DBMS PROJECT

S-MART: An Online Retail Store

Documentation

Maanas Gaur RollNo : 2021537
Lakshya Kumar RollNo : 2021536
Group Number : 157

Transactions:

Transaction1(T1): R1(x),W1(x)
Transaction2(T2): R2(x)
Transaction3(T3): R3(x),W3(x)

Schedules:

Schedule1(S1): R1(x);R3(x);W1(x);R2(x);W3(x)
Schedule2(S2): R3(x);R2(x);W3(x);R1(x);W1(x)

Tables:

Schedule1

T1	T2	T3
R1(x)		
		R3(x)
W1(x)		
	R2(x)	
		W3(x)

Conflicts: R1(x)-W3(x) : Read-Write
R3(x)-W1(x) : Read-Write
W1(x)-R2(x) : Write-Read
W1(x)-W3(x) : Write-Write

Schedule2

T1	T2	T3
		R3(x)
	R2(x)	
		W3(x)
R1(x)		
W1(x)		

Conflicts: R3(x)-W1(x) : Read-Write
R2(x)-W3(x) : Read-Write
W3(x)-R1(x) : Write-Read
W3(x)-W1(x) : Write-Write

Testing for Conflict Serializability:

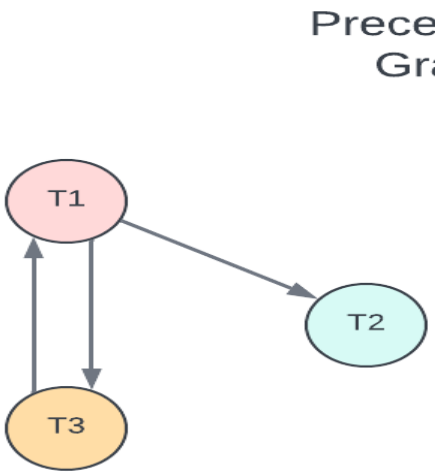

Fig1: S1

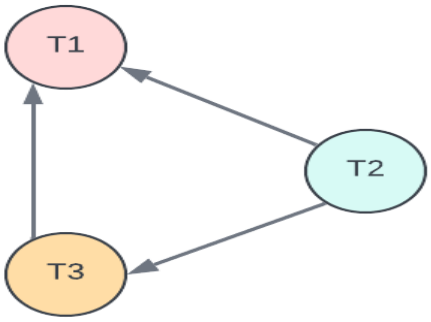

Fig2: S2

Note: S1 graph is drawn till a cycle is found

Conflicting Pairs:
Two operations inside a schedule are called conflicting if they meet these three conditions:

- ★ They belong to two different transactions.
- ★ They are working on the same data item.
- ★ One of them is performing the WRITE operation**.**

To check if a non-serial schedule is conflict serializable, the following steps can be taken:

Identify all conflicting operations within the schedule and list them.
Construct a Precedence Graph where each transaction in the schedule is represented by a node.
For each conflicting operation pair (Xi(A) and Yj(A)), draw a directed edge from transaction Ti to Tj in the Precedence Graph. This edge ensures that Ti is executed before Tj.
Check the Precedence Graph for the presence of any cycles. If there are no cycles, the schedule is conflict serializable.
A schedule is conflict serializable if it can be transformed into a conflict equivalent serial schedule by swapping the conflicting operations. The Precedence Graph technique provides a method to determine whether such a transformation is possible by checking for the presence of cycles in the graph. If there are no cycles, then the schedule can be transformed into a serial schedule by simply executing transactions in the order determined by the Precedence Graph.

Because the precedence graph of the S1 schedule has a cycle, it is non-conflict serializable, whereas the precedence graph of the S2 schedule does not contain a cycle, making it conflict serializable, and we can derive an analogous serial schedule from the precedence graph that is T2->T3->T1.

Sql Transactions:

Transaction 1:
This transaction selects the row with the customer ID of 2 and displays its data. Then, it updates the email address for that customer to

'WeeWee@Gmail.com'. Finally, the transaction is committed, which means the changes made are permanently saved in the database.

```
START TRANSACTION;

SELECT * FROM customer WHERE Customer_ID=2;
UPDATE customer SET email = 'WeeWee@Gmail.com' WHERE Customer_ID=2;

COMMIT;
```

Transaction 2:
This transaction selects the Full_Name column from the row with the customer ID of 2 and displays it. No changes are made to the database. Then, the transaction is committed.

```
START TRANSACTION;

SELECT Full_Name FROM customer WHERE Customer_ID=2;

COMMIT;
```

Transaction 3:
This transaction selects the row with the customer ID of 2 and displays its data. Then, it updates the email address for that customer to 'JKRowling@Gmail.com'. Finally, the transaction is committed, which means the changes made are permanently saved in the database.

```
START TRANSACTION;

SELECT * FROM customer WHERE Customer_ID=2;
UPDATE customer SET email = 'JKRowling@Gmail.com' WHERE Customer_ID=2;

COMMIT;
```

Locking:

Types of Lock:

A shared lock, denoted as S(x) where x is data item, allows multiple users to read the data but restricts them from making any changes to it. This means that while a shared lock is in place, data can only be read but not modified.

On the other hand, an **exclusive lock**, denoted as X(x) where x is data item, allows a single user to both read and modify the data. This means that while an exclusive lock is in place, the user holding the lock has exclusive access to the data, and no other users can read or modify it until the lock is released.

2PL protocol: 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.

Rigorous 2PL:

The lock must be 2-Phase, and all Shared(X) and Exclusive(X) locks held by the transaction must not be released until the Transaction Commits.

Ensuring:

1.Schedule is Serializable

2.Schedule is Recoverable

3.Schedule is Cascadeless

Schedule1

T1	T2	T3
X(x)		
R1(x)		
		X(x)
		R3(x)
X(x)		
W1(x)		
commit		
U-X(x)		
	S(x)	
	R2(x)	
	commit	
	U-S(x)	
		W3(x)

		commit
		U-X(x)

Reference: https://www.scaler.com/topics/dbms/serializability-in-dbms/