Internal Assessment Test 2 – May 2025

<u>Question Paper</u>

Sub:	**Database Management System**	**Sub Code:**	**BCS403**	**Branch**	**CSE**

	<u>Answer any FIVE FULL Questions</u>	MARKS	CO	RBT
1	Why concurrency control and recovery are needed in DBMS? Explain the different problems occur in concurrent transactions?	[10]	CO4	L2
2	Briefly explain 2PL (2 phase locking) protocol for concurrency control. How does it guarantee serializability? Explain with example.	[10]	CO5	L2

3	Consider the three transactions T1, T2,T3 and schedules S1 and S2 given below. Draw the serializability (precedence) graphs for S1 and S2 and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).
	T1: r1(X);r1(Z);w1(X);
	T2: r2(Z);r2(Y);w2(Z);w2(Y);
	T3: r3(X);r3(Y);w3(Y);
	S1: r1(X),r2(Z);r1(Z);r3(X);r3(Y);w1(X);w3(Y);r2(Y);w2(Z);w2(Y);
	S2: r1(X);r2(Z);r3(X);r1(Z);r2(Y);r3(Y);w1(X);w2(Z);w3(Y);w2(Y);
		[10]	CO4	L3

4	Explain 3NF, BCNF, 4NF & 5NF with suitable examples.	[10]	CO4	L2
5	• The **orchestras table** stores all orchestras. The columns are→ id, name, rating, city_origin, country_origin, and year in which the orchestra was founded. • The **concerts table** contains all concerts played by the orchestras. The columns are→ id, city, country, year, rating, and orchestra_id (references the orchestras table). • The **members table** stores the members of (i.e. musicians playing in) each orchestra. The columns are→ id, name, position (i.e. the instrument played), wage, experience, and orchestra_id (references the orchestras table). Write SQL Statements for the following questions: i) Show the name and position of orchestra members who earn more than the average wage of all violinists. ii) Show the names of orchestras that were created after the 'Chamber Orchestra' and have a rating greater than 7.5. iii) Create a view of all orchestras which are originated in India. iv) Update experience of all the members to 10 whose position is piano player	[10]	CO3	L3
6	Write short note on : i) The CAP Theorem. ii) Document-Based NOSQL Systems	[5+5]	CO6	L2

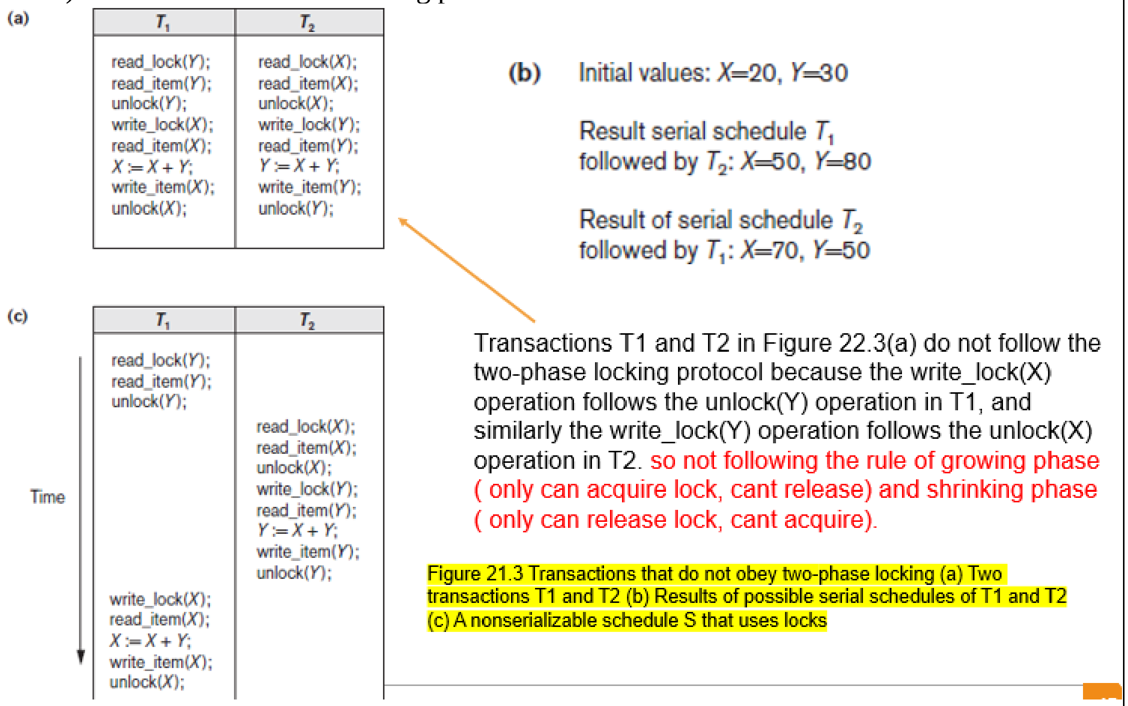
Answer Scheme and Solution

1	**Why concurrency control and recovery are needed in DBMS? Explain the different problems occur in concurrent transactions?**	[10]	CO4	L2
	Basically, whenever a transaction is submitted to a DBMS for execution, the operating system is responsible for making sure or to be confirmed that all the operations which need to be performed in the transaction have been completed successfully and their effect is either recorded in the database or the transaction doesn't affect the database or any other transactions. The DBMS must not permit some operation of the transaction T to be applied to the database while other operations of T are not. This basically may happen if a transaction fails after executing some of its operations but before executing all of them. **Here are some of the reasons why recovery is needed in DBMS.** • System failures: The DBMS can experience various types of failures, such as hardware failures, software bugs, or power outages, which can lead to data corruption or loss. Recovery mechanisms can help restore the database to a consistent state after such failures. • Transaction failures: Transactions can fail due to various reasons, such as network failures, deadlock, or errors in application logic. Recovery mechanisms can help roll back or undo the effects of such failed transactions to ensure data consistency. • Human errors: Human errors such as accidental deletion, updating or overwriting data, or incorrect data entry can cause data inconsistencies. Recovery mechanisms can help recover the lost or corrupted data and restore it to the correct state. • Security breaches: Security breaches such as hacking or unauthorized access can compromise the integrity of data. Recovery mechanisms can help restore the database to a consistent state and prevent further data breaches. • Hardware upgrades: When a DBMS is upgraded to a new hardware system, the migration process can potentially lead to data loss or corruption. Recovery mechanisms can help ensure that the data is successfully migrated and the integrity of the database is maintained. • Natural disasters: Natural disasters such as earthquakes, floods, or fires can damage the hardware on which the database is stored, leading to data loss. Recovery mechanisms can help restore the data from backups and minimize the impact of the disaster. • Compliance regulations: Many industries have regulations that require businesses to retain data for a certain period of time. Recovery mechanisms can help ensure that the data is available for compliance purposes even if it was deleted or lost accidentally. • Data corruption: Data corruption can occur due to various reasons such as hardware failure, software bugs, or viruses. Recovery mechanisms can help restore the database to a consistent state and recover any lost or corrupted data. ***Several problems can occur when concurrent transactions execute in an uncontrolled manner*** Types of problems we may encounter: 1. The Lost Update Problem[WW conflict] 2. The Temporary Update/ Dirty Read Problem[WR conflict] 3. The Incorrect Summary Problem 4. The Unrepeatable Read Problem [RW conflict]			
2	**Briefly explain 2PL (2 phase locking) protocol for concurrency control. How does it guarantee serializability? Explain with example.**	[10]	CO5	L2
	Two-phase locking (2PL) protocol divides the execution phase of a transaction into three parts. In the first part, when the transaction starts executing, it seeks permission for the locks it requires.			

It is a concurrency control method that guarantees serializability
The second part is where the transaction acquires all the locks.
As soon as the transaction releases its first lock, the third phase starts.

- In this phase, the transaction cannot demand any new locks.
- It only releases the acquired locks.
- The protocol utilizes locks, applied by a transaction to data, which may block other transactions from accessing the same data during the transaction's life.

A transaction is said to follow the two-phase locking protocol if all locking operations (read_lock, write_lock) precede the first unlock operation in the transaction
□ Such a transaction can be divided into two phases:
□ **Expanding or growing (first) phase**, during which new locks on items can be acquired but none can be released
□ **Shrinking (second) phase**, during which existing locks can be released but no new locks can be acquired
□ If lock conversion is allowed, then upgrading of locks (from read-locked to write-locked) must be done during the expanding phase, and downgrading of locks (from write-locked to read-locked) must be done in the shrinking phase.


(a)

T_1	T_2

read_lock(Y);
read_item(Y);
unlock(Y);
write_lock(X);
read_item(X);
$X := X + Y$;
write_item(X);
unlock(X);
	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	$Y := X + Y$;
	write_item(Y);
	unlock(Y);

(b) Initial values: X=20, Y=30

Result serial schedule T_1 followed by T_2: X=50, Y=80

Result of serial schedule T_2 followed by T_1: X=70, Y=50

(c)

T_1	T_2

read_lock(Y);
read_item(Y);
unlock(Y);	

	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	$Y := X + Y$;
	write_item(Y);
	unlock(Y);

write_lock(X);
read_item(X);
$X := X + Y$;
write_item(X);
unlock(X);	

Transactions T1 and T2 in Figure 22.3(a) do not follow the two-phase locking protocol because the write_lock(X) operation follows the unlock(Y) operation in T1, and similarly the write_lock(Y) operation follows the unlock(X) operation in T2. so not following the rule of growing phase (only can acquire lock, cant release) and shrinking phase (only can release lock, cant acquire).

Figure 21.3 Transactions that do not obey two-phase locking (a) Two transactions T1 and T2 (b) Results of possible serial schedules of T1 and T2 (c) A nonserializable schedule S that uses locks

- If we enforce two-phase locking, the transactions can be rewritten as T1’ and T2’ as shown in Figure 22.4.
- Now, the schedule shown in Figure 22.3(c) is not permitted for T1_ and T2_ (with their modified order of locking and unlocking operations) under the rules of locking because T1_ will issue its write_lock(X) before it unlocks item Y; consequently, when T2_ issues its read_lock(X), it is forced to wait until T1_ releases the lock by issuing an unlock (X) in the schedule.

Here during growing phase only acquiring locks, not releasing any and during shrinking phase only releasing locks, not acquiring any lock, so satisfies 2PL property.

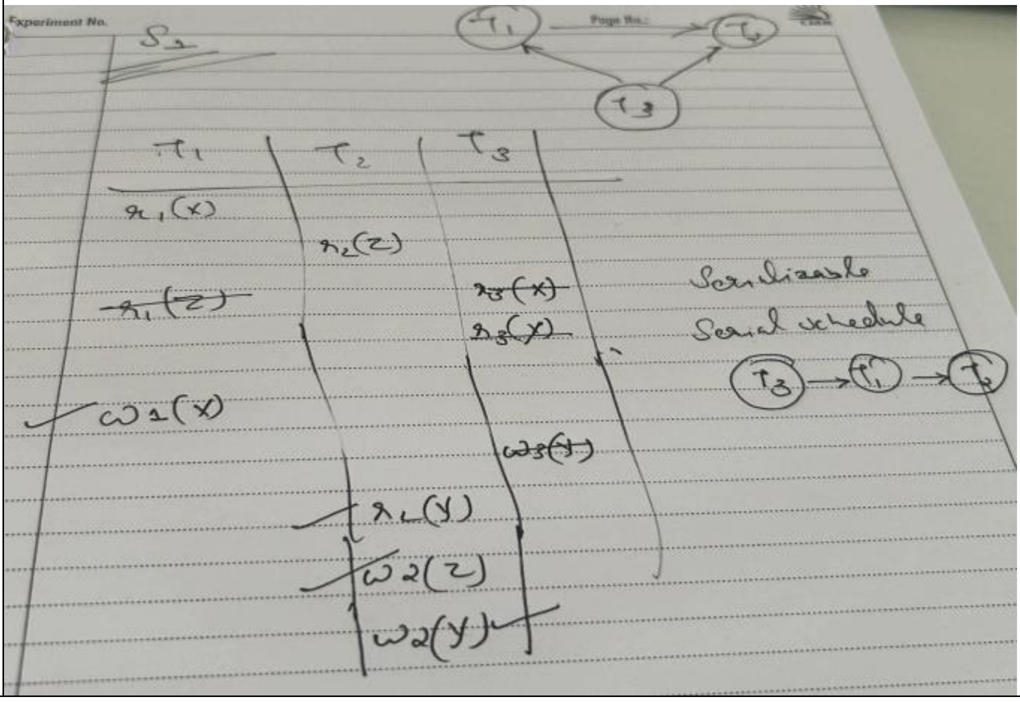
T_1'	T_2'

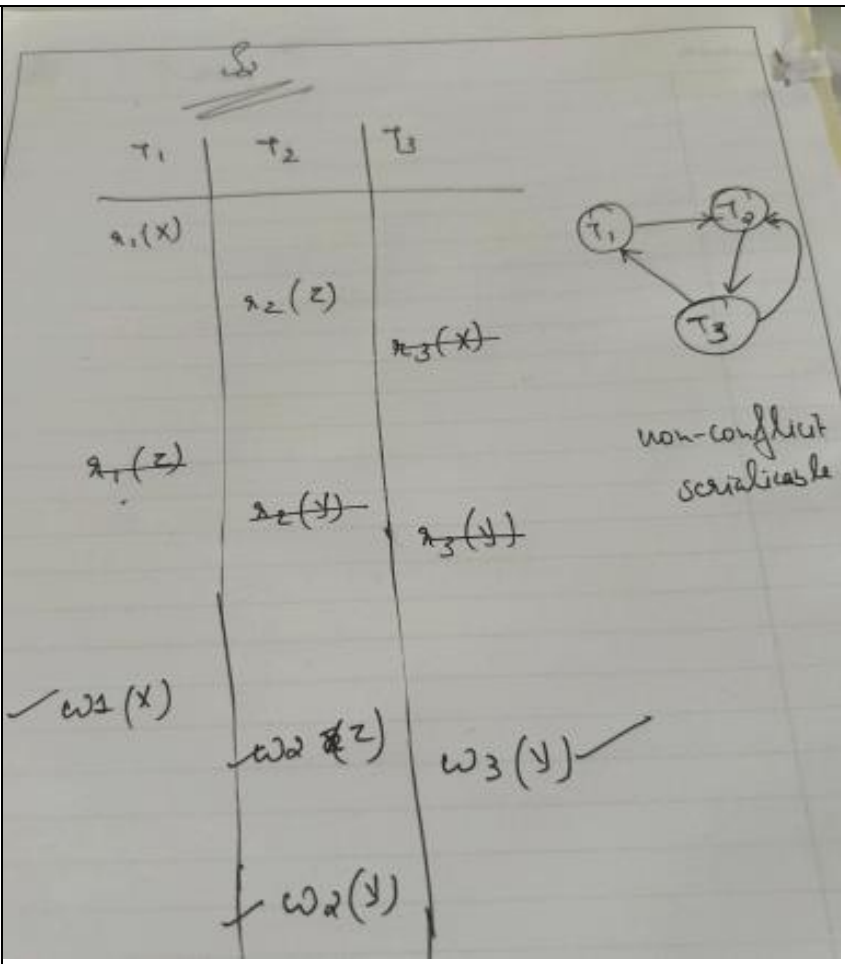
read_lock(Y);
read_item(Y);
write_lock(X);
unlock(Y)
read_item(X);
$X := X + Y$;
write_item(X);
unlock(X);
	read_lock(X);
	read_item(X);
	write_lock(Y);
	unlock(X)
	read_item(Y);
	$Y := X + Y$;
	write_item(Y);
	unlock(Y);

Figure 22.4
Transactions T_1' and T_2', which are the same as T_1 and T_2 in Figure 22.3, but follow the two-phase locking protocol. Note that they can produce a deadlock.

- **If every transaction in a schedule follows the two-phase locking protocol, schedule guaranteed to be serializable**

3	**Consider the three transactions T1,T2,T3 and schedules S1 and S2 given below. Draw the serializability (precedence) graphs for S1 and S2 and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).**
	T1: r1(X);r1(Z);w1(X);
	T2: r2(Z);r2(Y);w2(Z);w2(Y);
	T3: r3(X);r3(Y);w3(Y);
	S1: r1(X), r2(Z);r1(Z);r3(X);r3(Y);w1(X);w3(Y);r2(Y);w2(Z);w2(Y);
	S2: r1(X);r2(Z);r3(X);r1(Z);r2(Y);r3(Y);w1(X);w2(Z);w3(Y);w2(Y);
		[10]	CO4	L3



T1	T2	T3
r1(X)		
	r2(Z)	
		~~r3(X)~~
~~r1(Z)~~		
	~~r2(Y)~~	
		~~r3(Y)~~
w1(X)		
	w2(Z)	
		w3(Y)
	w2(Y)	

$$\begin{tikzcd} T_1 \arrow[r] & T_2 \arrow[d] \\ & T_3 \arrow[ul] \arrow[u, bend right] \end{tikzcd}$$

non-conflict serializable

4	**Explain 3NF, BCNF, 4NF & 5NF with suitable examples.**	[10]	CO4	L2

<u>Third Normal Form (3NF)</u>

A table is said to be in the Third Normal Form when,

1. It satisfies the First Normal Form and the Second Normal form.
2. And, it doesn't have Transitive Dependency.

<u>Boyce-Codd Normal Form (BCNF)</u>

- **Boyce and Codd Normal Form** is a higher version of the Third Normal Form.
- This form deals with a certain type of anomaly that is not handled by 3NF.
- A 3NF table that does not have **multiple overlapping candidate keys** is said to be in BCNF.
- For a table to be in BCNF, the following conditions must be satisfied:
  - R must be in the 3rd Normal Form
  - and, for each functional dependency (X → Y), X should be a Super Key.

Fourth Normal Form (4NF)

A table is said to be in the Fourth Normal Form when,

1. It is in the Boyce-Codd Normal Form.
2. And, it doesn't have Multi-Valued Dependency.

Fifth Normal Form (5NF)

- The fifth normal form is also called the **PJNF - Project-Join Normal Form**
- It is the most advanced level of Database Normalization.
- Using Fifth Normal Form you can fix **Join dependency** and reduce data redundancy.
- It also helps in fixing **Update anomalies** in DBMS design.

5 [10] CO3 L3

- **The orchestras table stores all orchestras. The columns are→ id, name, rating, city_origin, country_origin, and year in which the orchestra was founded.**
- **The concerts table contains all concerts played by the orchestras. The columns are→ id, city, country, year, rating, and orchestra_id (references the orchestras table).**
- **The members table stores the members of (i.e. musicians playing in) each orchestra. The columns are→ id, name, position (i.e. the instrument played), wage, experience, and orchestra_id (references the orchestras table).**

Write SQL Statements for the following questions:

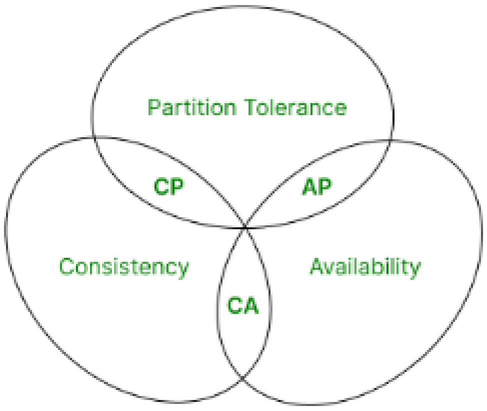
i. Show the name and position of orchestra members who earn more than the average wage of all violinists.
ii. Show the names of orchestras that were created after the 'Chamber Orchestra' and have a rating greater than 7.5.
iii. Create a view of all orchestras which are originated in India.
iv. Update experience of all the members to 10 whose position is piano player

```
i)
SELECT name, position
FROM members
WHERE wage > (SELECT AVG(wage)
              FROM members
              WHERE position = 'violin');
ii)

SELECT name
FROM orchestras
WHERE year > (SELECT year FROM orchestras
              WHERE name = 'Chamber Orchestra')
AND rating > 7.5;

iii)
```

Create view India_Orchestra as

	(select * from orchestras where country_origin= “India”); iv) update members set experience=10 where position= “piano player”;			
6	**Write short note on :** **iii) The CAP Theorem.** **iv) Document-Based NOSQL Systems**	[5+5]	CO6	L2
	The CAP theorem, or CAP principle, is a central foundation for comprehending these trade-offs in distributed systems. The CAP theorem emphasizes the limitations that system designers have while addressing distributed data replication. It states that only two of the three properties—**consistency, availability, and partition tolerance**—can be concurrently attained by a distributed system.  Partition Tolerance CP AP Consistency Availability CA *Venn diagram of CAP theorem* **Document databases offer a variety of advantages, including:** • An intuitive data model that is fast and easy for developers to work with • A flexible schema that allows for the data model to evolve as application needs change • The ability to horizontally scale out Because of these advantages, document databases are general-purpose databases that can be used in a variety of use cases and industries. Document databases are considered to be non-relational (or NoSQL) databases. Instead of storing data in fixed rows and columns, document databases use flexible documents. Document databases are the most popular alternative to tabular, relational databases.			