CONCURRENCY CONTROL PROTOCOLS :-

1) Lock based Protocols
   a) 2 phase locking protocols (2PL)
      - basic 2PL
      - Conservative 2PL
      - Strict 2PL
      - Rigorous 2PL
   b) Graph based protocols.
2) Time-stamp based protocol.
   a) time stamp ordering protocol
   b) Thomas write rule
3) Multiple granularity protocol
4) Multi version protocol.
   a) multi-version 2PL
   b) multi-version Time stamp ordering.

LOCK based Protocol :-

lock data item before we access it and the lock must be done appropriately.

- If we read data, we allow other transaction to do some other operation.
- but if we write data, we dont allow

Scanned by CamScanner

- other transactions till we finish it.

types of locks :-

- Shared lock - read lock - Lock-R(A)
- Exclusive lock - write lock - Lock-W(A).
  ↳ when we write (no one is able to read or write.

(exclusive locking)

	S	X
S	✓	X
X	X	X

Compatibility.

any no of transaction can read at a time.

T_1 : $A \xrightarrow{50} B$

T_2 : $A + B$

T_1	T_2
L-S(A)	
Read(A)	
U-S(A)	
L-X(A)	
A = A-50	
write(A)	
U-X(A)	
	A + B(250)

A = 100 50
B = 200 250
300 300

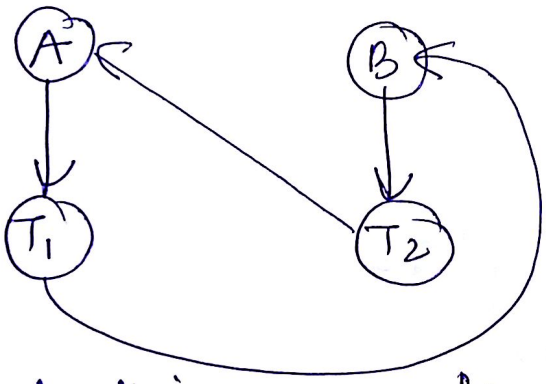
Inconsistent.
unlocking too early can give us inconsistent result!

Scanned by CamScanner

2- There may be deadlock.

T_1	T_2
L-X(A)	
	L-X(B)
L-X(B)	
	L-X(A).

data items A locked by transaction T_1
data items B " " " T_2



Simple locking can lead to deadlock.

→ They might not guarantee any serializibility.

T_1	T_2
L-X(A)	
W(A)	
U-X(A)	
	L-S(A)
	R(A)
	U-S(A)
L-X(A)	
W(A)	
U-X(A)	

not conflict serializibility

2-phase locking protocol:-

- It requires locking & unlocking can be done in 2 phases.
  (i) Growing phase
  (ii) Shrinking phase
- Transaction can obtain locks but not release locks in growing phase.
- Transaction can release the locks but not obtain the locks in shrinking phase.

T_1	T_2
W(A)	
	R(A)
W(A)	

T_1	T_2	T_3
	L-S(A)	
		L-S(A)
	L-X(B)	
	U(A)	
		L-X(C)
	U(B)	
L-S(B)		
		U(A)
		U(C)
L-X(A)		
U(A)		
U(B)		

Scanned by CamScanner

L-X(A)	
W(A)	
	W(A)
	W(B)
L-X(B)	
W(B)	
LP.	

$\boxed{T_2\ T_3\ T_1}$

Disadvantages

- There may be need of cascading aborts.

T_1	T_2
L-X(A)	
W(A)	
L-X(B)	
W(B)	
⋮	
U(A)	
U(B)	
	L-S(A)
	R(A)
Rollback	

T_1	T_2
L-X(A)	
	L-X(B)
L-X(B)	
	L-X(A)

Scanned by CamScanner

Variations in 2 Phase locking :-

1) **Strict 2-Phase locking Protocol :-**
which requires in addition to locking being 2-phase all exclusive mode locks must be held untill transaction commits.

2) **Rigorous 2-Phase locking Protocol :-**
which requires that in addition to locking being 2-phase all locks must be hold untill the transaction commits.

3) **Conservative 2-Phase locking Protocol :-**
- which requires the transaction to obtain all the locks before it starts and releases all the locks after it commits.
  ↳ it avoids deadlock.

T_i	T_j
L-X (A)	
W(A)	L-S(A) × never grant.
⋮	R(A)
C	
U(A)	

recoverable, cascadless and strict.

Scanned by CamScanner

Conservative
↳ no deadlock, serializible, Strict.

→ All guarantee serializability.

disadvantage:-
↳ we are unnecessarily obtaining locks which can lead to starvation.
- Performance can decrease.

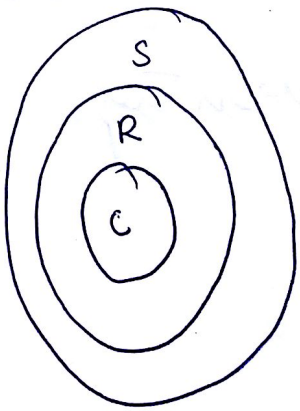
Q Which of the following schedules (transactions) are strict 2PL?

a) Lock-S(A)
R(A)
Lock-X(B)
R(B)
unlock(A)
W(A)
unlock(B)

b) Lock-S(A)
R(A)
unlock-X(B)
unlock(A)
R(B)
W(B)
commit
unlock(B)
↓
Strict

c) Lock-S(A)
R(A)
Lock-X(B)
R(B)
W(B)
commit
unlock(A)
unlock(B)
strict.

every rigorous is strict.



Scanned by CamScanner

Tree - based protocol:-

- Circular wait can be avoided.

Tree Protocol (Dependency Graph)

- is a tree using only exclusive locks.

each transaction Ti can lock ~~A~~ data items at most one and must observe the following rules.

(i) The Ist lock by Ti may be on any data items.

(ii) Subsequently an items can be locked by Ti only if the parent of A is concurrently locked by Ti.

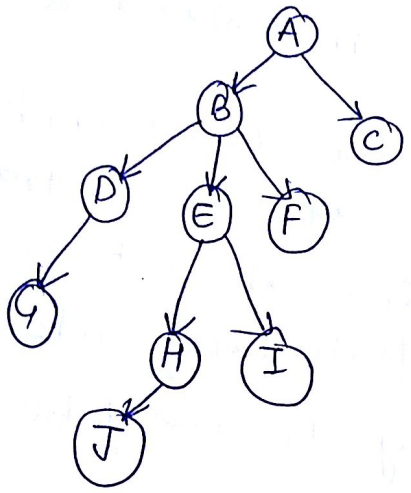
(iii) Data items may be unlocked at any time.

(iv) Subsequently relocking is not allowed.

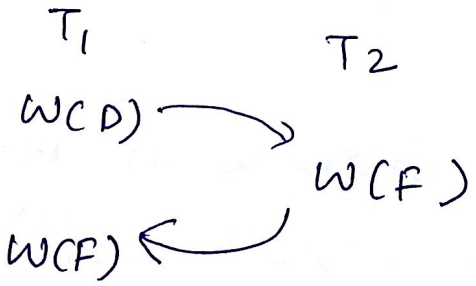
- Tree based protocol is both conflict serializable & deadlock free protocol.

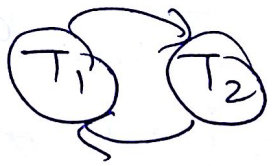
Advantages - easily unlocking
- more concurrency.

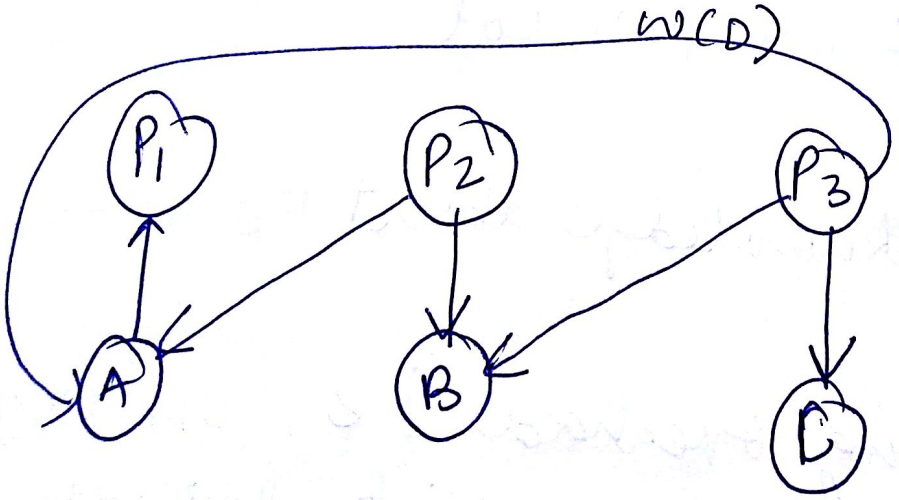
Scanned by CamScanner



- using tree we are imposing some order
- If someone wants to have access of both A & B. First we have to lock A then B.
- ABC then lock A then B or C any other because they are not ordered they are partially ordered.
- ABH A first then B then H.







Cycle

Scanned by CamScanner

- Circular wait can be avoided.

Definition :-

In a tree protocol the only lock instruction allowed is lock-X and each transaction Ti can lock a data item atmost once and must observe the following rules.

1) The first lock by Ti may be on any data item
2) Subsequently a data item can be locked by Ti only if the parent of the data item is currently locked by Ti
3) Data items may be unlocked at any time.
4) Data items that has been locked and unlocked by Ti can not subsequently released by Ti

Advantages :-

- Early unlocks cause increase in concurrency
- Ensures conflict serializability.
- it is deadlock free protocol.

disadvantages :-

- requires prior knowledge about the order of the data.
- unnecessary locking overhead i.e, in some cases lock the data items that it does not access.

Scanned by CamScanner

TIME-STAMP BASED PROTOCOLS:-

- each transaction T_i is assigned a unique timestamp, denoted by $Ts(T_i)$ prior to the beginning of execution

$Ts(T_i)$: transaction beginning time-stamp or system time or counter.

- logical timing value this transaction occurs before any other.

$$Ts(T_i) < Ts(T_j)$$

- schedule is serializable if T_i followed by T_j.

Every data item has Two time stamps:

- W-timestamp(Q) - denotes the largest TS value of any transaction that executes write(Q)
- R-timestamp(Q) denotes the largest TS value of any transaction that successfully executed read(Q).

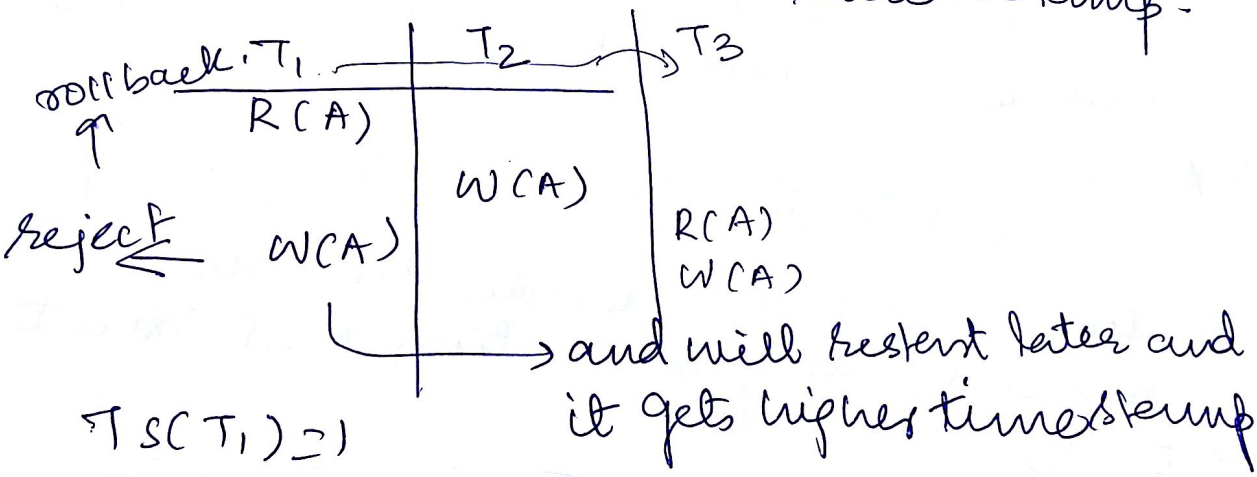
- time stamp ordering among the transactions is determined in advance based on their time stamps (i.e the time at which transaction enters into system for execution)
- If any transaction T_j enters after T_i, the relation between their time stamps

Scanned by CamScanner

- will be $TS(T_i) < TS(T_j)$ which means the resulting schedule must be equivalent to a serial schedule $T_i \rightarrow T_j$.

Time-Stamp ordering Protocol:-

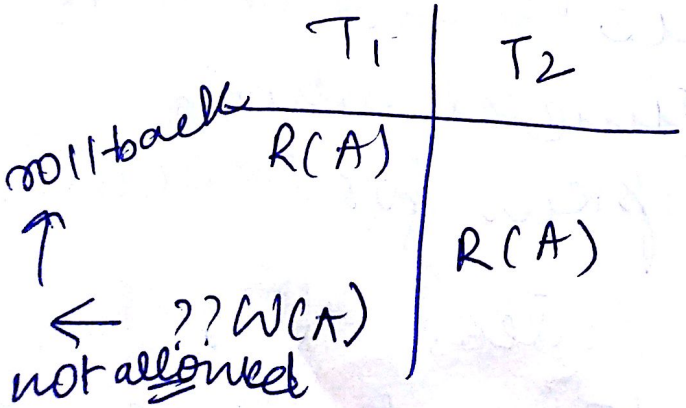
→ Time Stamp ordering Protocol ensures that any conflicting read and write operations are executed in time stamp order, if not such an operation is rejected and the transaction will be rolled back, the rolled back transaction will be restarted with a new time-stamp.



$TS(T_1) = 1$
$TS(T_2) = 2$

$$TS(T_1) < TS(T_2)$$

$$T_1 \rightarrow T_2.$$



$$TS(T_1) < TS(T_2)$$

$$T_1 \rightarrow T_2.$$

Scanned by CamScanner

Thomas Write Rule:-

Ignoring outdated writes is called the Thomas write rule.

T_1	T_2
✓ R(A)	
	W(A) ← check ??
W(A)	

$TS(T_1) = 1$ $TS(T_2) = 2$.

allow schedules which are conflict equivalent to T_1 followed by T_2

T_1	T_2
R(A)	
W(A)	
	W(A)

T_1	T_2
r(Q)	
	W(Q) com
W(Q) com ↙ ignore	

every conflicting operation should happen in the order of time stamp.

T_1 → W(A) will be rejected & will be rolled back.

Thomas write rules is bit flexible it doesn't allow unnecessary roll back.

- allows more no of schedules.

Scanned by CamScanner

$Ts(T_1) < Ts(T_2)$

	Not allowed	allowed
TOP	$R_2(A)$ $W_1(A)$ $W_2(A)$ $R_1(A)$ $W_2(A)$ $W_1(A)$	$R_1(A)$ $W_2(A)$ $R_2(A)$ $R_1(A)$ $W_1(A)$ $R_2(A)$. $W_1(A)$ $W_2(A)$.
TWR	$R_2(A)$ $W_1(A)$ $W_2(A)$ $R_1(A)$	$R_2(A)$ $R_1(A)$ $W_2(A)$ $W_1(A)$.

T_1	T_2
	$W(A)$
$W(A)$	

T_1	T_2
$W(A)$	
	$W(A)$

Scanned by CamScanner

Examples on Time-stamp ordering Protocol

$W_1(x)$		
	$W_2(x)$	
		$W_3(x)$
	$R_2(x)$	
		$W_3(x)$

not allowed under time stamp orthomas write rule

1) if T wants ~~to~~ write ~~on~~ on item (x) th
younger transaction has ab

7 7 10 7 7

3 3 3 3

Scanned by CamScanner