**(Need for Concurrency Control)**

**DBMS Concurrency Control**

Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database.

But before knowing about concurrency control, we should know about concurrent execution.

**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**

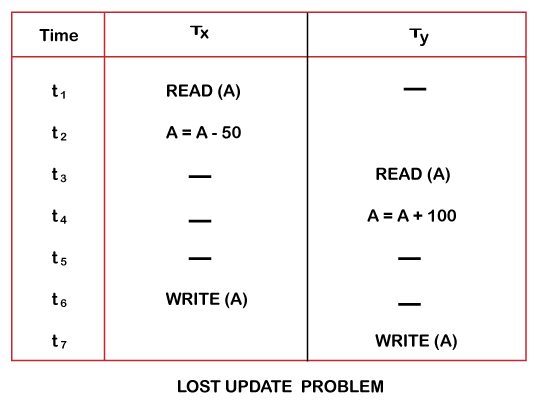
In a database transaction, the two main operations are **READ** and **WRITE** operations. So, there is a need to manage these two operations in the concurrent execution of the transactions as if these operations are not performed in an interleaved manner, and the data may become inconsistent. So, the following problems occur with the Concurrent Execution of the operations:

**Problem 1: Lost Update Problems (W - W Conflict)**

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 TX and TY, are performed on the same account A where the balance of account A is $300.**

* At time t1, transaction TX reads the value of account A, i.e., $300 (only read).
* At time t2, transaction TX deducts $50 from account A that becomes $250 (only deducted and not updated/write).
* Alternately, at time t3, transaction TY reads the value of account A that will be $300 only because TX didn't update the value yet.
* At time t4, transaction TY adds $100 to account A that becomes $400 (only added but not updated/write).
* At time t6, transaction TX writes the value of account A that will be updated as $250 only, as TY didn't update the value yet.
* Similarly, at time t7, transaction TY 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 TX is lost, i.e., $250 is lost.

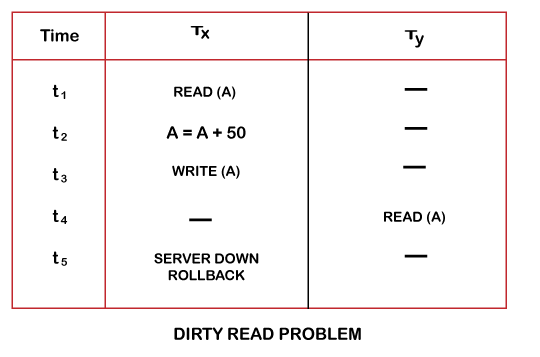
Hence data becomes incorrect, and database sets to inconsistent.

**Dirty Read Problems (W-R Conflict)**

The dirty read problem occurs *when one transaction updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction. There comes the Read-Write Conflict between both transactions.*

**For example:**

**Consider two transactions TX and TY in the below diagram performing read/write operations on account A where the available balance in account A is $300:**

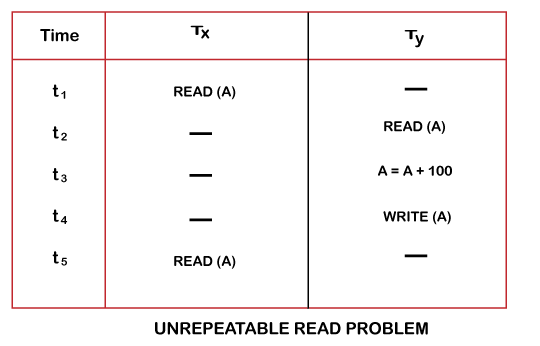
* At time t1, transaction TX reads the value of account A, i.e., $300.
* At time t2, transaction TX adds $50 to account A that becomes $350.
* At time t3, transaction TX writes the updated value in account A, i.e., $350.
* Then at time t4, transaction TY reads account A that will be read as $350.
* Then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

**Unrepeatable Read Problem (W-R Conflict)**

*Also known as Inconsistent Retrievals Problem that occurs when in a transaction, two different values are read for the same database item.*

**For example:**

**Consider two transactions, TX and TY, performing the read/write operations on account A, having an available balance = $300. The diagram is shown below:**

* At time t1, transaction TX reads the value from account A, i.e., $300.
* At time t2, transaction TY reads the value from account A, i.e., $300.
* At time t3, transaction TY updates the value of account A by adding $100 to the available balance, and then it becomes $400.
* At time t4, transaction TY writes the updated value, i.e., $400.
* After that, at time t5, transaction TX reads the available value of account A, and that will be read as $400.
* It means that within the same transaction TX, it reads two different values of account A, i.e., $ 300 initially, and after updation made by transaction TY, it reads $400. It is an unrepeatable read and is therefore known as the Unrepeatable read problem.

Thus, in order to maintain consistency in the database and avoid such problems that take place in concurrent execution, management is needed, and that is where the concept of Concurrency Control comes into role.

**Concurrency Control**

Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations and thus avoiding the inconsistencies in the database. Thus, for maintaining the concurrency of the database, we have the concurrency control protocols.

**Concurrency Control Protocols**

The concurrency control protocols ensure the *atomicity, consistency, isolation, durability* and *serializability* of the concurrent execution of the database transactions. Therefore, these protocols are categorized as:

* Lock Based Concurrency Control Protocol
* Time Stamp Concurrency Control Protocol
* Validation Based Concurrency Control Protocol

**Need for Concurrency Control in Executing Database Transactions**

**Why do we need concurrency control mechanisms in DBMS? Problems with concurrent execution of transactions**

**Need for Concurrency Control in Executing Database Transactions**

Concurrency or concurrent execution of transactions is about executing multiple transactions simultaneously. This is done by executing few instructions of one transaction then the next and so on. This way will increase the Transaction throughput (number of transactions that can be executed in a unit of time) of the system is increased.

When you execute multiple transactions simultaneously, extra care should be taken to avoid inconsistency in the results that the transactions produce. This care is mandatory especially when two or more transactions are working (reading or writing) on the same database items (data objects).

**For example**, one transaction is transferring money from account A to account B while the other transaction is withdrawing money from account A. these two transactions should not be permitted to execute in interleaved fashion like the transaction that are working on different data items. We need to serially execute (one after the other) such transactions.

If transactions are executed *serially*, i.e., sequentially with no overlap in time, no transaction concurrency exists. However, if concurrent transactions with interleaving operations are allowed in an uncontrolled manner, some unexpected, undesirable results may occur, such as:

1. The lost update problem: A second transaction writes a second value of a data-item (datum) on top of a first value written by a first concurrent transaction, and the first value is lost to other transactions running concurrently which need, by their precedence, to read the first value. The transactions that have read the wrong value end with incorrect results.
2. The dirty read problem: Transactions read a value written by a transaction that has been later aborted. This value disappears from the database upon abort, and should not have been read by any transaction ("dirty read"). The reading transactions end with incorrect results.
3. The incorrect summary problem: While one transaction takes a summary over the values of all the instances of a repeated data-item, a second transaction updates some instances of that data-item. The resulting summary does not reflect a correct result for any (usually needed for correctness) precedence order between the two transactions (if one is executed before the other), but rather some random result, depending on the timing of the updates, and whether certain update results have been included in the summary or not.

Most high-performance transactional systems need to run transactions concurrently to meet their performance requirements. Thus, without concurrency control such systems can neither provide correct results nor maintain their databases consistently.

**OTHER REFRENCES**

•      [Concurrency Control in DBMS (beginnersbook.com)](https://beginnersbook.com/2017/09/concurrency-control-in-dbms/#:~:text=When%20more%20than%20one%20transactions%20are%20running%20simultaneously,so%20that%20the%20integrity%20of%20data%20remains%20intact.)

•      [Concurrency Control in DBMS - GeeksforGeeks](https://www.geeksforgeeks.org/concurrency-control-in-dbms/)

•      [Why concurrency control is needed demonstrate with example? - Brainly.in](https://brainly.in/question/7335825)

**SUGGESTED BOOK REFERENCES**

* C.J.Date, “An Introduction to DatabaseSystems”, Addison Wesley.
* Thomas M. Connolly, Carolyn & E.Begg,“Database Systems: A Practical Approach to Design, Implementationand Management”, 5/E, University of Paisley, Addison-Wesley.
* Rob,”Database Principal Fundamental Design, Cengage Learning.