Concurrency Control

- In the concurrency control, the multiple transactions can be executed simultaneously
- It may affect the transaction result. It is highly important to maintain the order of execution of those transactions.

Problems of Concurrency Control

Several problems can occur when concurrent transactions are executed in an uncontrolled manner. Following are the the three problems in concurrency control.

1. Lost updates
2. Dirty read
3. Unrepeatable read.

1. Lost update problem

- when two transactions that access the same database items contains their operations in a way that makes the value of some database item incorrect, then the lost update ~~pro~~ problem occurs.
- If two transactions T1 and T2 read a record and then update it, then the effect of updating of the first record will be overwritten by the second update.

Example:

Transaction X	Time	Transaction Y
—	t1	—
Read A	t 2	Read A
—	t3	—
Update A	t4	update A
—	t5	—
—	t 6	

Here,

- At time t2, transaction - X reads A's value.
- At time t3, transaction - Y reads A's value.
- At time t4, transactions - X writes A's value on the basis of the value seen at time t2.
- At time t5, transactions - Y writes A's value on the basis of the value seen at time t3.
- So at time t5, the update of Transaction - X is lost because Transaction Y overwrites it without looking at its current value.
- Such type of problem is known as lost update Problem as update made by one transaction is lost ~~there~~ here.

2) Dirty Read

- The dirty read occurs in the case when one transaction updates an item of the database, and then the transaction fails for some reason. The updated database item is accessed by another transaction before it is changed back to the original value.
- A transaction T1 update a record ~~with~~ which is read by T2. If T1 aborts then T2 now has value which have never formed part of the stable database.

Example,

Transaction X	Time	Transaction Y
—	t1	—
—	t2	update A
Read A	t3	—
—	t4	Rollback
—	t5	—

- At time t2, transaction - Y writes A's value.
- At time t3, transaction - X read's A's value.
- At time t4, Transactions - Y rollbacks. So, It changes A's value back to that prior to t1.
- So, Transaction - X now contains a value which has never become part of the stable database.
- Such type of problem is known as Dirty Read problem, as one transaction reads a dirty value which ~~does~~ has not been committed.

3) Inconsistent Retrievals Problem

- Inconsistent Retrievals Problem is also known as unrepeatable read. When a transaction calculates some summary function over a set of data while the other transactions are updating the data, then the Inconsistent Retrievals Problem occurs.

- A transaction T1 reads a record and then does some other processing during which the transaction T2 updates the record. Now when the transaction T1 reads the record, then the new value will be inconsistent with the previous value.

Example:

Suppose two transactions operate on the Three accounts:

Account - 1	Account - 2	Account - 3
Balance - 200	Balance - 250	Balance - 150.

Transaction X	Time	Transaction Y
—	t 1	—
Real Balance of Acc-1 sum <-- 200	t 2	—
Read Balance of Acc-2 sum <-- sum + 250 = 450	t 3	—
—	t 4	Real Balance of Acc-3
—	t 5	Update Balance of Acc-3 150 --> 150 - 50 --> 100
—	t 6	Read Balance of Acc-1
—	t 7	Update Balance of Acc-1 200 --> 200+50 --> 250
Read Balance of Acc-3	t 8	COMMIT
Sum <-- Sum + 100 = 550	t 9	—

- Transaction -X is doing the sum of all balance whereas transaction -Y is transfering an amount 50 Account -1 to Account-3.)

- Here, transaction -X produces the result of 550 which is incorrect. If we write this produced result in the database, the database will become an inconsistem state because of the actual sum is 600.

- Here, transaction - X has seen an inconsistent state of the database.

Concurrency Control Protocol

It ensure atomicity isolation, and serializability of concurrent transactions. The concurrency control protocol can be divided into three ~~type~~ categories:

1. Lock based protocol
2. Time-stamp protocol
3. Validation based protocol.

Lock Based Protocol

In this type of protocol, any transaction cannot read or write data until it acquire an appropriate lock on it. There are two types of lock:

1. Shared lock

- It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.
- It can be shared between the transaction holds a lock, then it can't update the data on the data item.

2. Exclusive Lock:

- In the exclusive lock, the data item can be both reads as well as written by the transaction.
- This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

There are 4 types of lock Protocols available:

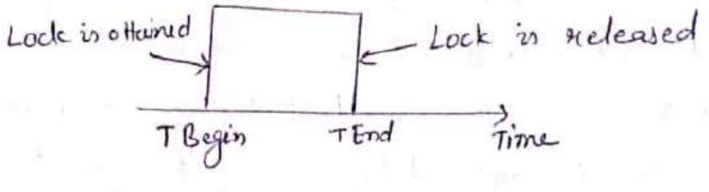
1. Simplistic Lock protocol

It is the simplest way of locking the data while transaction. Simplistic lock-based protocols allow all the transactions to get the lock on the data before insert or delete or update on it. It will unlock the data item after completing the transaction

2. Pre-claiming Lock Protocol

- Pre-claiming Lock protocols evaluate the transaction to list all the data items on which they need locks.
- Before initiating an execution of the transaction, it requests DBMS for all the lock on all those data items.

- If all the locks are granted then this protocol allows the transaction to begin. when the transaction is completed then it releases all the lock.
- If all the locks are not granted then this protocol allows the transaction to rolls back and waits until all the locks are granted.



3.) Two-phase locking (2PL)

- The two-phase locking protocol divides the execution phase of the transaction into three parts.
- In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires.
- In the second part, the transaction acquires all the locks. The third phase is started as soon as the transaction releases its first lock.
- In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.

There are two phases of 2PL:

Growing phase: In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

Shrinking phase: In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

In the below example, if lock conversion is allowed then the following phase can happen:

1. Upgrading of lock (from S(a) to X(a)) is allowed in growing phase.
2. Downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

Example:

	T1	T2
0	Lock-S(A)	
1		Lock-S(A)
2	Lock-X(B)	
3	—	—
4	UNLOCK(A)	
5		LOCK-X(C)
6	UNLOCK(B)	
7		UNLOCK(A)
8		UNLOCK(C)
9	—	—

The following way shows how unlocking and locking work with 2PL.

Transaction T1:

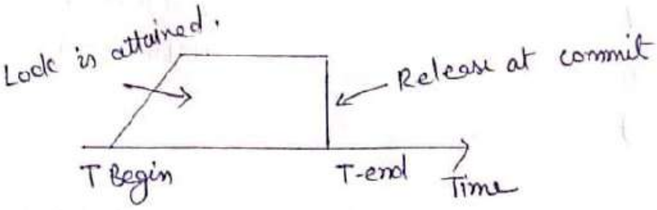
- Growing phase: from step 1-3
- Shrinking phase: from step 5-7
- Lock point : at 3

Transaction T2:

- Growing phase: from step 2-6
- Shrinking phase: from step 8-9
- Lock point: at 6.

4. Strict Two-Phase Locking (strict 2PL)

- The first phase of strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
- The only difference between 2PL and strict-2PL is that Strict 2PL does not release a lock after using it.
- Strict 2PL waits until the whole transaction to commit, and then it releases all the locks at a time.
- Strict 2PL protocol does not have shrinking phase of lock release.



It does not have cascading abort as 2 PL does.

Lecture No-44 5

Timestamp Ordering Protocol

- The Timestamp Ordering protocol is used to order the transactions based on their timestamps. The order of transaction is nothing but the ascending order of the transaction creation

- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.

- The lock based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But timestamp based protocols start working as soon as a transaction is created.

- Let's assume there are two transactions T_1 and T_2. Suppose the transaction T_1 has entered the system at 007 times and transaction T_2 has entered the system at 009 times. T_1 has the higher priority, so it executes first as it is entered the system first.

- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

- Basic Timestamp ordering protocol works as follows :—

1) Check the following condition whenever a transaction T_i issues a Read (X) operation :

  - if $W\_TS(X) > TS(T_i)$ then the operation is rejected.
  - if $W\_TS(X) <= TS(T_i)$ then the operation is executed.
  - Timestamps of all the data items are updated.

2) Check the following condition whenever a transaction T_i issues a Write (X) operation:

  - if $TS(T_i) < R\_TS(X)$ then the operation is rejected.
  - if $TS(T_i) < W\_TS(X)$ then the operation is rejected and T_i is rolled back Otherwise the operation is executed.

where,

TS(Ti) denotes the timestamp of the transaction Ti

R_TS(x) denotes the Read time-stamp of data item x.

W_TS(x) denotes the write time-stamp of data item x.

Advantage and Disadvantage of TO Protocol :—

- TO Protocol ensures serializability since the precedence graph is as follows :-

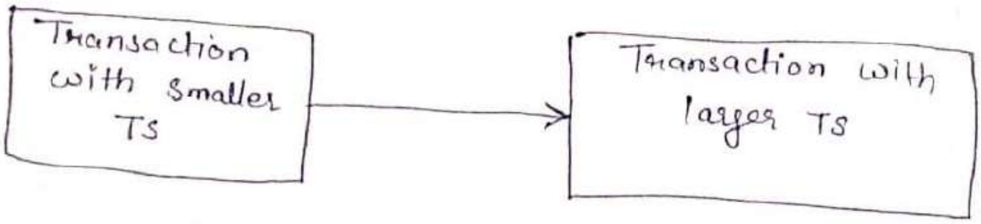


Fig - Precedence graph for TS ordering

- TS Protocol ensures freedom from deadlock that means no transaction ever waits.
- But the schedule may not be recoverable and may not even be cascade-free.

Validation Based Protocol

Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases:

(1) Read phase: In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.

(2.) Validation phase: In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.

(3) Write phase: If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Here each phase has the following different timestamps:

Start (Ti): It contains the time when Ti started its execution.

Validation (T_i): It contains the time when T_i finishes its read phase and starts its validation phase.

Finish (Ti): It contains the time when Ti finishes its write phase.

- This protocol is used to determine the time stamp for the transaction for serialization using the time stamp of the validation phase, as it is the actual phase which determines if the transaction will commit or roll back.
- Hence TS(T) = Validation (T).
- The serializability is determined during the validation process. It can't be decided in advance.
- while executing the transaction, it ensures a greater degree of concurrency control and also less number of conflicts.
- Thus it contains transactions which have less number of roll backs.

Multiple Granularity

- Granularity is the size of data item allowed to lock.
- Multiple granularity can be defined as hierarchically breaking up the database into blocks which can be locked.
- The Multiple granularity protocol enhances concurrency and reduces lock overhead.
- It maintains the track of what to lock and how to lock.
- It makes easy to decide either to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.
- Example –

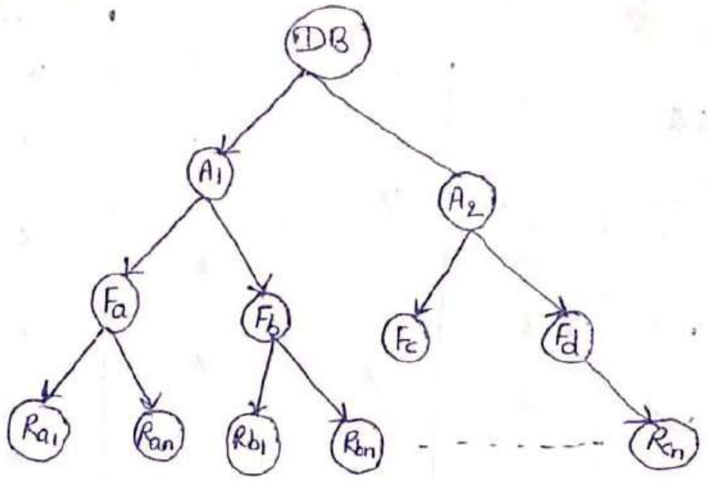


Fig – Multi granularity tree Hierarchy

The levels of the tree starting from the top level are as follows :–

1. Database
2. Area
3. File
4. Record

- There are three additional lock modes with multiple granularity.

Intention Mode Lock

- Intention - Shared (IS) – It contains explicit locking at a lower level of the tree but only with shared locks.

- Intention - Exclusive (IX) – It contains explicit locking at a lower level with exclusive or shared locks.

- Shared & Intention - Exclusive (SIX) :- In this lock, the node is locked in shared mode, and some node is locked in exclusive mode by the same transaction.

Compatibility Matrix with Intention lock

	IS	IX	S	SIX	X
IS	✓	✓	✓	✓	X
IX	✓	✓	X	X	X
S	✓	X	✓	X	X
SIX	✓	X	X	X	X
X	X	X	X	X	X

- It uses the intention lock modes to ensure serializability. It requires that if a transaction attempts to lock a node, then that node must follow these protocols :-

i) Transaction Ti should follow the lock - compatibility Matrix.

ii) Transaction Ti firstly locks the root of the tree. It can lock it in any mode.

iii) If Ti currenty has the parent of the node locked in either IX or IS mode, then the transaction Ti will lock a node in S or IS mode only.

iv) If Ti currenty has the parent of the node locked in either IX or SIX modes, then the transaction Ti will lock a node in X, SIX or IX mode only.

v) If Ti has not previously unlocked any node only, then the transaction Ti can lock a node.

vi) If Ti currently has none of the children of the node - locked only, then transaction Ti will unlock a node.

- Observe that in multiple granularity, the locks are acquired in top-down order, and locks must be released in bottom up order.