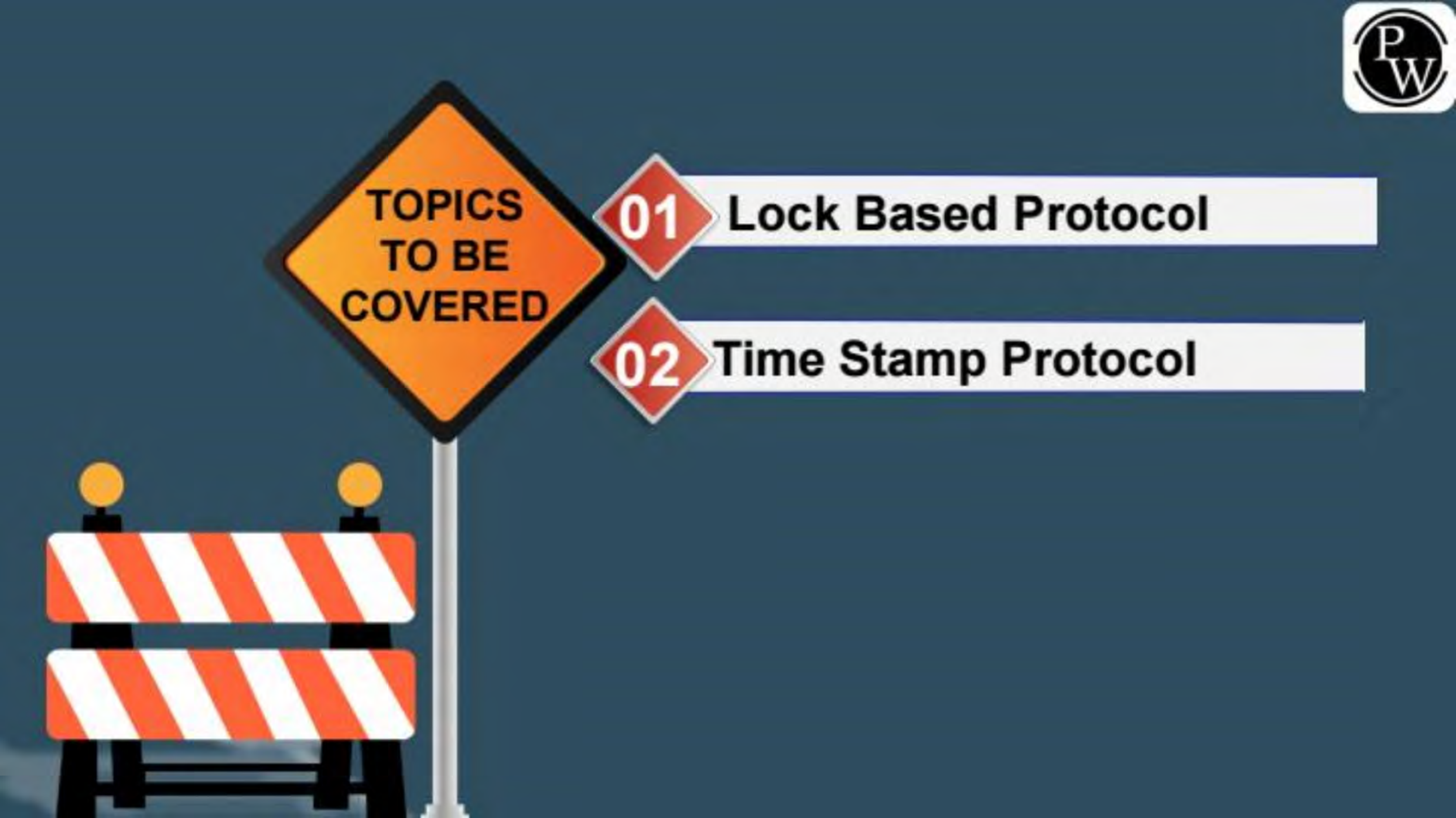
COMPUTER SCIENCE

Database Management System

Transaction & Concurrency Control

Lecture_8

Vijay Agarwal sir


TOPICS
TO BE
COVERED
01
Lock Based Protocol
02
Time Stamp Protocol

Lock Based Protocol.

Basic 2PL

Strict 2PL

Rigorious 2PL

Conservative 2PL.

2PL : Serializability order determine Based on Conflict operation at Run time.

With the Help of LOCK POINT.

TIMESTAMP BASED CONCURRENCY CONTROL

Time Stamp Protocal : [TSP]

- A unique Time Stamp value assigned to each transaction when they arrive in the System.
- Based on this time stamp determines the Serializability Order. [TSP, Predefined Serializability Order based on Time Stamp.

Time Stamp of T_1 : 10

Time Stamp of T_2 : 20

Serializability order : $T_1 \rightarrow T_2$

T_1 ← OLDER Transaction

T_2 ← Younger transaction.

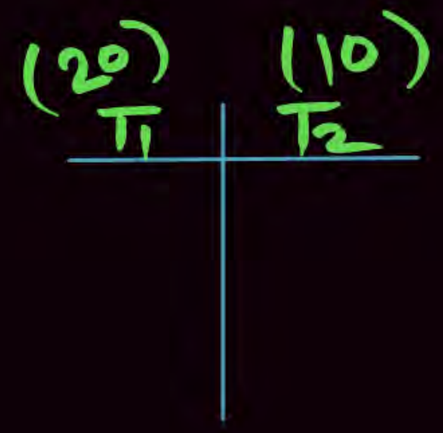
Transaction :	T_1	T_2	T_3	T_4	T_5	T_6
if Time Stamp of Transaction :	10	30	20	40	60	50

Serializability order :

OLDER Transaction

Younger (Newest) Transaction.

TS: Time Stamp.



eg

$$TS(T_1) > TS(T_2)$$

Serializability order : $T_2 \rightarrow T_1$

T_2: OLDER Transaction

T_1: younger Transaction

eg

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

Serializability order : $T_1 \rightarrow T_2$.

T_1: OLDER Transaction

T_2: younger transaction.

Timestamp-Based Protocols

- Each transaction T_i is issued a timestamp $TS(T_i)$ when it enters the system.
  - Each transaction has a unique timestamp
  - Newer (youngest) transaction have timestamp strictly greater than earlier (OLDER) ones
  - Timestamp could be based on a logical counter
    - Real time may not be unique
    - Can use (wall-clock time, logical counter) to ensure
- Timestamp-based protocols manage concurrent execution such that **time-stamp order = serializability order**
- Several alternative protocols based on timestamps

There are 2 Types of Time Stamp.

1. Transaction Time stamp — $TS(T_i)$ (Fixed)
2. Data Item Time Stamp. (Variable)
   1. Read-Time Stamp (Q); $RTS(Q)$
   2. Write Time stamp (Q): $WTS(Q)$

① Read_Time Stamp(Q); RTS(Q) : Denotes the Highest (youngest) Transaction Time Stamp that Perform Read(Q) operation successfully.

$T_1^{(10)}$	$T_2^{(20)}$	$T_3^{(30)}$
R(A)		
	R(A)	
		R(A)

Initially RTS = 0

RTS = 10, max[0,10]

RTS = 20, max[10,20]

RTS = 30 max[20,30]

RTS(A) = 30

RTS(A) = 30

RTS(A) = 30

Write Time Stamp; WTS(Q) : Denotes the Highest (younger) → largest Transaction Timestamp that perform write(Q) operation Successfully.

(10) T_1	(20) T_2	(30) T_3
W(A)		
	W(A)	
		W(A)

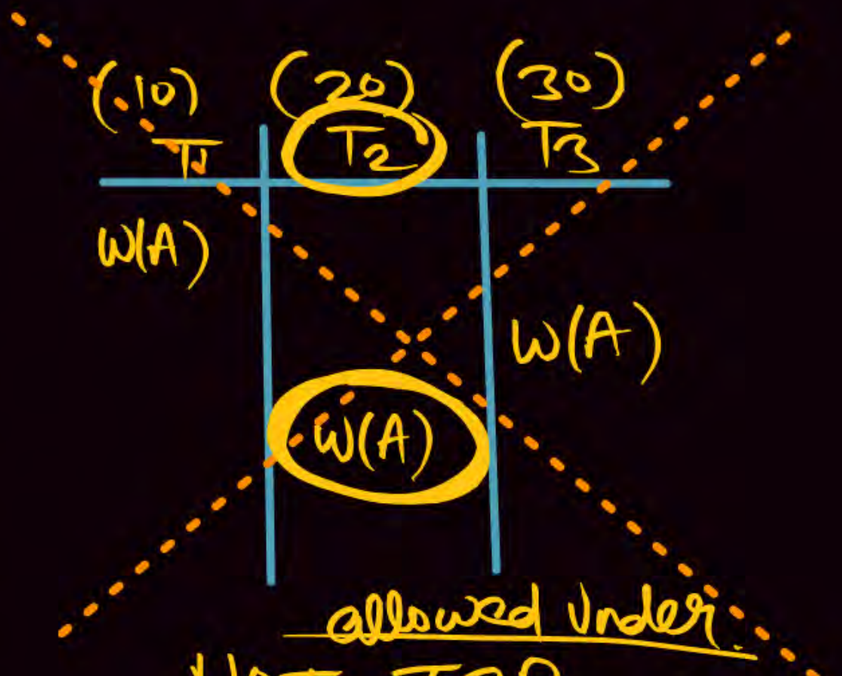
WTS = 0

WTS = 10 max(0,10)

WTS = 20 max(10,20)

WTS = 30 max(20,30)

WTS(A) = 30



TS(T_2) < WTS(A)

20 < 30 Not allowed

Timestamp-Based Protocols

The **timestamp ordering (TSQ) protocol**

- Maintains for each data Q two timestamp values:
  - **W-timestamp**(Q) is the largest time-stamp of any transaction that executed write (Q) successfully.
  - **R-timestamp** (Q) is the largest time-stamp of any transaction that executed **read** (Q) successfully.

V.V.V.Imp. Note

- Imposes rules on read and write operations to ensure that
  - any conflicting operations are executed in timestamp order
  - out of order operations cause transaction rollback

Conflict operation

Same Data Item

1. R(A) → W(A)
2. W(A) → R(A)
   W(A) → W(A)

Non Conflict operation

R(A) → R(A) && Different Data Item.

R(A) → W(B)

W(B) → R(A)

W(B) → W(A)

R-W

W-R

W-W

TSP

TS(Ti)

Data Item Time Stamp → RTS(Q)

Data Item Time Stamp → WTS(Q)

Ti: Read

TS(Ti) < WTS(Q)

Not allowed. ∴ Ti Rollback

Ti: Write

TS(Ti) < RTS(Q)

TS(Ti) < WTS(Q)

} Not allowed.

① Read(Q) : T_i

Transaction T_i wants to perform Read(Q) operation.

$TS(T_i) < WTS(Q)$; Read operation Reject & T_i Rollback. & Restart with New Stamp.

---

② T_i : write(Q) ; Transaction T_i wants to perform write(Q) operation.

$TS(T_i) < RTS(Q)$:
$TS(T_i) < WTS(Q)$: } write operation Reject & Transaction T_i Rollback & Restart with New Timestamp.

I: T_i – Read(Q) (Transaction T_i Issue R(Q) Operation)

(i) If TS (T_i) < WTS (Q): Read operation Reject & T_i Rollback. (Not allowed / Abort & Restart with New Transaction)

(ii) If TS (T_i) ≥ WTS(Q): Read operation is allowed

and Set Read – TS(Q) = max[RTS(Q), TS (T_i)]

II: T_i – Write(Q) (Transaction T_i Issue Write(Q) Operation)

(i) If TS (T_i) < RTS (Q): Write operation Reject & T_i Rollback.

(ii) If TS (T_i) < WTS(Q): Write operation Reject & T_i Rollback.

(iii) Otherwise execute write (Q) operation

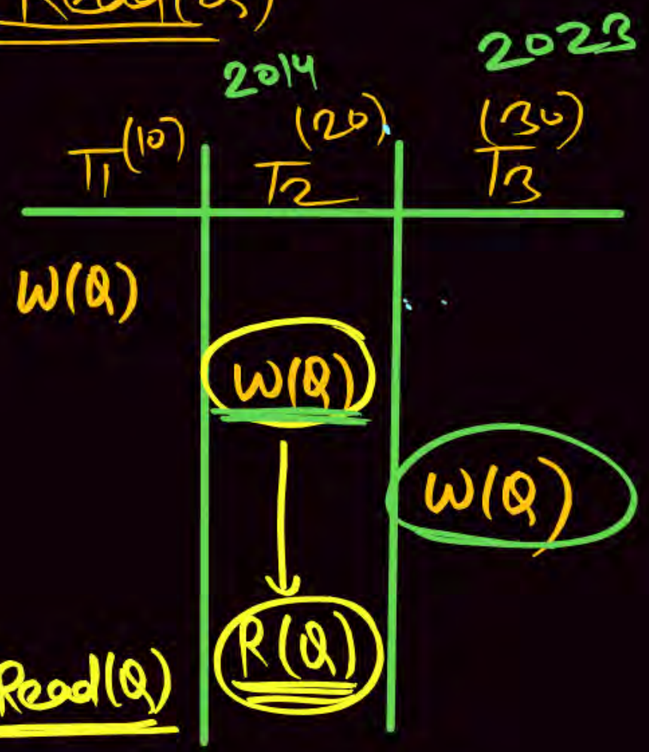
Set Read WTS(Q) = TS (T_i)

Timestamp-Based Protocols (Cont.)

- Suppose a transaction T_i issues a **read** (Q)
  1. If $TS(T_i) \leq$ **W**-timestamp (Q), then T_i needs to read a value of Q that was already overwritten.
     - Hence, the **read** operation is rejected, and T_i is rolled back. & Restart with New Timestamp.
  2. If $TS(T_i) \geq$ **W**-timestamp (Q), then the **read** operation is executed, and R-timestamp(Q) is set to.

     max(R-timestamp (Q), TS (T_i)).

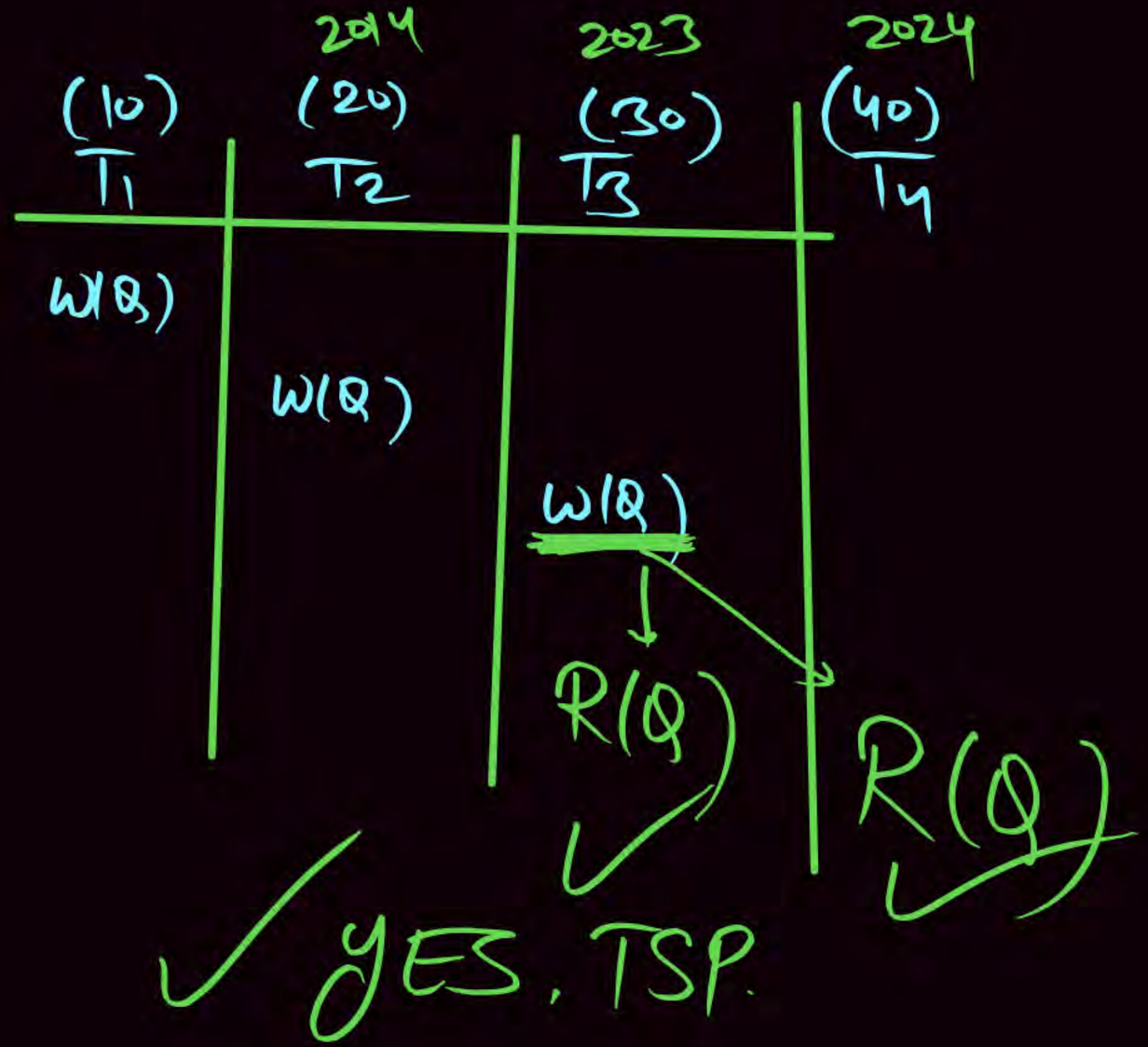
T_2: Read(Q)



$TS(T_2) < WTS(Q)$

$(20) < 30$

Not allowed under TSP.



YES, TSP.

Timestamp-Based Protocols (Cont.)

- Suppose a transaction T_i issues **write**(Q)
  1. If $TS(T_i) < $ **R**-timestamp(Q), then the value of Q that T_i is producing was needed previously, and the system assumed that the value would never be produced.
     - Hence, the **write** operation is rejected, and T_i is rolled back.
  2. If $TS(T_i) <$ **W**-timestamp (Q), then T_i is attempting to write an obsolete value of Q. [Obsolete write]
     - Hence, this **write** operation is rejected, and T_i is rolled back.
  3. Otherwise, the **write** operation is executed, and W-timestamp(Q) is set to $TS(T_i)$.

T_2 : write (Q)

(2014) (2023)

$TS(T_2) : 20$

$RTS(Q) : 30$

$TS(T_2) < RTS(Q)$

$20 < 30$

✗ Not allowed

(2)

$TS(T_2) < WTS(Q)$

$20 < 30$

✗ Not TSP

SUMMARY

$T_i : R(Q)$

T_i [Read(Q)]

$TS(T_i) < WTS(Q)$

Read operation Not allowed under TSP.

&& Transaction T_i Abort Rollback, Restart with New(Higher) Timestamp.

T_i write(Q)

① $TS(T_i) < RTS(Q)$

② $TS(T_i) < WTS(Q)$

then write operation Not allowed Under TSP & T_i Abort, Restart(T_i) with Higher timestamp.

Alternate Approach to Check allowed under TSP or Not.

TSP

$TS(T_1) < TS(T_2)$

If $TS(T_1) : 10$

If $TS(T_2) : 20$

Serializable order : $T_1 \rightarrow T_2$

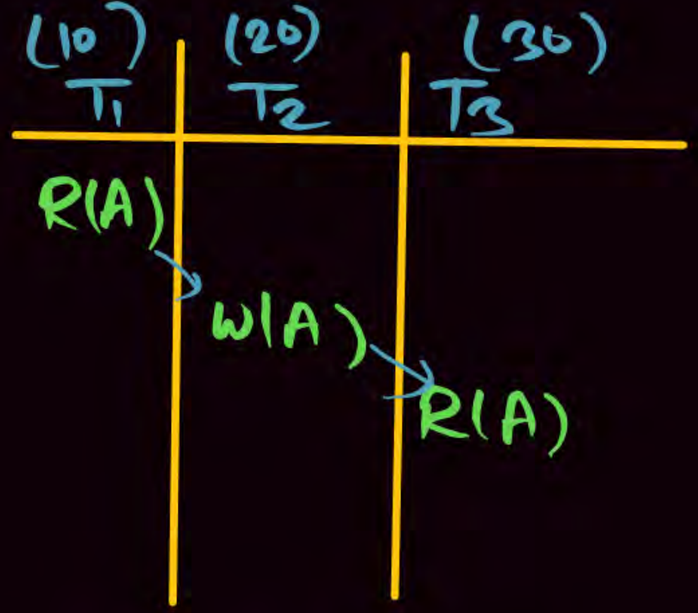
T_1 followed by T_2.

Note

Then all Conflicting operation must be executed in the order T_1 followed by T_2 (Same as Serializability order)

if YES then allowed under TSP

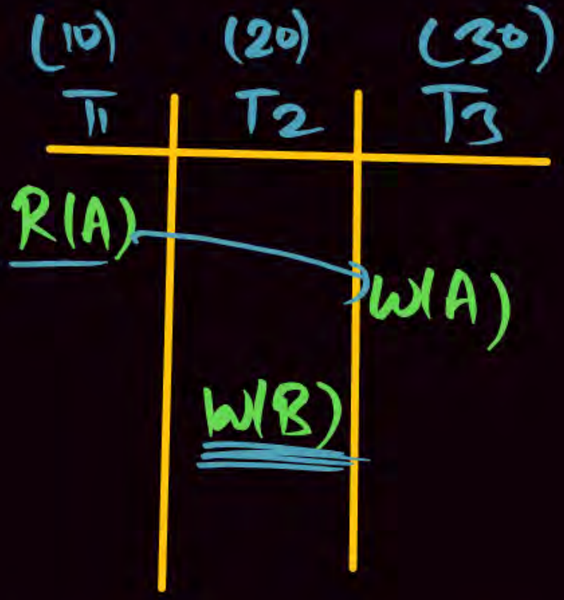
if No then Not allowed Under TSP.



Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

YES TSP



Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

YES

Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

YES TSP

Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

YES

Serializability order

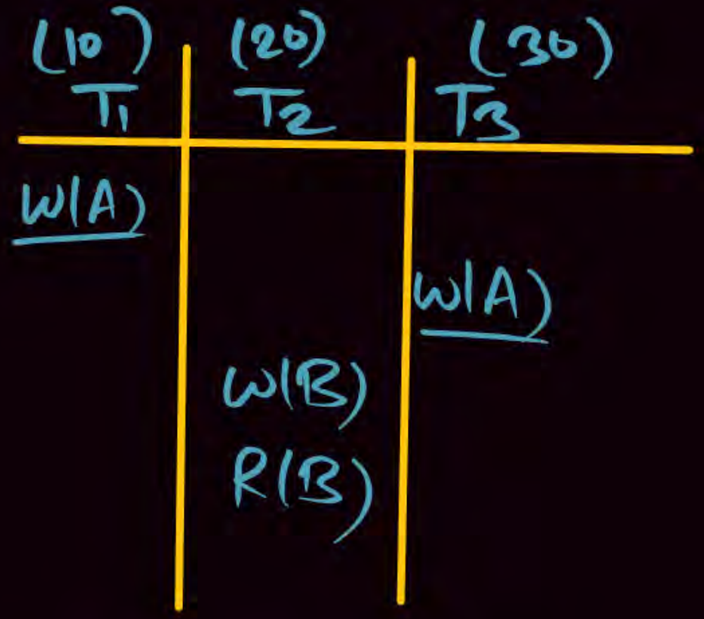
$T_1 \rightarrow T_2 \rightarrow T_3$

YES

Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

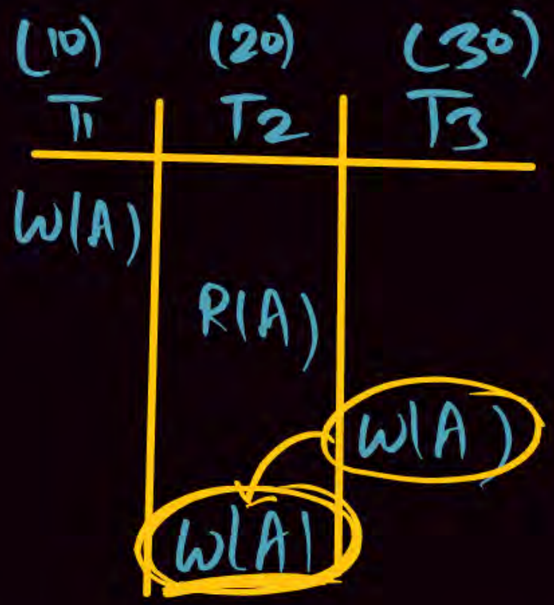
YES



Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

YES



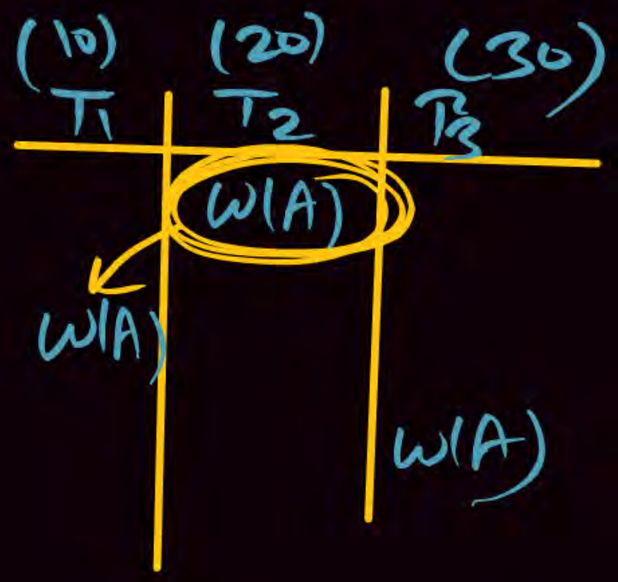
Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

Not allowed.

$TS(T_2) < WTS(A)$ Not allowed

T_2 Rollback & Restart

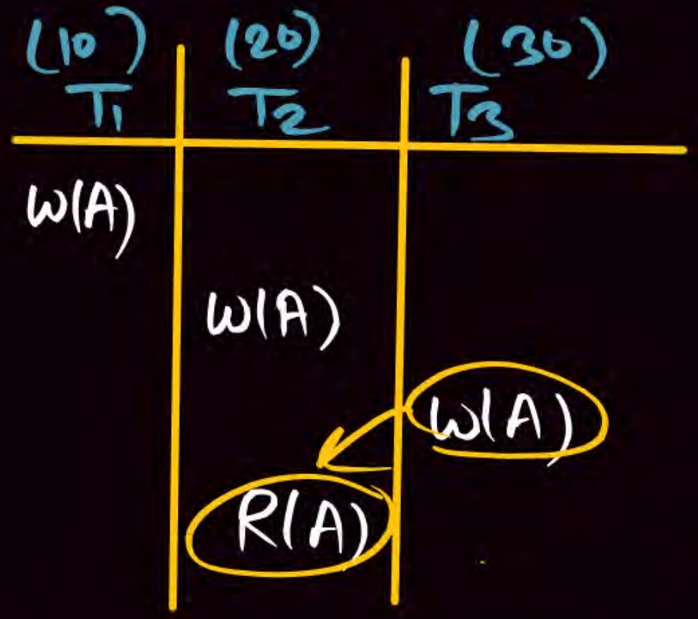


Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

Not TSP

T_1 Rollback

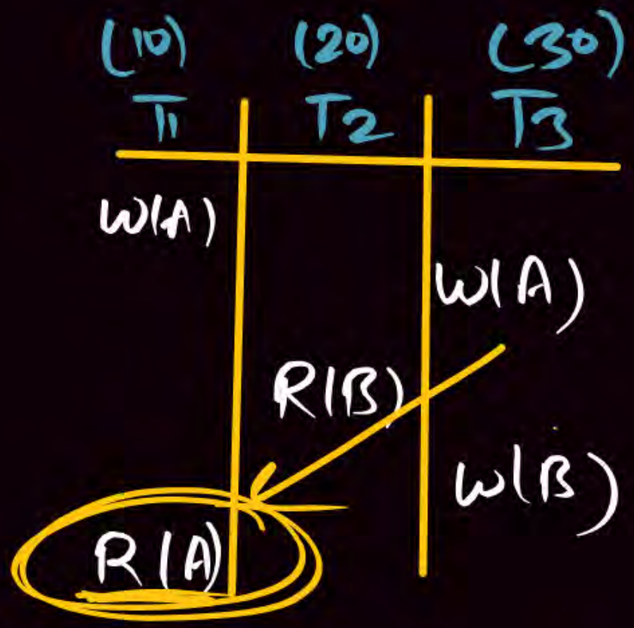


Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

Not TSP

T_2 Rollback



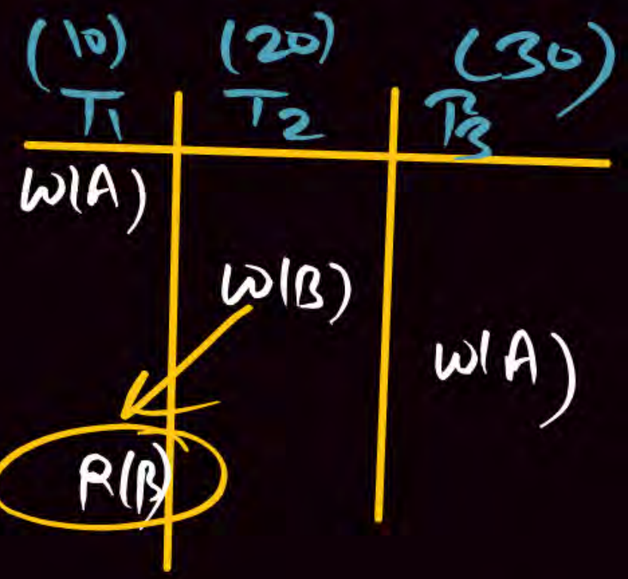
Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

Not TSP

$TS(T_1) < WTS(A)$

10 < 30; T_1 Rollback.



Serializability order

Not TSP $T_1 \rightarrow T_2 \rightarrow T_3$

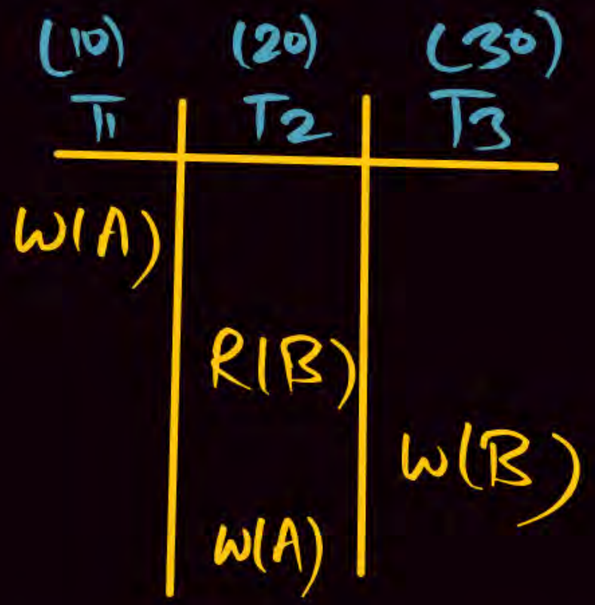
T_1 Restart with New (Highest Time stamp)

Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

Not TSP

T_2 Rollback



Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

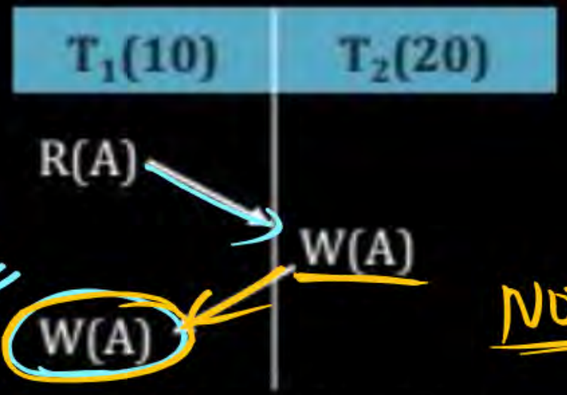
YES ✓ TSP

Serializability order

$T_1 \rightarrow T_2 \rightarrow T_3$

YES TSP.

(1)



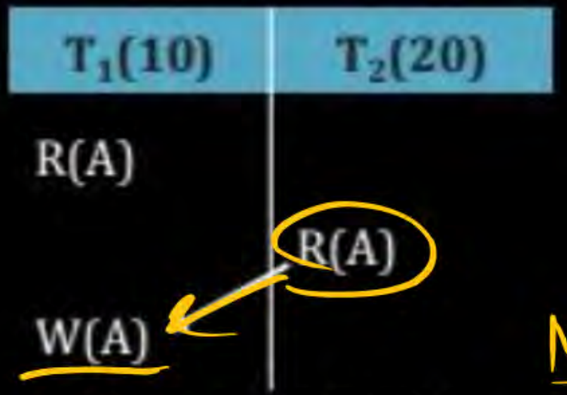
T1 Rollback

NOT TSP

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(2)

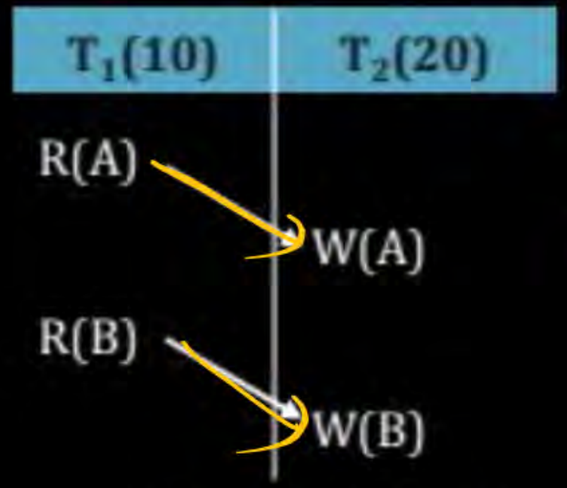


NOT TSP

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(3)

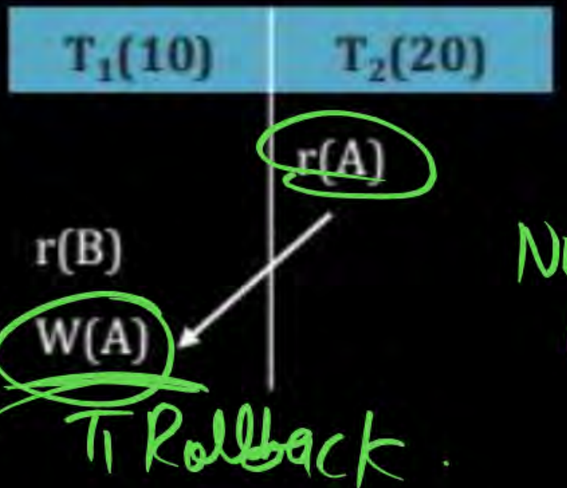


YES its allowed Under TSP

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(4)



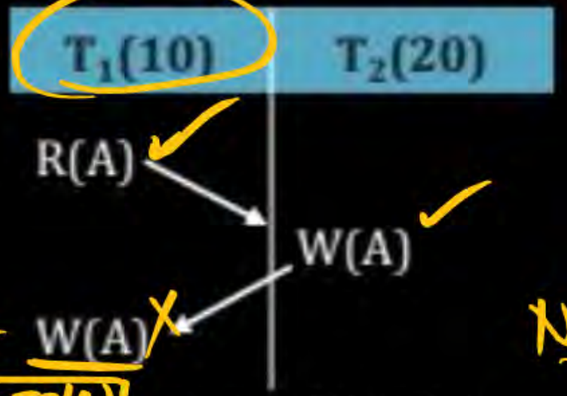
Not allowed Under TSP.

T1 Rollback.

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(1)



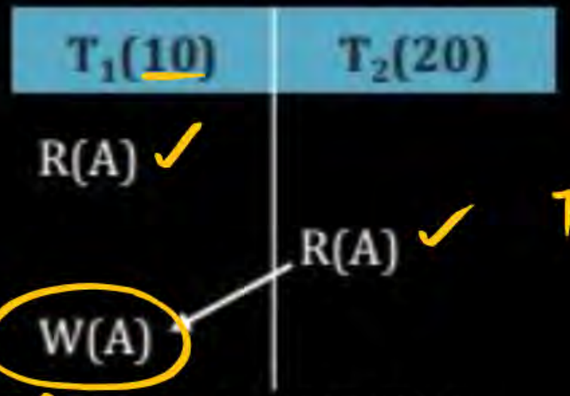
Not TSP

∴ Reject W(A)

TS(T1) < WTS(A)

T1 Rollback

(2)

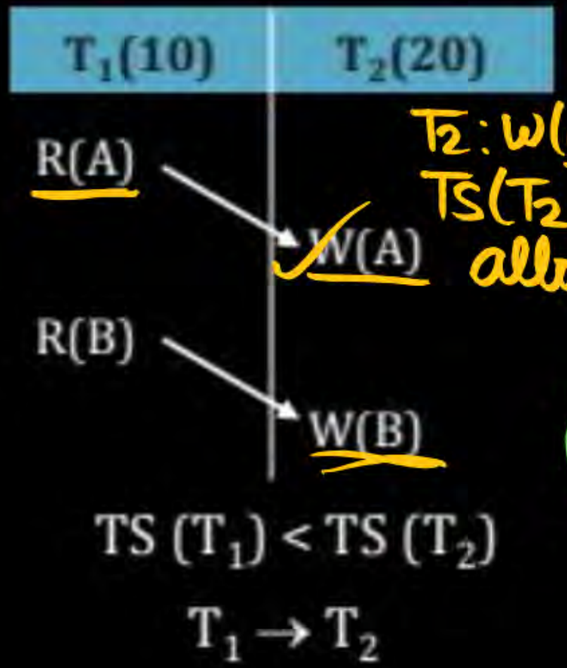


TS(T1) < RTS(A)

10 < 20

Reject & T1 Restart.

(3)



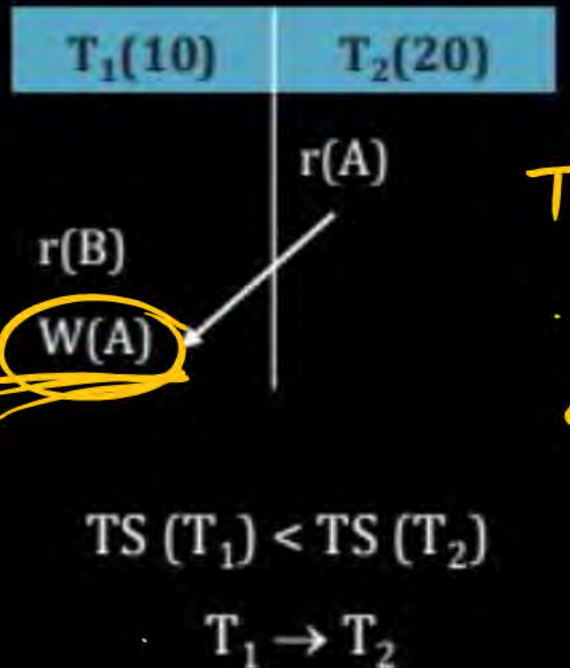
T2: W(A)

TS(T2) ≮ RTS(A)

allowed

YES its allowed

(4)



T1: W(A)

TS(T1) < RTS(A)

Reject & Rollback T1 & T1 Restart with new Timestamp.

Q Why Restart

New (Higher) Time Stamp ?

④

(10) T_1	(20) T_2
	R(A)
W(A)	

if again Restart with same Time Stamp

10 T_1	20 T_2
	R(A)
W(A)	

again Not allowed Under TSP again Rollback

if Higher time stamp

(20) T_2	>20 30 T_1
R(A)	
	W(A)

YES allowed Under TSP.

$T_2 \rightarrow T_1$

& Conflict operation $T_2 \rightarrow T_1$

Important Point about TSP

① If Schedule followed by TSP then Ensure Conflict Serializable

② TSP Not Ensure Recoverability.

& Cascading Rollback Possible.

$T_1 \rightarrow T_2$

allowed under TSP But Irrecoverable Schedule

Important Point about TSP

③ TSP Free from Deadlock.

④ Starvation May (or) May Not occur.

(W-W allowed only) (TWR)

Thomas write Rule (View Serializability)

Some Little Modification in TSP.

allowed obselete write. But obselete write Not allowed under TSP.

T_i : Read(Q) → Same as TSP.

$TS(T_i) < WTS(Q)$; Read operation Reject & T_i Rollback.

T_i : write(Q)

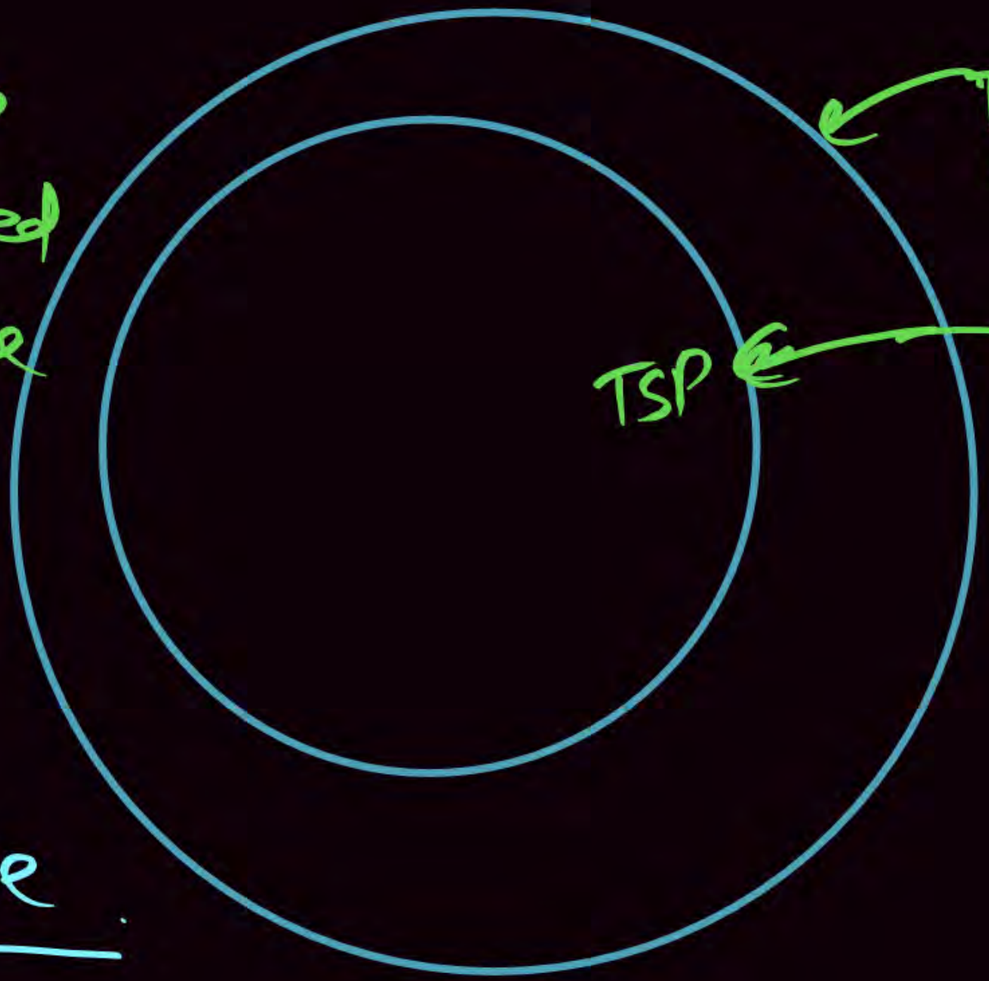
→ $TS(T_i) < WTS(Q)$; write operation & No Rollback (allowed TWR)
Ignored

obselete write allowed under TWR but Not TSP

$TS(T_i) < RTS(Q)$; Reject & T_i Rollback

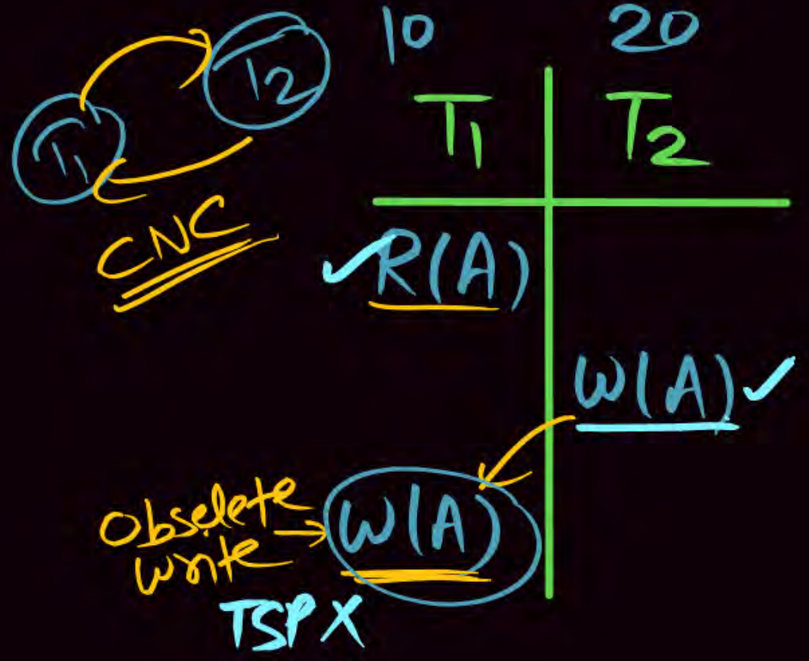
① S if allowed by TSP then already allowed by Thomas write Rule

② If some schedule Not allowed by TSP, then it may allowed by Thomas write Rule.



Thomas' Write Rule

- Modified version of the timestamp-ordering protocol in which obsolete **write** operations may be ignored under certain circumstances.
- When T_i attempts to write data item Q, if $TS(T_i) < W\text{-timestamp}(Q)$, then T_i is attempting to write an obsolete value of {Q}.
  - Rather than rolling back T_i as the timestamp ordering protocol would have don, this {**write**} operation can be ignored.
- Otherwise this protocol is the same as the timestamp ordering protocol. (Same as TSP)
- Thomas' Write Rule allows greater potential concurrency. → Not Conflict those also allowed
  - Allows some view-serializable schedules that are not conflict-serializable.



$TS(T_1) < WTS(A)$

$10 < 20$; Not allowed Under TSP.

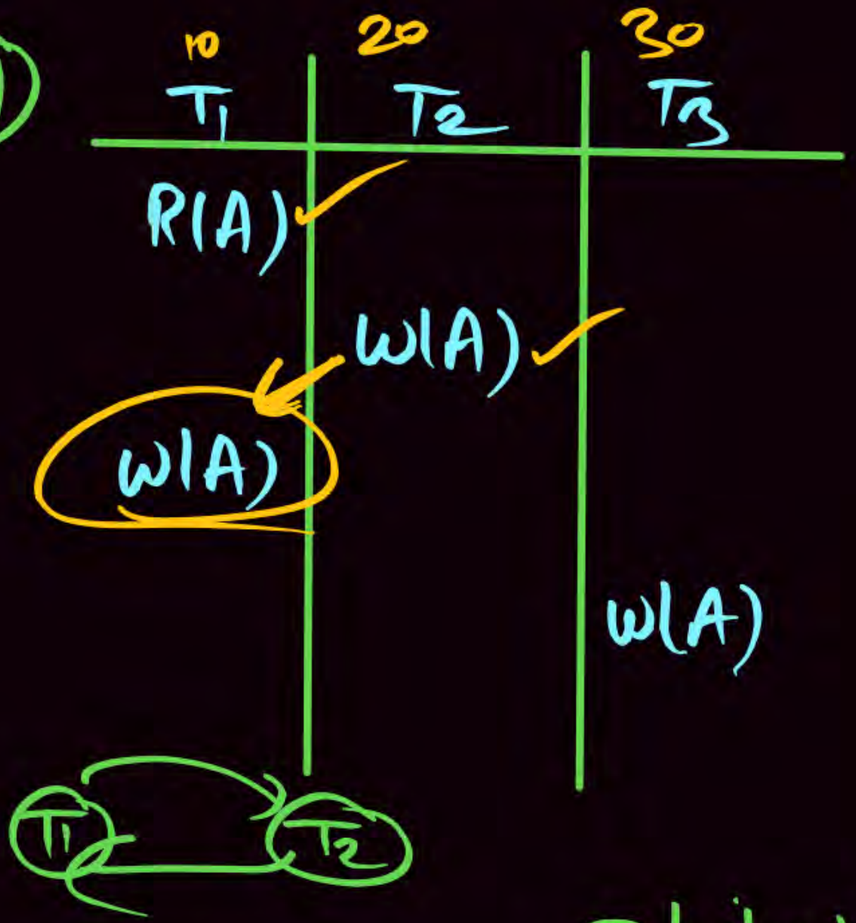
But allowed under Thomas write Rule.

$T_1 \Rightarrow W(A)$: Ignored, No Rollback

allowed Under TWR.

More Concurrency means this Schedule is Not Conflict serializable, this Not allowed Under TSP But allowed by Thomas Write Rule.

(Q1)



(T3)

CNC [Not Conflict]

Check TSP ?

10	20
TS(T1)	< WTS(A)

Reject T1 Rollback

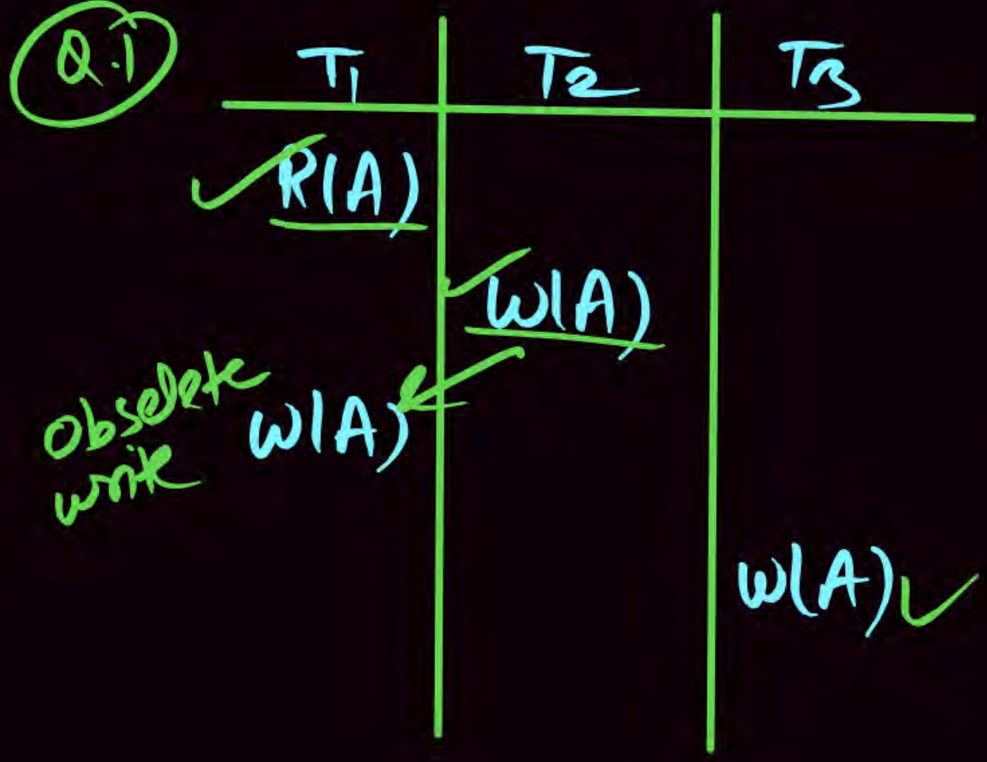
Not allowed under TSP.

But allowed under Thomas Write Rule.

But its View Serializable (T1, T2, T3)

P.T.O

Q.1



This allowed under Thomas write Rule.

$T_1 \rightarrow T_2$

Not TSP.

CHECK TWR ?

T_1 : W(A) $TS(T_1) < WTS(A)$ YES allowed in TWR

T_1 : R(A) $\Rightarrow$ $TS(T_1) < WTS(A)$ Not in TWR.

NOT TSP

NOT TWR.

Thomas Write Rule (View Serializability)

1. TS (T_i) < RTS(Q) : Rollback
2. TS(T_i) < WTS(Q) : Write operation is Ignored and No Roll back

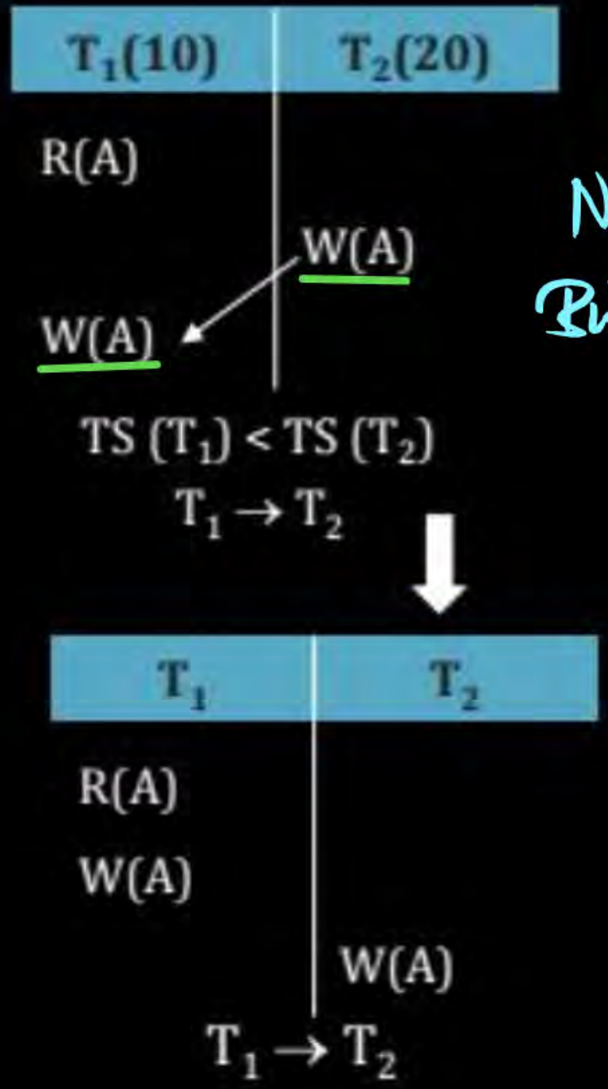
Same as TSP

Time Stamp Protocol: Ensure serializability deadlock free but starvation possible

Deadlock Prevention Algorithm

(1) Wait-Die (2) Wound-wait

Older Younger

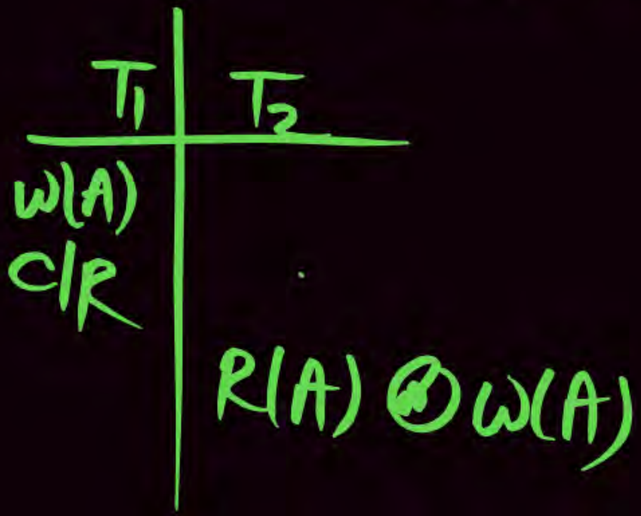


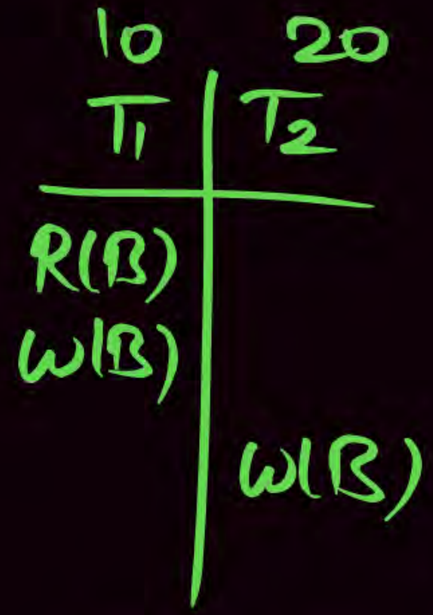
Not in TSP But in TWR.

STRICT Time Stamp ordering Protocol = TSP + Strict Schedule.

- Ensure Serializability
- Ensure Recoverability
- Cascadeless (No Cascading Rollback)
- Strict Recoverable

STRICT | STRICT Recoverable





YES TSP.

YES TWR.

Q.

In which one of the following Lock Scheme Deadlock cannot occur?

[MCQ]

A. Basic 2PL

B. Strict 2PL

C. Conservative 2PL

D. Rigorous 2PL

Q.

Consider the following statement about lock–based protocol

(A) 2 PL (2phase locking) protocol Ensure view serializability

(B) 2PL ensure recoverability &No cascading rollback.

(C) Strict 2 PL ensure recoverability & no cascading rollback.

(D) Strict 2 PL avoids deadlock (not suffering from deadlock).

How many numbers of above statement are correct?

[MCQ]

A 1

B 2

C 3

D 4

Q.

[MSQ]

Consider the following Schedule:

$r_1(x)\ r_2(y)\ r_2(x)\ w_1(z)\ r_1(y)\ w_3(y)\ r_3(z)\ w_2(y)\ w_3(x)$

which of the following time stamp ordering Not allows to execute schedule using Thomas Write rule time stamp Ordering Protocol?

A $(T_1, T_2, T_3) = (20, 30, 10)$

B $(T_1, T_2, T_3) = (10, 20, 30)$

C $(T_1, T_2, T_3) = (10, 30, 20)$

D $(T_1, T_2, T_3) = (30, 20, 10)$

MCQ

Consider the following two statements about database transaction schedules:

I. Strict two-phase locking protocol generates conflict serializable schedules that are also recoverable.

II. Timestamp-ordering concurrency control protocol with Thomas Write Rule can generate view serializable schedules that are not conflict serializable.

Which of the above statements is/are True?

[GATE-2019-CS: 1M]

A. I only

B. II only

C. Both I and II

D. Neither I nor II

MCQ

Consider the following partial Schedule S involving two transactions T1 and T2. Only the read and the write operations have been shown. The read operation on data item P is denoted by read (P) and the write operation on data item P is denoted by write (P).

Suppose that the transaction T1 fails immediately after time instance 9. Which one of the following statements is correct?

Time	Transaction-id	
	T 1	T 2
1.	read(A)	
2.	write(A)	
3.		read(C)
4.		write(C)
5.		read(B)
6.		write(B)
7.		read(A)
8.		commit
9.	read(B)	

[GATE-2015-CS: 2M]

A. T2 must be aborted and then both T1 and T2 must be restarted to ensure transaction atomicity

B. Schedule S is non recoverable and cannot ensure transaction atomicity

C. One T2 must be aborted and then restarted to ensure transaction atomicity

D. Schedule S is recoverable and can ensure atomicity and nothing else needs to be done

MCQ

Which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock?

I. 2-phase locking

II. Time-stamp ordering

[GATE-2010-CS: 1M]

A I only

B II only

C Both I and II

D Neither I nor II

THANK YOU!