Concurrent Execution

If more than one transactions are executed at the same time then they are said to be executed concurrently.

Need for concurrency control:

- Isolation (+ consistency) ⇒ concurrency control
- Multiple transactions may want to access and modify the same resource.
- whenever multiple processes share resources there is need to schedule the access.

concurrency Control Techniques:

To avoid concurrency related problems and to maintain consistency, some rules (protocols) need to be made to increase the efficiency.

Lock-Based Protocols

To access any data item, transaction have to obtain lock on it.

Types of Lock

There are two types of locks that can be implemented on a transaction.

(i) **Shared lock:** Shared lock is Read-Only lock. If a transaction T_i has obtained shared lock on data item A then T_i can read A but can not modify A, and is represented by S and T_i is said to be in shared lock mode.

(ii) **Exclusive lock:** Exclusive lock is Read-Write lock. If a transaction T_i has obtained exclusive lock on data item A then T_i can both read and modify (write) A. & is denoted by X and T_i is said to be in Exclusive lock mode.

concurrency control manager: The working of concurrency control Manager is to grant locks to different transactions. It is basically a programme.

Compatible lock modes

	S	X
S	Possible	Not Possible
X	Not Possible	Not Possible

Lock to Be granted.

Lock-X(A); (Exclusive Lock, we want to both read A's value and modify it).

Read A;
A = A-100;
Write A;
Unlock (A); (Unlocking A after the modification is done).

Two Phase Locking Protocol

The use of locks has helped us to create current schedule. The Two Phase Locking Protocal defines the rules of how to acquire the locks on a data item and how to release the locks.

The two Phase locking Protocal assumes that a transaction can only be in one of two phases.

Growing Phase: In this phase the transaction can only acquire locks, but cannot release any lock. The transaction the growing phase as soon as it acquires the first lock it wants.

It can not release any lock at this phase even if it has finished working with a locked data item

Ultimately the transaction reaches a point where all the lock it may need has been acquired. This point is called Lock Point.

Shrinking Phase: After Lock point has been reached, the transaction enters the shrinking phase. In this phase transaction can only release locks, but can not acquire any new lock.

There are Two different versions of Two Phase Locking Protocol. one is called the Strict Two Phase locking protocol and the other one is called Rigorous Two Phase Locking Protocol.

Strict Two Phase Locking Protocol

In this protocol, a transaction may release all the shared locks after lock Point has been reached, but it cannot release any of the exclusive locks until the transaction commits. This protocol helps in creating cascade less schedule.

Cascading Rollback : Phenomenon of rolling back a child transaction if the parent transaction is rolled back, which causes a tremendous loss of processing power & execution time.

T_1

Lock-X(A)
Read A;
A = A-100;
Write A;
Unlock(A)

Lock-X(B)
Read B;
B = B+100;
Write B;
Unlock(B)

T_2

Lock-S(A)
Read A;
Temp = A * 0.1;
Unlock(A)
Lock-X(C)
Read C;
C = C + Temp;
Write C;
Unlock(C)

Rigorous Two Phase Locking Protocol

In Rigorous Two Phase locking Protocol a transaction is not allowed to release any lock (either shared or exclusive) until it commits. This means that until the transaction commits, other transaction might acquire a shared lock on a data item on which the uncommitted transaction has a shared lock; but can not acquire any lock on a data item on which the uncommitted transaction has an exclusive lock.

Timestamp Ordering Protocol

A timestamp denotes a specific time on which the transaction or data item has been activated in any way.

The timestamp of a data item can be of the following two types:

W-timestamp (Q): This means the latest time when the data item Q has been written into.

R-timestamp (Q): This means the latest time when the data item Q has been read from.

These two timestamps are updated each time a successful read/write operation is performed on the data item Q.

For Read Operations:

1. If TS(T) < W-timestamp(Q), then the transaction T is trying to read a value of data item Q which has already been overwritten by some other transaction. Hence the value which T wanted to read from Q does not exist there anymore, & T would be rolled back.

2. If TS(T) >= W-Timestamp(Q), then the transaction T is trying to read a value of data item Q which has been written and committed by some other transaction earlier. Hence T will be allowed to read the value of Q and R-timestamp of Q should be updated to TS(T).

For Write operations:

(1) If TS(T) < R-timestamp(Q), then it means that the system has waited too long for transaction T to write its value, and the delay has become so great that it has allowed another transaction to read the old value of data item Q. In such a case T has lost its relevance & will be rolled back.

(2). else if TS(T) < W-timestamp(Q), then transaction T has delayed so much that the system has allowed another transaction To write into the data item Q. in such a case too, T has lost its relevance & will be rolled back.

(3) otherwise the system executes transaction T and updates the W-timestamp of Q to TS(T).

Validation based protocal:

The two concurrency control techniques, one is locking based concurrency control, another is timestamping based concurrency technique. but in both the techniques, we have certain level of drawbacks.

In locking based techniques the item being accessed must be locked, it performs the operations in two steps first lock all the data items then secondly data items are modified.

In time stamp based techniques, each transaction is assigned a timestamp value and this transaction timestamp is checked against the read and write timestamp of the item. Both techniques causes the transaction slow due to these overheads.

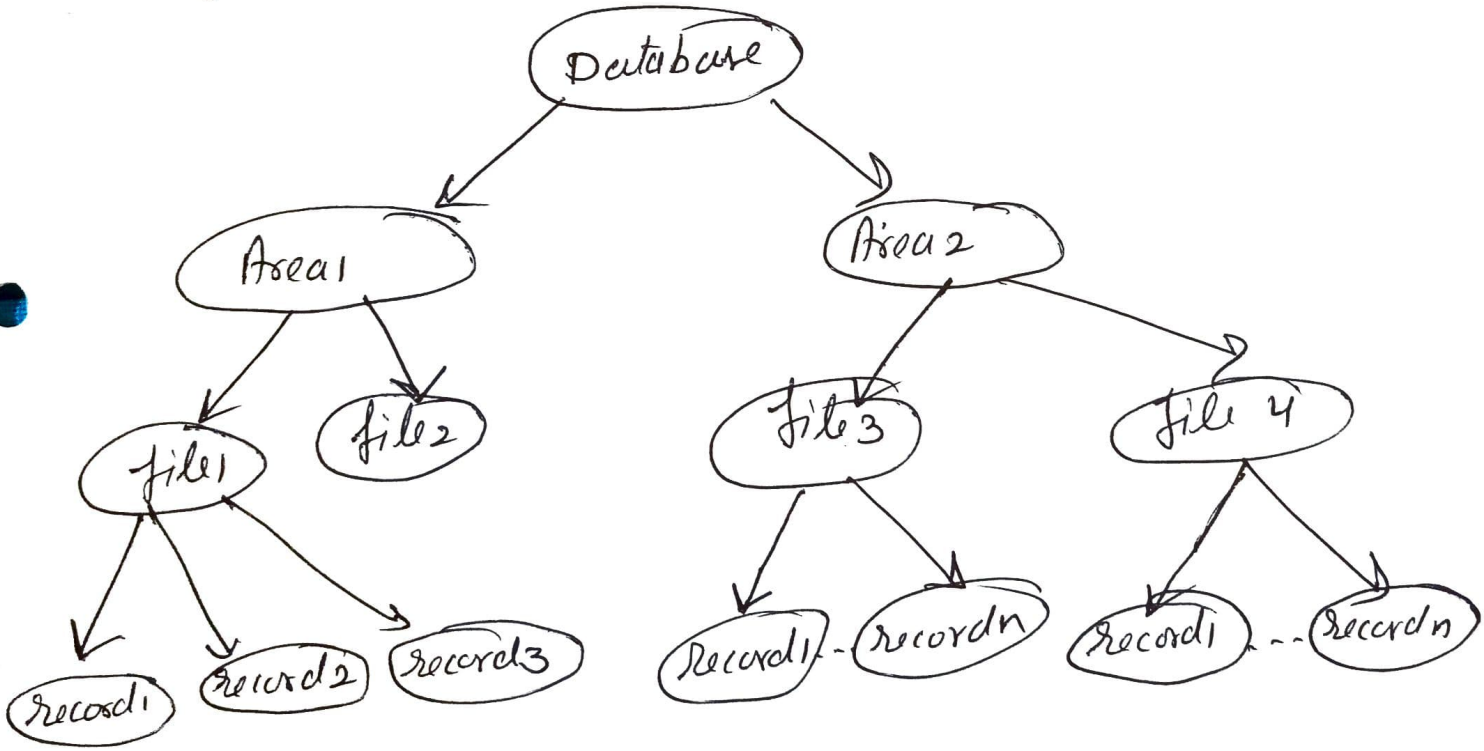
Three Phases of Validation (optimistic) based concurrency control techniques —

(1) Read Phase
(2) Validation Phase
(3) write Phase

Multiple granularity

Granularity means size of the data item being locked. Granularity is considered as the major factor concurrency control because it can effect the performance of concurrency and recovery.

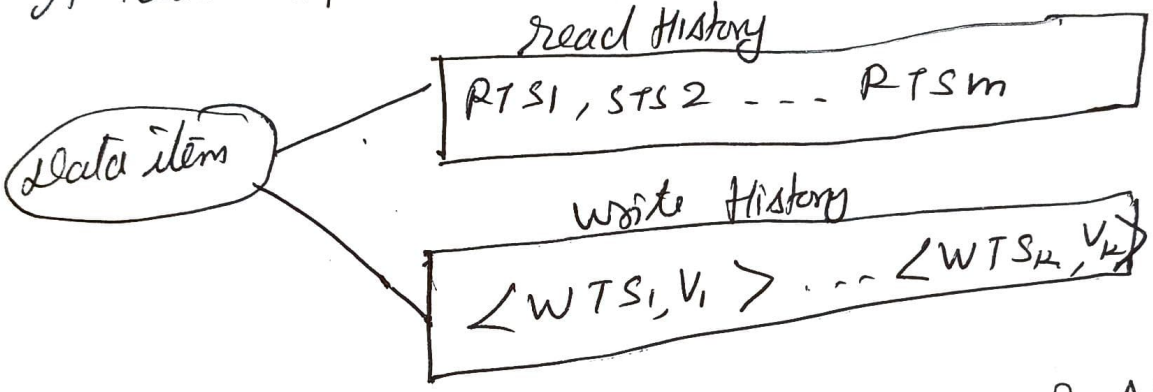
If the granularity of a data item is very large then overhead of locking is very low but it will effect the concurrency.



Multiversion

In multiversion concurrency control scheme each write (x) operation creates a new version of the item x.

- A TS(Xn) does not overwrite old value of a data item x.
- A read operation is never rejected.



The W-timestamp & R-timestamp of Nth Version: In this method several versions $x_1, x_2, x_3 \ldots x_n$ of each data item x are maintained for each version, the value of version Xn.

(i) Read-TS (Xn) or [R-TS (Xn)]

(ii) Write-TS (Xn) or [W-TS (Xn)]

Recovery with concurrent transaction

we discuss here how we can modify and extend the log-based recovery scheme to deal with multiple concurrent transaction.

(i) Interaction with concurrency control :→ The recovery scheme depends mostly on the concurrency control scheme that is used to rollback a failed transaction, we must undo the updates performed by the transaction.

(ii) Transaction Rollback : We rollback a failed transaction T_i by using log.

(iii) check Points : We use check points to reduce the number of log records that the system must scan when it recovers from a crash.

(iv). Restart Recovery — through UNDO-LIST & REDO-List.