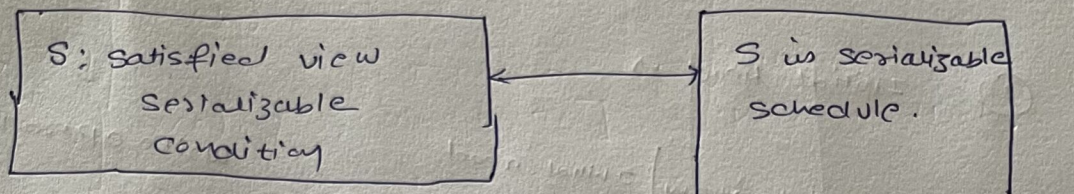
1

View serializability schedule testing conditions are sufficient and necessary conditions for serializable testing.

(both sufficient and necessary

(or)

if and only if



[T] ⟷ [T]

[F] ⟷ [F].

View serializable schedule:- Schedule S is view serializable schedule, if and only if there exist some serial schedule [S'] which is view equal to given schedule (S).

Example: Test - view serializable (or) not.

S:

T_1	T_2	T_3
	$R_2(A)$	
	$R_2(B)$	
$w_1(A)$		
		$R_3(A)$
$w_1(B)$		
	$w_2(B)$	
		$w_3(B)$

T_1	T_2	T_3
	$R_2(A)$	
	$R_2(B)$	
	$w_2(B)$	
$w_1(A)$		
$w_1(B)$		
		$R_3(A)$
		$w_3(B)$.

$T_2 \rightarrow T_1 \rightarrow T_3$

(1) Final write :-

Given schedule

Equal serial schedule

A: T_1

B: $T_1, T_2, (T_3)$

final write

$(T_1, T_2) \rightarrow T_3$

or $T_1 \cdot T_2\ T_3$ & $T_2\ T_1\ T_3$

(2) Initial Read =>

data item	Initial read
A	T_2
B	T_2

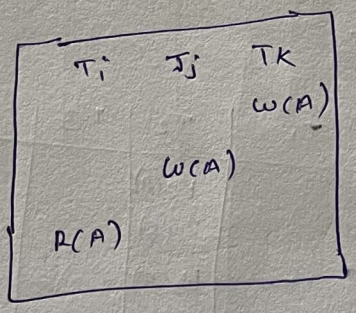
write operation

$T_1 \Leftrightarrow T_2 \rightarrow T_1$ in serial order

$(T_1, T_2\ T_3) \Leftrightarrow T_2 \rightarrow (T_1, T_3)$ in serial.

[updated Read]

(3) write - Read



$w_j(A) \rightarrow R_i(A)$ T_K also writes A

In equal serial order =>

$T_j \rightarrow T_i$ and

T_K should not b/w T_j & T_i

$T_j \rightarrow T_i$

T_K

Given Schedule: $T_1 : w(A) \rightarrow T_3 : R(A)$

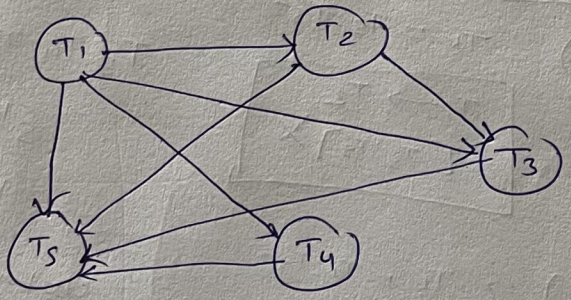
$T_K \neq \emptyset$ => $T_1 \rightarrow T_3$

=> S is view serializable schedule. equal serial schedule. $T_2 \rightarrow T_1 \rightarrow T_3$

T_1 S_B :

T_1	T_2	T_3	T_4	T_5
$r_1(A)$				
		$r_3(D)$		
$w_1(B)$				
	$r_2(B)$			
		$w_3(B)$		
			$r_4(B)$	
	$w_2(C)$			
				$r_5(C)$
			$w_4(E)$	
				$r_5(E)$
				$w_5(B)$

(a) Find # of conflict equal serial order :-



S' : T_1 ; T_2 ; T_3 ; T_4 ; T_5 Conflict equal to S.

(b) Find view equal serial orders :-

(1) Final write :-

A : —

B : $T_1 : T_3 : T_5$ ⇒ $T_1, T_3 \rightarrow T_5$

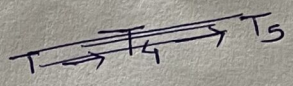
C : T_2

D : —

E : (T_4)

(2) Initial Read

DI	IR	writes
A	T_1	—
B	—	T_1 T_3 T_5
C	—	T_2
D	T_3	—
E	—	T_4



(3) Write – Read sequence :-

* $w_1(B) \rightarrow R_2(B)$ $T_k = T_3\ T_5$

$T_1 \rightarrow T_2$

$[T_3 : T_5]$ ✗

$T_3\ T_4\ T_1\ T_2\ T_5$ | $T_1\ T_2\ T_3\ T_4\ T_5$

S' | S'

both ~~are~~ serial are view ~~equi~~ equal with S_3

* $w_3(B) - R_4(B)$ ⇒ $T_k = T_1\ T_5$

= $T_3 \rightarrow T_4$

$[T_1, T_5]$

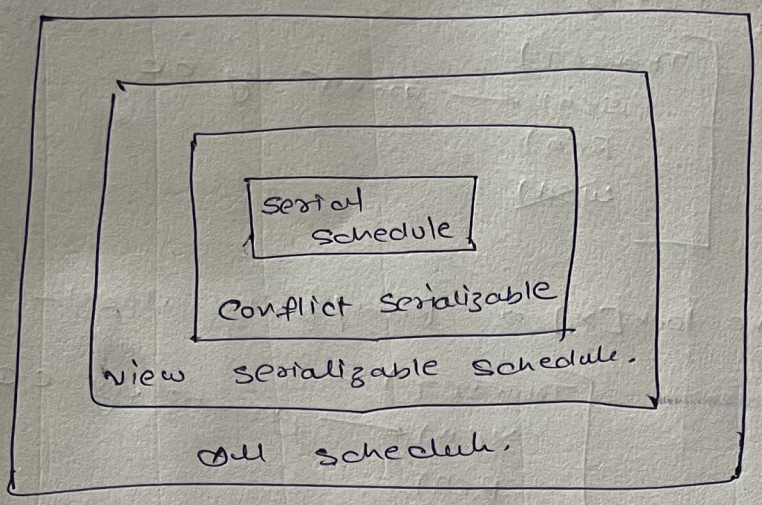
✗

* $w_2(C) \rightarrow R_5(C)$; $T_k : \emptyset$

= $T_2 \rightarrow T_5$

* $w_4(E) \rightarrow R_5(E)$, $T_k : \emptyset$

$T_4 \rightarrow T_5$



$$\{\text{Non serializable schedule}\} = \{\text{All schedule}\} - \{\text{view serializable}\}$$

Concurrency + Control + protocol :- using concurrency control protocol Non-serializable schedule (or) Not strict recoverable schedule should not be allowed to execute [discarded].

⇒ Schedule (S) free from all problem only if S is Serializable schedule (and) S is strict Recoverable schedule.

* Locking + protocol :- Lock to the data item.

Trans (T_1)
$lock_1(A)$ ← grants by c.c.
$R(A)$
$W(A)$
$unlock_1(A)$
$lock_2(B)$ ← grants
$R(A)$
$W(B)$
$unlock_2(B)$

* Shared- Exclusive lock :-

Shared lock [S] => Read only lock

Trans (T)
S(A) ← grants by cc
R(A)
~~W(A)~~

Exclusive lock [X] ⇒ Read / write lock) -

Trans (T)
X(A) ← grants by cc
R(A)
W(A)

* lock compatability =>

	S	X
S	Yes	NO
X	NO	NO

← Hold by some trans.

↑ Requesting by other transaction

~~Non ser~~

→ Non serializable schedule

T_1	T_2
S(A)	
$R_1(A)$	
U(A)	X(A)
	$W_2(A)$
	U(A)
	X(B)
	$W_2(B)$
	U(B)
S(B)	
$R_1(B)$	
U(B)	

=> Shared exclusive locking and compatability table not sufficient for serializability and strict serializable schedule.

* Two phase locking protocol :- [using 2PL can ensure serializability]

Trans(T)

locking phase [growing phase]
- S(A)
- X(B)
- X(C)
- X(D)
- S(E)

First unlock of Tran (T)

unlocking phase [shrinking phase]
- U(B)
- U(E)
- U(A)
- U(C)
- U(D)

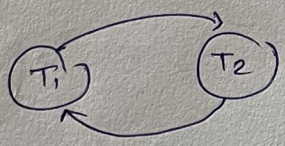
* According to 2PL Transaction T Not allowed to request lock of any data item if T performs any unlock operation.

* Lock point of Transaction (T), Means position of last lock or first unlock of Transaction(T).

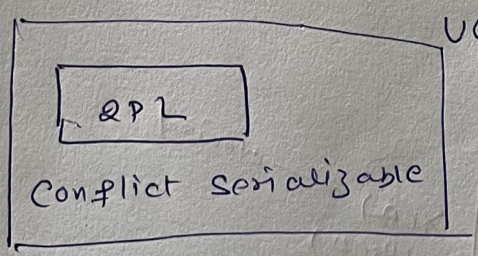
⇒ If schedule (S) is Not conflict serializable schedule then (S) Not allowed to execute by 2PL.

⇒ If schedule allowed to execute by 2PL then Schedule is conflict serializable schedule.

⇒ Every 2PL schedule is conflict serializable schedule, but every conflict serializable schedule may not be an 2PL.

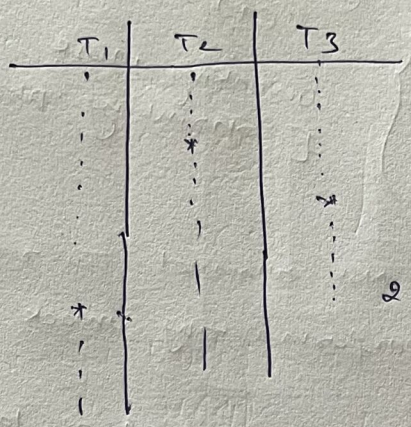


T_1	T_2
S(A)	
$R_1(A)$	
S(B)	X(A)
U(A)	
	~~X(A)~~
	$w_2(A)$
	X(B) ← denied
	$w_2(B)$
$R_1(B)$	U(A) →
U(B)	



⇒ If schedule is allowed by 2PL then schedule is Conflict serializable schedule, and equal serial schedule is based on lock points orders.

Ex:

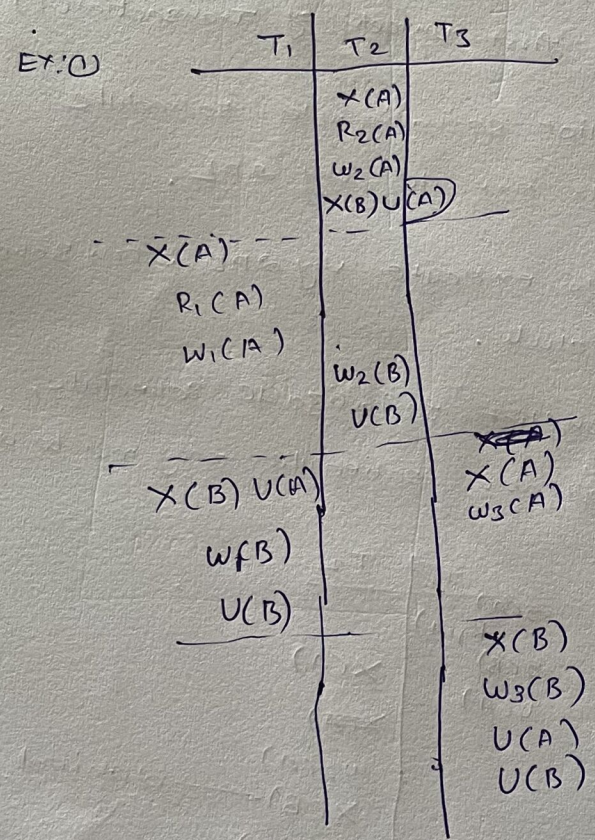


Equal serial schedule

$T_2 \rightarrow T_3 \rightarrow T_1$

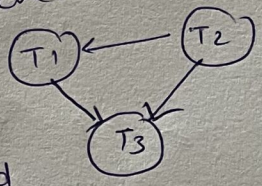
[Lock points order].

Ex: (1)



① 2PL then

② Conflict serializable schedule



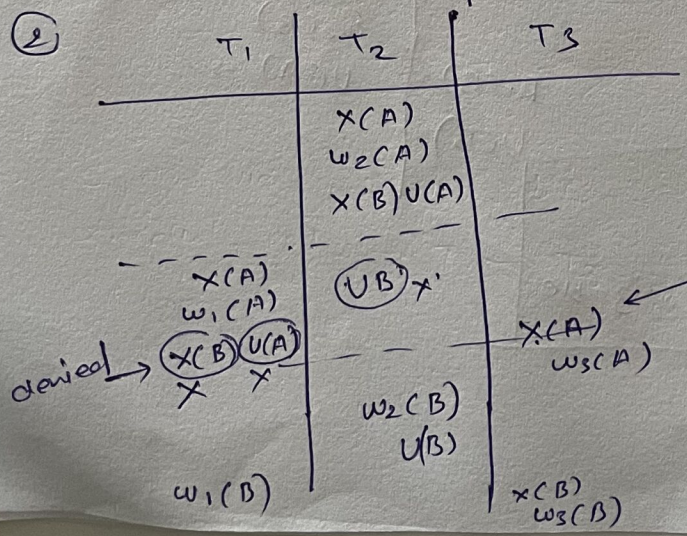
and

③ Equal serial schedule.

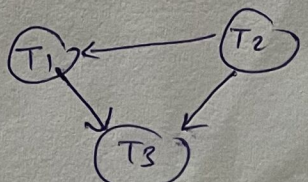
$T_2 \rightarrow T_1 \rightarrow T_3$

(last lock point precedence) ;

(2)



Not in 2PL ✗



Conflict S.S. ✓

* Limitations of 2PL protocol ⇒ (1) 2PL constraint may leads to deadlock.

[S(A) S(B) U(A) U(B)] T_1 : $R_1(A)$ $R_1(B)$

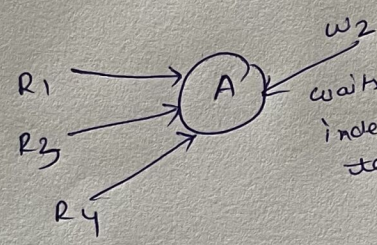
T_2 : $W_2(B)$ $W_2(A)$

~~[X(B) X(A) U(B) U(A)]~~

[X(B) X(A) U(B) U(A)]

T_1	T_2
S(A)	
$R_1(A)$	
	X(B)
	$W_2(B)$
denied → S(B)	
[T_1 waiting T_2]	X(A) [T_2 waiting T_1] so here deadlock.

(2) 2PL/SX, locking conditions may leads to starvation.



T_2	T_1	T_3	T_4	T_5
	S(A)			
denied X(A) bcz of T_1	⋮			
	U(A)	S(A)		
bcz T_3 → X(A)		⋮		
		U(A)	S(A)	
bcz T_4 → X(A)			⋮	
⋮				

(3) 2PL condition may leads to irrecoverable schedule, cascading rollback, and lost update problem.

T_1	T_2
X(A)	
W(A)	
U(A)	
	S(A)
	R(A)
	U(A)
	C_2
C_1	

→ Allowed by 2PL

→ Irrecoverable schedule.

* strict 2PL protocol ⇒ Basic 2PL conditions and every exclusive lock of Transaction (T) must hold until C/R of Transaction T.

[strict recoverability]

Trans (T1)
- S(A)
- X(B)
- S(C)
- X(D)
- }
- U(A)
- U(C)
- commit
- U(B)
- U(D)