Paper Name: Database Management Systems

Topic: Concurrency Problem in DBMS

Paper Code: BCA CC410

By Ms. Manisha Prasad

Head, Assistant Professor,

Department of Computer Science

Patna Women's College

Concurrency Problems in DBMS Transactions

We know that in real world applications the transactions cannot occur one by one or serially, otherwise it would cause inordinate delays. Therefore we need to have multiple transactions executing simultaneously or concurrently.

When multiple transactions execute concurrently in an uncontrolled or unrestricted manner, then it might lead to several problems. These problems are commonly referred to as concurrency problems and can lead to Inconsistency in the Database environment. The concurrency problems that can occur in database are as follows:

1. **Dirty Read Problem:**
   Dirty read problem occurs when one transaction updates an item and fails. But the updated item is used by another transaction before the item is changed or reverted back to its previous value.

   Example:

T1	**T2**
R(X)	
X=X+1	
W(X)	
	R(X)
	X=X+M
	W(X)
R(Y)	
ABORTS DUE TO SOME FAILURE AND ROLL BACK	

   In the above example, Let us assume that initial value of X is 10, T1 changed the value of X to 11 by incrementing it and writing it in the local buffer, Now transaction T2 reads the value of X as 11 and starts doing its work. Now if transaction T1 fails for some reason, then the Transaction T1 will roll back (**Atomicity Property**) and all data items will be restored to its previous values. But transaction T2 has already read the incorrect value of X and proceeded further.

2. **Unrepeatable Read Problem:**

The unrepeatable problem occurs when two or more read operations of the same transaction read different values of the same variable.
Example:

T1	**T2**
R(X)	
	R(X)
X=X+1	
W(X)	
	R(X)

In the above example, once transaction T2 reads the variable X, a write operation in transaction T1 changes the value of the variable X. Now in the subsequent read operation Transaction T2 reads a different value of the same data item.
For ex:- if the initial value of X was 10 both T1 and T2 read the value 10. Then T1 incremented the value of X by 1 and made it 11 and wrote it in the local buffer. When a another read operation is performed by transaction T2, it reads the new value of X, i.e. 11 which was updated by transaction T1.
So for Transaction T2 there will be a confusion as to how the value of X changed without any operation done by it.
We have to remember that the in the basic property of transaction (ACID Property), ISOLATION property states that each transaction should run in isolation of other transaction.

3. **Phantom Read Problem:**
The Phantom read problem occurs when a transaction reads a variable once but when it tries to read that same variable again, an error occurs saying that the variable does not exist.
Example:

T1	**T2**
R(X)	
	R(X)
Delete (X)	
	R(X)

In the above example, once transaction T2 reads the variable X, transaction T1 deletes the variable X without transaction T2's knowledge. Thus, when transaction T2 tries to read X, it is not able to do so.

4. **Lost Update Problem(write –write conflict):**

In the lost update problem, update done to a data item by a transaction is lost as it is overwritten by the update done by another transaction.

Example:

T1	**T2**
R(X)	
	R(X)
X=X+1	
W(X)	
	X= X+ 10
	W(X)
	COMMIT
COMMIT	

We already know that till the Transaction is not committed , all the operations are performed in the local buffer or the main memory and changes are done here by the transactions (as mentioned in Transaction States) . Once all the operations are successful in the transaction , then it is committed which means the changes are made permanently in the Database.

In the above example, We assume that initial value of X was 10. Transaction T1 changes the value of X to 11and writes it . however in the mean time Transaction T2 changes the value of X to 20 , so the value of X gets overwritten by the update done by transaction T2 on X, and commits first making this value permanent in the Database. Therefore, the update done by transaction T1 is lost because according to Transaction T1 the value should have been 11 whereas the value of X in the database has become 20.