CONCURRENCY CONTROL

Concurrency control is the process of managing simultaneous execution of transactions in a shared database, to ensure the serializability of transactions, is known as concurrency control.

• Process of managing simultaneous operations on the database without having them interfere with one another.

• Prevents interference when two or more users are accessing database simultaneously and at least one is updating data.

• Although two transactions may be correct in themselves, interleaving of operations may produce an incorrect result.

Need for Concurrent Execution in DBMS

- In a multi-user system, multiple users can access and use the same database at one time, which is known as the concurrent execution of the database. It means that the same database is executed simultaneously on a multi-user system by different users.
- While working on the database transactions, there occurs the requirement of using the database by multiple users for performing different operations, and in that case, concurrent execution of the database is performed.
- The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database. Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.

Problems with Concurrent Execution

Simultaneous execution of transactions over a shared database can create several data integrity and consistency problems.

• lost updated problem

• Temporary updated problem

• Incorrect summary problem

Problem 1: Lost Update Problems

The problem occurs when two different database transactions perform the read/write operations on the same database items in an interleaved manner (i.e., concurrent execution) that makes the values of the items incorrect hence making the database inconsistent.

For example:

Consider the below diagram where two transactions T_X and T_Y, are performed on the same account A where the balance of account A is $300.

Time	T_x	T_y
t_1	READ (A)	—
t_2	A = A - 50	
t_3	—	READ (A)
t_4	—	A = A + 100
t_5	—	—
t_6	WRITE (A)	—
t_7		WRITE (A)

- At time t1, transaction T_X reads the value of account A, i.e., $300 (only read).
- At time t2, transaction T_X deducts $50 from account A that becomes $250 (only deducted and not updated/write).
- Alternately, at time t3, transaction T_Y reads the value of account A that will be $300 only because T_X didn't update the value yet.
- At time t4, transaction T_Y adds $100 to account A that becomes $400 (only added but not updated/write).
- At time t6, transaction T_X writes the value of account A that will be updated as $250 only, as T_Y didn't update the value yet.
- Similarly, at time t7, transaction T_Y writes the values of account A, so it will write as done at time t4 that will be $400. It means the value written by T_X is lost, i.e., $250 is lost.
- Hence data becomes incorrect, and database sets to inconsistent.

2. Temporary updated problem

- This problem occurs when one transaction updates a database item and then the transaction fails for some reason. The updated item is accessed by another transaction before it is changed back to its original value.
- Occurs when one transaction can see intermediate results of another transaction before it has committed.

Time	T_x	T_y
t_1	READ (A)	—
t_2	A = A + 50	—
t_3	WRITE (A)	—
t_4	—	READ (A)
t_5	SERVER DOWN ROLLBACK	—

- At time t1, transaction T_X reads the value of account A, i.e., $300.
- At time t2, transaction T_X adds $50 to account A that becomes $350.
- At time t3, transaction T_X writes the updated value in account A, i.e., $350.
- Then at time t4, transaction T_Y reads account A that will be read as $350.
- Then at time t5, transaction T_X rollbacks due to server problem, and the value changes back to $300 (as initially).
- But the value for account A remains $350 for transaction T_Y as committed, which is the dirty read and therefore known as the Dirty Read Problem.

Incorrect summary problem

- If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
- Occurs when transaction reads several values but second transaction updates some of them during execution of first.

T1	T2
|	Sum:=0 Read(A) Sum:=sum+A Read(B) Sum:=sum+B . . . Read(X) Sum:=sum+X
Read(B) B:=B-N Write(B)	
Read(X) X:=X-N Write(X)	

Example:

- T6 is totaling balances of account x (£100), account y (£50), and account z (£25).
- Meantime, T5 has transferred £10 from bal(x) to bal(z), so T6 now has wrong result (£10 too high).
- Problem avoided by preventing T6 from reading bal(x) and bal(z) until after T5 completed updates.