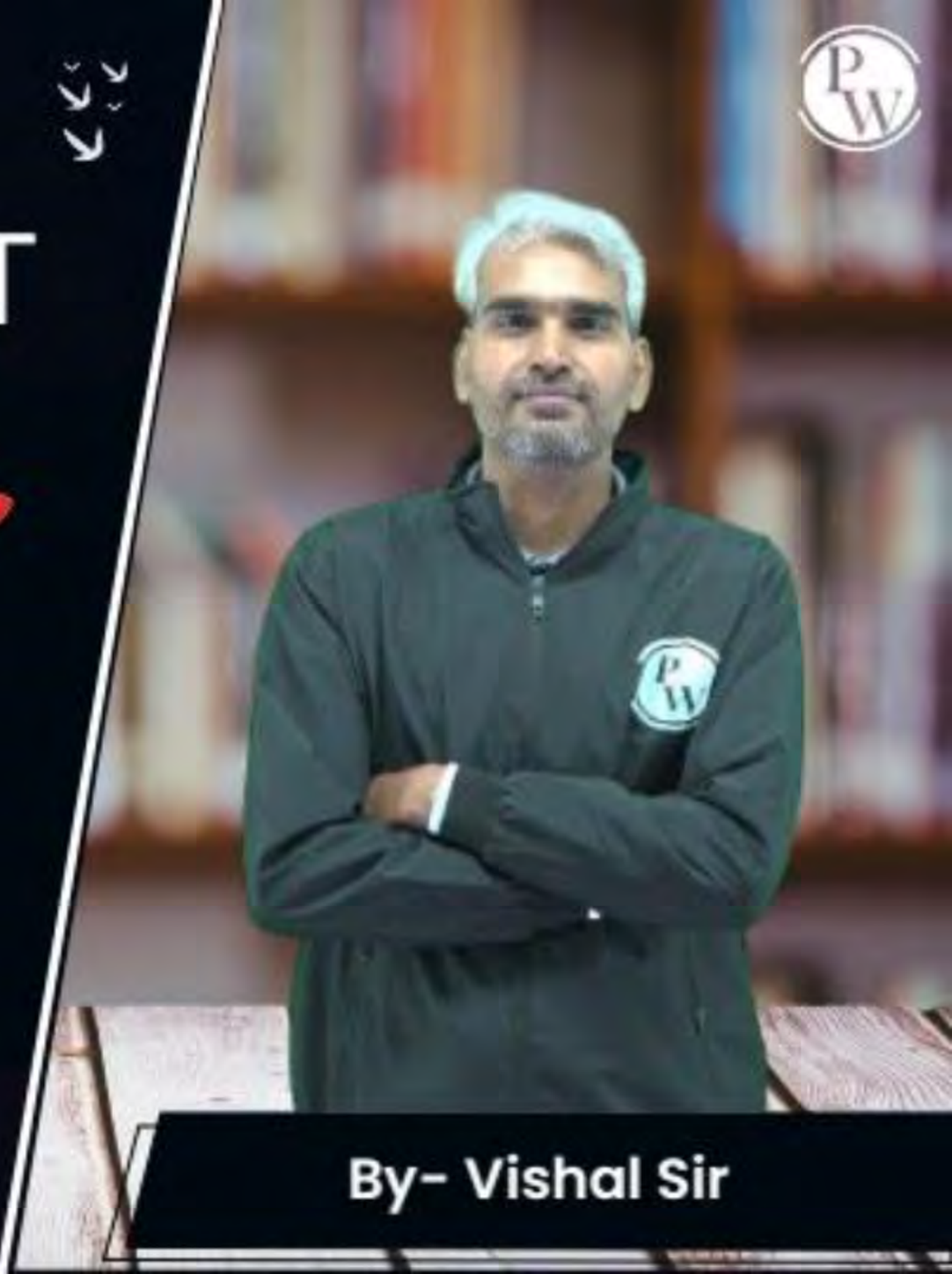
Computer Science & IT

Database Management System

Transaction & concurrency control

Lecture No. 08



By- Vishal Sir

Recap of Previous Lecture

- Topic: View serializable schedule
- Topic: Concurrency control component
- Topic: Simple use of shared and exclusive locks
- Topic: Two phase lock (2PL)

Slide

Topics to be Covered

- Topic — Basic 2PL
- Topic — Strict 2PL
- Topic — Conservative 2PL
- Topic — Rigorous 2PL

Slide

Concurrency Control Component

Topic : Concurrency Control Protocols

- Concurrency Control Protocols are responsible for avoiding the execution of non-serializable schedules.

- Concurrency Control Protocols are suggested to ensure serializability

  * If schedule is allowed to execute using a valid Concurrency Control Protocol, then schedule is a Serializable Schedule
    Converse of the statement need not be true

Topic : Concurrency Control Protocols

- There are two types of concurrency control protocols.

1. Lock based Protocols

   Two Phase Locking Protocol (2PL)

   Versions of 2PL
   - (i) Basic 2PL
   - (ii) Strict 2PL
   - (iii) Conservative 2PL
   - (iv) Rigorous 2PL

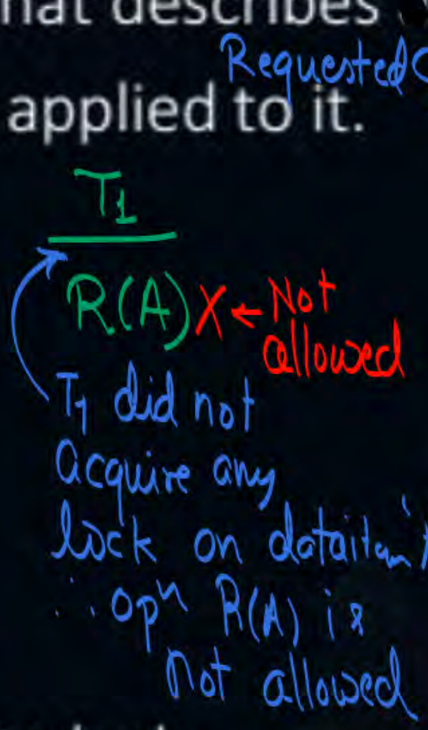
2. Time-stamp based Protocols
   - (i) Basic Time stamp ordering Protocol.
   - (ii) Thomas write Time-stamp ordering Protocol

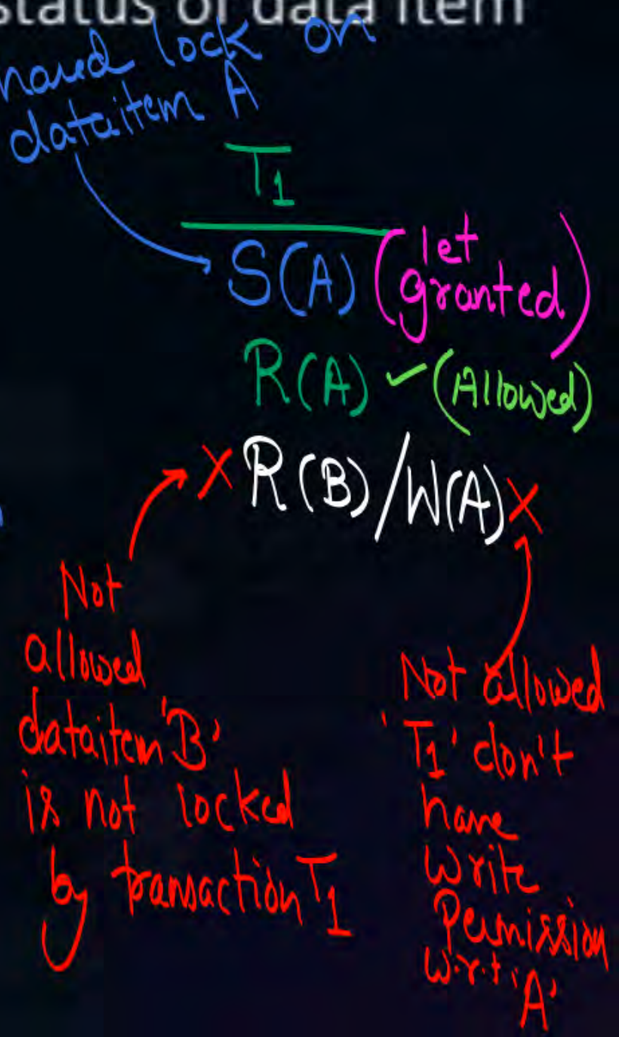
Topic : Lock Based Protocols

- Lock is a variable associated with a data item that describes status of data item with respect to possible operation that can be applied to it.

- There are two types of locks used,
  1. Shared LOCK [S]: It is read only lock.
  2. Exclusive LOCK [X]: It is read as well as write lock.

If transaction wants to perform only read opn on a dataitem, then transaction will request for a shared lock on that dataItem





Topic : Lock Based Protocols

Lock is a variable associated with a data item that describes a status of data item with respect to possible operation that can be applied to it.

- There are two types of locks used,
  1. Shared LOCK [S]: It is read only lock.
  2. Exclusive LOCK [X]: It is read as well as write lock.

T
W(A)
Not allowed

Requested for Exclusive lock on dataitem 'A'

T
X(A) (lst granted)
W(A) ✓(Allowed)
S(B) (granted)
R(B) ✓(Allowed)
R(A) ✓(Allowed)
R(C)/W(B)
Not allowed

{Transaction can request for an Exclusive lock on dataitem A only if transaction wants to perform at least one write oph on dataitem 'A'}

Topic : Lock Compatibility Table

Locks held by Transaction Ti

data item 'A'	Shared lock 'S'	Exclusive lock 'X'
Shared lock 'S'	(✓) Allowed	Not allowed
Exclusive lock 'X'	Not allowed	Not allowed

locks requested by transaction Tj

Note :- To ensure serializability a non-serializable schedule must not be allowed to execute.

Topic : Simple use of shared and exclusive locks

↳ i.e, No restriction { i.e, dataitems can be locked / Unlocked in any order at any time }

Consider the following Schedule.

'S'

T_1	T_2
$X(A)$ ✓	
$R_1(A)$ ✓	
$W_1(A)$ ✓	
Unlock 'A' ⟶ $U(A)$	
	$S(A)$ ✓
	$R_1(A)$ ✓
	$S(B)$ ✓
	$R_2(B)$ ✓
	$U(A)$
$X(B)$ ✓	$U(B)$
$R_1(B)$ ✓	
$W_1(B)$ ✓	
$U(B)$	

Given schedule is a non-serializable schedule

↓

But it is allowed to Execute using Simple use of Shared and Exclusive locks

A non-serializable schedule may be allowed to execute using Simple use of Shared & Exclusive locks

∴ Simple use of Shared & Exclusive lock does not Ensure Serializability.

Topic : Basic Two Phase Locking Protocol

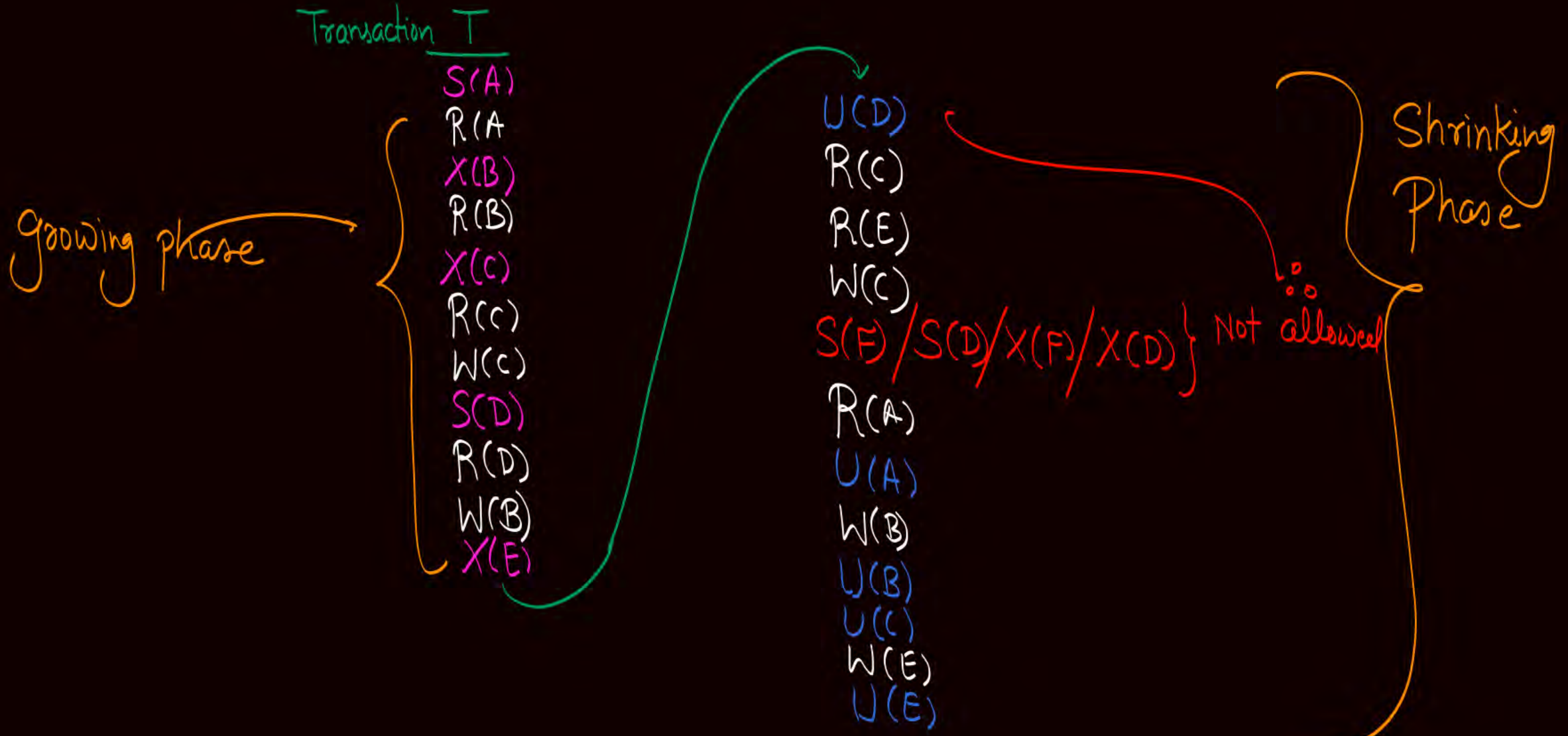
There are two phases in Two phase locking protocol

1. Growing phase / Lock acquiring phase
2. Shrinking phase / Lock releasing phase

In Two Phase Locking Protocol, transaction T is allowed to request for a lock only if transaction T has not performed any unlock operation.

on any dataitem

→ The moment transaction perform any unlock operation it can request for a lock on any dataitem


Transaction T
S(A)
R(A
X(B)
R(B)
X(C)
R(C)
W(C)
S(D)
R(D)
W(B)
X(E)
Growing phase
U(D)
R(C)
R(E)
W(C)
S(F)/S(D)/X(F)/X(D)} Not allowed
R(A)
U(A)
W(B)
U(B)
U(C)
W(E)
U(E)
Shrinking Phase

H.W.Q: Check whether the schedule is allowed to execute using 2PL or not?

S1

T_1	T_2
$R_1(A)$	
$W_1(A)$	
	$R_2(A)$
	$R_2(B)$
$R_1(B)$	
$W_1(B)$	

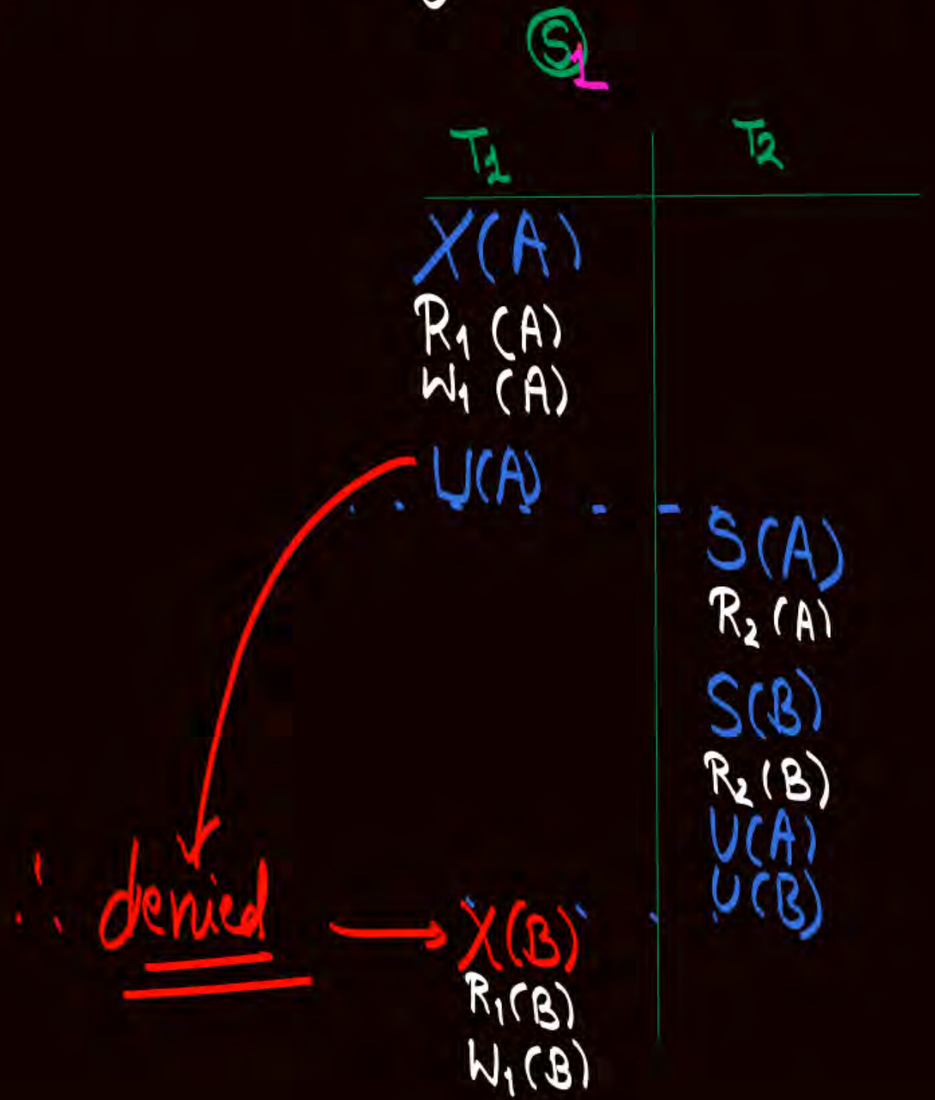
H.W.Q: Check whether the schedule is allowed to execute using 2PL or not?

S1

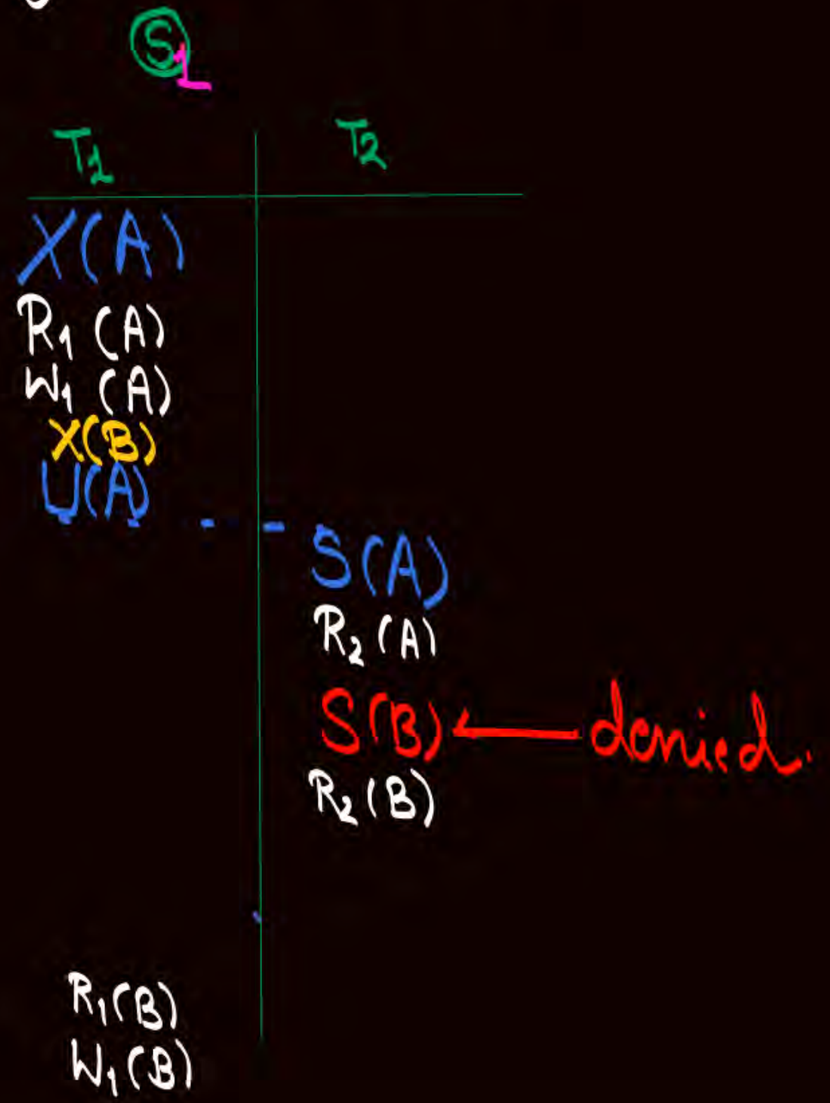
T_1	T_2
X(A)	
$R_1(A)$	
$W_1(A)$	
⋮	
	S(A) ← denied
	$R_2(A)$
	$R_2(B)$
$R_1(B)$	
$W_1(B)$	

∴ Transaction T_2 will will for a time out period

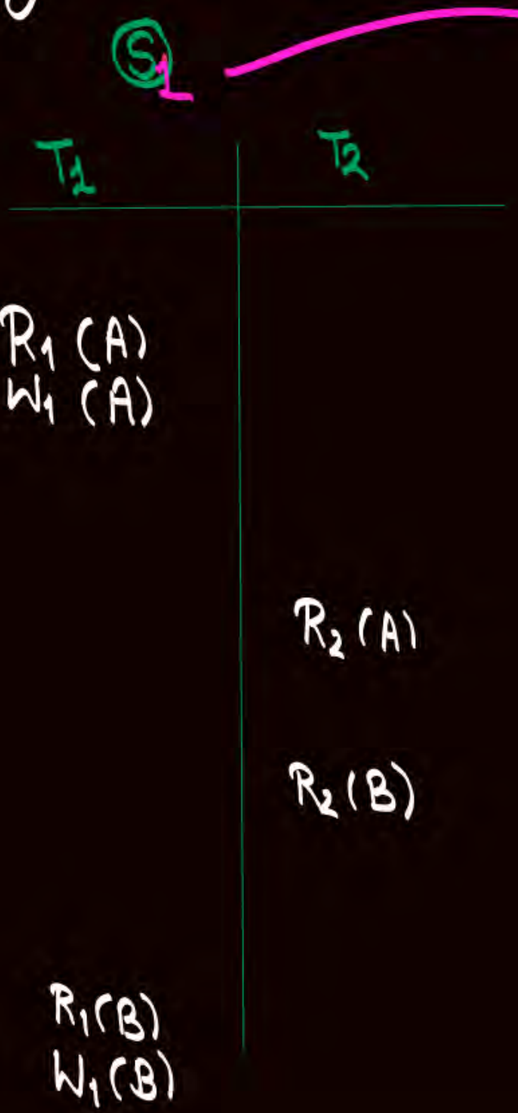
H.W Q: Check whether the schedule is allowed to execute using 2PL or not?



H.W.Q: Check whether the schedule is allowed to execute using 2PL or not?

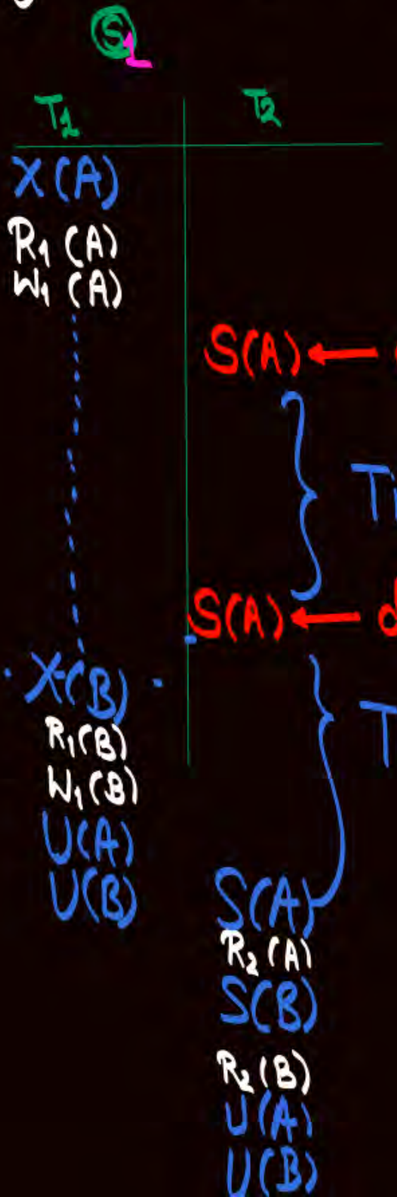


H.W.Q: Check whether the schedule is allowed to execute using 2PL or not?



It is a non-serializable schedule
&
Not allowed by
Basic 2PL

H.W.Q. Check whether the schedule is allowed to execute using 2PL or not ?



S(A) ← denied

Time out

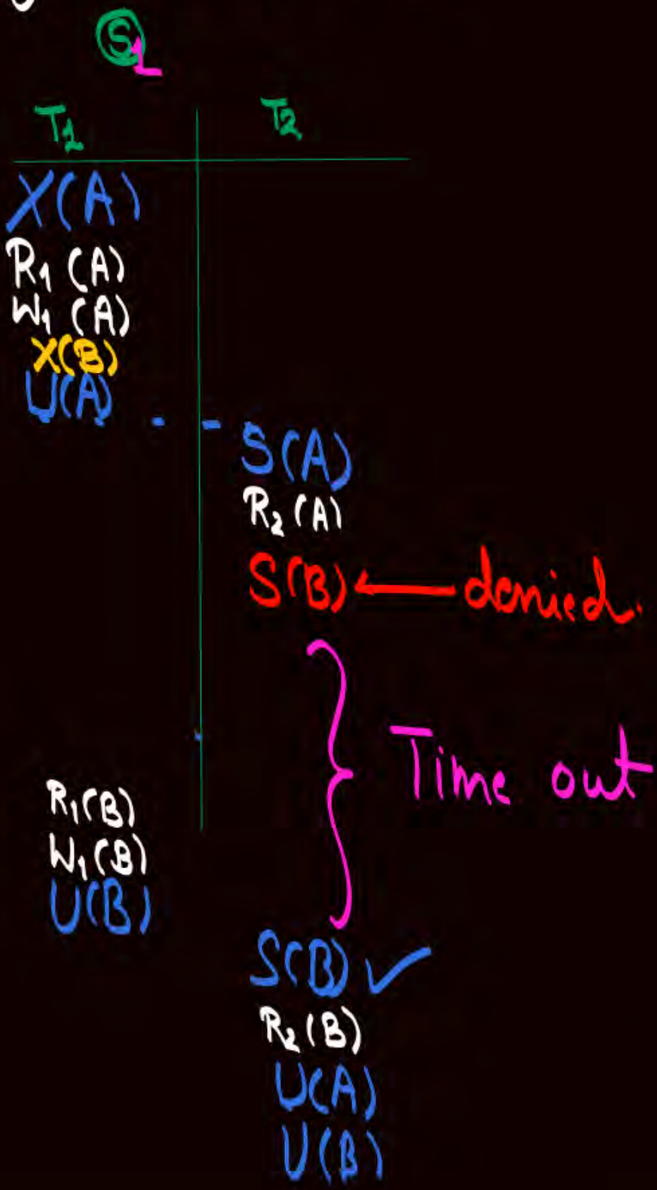
S(A) ← denied

Time out

∴ Transaction T_2 will will for a time out period

Using Basic 2PL this may be the order of Execution but it is not same as the given schedule

H.W.Q. Check whether the schedule is allowed to execute using 2PL or not?



This is also a possible sequence using basic 2PL but again schedule is not same as given schedule

Q:- Check whether the following Schedule is allowed using Basic 2PL or not?

(S)

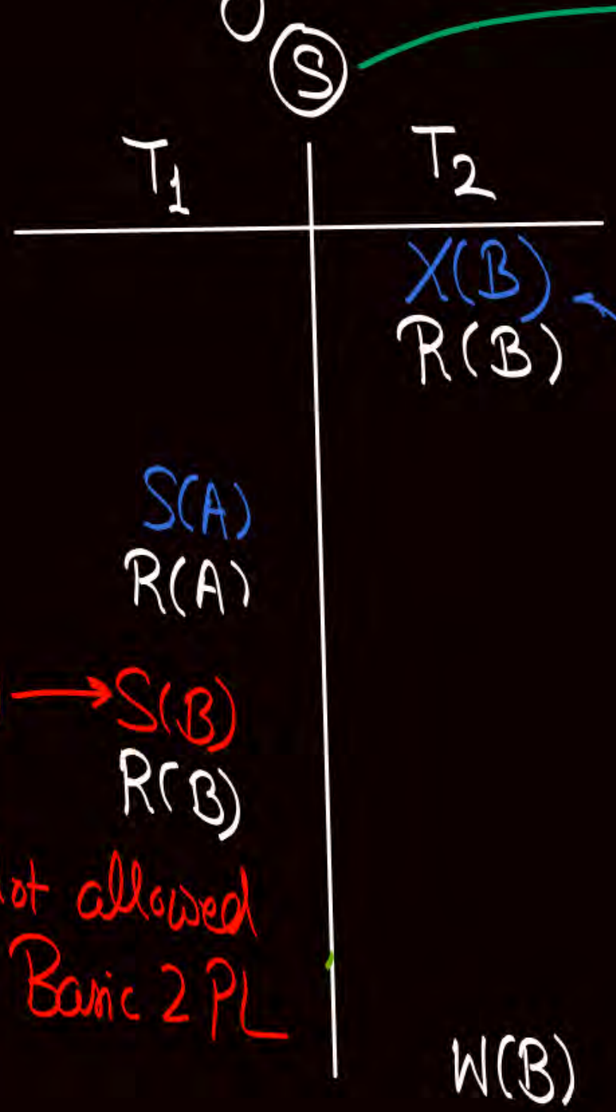
T_1	T_2
X(A)	
$R_1(A)$	
$W_1(A)$	
X(B)	
U(A)	
	S(A)
	$R_2(A)$
$R_1(B)$	
$W_1(B)$	
U(B)	
	S(B)
	$R_2(B)$
	U(A)
	U(B)

Acyclic ∴ C.S.S.

↓

and it is allowed to execute using Basic 2PL

Q: Check whether the schedule is allowed to execute using Basic 2PL or not?



Until T_2 Execute all opn on dataitem 'B' it will not unlock 'B'

Precedence graph

Acyclic ⇒ ∴ C.S.S.

But it is not allowed to execute using Basic 2PL

Q:. Check whether the schedule is allowed to execute using Basic 2PL or not ?

(S)

It is not a C.S.S., but it is a Serializable schedule,
& Equivalent serial schedule is $T_1 \rightarrow T_2$

X(B) ← denied → ∴ Not allowed by Basic 2PL

Note:- ① If schedule is allowed to execute using Basic 2PL then schedule is guaranteed to be a Conflict serializable schedule (Converse of the statement need not be true)

{ i.e. Basic 2PL ensure Conflict serializability }

i.e. if schedule is not a C.S.S., then it will never be allowed to execute using Basic 2PL

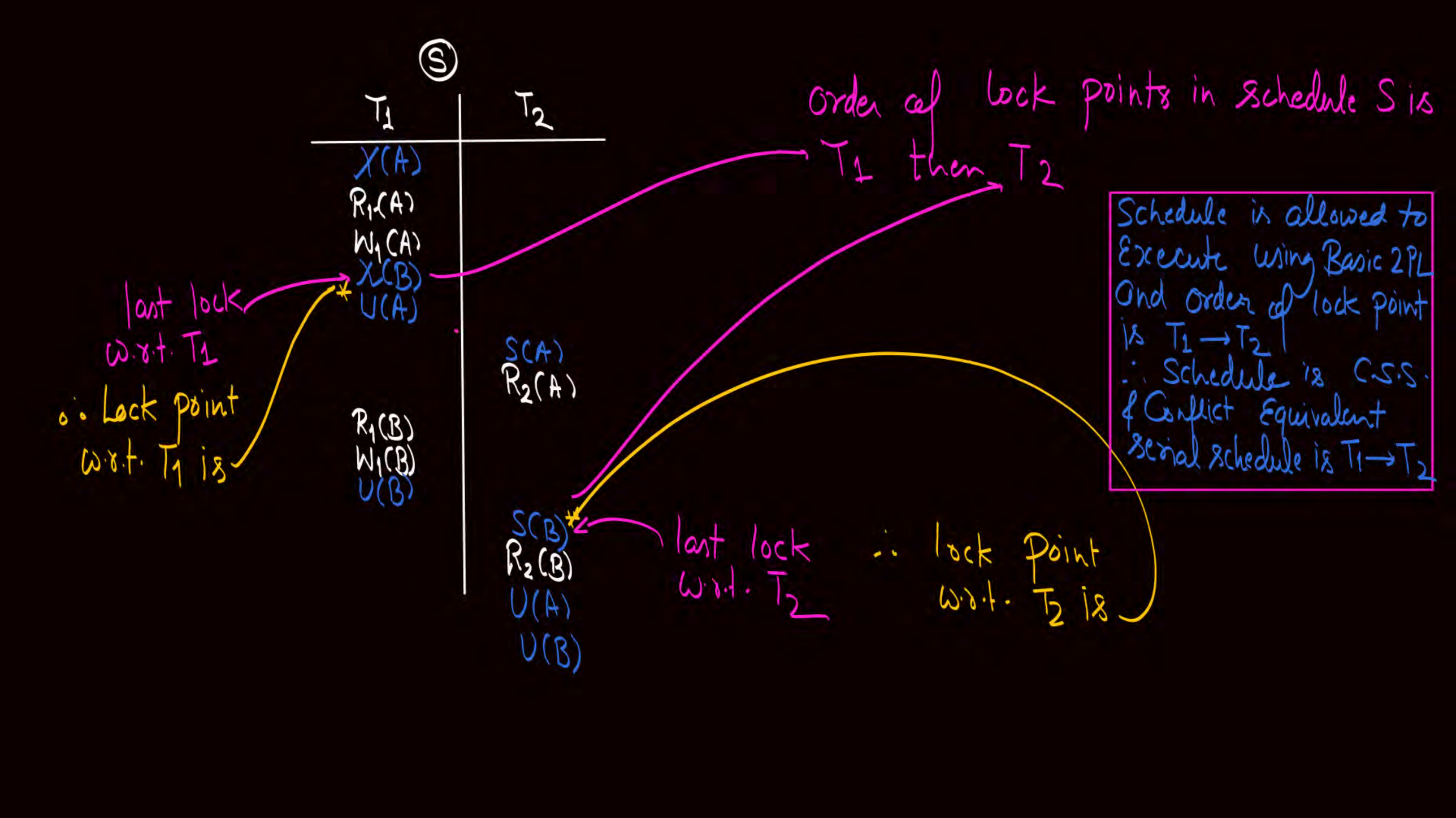
Topic : Basic Two Phase Locking Protocol

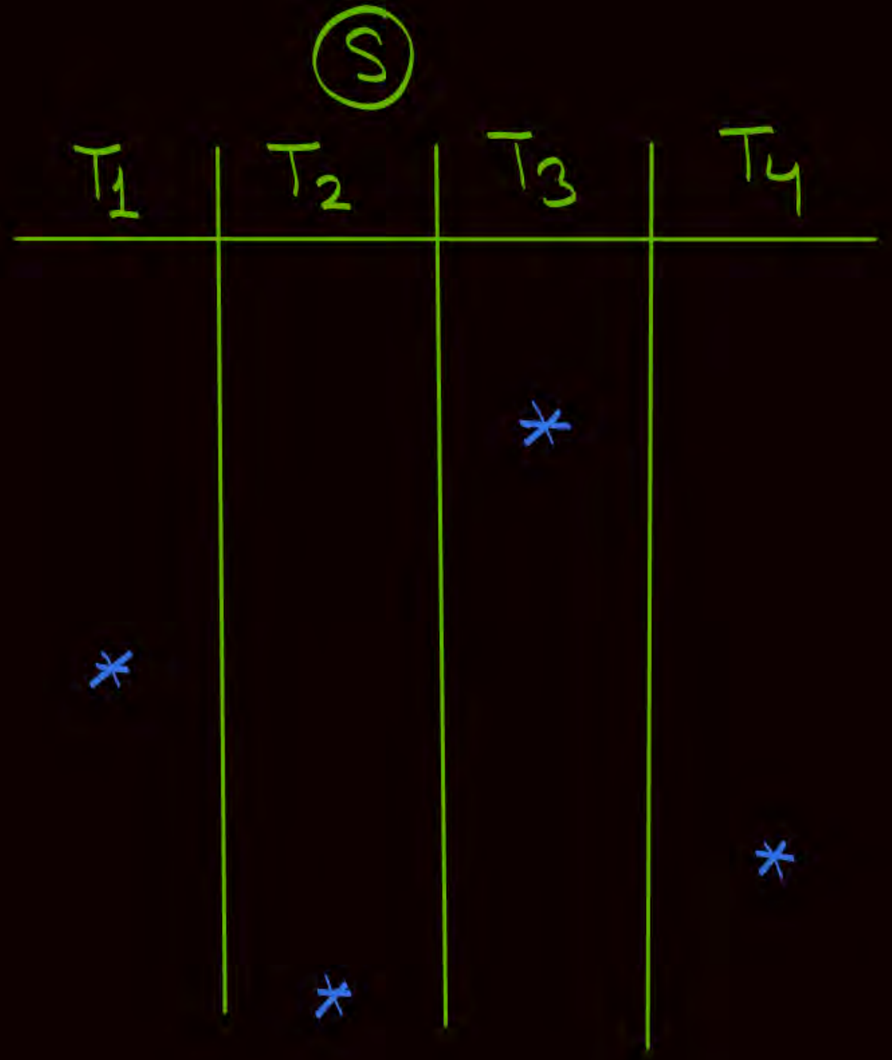
→ More Precisely Conflict serializability

1. 2PL ensures serializability, i.e., a non-serializable schedule is never allowed to execute using 2PL.
2. If schedule is allowed to execute using 2PL, then schedule is conflict serializable schedule, but every conflict serializable schedule need not be allowed to execute by 2PL.
3. If schedule is not a conflict serializable schedule, the that schedule is not allowed to execute using 2PL.
4. If schedule is allowed to execute using 2PL, then schedule is conflict serializable schedule and conflict equivalent serial schedule can be given by order of lock points.

Topic : Lock Point

The Point at which the growing phase ends, i.e., the point at which transaction takes the final lock it needs to carry on its work. is defined as lock point of that transaction


S
T1
T2
X(A)
R1(A)
W1(A)
XL(B)
U(A)
S(A)
R2(A)
R1(B)
W1(B)
U(B)
S(B)
R2(B)
U(A)
U(B)
last lock w.r.t. T1
∴ Lock point w.r.t. T1 is
Order of lock points in schedule S is T1 then T2
last lock w.r.t. T2
∴ lock Point w.r.t. T2 is
Schedule is allowed to Execute using Basic 2PL And order of lock point is T1 → T2
∴ Schedule is C.S.S. & Conflict Equivalent serial schedule is T1 → T2



Let Schedule S is allowed by basic 2PL, and '*' represent the lock point in the corresponding transaction, then

Schedule will be a C.S.S. and Conflict Equivalent Serial Schedule will be $T_3 \rightarrow T_1 \rightarrow T_4 \rightarrow T_2$

Topic : Basic 2PL with lock upgrading

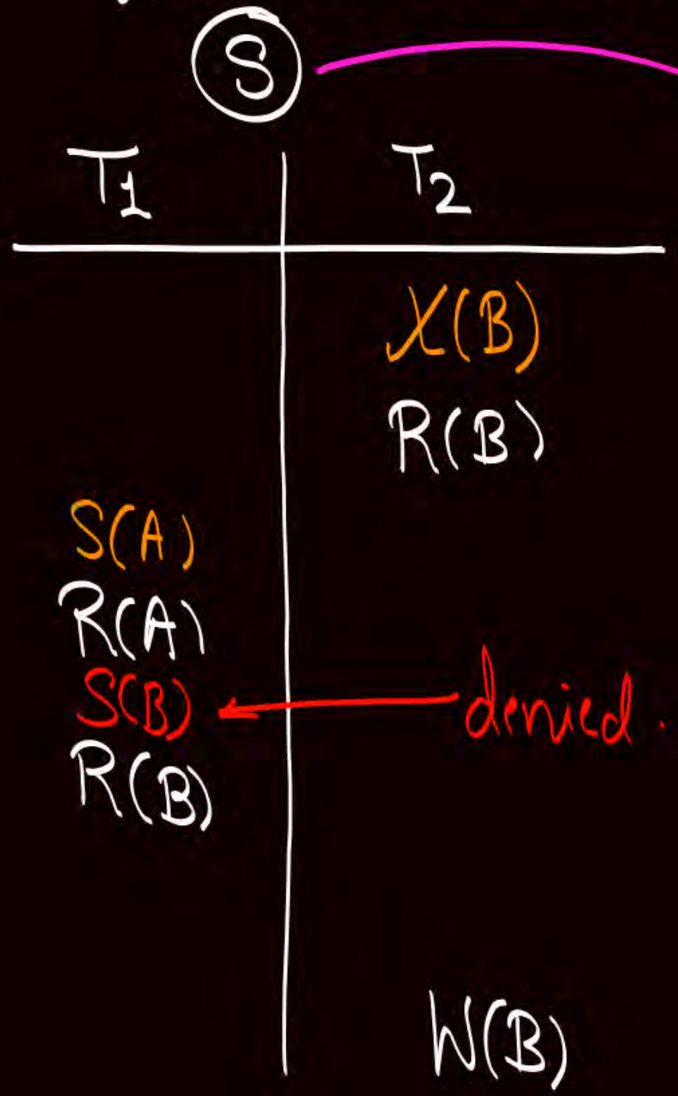
If lock conversion (upgrading/downgrading) is allowed, then we can upgrade shared lock into exclusive locks in the Growing Phase, and we can downgrade from the exclusive lock to shared lock in the shrinking phase.

Upgrade ⇒ Shared to Exclusive

Downgrade ⇒ Exclusive to Shared

- If transaction is already in its Shrinking Phase then it is not allowed to upgrade from Shared to Exclusive.
- If transaction is not already in its Shrinking Phase then also it can perform downgrade opn, but the moment it performs any downgrade opn its Shrinking Phase will start.

Q:- Check whether the following schedule is allowed to execute using Basic 2PL or not ?



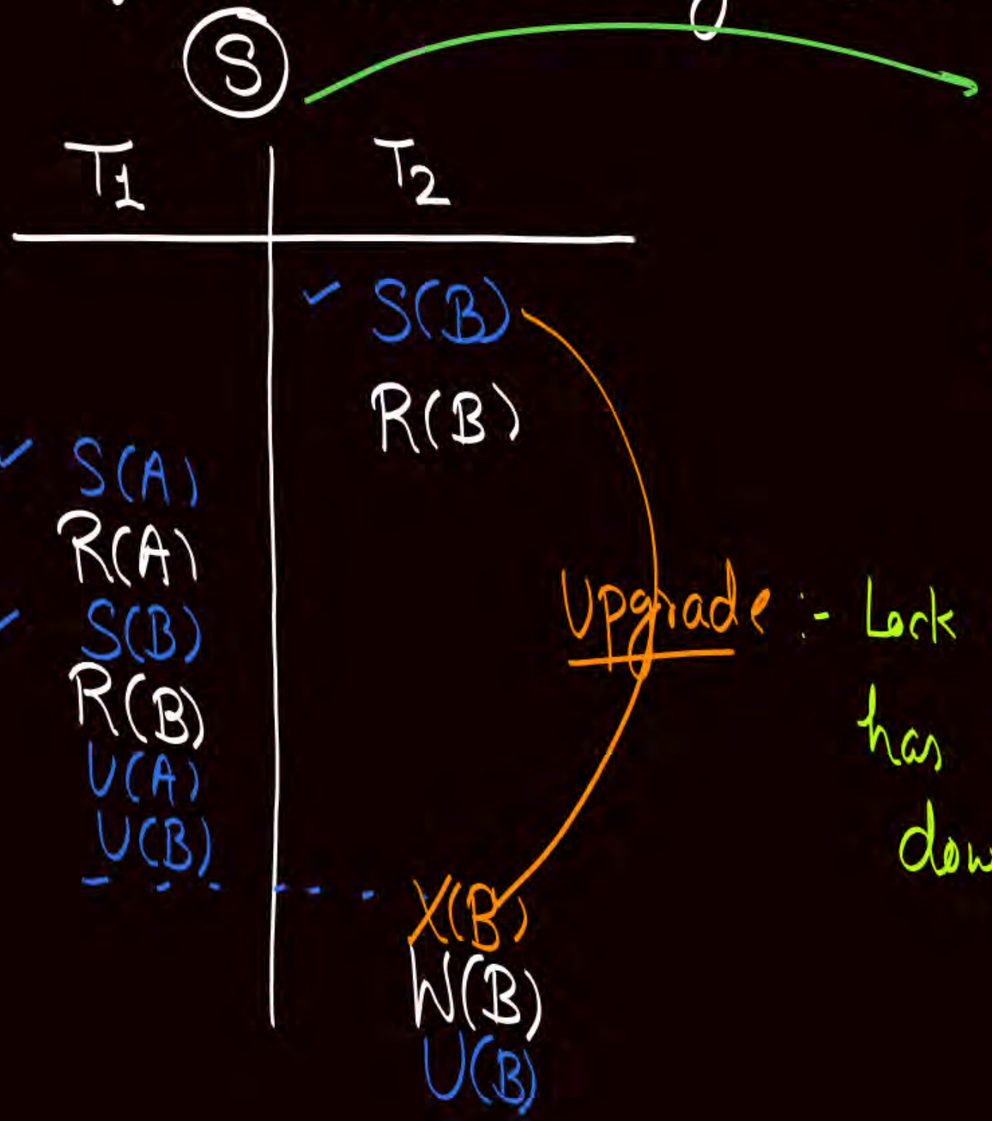
denied

It is a Conflict Serializable Schedule

↓

& Not allowed to execute using basic 2PL without lockupgrading

Q:- Check whether the following schedule is allowed to execute using Basic 2PL with lock upgrading or not?



It is a C.S.S.

↓

Not allowed by Basic 2PL without lock upgrade

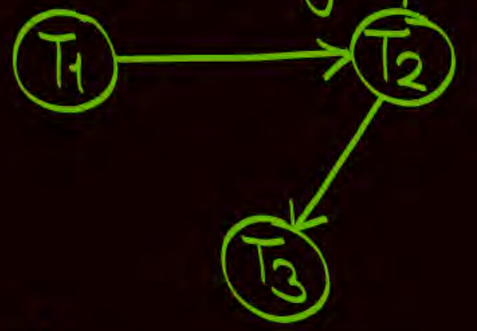
↓

But allowed to execute using Basic 2PL with lock upgrading

Upgrade :- Lock upgrading is allowed because transaction T_2 has not yet performed any Unlock or downgrade operation

Q:- Check whether the following schedule is allowed to execute using Basic 2PL or not ?

(S) (without lock upgrade) → Precedence graph

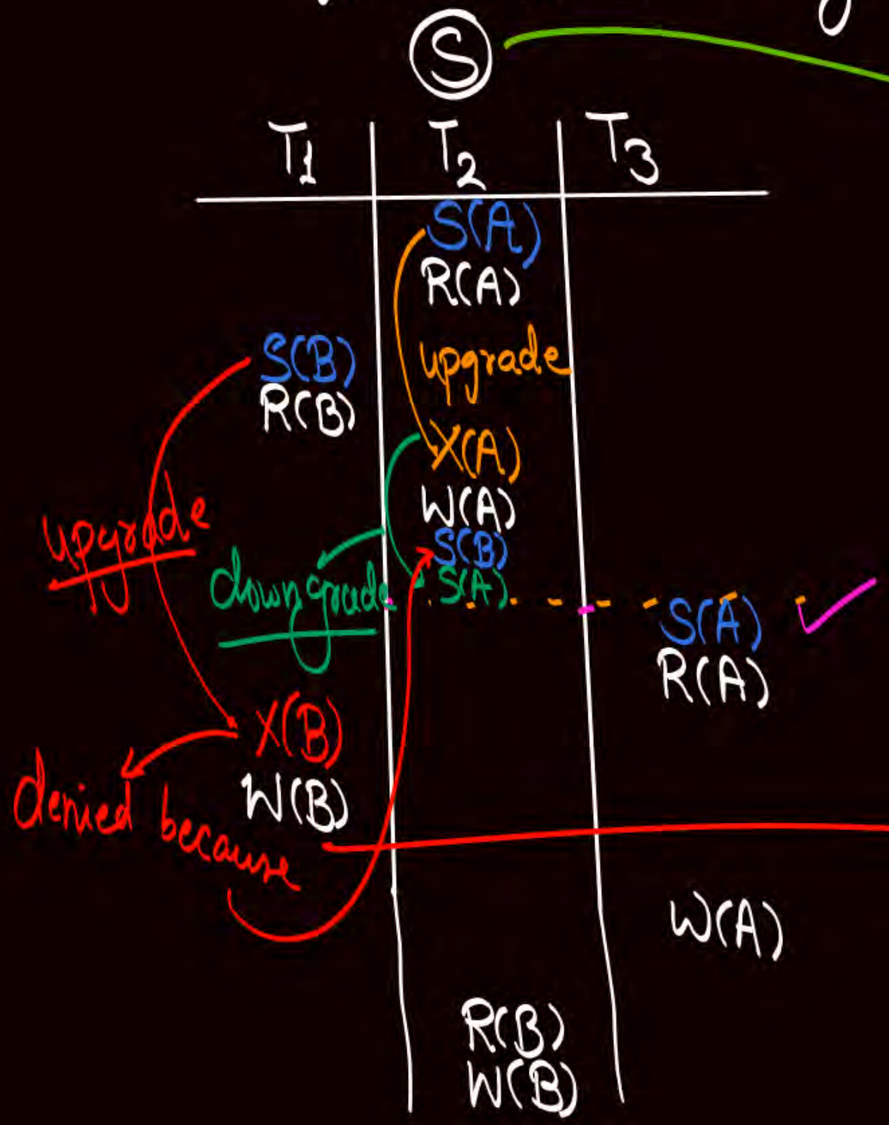


⇒ Acyclic
∴ C.S.S.

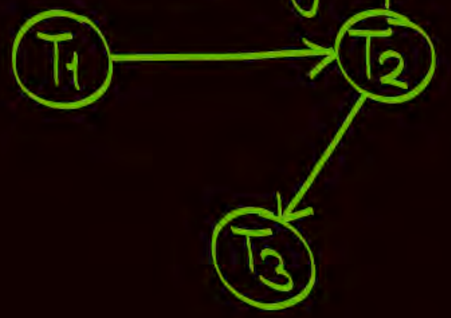
T_1	T_2	T_3
	X(A)	
	R(A)	
X(B)		
R(B)		
	W(A)	
	X(B) ← denied	
	U(A)	
		X(A)
		R(A)
W(B)		
		W(A)
	R(B)	
	W(B)	

∴ Not allowed using Basic 2PL without lock upgrading

Q:- Check whether the following schedule is allowed to execute using Basic 2PL with lock upgrading or not?



Precedence graph



⇒ Acyclic

∴ C.S.S.

⇓

But not allowed to Execute using Basic 2PL with lock upgrading as well

∴ Not allowed by Basic 2PL with lock upgrading as well.

Note:- ① If lock upgrading is allowed, then we may execute some extra conflict serializable schedules Compared to Basic 2PL without lock upgrading, but still there remains some conflict serializable schedules that are not allowed by Basic 2PL even when lock upgrading is allowed

② If schedule is not a conflict serializable schedule then it is never allowed by any type of 2PL, does not matter whether lock upgrading is allowed or not.

Topic : Problems possible with Basic 2PL

A schedule that is allowed to execute using basic 2PL protocol may suffer from,

1. Irrecoverability { Can be solved by Strict-2PL }
2. Deadlock { Can be solved by Conservative-2PL }
3. Starvation { No solution }

Topic : Irrecoverability with Basic 2PL

S → It is a Conflict Serializable schedule which is allowed to execute using Basic 2PL But it suffer from irrecoverability.

T_1	T_2
X(A)	
R(A)	
W(A)	
X(B)	
S(C)	
U(A)	
	S(A)
	R(A)
R(B)	
W(B)	
U(B)	
	S(B)
	R(B)
	U(A)
	U(B)
	Commit
R(C)	
U(C)	
Commit	

Precedence graph

Acyclic ∴ C.S.S.

Uncommitted Read

T_2 depends on T_1 & T_2 Commit before T_1 } ∴ Irrecoverable Schedule

Unsafe → { R(C), U(C), Commit

Topic : Deadlock with Basic 2PL

S

T_1	T_2
S(A)	
R(A)	
	X(B)
	W(B)
S(B) ← denied	
R(B)	
	X(A) ← denied
	W(A)

T_1 depends on T_2 for resource

T_2 depends on T_1 for resource

Time out

Time out

Dependency graph w.r.t resource

Cyclic dependency graph

∴ Deadlock

Topic : Starvation with Basic 2PL

T_1	T_2	T_3	T_4	T_5
	S(A)			
denied because of T_2 → X(A)				
Time Out		S(A)		
	U(A)			
denied because of T_3 → X(A)				
Time Out		U(A)		
denied because of T_4 → X(A)			X(A)	
Time Out			U(A)	
denied because of T_5 → X(A)				S(A)

If this keeps happening then we say that transaction T_1 is under Starvation

Topic : 2PL Classification

There are different versions of 2PL

1. Basic 2PL (Already done)
2. Strict - 2PL
3. Conservative - 2PL
4. Rigorous - 2PL

Topic : Strict 2PL

Basic-2PL restriction

↓

A transaction T can request for a lock on any data item only if it has not performed any Unlock operation

It will Ensure Serializability

+

Strict - Recoverability Condⁿ

↓

T_1	T_2
W(A)	
	.
	.
	.
Commit/Rollback	
	R(A)/W(A)

It Will Ensure Strict Recoverability

=

Strict - 2PL

T_1	T_2
X(A)	
	.
	.
	.
Commit/Rollback U(A)	
	S(A)/X(A)

It will Ensure Serializability as well as Strict recoverability

Topic : Strict 2PL

- Strict-2PL is a 2PL protocol with the restriction that Every Exclusive lock acquired by any transaction can be unlocked only after the Commit operation of that transaction

Shared locks can be unlocked at any time as per the restriction of 2PL

Strict - 2PL

T_1	T_2
X(A)	
Commit/Rollback U(A)	
	S(A)/X(A)

It will Ensure Serializability as well as Strict recoverability

Topic : Strict 2PL

Strict-2PL is → Free from

1. Irrecoverability
2. Cascading rollback problem
3. Lost-update problem

→ Not free from

1. Deadlock
2. Starvation

2 mins Summary

Topic Basic 2PL

Topic Strict 2PL

Topic Conservative 2PL

Topic Rigorous 2PL

THANK - YOU