locks.	4	3	6	2	2.6.1
Part – C (2 x 12 = 24 Marks)Answer All						
16.	a. Consider a small library system that tracks information about books, authors, and the borrowers who check out the books. Each book has a unique ISBN number, a title, a publication year, and a publisher. Each author has a unique author ID, a name, and a birth year. Each borrower has a unique borrower ID, a name, an address, and a phone number. Each book can be checked out by one or more borrowers, and each borrower can check out one or more books. For each instance of a book being checked out by a borrower, the system records the date the book was checked out and the date it was returned. Based on this scenario, can	12	4	5	3	3.6.2

you provide a table design that represents the data in 3NF? Discuss the concept of Functional dependency involved in 3NF?

A table design that represents the data in 3NF:
Table 1: Books
ISBN (primary key)
Title
Publication year
Publisher
Table 2: Authors
Author ID (primary key)
Name
Birth year
Table 3: Borrowers
Borrower ID (primary key)
Name
Address
Phone number
Table 4: Book Borrowings
Borrowing ID (primary key)
ISBN (foreign key referencing Books table)
Borrower ID (foreign key referencing Borrowers table)
Check out date
Return date

In this design, we have split the data into separate tables based on their unique attributes and relationships. This table structure eliminates redundancy and ensures data integrity by avoiding data duplication and maintaining relationships between entities.

Transitive dependency : (Explanation)
A->B (B depends on A)
B->C
A->C

The derived dependency is called transitive dependency when such dependency becomes improbable (i.e.) non-prime attribute depends on another non-prime attribute.

Eg:
Rollno->marks
Marks->grade
Rollno->grade

(or)

b. Consider the relational database given below where primary keys are underlined. Give an expression in Relational Algebra to express the following Queries:

employee (person_name, street, city)
works (person_name, company_name, salary)
company (company_name, city)
manages (person_name, manager_name)

i.)Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000 per annum.

select * from employee
where person_name in
(select person_name from Works
where company_name=’First Bank Corporation’ and salary>10000

	select E.person_name, street, city **from Employee as E, Works as W** **where E.person_name=W.person_name and W.company_name='First Bank Corporation'** **and W.salary>10000** ii.)Find the names of all employees in this database who live in the same city as the company for which they work. **select E.person_name** **from Employee as E, Works as W, Company as C** **where E.person_name=W.person_name and E.city=C.city** **and W.company_name=C.company_name** iii.) Find names of all employees who earn more than every employee of Small Bank Corporation **select person_name from Works** **where salary > all** **(select salary from Works** **where company_name='Small Bank Corporation')** **select person_name from Works** **where salary>(select max(salary) from Works** **where company_name='Small Bank Corporation')** iv.)Find the employees of small Bank Corporation who earn maximum salary **select company_name** **from Works** **group by company_name** **having avg(salary)>(select avg(salary)** **from Works** **where company_name='First Bank Corporation')**					
17.	a. Deadlock is said to be one of the most feared complications in DBMS as no task ever gets finished and is in waiting state forever. Illustrate the situation with an example. Discuss the methods used to prevent and detect the deadlock. System is deadlocked if there is a set of transactions such that every transaction in the set is waiting for another transaction in the set. **Deadlock prevention** protocols ensure that the system will *never* enter into a deadlock state. Some prevention strategies : Require that each transaction locks all its data items before it begins execution (predeclaration). Impose partial ordering of all data items and require that a transaction can lock data items only in the order specified by the partial order. Following schemes use transaction timestamps for the sake of deadlock prevention alone. **wait-die** scheme — non-preemptive older transaction may wait for younger one to release data item (older means smaller timestamp) . Younger transactions never wait for older ones; they are rolled back instead. a transaction may die several times before acquiring needed data item **wound-wait** scheme — preemptive older transaction *wounds* (forces rollback) of younger transaction instead of waiting for it. Younger transactions may wait for older ones. may be fewer rollbacks than *wait-die* scheme. **Deadlock Detection:**	12	4	6	3	3.6.2

Deadlocks can be described as a *wait-for graph*, which consists of a pair $G = (V,E)$,
V is a set of vertices (all the transactions in the system)
E is a set of edges; each element is an ordered pair $T_i \rightarrow T_j$.
If $T_i \rightarrow T_j$ is in E, then there is a directed edge from T_i to T_j, implying that T_i is waiting for T_j to release a data item.
When T_i requests a data item currently being held by T_j, then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph. This edge is removed only when T_j is no longer holding a data item needed by T_i.
The system is in a deadlock state if and only if the wait-for graph has a cycle. Must invoke a deadlock-detection algorithm periodically to look for cycles.

(or)

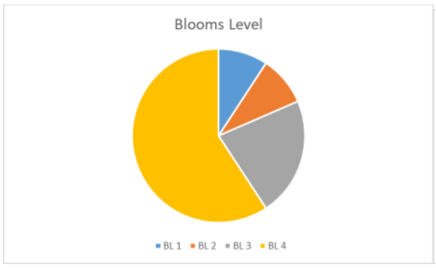
b.i.) Illustrate the methods used for log based recovery with examples?

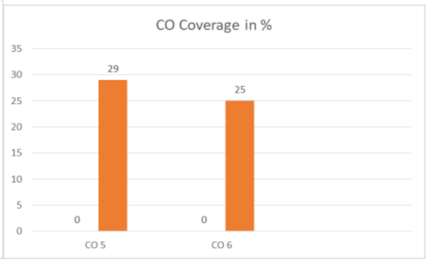
A **log** is kept on stable storage.The log is a sequence of **log records**, and maintains a record of update activities on the database.
When transaction T_i starts, it registers itself by writing a
<T_i **start**>log record
Before T_i executes **write**(X), a log record
<T_i,X,V_1,V_2>
is written, where V_1 is the value of X before the write (the **old value),** and V_2 is the value to be written to X (the **new value)**.
When T_i finishes it last statement, the log record <T_i **commi**t> is written.
Two approaches using logs
Deferred database modification
Immediate database modification
The **immediate-modification** scheme allows updates of an uncommitted transaction to be made to the buffer, or the disk itself, before the transaction commits
Update log record must be written *before* database item is written
We assume that the log record is output directly to stable storage
Output of updated blocks to stable storage can take place at any time before or after transaction commit
Order in which blocks are output can be different from the order in which they are written.
The **deferred-modification** scheme performs updates to buffer/disk only at the time of transaction commit
Simplifies some aspects of recovery
But has overhead of storing local copy

ii.) Suppose you have two transactions T1 and T2, which operate on a bank database containing two accounts: Account A and Account B. Transaction T1 transfers $100 from Account A to Account B, while transaction T2 transfers $50 from Account B to Account A. Can these transactions be executed concurrently while maintaining serializability? If not, why?
No, these transactions cannot be executed concurrently while maintaining serializability because they create a cyclic dependency or "cycle" in the transaction dependency graph.
When T1 transfers $100 from Account A to Account B and T2 transfers $50 from Account B to Account A, they create a cycle in the dependency graph, as follows:

	T1 reads from Account A T1 writes to Account B T2 reads from Account B T2 writes to Account A This cycle means that the transactions are not serializable, because there is no way to order the transactions without violating their dependencies. In other words, if we execute T1 before T2, we will get one set of values for the accounts, and if we execute T2 before T1, we will get a different set of values. Therefore, we need to prevent this cycle from occurring to ensure serializability. One way to avoid the cycle is to use a locking mechanism to ensure that only one transaction can access an account at a time.					

Course Outcome (CO) and Bloom’s level (BL) Coverage in Questions


Blooms Level
BL 1
BL 2
BL 3
BL 4


CO Coverage in %
CO 5
CO 6