
COMPUTER SCIENCE
Database Management System
Transaction & Concurrency Control
Lecture_11
GATE WALLAH
Vijay Agarwal sir

TOPICS TO BE COVERED

01 Time Stamp Protocol.

Lock Based Protocol.

- Basic 2PL
  - Growing Phase (Acquire lock)
  - Shrinking Phase [Release lock]
- STRICT 2PL.
- Rigorious 2PL
- Conservative 2PL.

Conservative 2PL.

STARVATION

A Suffer from

- → 2PL
- → C.S.S
- → Recoverable, Cascadeless, Strict Recoverable
- → No Cascading Rollback
- → No Deadlock

1	2	3	4	5
Lock-S(A)	Lock-S(A)	Lock-S(A)	Lock-S(A)	Lock-S(A)
R(A)	R(A)	R(A)	Lock-X(B)	R(A)
Lock-X(B)	Lock-X(B)	Lock-X(B)	R(A)	Unlock(A)
R(B)	Unlock(A)	R(B)	R(B)	Lock-X(B)
Unlock(A)	R(B)	W(B)	W(B)	R(B)
W(B)	W(B)	Commit	Commit	W(B)
Unlock(B)	Commit	Unlock(A)	Unlock(A)	Unlock(B)
Commit	Unlock(B)	Unlock(B)	Unlock(B)	Commit

TIMESTAMP BASED CONCURRENCY CONTROL

Time Stamp Protocal.

↳ unique Timestamp value is assigned to each transactions whenever they enter into System.

eg

$TS(T_1) = 10$

$TS(T_3) = 20$

$TS(T_2) = 30$

$TS(T_4) = 40$

$TS(T_6) = 50$

$TS(T_5) = 60$

Serializablity Order.

10		20		30		40		50		60
T_1	$\rightarrow$	T_3	$\rightarrow$	T_2	$\rightarrow$	T_4	$\rightarrow$	T_6	$\rightarrow$	T_5

OLDER Transaction

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 transaction have timestamp strictly greater than earlier 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

Q1

TS(T_1) : 10

TS(T_2) : 20

TS(T_3) : 30

$$\overset{10}{T_1} \longrightarrow \overset{20}{T_2} \longrightarrow \overset{30}{T_3}$$

Serializable order

$\langle T_1, T_2, T_3 \rangle$

Q2 TS(T_1) : 20

TS(T_2) : 10

$$\overset{10}{T_2} \longrightarrow \overset{20}{T_1}$$

$\langle T_2, T_1 \rangle$

T_1 followed by T_2.

eg TS(T_1) : 10

TS(T_3) : 20

TS(T_2) : 30

$$\overset{10}{T_1} \longrightarrow \overset{20}{T_3} \longrightarrow \overset{30}{T_2}$$

OLDER Transaction

Younger (New) Transaction.

$\langle T_1, T_3, T_2)$

eg) $TS(T_1) : 10$

$TS(T_2) : 20$

$$\overset{10}{T_1} \longrightarrow \overset{20}{T_2}$$

$\langle T_1, T_2 \rangle$

2 Type of Time Stamp.

1. Transaction Time Stamp $TS(T_i)$

2. Time Stamp of Data Item

(i) Read Time Stamp [$RTS(Q)$]

(ii) Write Time Stamp [$WTS(Q)$]

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

Initially $RTS(A) = 0$ [max(0, 10)]

$RTS(A) = 10$ [max(10, 20)]

$RTS(A) = 20$

max(20, 30)

$\boxed{RTS(A) = 30}$

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

$WTS(A) = 0$ (max(0, 10)

$WTS(A) = 10$

$WTS(A) = 20$ max(10, 20)

$\boxed{WTS(A) = 30}$ max(20, 30)

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.V Imp

- 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

V. V. V. Imp

TSP

Conflict operation order

Must be same Transaction timeStamp order

then Perform Read & Write operation

Successfully, Otherwise Rollback the transaction

Conflict operation

Same Data Item

& Different Transaction

T_i		T_j
R(A)	—	W(A)
W(A)	—	R(A)
W(A)	—	W(A)

Conflict operation

T_i → Read(Q)

① $TS(T_i) < WTS(Q)$

Read Reject & T_i Rolled back

T_i → Write(Q)

① $TS(T_i) < RTS(Q)$

② $TS(T_i) < WTS(Q)$

Write operation Reject & T_i Rollback

Conflict oper

R(A) – W(A)

W(A) – R(A)

W(A) – W(A)

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.

(ii) If TS (T_i) $\geq$ 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.

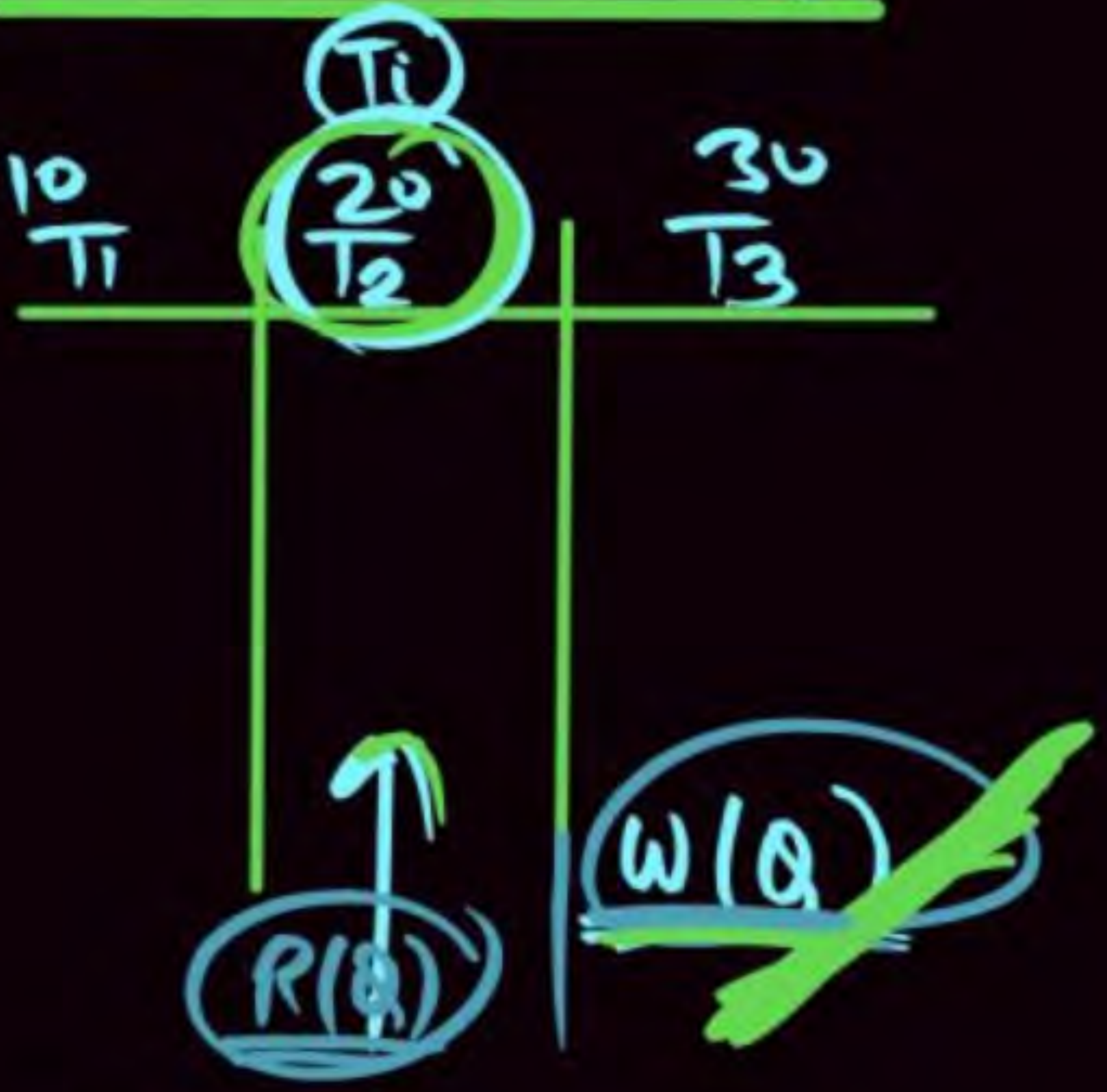
(ii) If TS (T_i) < WTS(Q): Write operation Reject & T_i Rollback.

(iii) Otherwise execute write (Q) operation

Set Read WTS(Q) = TS (T_i)

Read

1. $TS(T_i) < WTS(Q)$



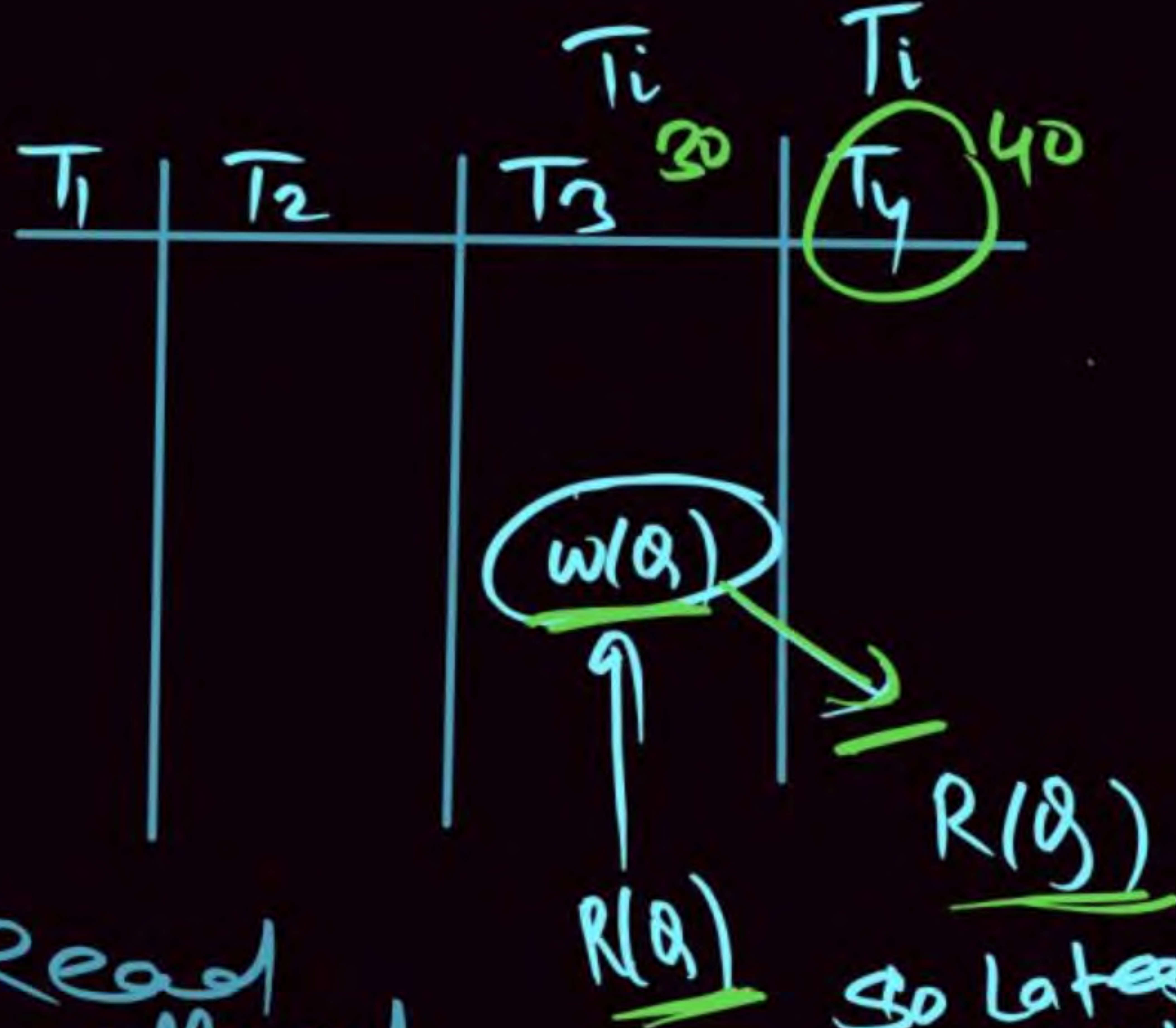
Assume $TS(T_i) = T_2 = 20$

$WTS(Q) = 30$

$TS(T_i) < WTS(Q)$

$20 < 30$

Read Not allowed.



Read allowed

So Latest (updated correct) Value Read.

T_i (T_2) write(A)

T_1	T_2 (20)	T_3 (30)
	~~×~~	
		W(A) ✓

$TS(T_2) = 20$

$\boxed{WTS(A) = 30}$

$TS(T_2) < WTS(A)$

(20) ← (T_2) ⇒ write(A)

T_1 (10)	T_2 (20)	T_3 (30)
	~~×~~	
		R(A)
	W(A)	

$\boxed{RTS(A) = 30}$

$TS(T_2) = 20$

$TS(T_2) < RTS(A)$

Timestamp-Based Protocols (Cont.)

20 < 30

- 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.

2. If $TS(T_i) \geq$ **W**-timestamp (Q), then the **read** operation is executed, and R-timestamp(Q) is set to.

    $\boxed{\textbf{max}(\text{R-timestamp (Q)}, \text{TS } (T_i)).}$

Timestamp-Based Protocols (Cont.)

T_2 30

- 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.
     - 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 $TS(T_i)$.

I^{st} Approach

$TS(T_i)$: Read(A)

① If $TS(T_i) < WTS(A)$; Read operation Reject, (Not allowed) & T_i Rollback.

$TS(T_i)$: Write(A)

① If $TS(T_i) < RTS(A)$
② If $TS(T_i) < WTS(A)$

write Operation Reject, (Not allowed) & T_i Rollback.

Obselete write

2nd Approach

V. V. V. Imp

TSP

Conflict operation order
Must be same Transaction timestamp order
then Perform Read & write operation
Successfully, Otherwise Rollback the transaction.

Conflict operation

Same Data Item
& Different Transaction

Ti	Tj
R(A) —	W(A)
W(A) —	R(A)
W(A) —	W(A)

Conflict operation

(1)

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

$TS(T_1) < TS(T_2)$

$W_2(A) - W_1(A)$
$T_2 \to T_1$

$\boxed{T_1 \to T_2}$

NOT TSP.

(2)

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

$TS(T_1) < TS(T_2)$

$\boxed{T_1 \to T_2}$

$R_2(A) - W_1(A)$
$T_2 \to T_1$

OR

$TS(T_1) = 10$ (write)
$RTS(A) = 20$
$TS(T_1) < RTS(A)$
X write Reject

(3)

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

$TS(T_1) < TS(T_2)$

$\boxed{T_1 \to T_2}$

$R_1(A) - W_2(A)$
$R_1(B) - W_2(B)$
$T_1 \to T_2$
YES TSP

(4)

$T_1(10)$	$T_2(20)$
	r(A)
r(B)	
W(A)	

$\boxed{TS(T_1) < TS(T_2)}$

$\boxed{T_1 \to T_2}$

$r_2(A) - W_1(A)$
$T_2 \to T_1$
Not in TSP.

$$\overset{10}{TS(T_1)} < \overset{20}{TS(T_2)}$$

$$\boxed{T_1 \rightarrow T_2}$$

(4) $RTS(A) = 20$

$TS(T_1) = 10$

write

$$\underbrace{\underset{10}{TS(T_1)} < \underset{20}{RTS(A)}}_{\text{write Not allowed.}}$$

①

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

write $TS(T_1) = 10$

$WTS(A) = 20$

write $\overset{10}{TS(T_1)} < \overset{20}{WTS(A)}$; Reject

Not TSP

$RTS(A) = 10$

$TS(T_2) = 20$

$TS(T_i) \nless RTS(A)$

write allowed

& $WTS(A) = 20$

(1)

10 20

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

$TS(T_1) < TS(T_2)$

$\boxed{T_1 \rightarrow T_2}$

(2)

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

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(3)

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

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

(4)

$T_1(10)$	$T_2(20)$
	r(A)
r(B)	
W(A)	

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

Thomas Write Rule (View Serializablity)

⇓

Obselete Write ✓

Write(Q)

(1) $\boxed{TS(T_i) < WTS(Q)\text{; Write operation Ignored \& No Rollback}}$

& Same as TSP.

Read → same as TSP.

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-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.
- Thomas' Write Rule allows greater potential concurrency.
  - Allows some view-serializable schedules that are not conflict-serializable.

Thomas Write Rule (View Serializability)

Write

Reject

1. $TS(T_i) < RTS(Q)$: Rollback
2. $\boxed{TS(T_i) < WTS(Q)}$: 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

10 20

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

$T_1 \rightarrow T_2$

$R_1(A) - W_2(A)$ ✓

$W_2(A) - W_1(A)$

$T_2 \rightarrow T_1$

$TS(T_1) < TS(T_2)$

$T_1 \rightarrow T_2$

$TS(T_1) < WTS(A)$

allowed under Thomas Write Rule

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

$T_1 \rightarrow T_2$

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. Consider the following Schedule: **[MSQ]**

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

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

a

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

10 T3	20 T1	30 T2
	r(x)	
		r(y) r(x)
	w(z) r(y)	
w(y) r(z)		
		w(y)
w(x)		

$T_3 \rightarrow T_1 \rightarrow T_2$

$r_1(x) - w_3(x)$

$T_1 \rightarrow T_3$

Not TSP & Thomas Write

Q. Consider the following Schedule: **[MSQ]**

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

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

C

10 T_1	20 T_3	30 T_2
r(x)		
		r(y)
		r(x)
w(z)		
r(y)		
	w(y)	
	r(z)	
		w(y)
	w(x)	

$T_1 \rightarrow T_3 \rightarrow T_2$

$r_2(y) - w_3(y)$

$T_2 \rightarrow T_3$

But order was

$T_3 \rightarrow T_2$

Not TSP &

Thomas write

Q. Consider the following Schedule: [MSQ]

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

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

(10) T_1	(20) T_2	(30) T_3
r(x)		
	r(y)	
	r(x)	
w(z)		
r(y)		
		w(y)
		r(z)
	w(y)	
		w(x)

$\overset{10}{T_1} \rightarrow \overset{20}{T_2} \rightarrow \overset{30}{T_3}$

$w_3(y) - w_2(y)$

$T_3 \rightarrow T_2$

Not in TSP But allowed under Thomas write Rule (obselete write)

Q.

Consider the following statements:

S_1: All strict recoverable schedule are serial.

S_2: All recoverable schedules are conflict serializable.

S_3: All strict schedules are conflict serializable.

S_4: All conflict serializable schedules are free from cascading rollbacks.

Which of the following is true?

(a) Only S_1 and S_4

(b) Only S_2, S_3 and S_4

(c) Only S_2 and S_4

(d) None of these

Q.

Consider the following transaction:

T_1: $R_1(x)\ W_1(x)\ R_1(y)\ W_1(y)$

T_2: $W_2(y)\ W_2(x)$

The number of non-serial schedules between T_1 and T_2 which are serializable?

(a) 2 (b) 13

(c) 15 (d) None of these

THANK
YOU!