Vivekanand Education Society’s Institute of Technology, Chembur, Mumbai, Department Of Computer Engineering, Year:2019-20 (Odd Sem) Test No.- 2 Solutions

Class : Third Year	**Division: A, B and C**
Semester :V	**Subject: Database Management System**
Date: 22/10/2019	**Time: 1 hr**

<table>
<tr><td>Q.1</td><td></td><td>(Attempt any five of the following)</td></tr>
<tr><td></td><td>a)</td><td>Explain DCL and TCL commands with syntax and suitable example.

DCL(Data Control Language) : DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.
Examples of DCL commands:

GRANT-gives user’s access privileges to database.
REVOKE-withdraw user’s access privileges given by using the GRANT command.

TCL(transaction Control Language) : TCL commands deals with the transaction within the database.
Examples of TCL commands:

COMMIT– commits a Transaction.
ROLLBACK– rollbacks a transaction in case of any error occurs.
SAVEPOINT–sets a savepoint within a transaction.
SET TRANSACTION–specify characteristics for the transaction.</td></tr>
<tr><td></td><td>b)</td><td>List and explain different Anomalies.

Insertion anomaly: If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow inserting in referencing relation. For Example, If we try to insert a record in STUDENT_COURSE with STUD_NO =7, it will not allow.

Deletion Anomaly:A deletion anomaly is the unintended loss of data due to deletion of other data.

Update Anomaly: An update anomaly is a data inconsistency that results from data redundancy and a partial update.

If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation.

For Example, If we try to delete a record from STUDENT with STUD_NO =1, it will not allow.</td></tr>
</table>

c) **Given a relation R (X,Y,W,Z,Q) and the set of FDs F={X→ Z, Y→ Q, ZQ → W}. Find X^+ , Y^+, W^+ and candidate key for the relation.**

Ans:

X^+ ={X,Z }

Y^+ ={YQ }

W^+ ={W }

CANDIDATE KEY :XY

d) **Explain deadlock in DBMS with example.**

A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks.

For example: In the student table, transaction T1 holds a lock on some rows and needs to update some rows in the grade table. Simultaneously, transaction T2 holds locks on some rows in the grade table and needs to update the rows in the Student table held by Transaction T1.

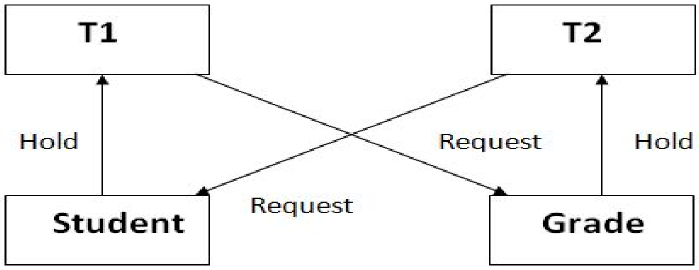

Figure: Deadlock in DBMS

Now Transaction T1 is waiting for T2 to release its lock and similarly, transaction T2 is waiting for T1 to release its lock. All activities come to a halt state and remain at a standstill. It will remain in a standstill until the DBMS detects the deadlock and aborts one of the transactions.

e) **Explain cascaded rollback with suitable example.**

If in a schedule, failure of one transaction causes several other dependent transactions to rollback or abort, then such a schedule is called as a **Cascading Schedule** or **Cascading Rollback** or **Cascading Abort**.

T1	T2	T3	T4
R (A)			
W (A)			
	R (A)		
	W (A)		
		R (A)	
		W (A)	
			R (A)
			W (A)
Failure			

Cascading Recoverable Schedule

Here,

- Transaction T2 depends on transaction T1.
- Transaction T3 depends on transaction T2.
- Transaction T4 depends on transaction T3.

In this schedule,

- The failure of transaction T1 causes the transaction T2 to rollback.
- The rollback of transaction T2 causes the transaction T3 to rollback.
- The rollback of transaction T3 causes the transaction T4 to rollback.

Such a rollback is called as a **Cascading Rollback**.

f) **Explain conflict serializability with example**

If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are **conflict equivalent**.

We say that a schedule S is **conflict serializable** if it is conflict equivalent to a serial schedule

Non-serial schedule

T1	T2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

Schedule S1

Serial Schedule

T1	T2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)

Schedule S2

Q.2 a) Consider the given schema and write the SQl queries for given statements.

Suppliers(sid: integer, sname: varchar(50), address: varchar(60))
Parts(pid: integer, pname: varchar(50), color: varchar(20))
Catalog(sid: integer, pid: integer, cost: real)

(a) Find the sids of suppliers who supply a red part and a green part.

```
SELECT DISTINCT C.sid
FROM Catalog C, Parts P
WHERE C.pid = P.pid AND P.color = 'Red'
INTERSECT
SELECT DISTINCT C1.sid
FROM Catalog C1, Parts P1
WHERE C1.pid = P1.pid AND P1.color = 'Green'
```

(b) Find the price of the least expensive red part.

```
SELECT cost
FROM catalog
WHERE cost =  (SELECT MIN(cost) FROM catalog);
```

(c) Find the snames of suppliers who supply some red part.

```
SELECT S.sname
FROM Suppliers S, Parts P, Catalog C
WHERE P.color='red' AND C.pid=P.pid AND C.sid=S.sid
```

OR

	b)	**Consider the given schema and associated functional dependencies. Decompose given schema into 1NF, 2NF and 3NF.** **Project(p_id,p_name,emp_id,emp_name,emp_dept,emp_hr_rate, total _hrs_worked)** **p_id→ p_name** **emp_id→ emp_name, emp_dept** **emp_dept→ hr_rate** **p_id, emp_id→ total_hrs_worked**
Q.3	a)	**Define transaction and also explain ACID properties of it.** A **transaction** is a unit of program execution that accesses and possibly updates various data items. To preserve the integrity of data the database system must ensure: • **Atomicity.** Either all operations of the transaction are properly reflected in the database or none are. • **Consistency.** Execution of a transaction in isolation preserves the consistency of the database. • **Isolation.** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions. ◦ That is, for every pair of transactions T_i and T_j, it appears to T_i that either T_j, finished execution before T_i started, or T_j started execution after T_i finished. • **Durability.** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.
		OR
	b)	**Explain 2 PL with proper example and state whether it is free from deadlock or not?** Two Phase Locking – A transaction is said to follow Two Phase Locking protocol if Locking and Unlocking can be done in two phases. Growing Phase: New locks on data items may be acquired but none can be released. Shrinking Phase: Existing locks may be released but no new locks can be acquired. Note – If lock conversion is allowed, then upgrading of lock(from S(a) to X(a)) is allowed in Growing Phase and downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

	T_1	T_2
1	LOCK-S(A)	
2		LOCK-S(A)
3	LOCK-X(B)	
4	....	......
5	UNLOCK(A)	
6		LOCK-X(C)
7	UNLOCK(B)	
8		UNLOCK(A)
9		UNLOCK(C)
10	....	......

Transaction T1:

- Growing Phase is from steps 1-3.
- Shrinking Phase is from steps 5-7.
- Lock Point at 3

Transaction T2:

- Growing Phase is from steps 2-6.
- Shrinking Phase is from steps 8-9.
- Lock Point at 6

2-PL ensures Conflict Serializability but ***does not*** prevent Cascading Rollback and Deadlock.

~ All the best!!! ~

b) Insertion anomaly: If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow inserting in referencing relation.

A deletion anomaly is the unintended loss of data due to deletion of other data.

An update anomaly is a data inconsistency that results from data redundancy and a partial update.

c) Given a relation R (X,Y,W,Z,Q) and the set of FDs F={X→ Z, Y→ Q, ZQ → W}. Find $\mathbf{X^+}$ **,** $\mathbf{Y^+}$**,** $\mathbf{W^+}$ and candidate key for the relation.

X^+ ={X,Z }
Y^+ ={YQ }
W^+ ={W }
CANDIDATE KEY :XY

d)Explain deadlock in DBMS with example
Consider the partial schedule

T_3	T_4
lock-x (*B*)	
read (*B*)	
B := *B* - 50	
write (*B*)	
	lock-s (*A*)
	read (*A*)
	lock-s (*B*)
lock-x (*A*)	

Neither T3 nor T4 can make progress — executing lock-S(B) causes T4 to wait for T3 to release its lock on B, while executing lock-X(A) causes T3 to wait for T4 to release its lock on A.
Such a situation is called a deadlock.
To handle a deadlock one of T3 or T4 must be rolled back
and its locks released.

3a)Define transaction and also explain ACID properties of it