Overview

When several transactions execute concurrently without any rules and protocols, various problems arise that may harm the data integrity of several databases. These problems are known as **concurrency control problems**. Therefore several rules are designed, to maintain consistency in the transactions while they are executing concurrently which are known as concurrency control protocols.

What is concurrency control in DBMS?

A transaction is a single reasonable unit of work that can retrieve or may change the data of a database. Executing each transaction individually increases the *waiting time* for the other transactions and the overall execution also gets delayed. Hence, to increase the throughput and to reduce the waiting time, transactions are executed concurrently.

Example: Suppose, between two railway stations, **A** and **B**, 5 trains have to travel, if all the trains are set in a row and only one train is allowed to move from station **A** to **B** and others have to wait for the first train to reach its destination then it will take a lot of time for all the trains to travel from station **A** to **B**. To reduce time all the trains should be allowed to move concurrently from station **A** to **B** ensuring no risk of collision between them.

When several transactions execute simultaneously, then there is a risk of violation of the data integrity of several databases. **Concurrency Control in DBMS** is a procedure of managing simultaneous transactions **ensuring their atomicity, isolation, consistency and serializability**.

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.
- **Concurrency Control Problems**
- Several problems that arise when numerous transactions execute simultaneously in a random manner are referred to as Concurrency Control Problems.
- Dirty Read Problem
- The dirty read problem in DBMS occurs when *a transaction reads the data that has been updated by another transaction that is still uncommitted.* It arises due to multiple uncommitted transactions executing simultaneously.
- **Example:** Consider two transactions A and B performing read/write operations on a data DT in the database DB. The current value of DT is 1000: The following table shows the read/write operations in A and B transactions.

Time	**A**	**B**
T1	READ(DT)	------

Time	A	B
T2	DT=DT+500	------
T3	WRITE(DT)	------
T4	------	READ(DT)
T5	------	COMMIT
T6	ROLLBACK	------

- Transaction A reads the value of data DT as 1000 and modifies it to 1500 which gets stored in the temporary buffer. The transaction B reads the data DT as 1500 and commits it and the value of DT permanently gets changed to 1500 in the database DB. Then some server errors occur in transaction A and it wants to get rollback to its initial value, i.e., 1000 and then the dirty read problem occurs.
- Unrepeatable Read Problem
- The unrepeatable read problem occurs when two or more different values of the same data are read during the read operations in the same transaction.
- **Example:** Consider two transactions A and B performing read/write operations on a data DT in the database DB. The current value of DT is 1000: The following table shows the read/write operations in A and B transactions.

Time	A	B
T1	READ(DT)	------
T2	------	READ(DT)

Time	**A**	**B**
T3	DT=DT+500	------
T4	WRITE(DT)	------
T5	------	READ(DT)

- Transaction A and B initially read the value of DT as 1000. Transaction A modifies the value of DT from 1000 to 1500 and then again transaction B reads the value and finds it to be 1500. Transaction B finds two different values of DT in its two different read operations.
- Phantom Read Problem
- In the phantom read problem, data is read through two different read operations in the same transaction. In the first read operation, a value of the data is obtained but in the second operation, an error is obtained saying the data does not exist.
- **Example:** Consider two transactions A and B performing read/write operations on a data DT in the database DB. The current value of DT is 1000: The following table shows the read/write operations in A and B transactions.

Time	**A**	**B**
T1	READ(DT)	------
T2	------	READ(DT)
T3	DELETE(DT)	------
T4	------	READ(DT)

- Transaction B initially reads the value of DT as 1000. Transaction A deletes the data DT from the database DB and then again transaction B reads the value and finds an error saying the data DT does not exist in the database DB.
- Lost Update Problem
- The Lost Update problem arises when an update in the data is done over another update but by two different transactions.
- **Example:** Consider two transactions A and B performing read/write operations on a data DT in the database DB. The current value of DT is 1000: The following table shows the read/write operations in A and B transactions.

Time	**A**	**B**
T1	READ(DT)	------
T2	DT=DT+500	------
T3	WRITE(DT)	------
T4	------	DT=DT+300
T5	------	WRITE(DT)
T6	READ(DT)	------

- Transaction A initially reads the value of DT as 1000. Transaction A modifies the value of DT from 1000 to 1500 and then again transaction B modifies the value to 1800. Transaction A again reads DT and finds 1800 in DT and therefore the update done by transaction A has been lost.

- Incorrect Summary Problem
- The Incorrect summary problem occurs when there is an incorrect sum of the two data. This happens when a transaction tries to sum two data using an aggregate function and the value of any one of the data get changed by another transaction.
- **Example:** Consider two transactions A and B performing read/write operations on two data DT1 and DT2 in the database DB. The current value of DT1 is 1000 and DT2 is 2000: The following table shows the read/write operations in A and B transactions.

Time	**A**	**B**
T1	READ(DT1)	------
T2	add=0	------
T3	add=add+DT1	------
T4	------	READ(DT2)
T5	------	DT2=DT2+500
T6	READ(DT2)	------
T7	add=add+DT2	------

- Transaction A reads the value of DT1 as 1000. It uses an aggregate function SUM which calculates the sum of two data DT1 and DT2 in variable add but in between the value of DT2 get changed from 2000 to 2500 by transaction B. Variable add uses the modified value of DT2 and gives the resultant sum as 3500 instead of 3000.