UNIT-V (Concurrency Control Techniques)

-!- **Concurrency Control:-** Concurrency control is a process in which multiple transactions are executed at a time.

Advantages:-

(1) Reduces waiting time
(2) Response time decreases
(3) CPU utilization increases
(4) Efficiency increases

=> **Problems in Concurrency Control:-**

(1) Dirty Read Problem
(2) Unrepeatable Read Problem
(3) Lost Update Problem
(4) Phantom Read "

} Discussed in Unit-4.

!- **Techniques for Concurrency Control:-**

=> Concurrency Control Techniques => It is a process of managing simultaneous execution of transaction in shared database-

Purpose:-

(i) To enforce isolation
(ii) To preserve database inconsistency
(iii) To resolve conflicts (RW, WR, WW)

Techniques for Concurrency Control / Protocols –

1. Locking (Lock-Based Protocol)
2. Time-stamp based protocol
3. ~~Optimistic~~ Validation based protocol
4. Multiversion schemes.

(1) Locking Technique :– (Lock-Based Protocol)

→ A "Lock" gurantees exclusive use of a data item to a current transaction.

↳ Lock is used to access data item (Lock Acquire)

↳ After completion of transaction (Release lock)

⇒ A lock is a variable associated with each data item that indicates whether a read or write operation can be applied to the data item.

Types of Lock : 2 Types

(1) Shared lock (Lock-S)

→ Read data item value. cannot write.

→ Any no. of transactions can acquire shared locks on an data item.

(2) Exclusive lock (Lock-X)

→ ~~Update the~~ It is used for both Read & Write.

→ Only one transactions can acquire exclusive lock on a data item at a time.

Compatibility b/w Lock Modes :–

T_i \ T_j	S	X
S	✓	X
X	X	X

Compatibility Matrix

Note: Any no. of transactions can hold shared lock on an item But Exclusive lock can be hold only by one transactions at a time.

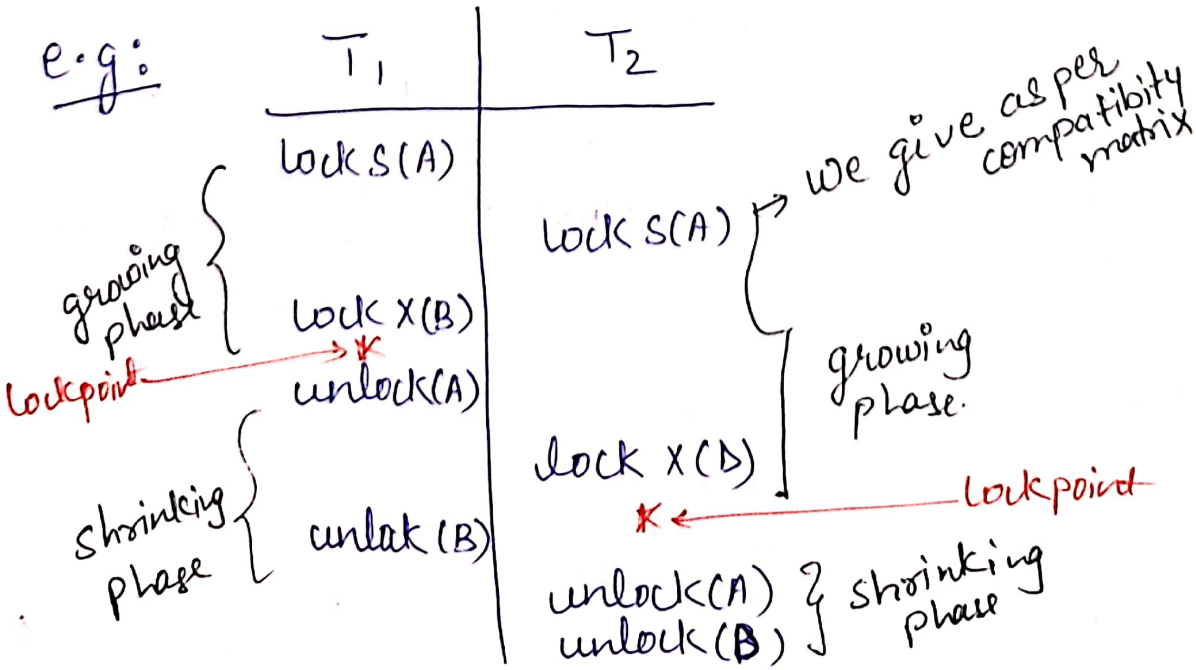
Two-Phase Locking Protocol (2PL):—

It is a lock based concurrency control technique that requires each transaction to be divided into two phases —

(1) Growing/expanding phase — In this phase, the transaction acquires all the locks it needed. Here, the number of locks held by a transaction increases from 0 to maximum.
→ Transaction cannot release any lock during this phase.

(2) Shrinking Phase — In this phase, the transaction releases all the locks. The no. of locks held by transaction uses from max. to zero.
→ Transaction cannot acquire any lock during this phase.

e.g:



→ A transaction can unlock a data item Q by unlock(Q) instruction.

Example:-

T_1	T_2	A	B
		100	200
lock-X(B)	lock-S(A)		
200 read(B)	read(A) 150		
150 B = B-50	unlock(A)		
150 write(B)	lock-S(B)		
unlock(B)	read(B) 150		
lock-X(A)	unlock(B)		
100 read(A)	display(A+B) 300		
150 A = A+50			
150 write(A)			
unlock(A)			

We assume T_1 then T_2 (serial execution)

A = 150, B = 150

A+B = 300

⇒ Now, if we assume concurrent execution of T_1 & T_2

A=100, B=200

T_1	T_2
lock-X(B)	
200 read(B)	
150 B = B-50	
150 write(B)	
unlock(B)	
	lock-S(A)
	read(A) 100
	unlock(A)
	lock-S(B)
	read(B) 150
	unlock(B)
	display(A+B) 250
lock-X(A)	
100 read(A)	

T_1

A = A+50 150

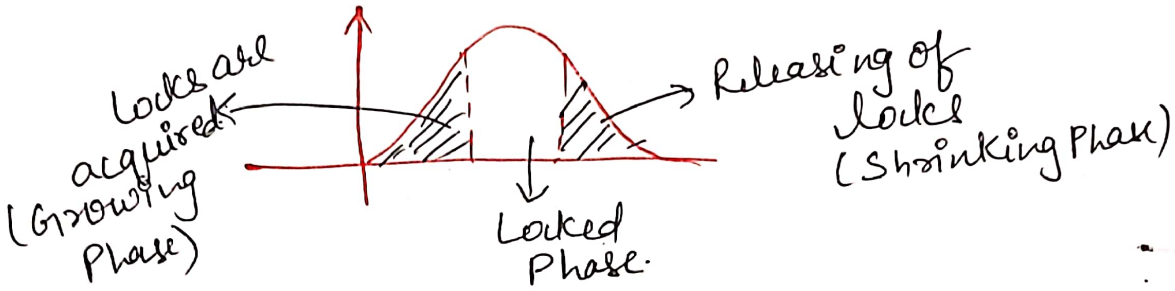
write(A) 150

unlock(A) 150

Ans: A+B = 250

∴ Here, no proper use of locks leads to inconsistency.

LOCK POINT:- It is a point at which a transaction has obtained its final lock.



* 2PL enforces serializability but may reduces concurrency due to foll. reasons —
  (i) Holding lock Un-necessarily.
  (ii) Locking too easily
  (iii) Penalty to other transaction.

* Variations of 2PL :—

(i) Conservative (Static) 2PL	(ii) Strict 2PL	(iii) Rigrous 2PL
↳ Acquire all locks before it starts.	↳ Exclusive lock cannot be released until commit.	↳ does not released both shared / exclusive until commit.
↳ Release all locks after commit	↳ Helps in cascadless schedules (avoid cascading roll back)	↳ Avoids cascading roll back.
↳ It avoids cascading Rollback.	↳ Deadlock may occur	↳ Deadlock may occur.
↳ It is also deadlock free.		

(2) Time Stamping Based Protocol :-

=> Timestamp: is a tage that can be attached to any transaction or data item, which denotes specific time on which the transaction or the data item had been activated in any way.

* Timestamp is a unique value assign to every transh. when they enter in sys-em
* Timestamp of each transaction can be genrated either by system time or logical counter.
* Transaction with lower timestamp has higher priority.

-> Timestamp of trans. T_i is denoted by $T_s(T_i)$.

-> $T_s(T_i) < T_s(T_j)$ => It means transaction T_j enters after T_i in the system.

* To implement this scheme, we associate two time stamp values with each data item(Q):-

(i) read - $T_s(Q)$:- read Timestamp of Q.
This is the largest timestamp among all the timestamps of transactions that have sucessfully executed read (Q) opn.

(ii) write - $T_s(Q)$:- write timestamp of Q.
This is the largest timestamp among all the timestamps of transactions that have sucessfully executed write (Q).

* largest timestamp give to the last transaction.

* Whenever new read(Q) or write(Q) operations are executed read-Ts(Q) & write-Ts(Q) are updated.

Basic Timestamp Ordering Protocol (Rules of) :-

Timestamp ordering protocol works as follows :-

(1) Transaction T_i issues a Read(A) operation

(a) if write-Ts(A) > Ts(T_i), then Rollback T_i

(b) Otherwise execute Read(A) op^n

Set Read-Ts(A) = Max{Read-Ts(A), Ts(T_i)}

(2) Transaction T_i issues Write(A) op^n –

(a) if Read-Ts(A) > Ts(T_i) then Rollback T_i

(b) if Write-Ts(A) > Ts(T_i) then Rollback T_i

(c) Otherwise execute write(A) op^n.

Set Write-Ts(A) = Ts(T_i)

Example:

100 ← older	200	300 ← younger
T_1	T_2	T_3
R(A)		
	R(B)	~~R(A)~~ R(B)
W(C)		
	~~R(A)~~	
R(C)		
	W(B)	

~~Read-Ts(A) = 30~~

Soln:

	A	B	C
		200	
Read-Ts	~~0~~ 100	~~0~~ 200	~~0~~ 100
Write-Ts	0	0	~~0~~ 100

(1) [Write-Ts(A) > TS(Ti)] DO

~~200~~ > 100 (False), Nothing

Set Read-Ts(A) = 100

0 > 200

Read Ts(B) = 200

(2) Ti issues write(A) –

~~Ts~~ Write Ts(C) > Ts(Ti)

0 > 300 NO

~~write~~ read-Ts(C) > Ti NO

then ~~T0~~ set write Ts(C) = T1

so. Validate(Ti) < Validate(Tj) : It ensures that

(3) Validation based Protocol (Optimistic Concurrency control technique)

In this type of ~~sche~~ technique, no checking is done. It means, it w

⇒ It is based on the assumption that conflict is rare and it is more efficient to allow transactions to proceed without imposing delays to ensure serializability.

⇒ It is also known as Optimistic Concurrency control technique.

⇒ In this technique no checking is done while the transaction is been executed. Until the transaction end is reached updates in the trans^n are not applied directly to the d/b. All updates are applied to local copies of data items kept ~~in~~ for the trans^n.

* There are three phases for every transaction—
  ① Read Phase ② Validation phase ③ Write phase.

(i) Read Phase:- At this phase, transaction read data items, values from database (committed data items) ~~& store them in~~ Updates are only applied to local data versions.

(ii) Validation Phase:- Checking is performed to make sure that there is no violation of serializability when the transaction

updates are applied to the database.

(iii) Write Phase:- On the success of the validation phase, the transaction updates are applied to the database, otherwise the updates are discarded and the transaction is slowed down.

* Timestamp is used to determine when to start Validation Test: There are three timestamps associated with every Trans°: T_i.

(a) START (T_i) : It gives time when T_i start execution.

(b) VALIDATION (T_i): It gives time when T_i finish its Read phase & start validation phase.

(c) FINISH (T_i) :- It gives time when T_i finished its execution or write phase.

* If transaction failed in Validation Test then its Aborted & Rollback.

⇒ There are the following conditions which must be satisfy to clear the validation Test —

(i) Finish (T_i) < Start (T_j) : It means T_i is older and get finish befor T_j starts. (No Overlap)

(ii) Finish (T_i) < Validate (T_j) :- It ensures actual write by T_i & T_j will not overlap

(iii) Validate (Ti) < Validate (Tj) : It ensures that Ti has completed read phase before Tj complete read phase.

Advantages :-

(1) Maintains Serializability
(2) Free from cascade Rollback
(3) Less Overhead than other protocols
(4) Avoid Deadlock

Disadvantages :-

(1) Starvation of long Transactions due to conflicting transactions.