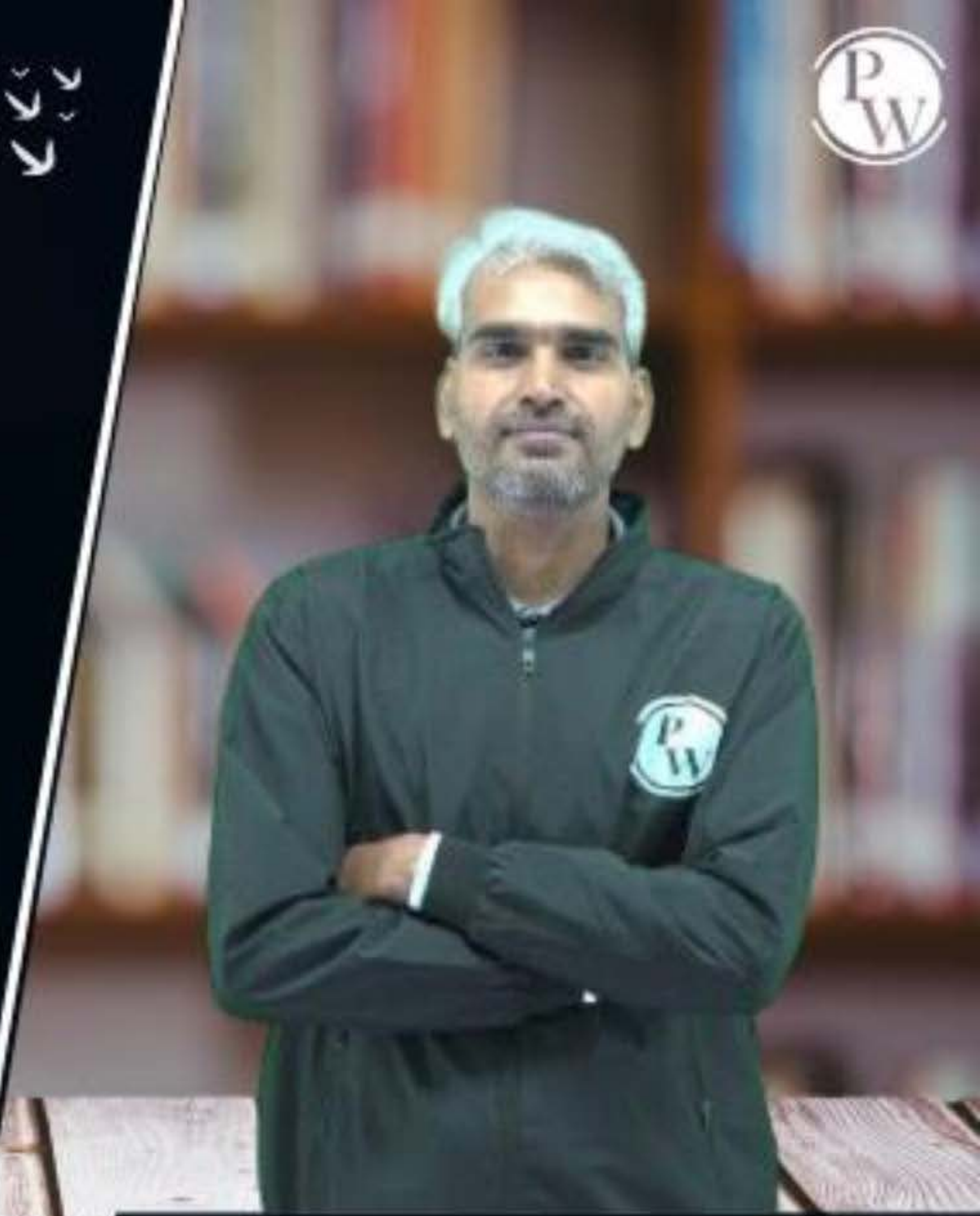
Computer Science & IT

Database Management System

Transaction & concurrency control

Lecture No. 10



By- Vishal Sir

Recap of Previous Lecture

✓ Topic Conservative 2PL

✓ Topic Rigorous 2PL

✓ Topic Basic time stamp ordering protocol

Slide

Topics to be Covered

- ✓ Topic — Basic time stamp ordering protocol
- ✓ Topic — Time stamp ordering protocol with Thomas write rule
- ✓ Topic — Wait-die protocol
- ✓ Topic — Wound-wait protocol

} Deadlock prevention Algorithms

Slide

Time Stamp Ordering Protocols

Topic : Time stamp ordering protocols

- There are two different versions of time stamp ordering protocol

  ✓ (1) Basic time stamp ordering protocol

  ✓ (2) Time stamp ordering protocol with Thomas Write Rule

* Time Stamp :-

Time stamp is a unique value assigned by DBMS to each transaction in ascending order.

- Let T_1 and T_2 are two transactions in the system, such that

$$\underset{TS(T_1)}{\text{Time stamp of } T_1} < \underset{TS(T_2)}{\text{Time Stamp of } T_2}$$

then, T_1 is the old transaction & T_2 is the younger transaction

- Read time stamp of data item 'A'

  RTS(A): It is the highest time stamp value among the time stamps of the transactions that has Performed Read(A) opⁿ successfully.

- Write time stamp of dataitem 'A'

  WTS(A): It is the highest time stamp value among the time stamps of the transactions that has Performed Write(A) opⁿ successfully.

* Initially,

$$\left.\begin{array}{l} RTS(A) = 0 \\ WTS(A) = 0 \end{array}\right\} \text{for all dataitems 'A'}$$

eg: RTS(A) & WTS(A) :-

$TS(T_1)=10$ $TS(T_2)=20$ $TS(T_3)=30$ $TS(T_4)=40$

	T_1	T_2	T_3	T_4	RTS(A)	WTS(A)
Initially →					0	0
After R(A) opn of transaction T_1 →	R(A)				10	0
			R(A)		30	0
		W(A)			30	20
		R(A)			30	20
				W(A)	30	40
			W(A)		30	40

Topic : Basic Time stamp ordering protocol

(B.T.S.O.P.)

- A schedule is allowed to execute using B.T.S.O.P. if and only if schedule is a Conflict serializable schedule and Conflict equivalent serial schedule is based on time stamp ordering of the transaction

eg Conside the Schedule 'S' with time stamp ordering as specified along with transactions

S

✓ T_1	✓ T_2	✓ T_3
$TS(T_1)=20$	$TS(T_2)=10$	$TS(T_3)=30$

We can observe that the time stamp ordering is

$$TS(T_2) < TS(T_1) < TS(T_3)$$

i.e; $T_2 \to T_1 \to T_3$

Schedule 'S' will be allowed to Execute using B.T.S.O.P. if and only if schedule 'S' is a Conflict Serializable Schedule and Conflict Equivalent Serial Schedule is $T_2 \to T_1 \to T_3$

(i.e; Based on Time Stamp Ordering)

Basic time stamp ordering protocol Conditions:-

- Let T_1 & T_2 are two transactions such that $\overset{old}{TS(T_1)} < \overset{young}{TS(T_2)}$

① When transaction T_1 issue a Read(A) op^n

(i)

T_1	T_2
	R(A)
R(A)	

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp ordering is $T_1 \rightarrow T_2$

If transaction T_1 is allowed to peform this R(A) operation, then also the behaviour of this schedule will be Conflict Equivalent to Serial schedule based on time stamp ordering of transactions (i.e, $T_1 \rightarrow T_2$)

∴ Transaction T_1 is allowed to peform this R(A) op^n

- Read - Read op^n will never create any problem

- If $RTS(A) > TS(T_1)$, then T_1 is allowed to peform R(A) op^n

(ii)

T_1	T_2
	W(A)
R(A)	

Rollback T_1

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp ordering is $T_1 \rightarrow T_2$

If transaction T_1 is allowed to peform this R(A) operation then the behaviour of this schedule will not be Conflict Equivalent to Serial schedule based on time stamp ordering of transactions (i.e, $T_1 \rightarrow T_2$)

∴ T_1 is not allowed to peform this R(A) op^n and we will rollback transaction T_1.

* If $WTS(A) > TS(T_1)$, then T_1 is not allowed to peform this R(A) op^n & Rollback T_1.

Basic time stamp ordering protocol Conditions:-

Let T_1 & T_2 are two transactions such that $TS(T_1) < TS(T_2)$

② When transaction T_1 issue a Write(A) op^n

(i)

T_1	T_2
⋮	R(A)
W(A)	

Rollback

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp ordering is
$T_1 \rightarrow T_2$

If transaction T_1 is allowed to perform this W(A) op^n then behaviour of schedule will not be Conflict Equivalent to serial schedule based on time stamp ordering (i.e, $T_1 \rightarrow T_2$)

∴ T_1 is not allowed to perform this W(A) op^n, hence rollback T_1

* If transaction T_1 issue a W(A) op^n and if $RTS(A) > TS(T_1)$, then Rollback $\overline{T_1}$

(ii)

T_1	T_2
⋮	W(A)
W(A)	

Rollback

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp ordering is
$T_1 \rightarrow T_2$

If transaction T_1 is allowed to perform this W(A) op^n then behaviour of schedule will not be Conflict Equivalent to serial schedule based on time stamp ordering (i.e, $T_1 \rightarrow T_2$)

∴ T_1 is not allowed to perform this W(A) op^n, hence rollback T_1

* If transaction T_1 issue a W(A) op^n and if $WTS(A) > TS(T_1)$, then Rollback $\overline{T_1}$

- ## Basic time stamp ordering Protocol Condition :-

Let T_1 & T_2 are two transactions s.t. $TS(T_1) < TS(T_2)$

ie, T_1 is older than T_2

1. Let T_1 issue Read(A) opⁿ :-

   (a) If $WTS(A) > TS(T_1)$, then rollback T_1

   (b) Otherwise T_1 is allowed to peform this R(A) opⁿ

   ∴ T_1 will peform this R(A) opⁿ, and set $RTS(A) = Max(RTS(A), TS(T_1))$

2. Let T_1 issue Write(A) opⁿ :-

   (a) If $RTS(A) > TS(T_1)$, then Rollback T_1

   & (b) If $WTS(A) > TS(T_1)$, then Rollback T_1

   (c) Otherwise, T_1 will Peform this W(A) opⁿ, and set $WTS(A) = TS(T_1)$

Q:- Consider the following schedule

T_1	T_2	T_3
R(A)		
	R(B)	
W(C)		
		R(B)
		R(C)
	W(B)	
		W(A)

Which of the following time stamp ordering of the transaction will allow the schedule to be executed by B.T.S.O.P.

(a) T.S.$(T_1, T_2, T_3) = (30, 10, 20)$

(b) T.S.$(T_1, T_2, T_3) = (30, 20, 10)$

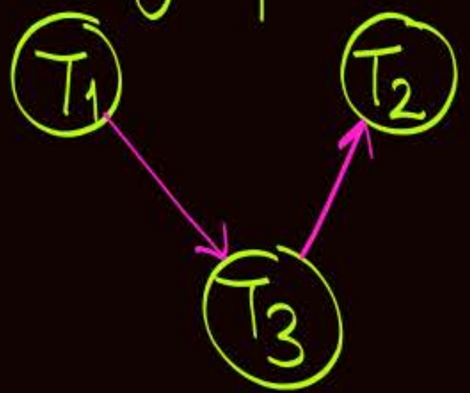
(c) T.S.$(T_1, T_2, T_3) = (10, 20, 30)$

(d). T.S.$(T_1, T_2, T_3) = (10, 30, 20)$

Q:- Consider the following schedule

T_1	T_2	T_3
R(A)		
	R(B)	
W(C)		
		R(B)
		R(C)
	W(B)	
		W(A)

Precedence graph



⇒ Acyclic Precedence graph of Schedule is Conflict Serializable schedule and Conflict Equivalent serial schedule is $T_1 \rightarrow T_3 \rightarrow T_2$

Hence this schedule will be allowed by B.T.S.O.P. if and only if Time stamps of the transaction are w.r.t serial schedule $T_1 \rightarrow T_3 \rightarrow T_2$ i.e. $TS(T_1) < TS(T_3) < TS(T_2)$

⇓

∴ option D is true

(a) $T.S.(T_1, T_2, T_3) = (30, 10, 20)$

(b) $T.S.(T_1, T_2, T_3) = (30, 20, 10)$

(c) $T.S.(T_1, T_2, T_3) = (10, 20, 30)$

✓ (d). $T.S.(T_1, T_2, T_3) = (\underline{10}, \underline{30}, \underline{20})$

Q:- Consider the following schedule

ⓐ

30	10	20
T_1	T_2	T_3
R(A) RTS(A)=30		
	R(B) RTS(B)=10	
W(C) WTS(C)=30		
		WTS(B)=0 < TS(T_3) R(B) ∴ Allowed RTS(B)=20
		WTS(C)=30>TS(T_3) R(C) → ∴ Not allowed & Rollback T_3
	W(B)	
		W(A)

Which of the following time stamp ordering of the transaction will allow the schedule to be executed by B.T.S.O.P.

ⓐ T.S.(T_1, T_2, T_3) = (30, 10, 20)

ⓑ T.S.(T_1, T_2, T_3) = (30, 20, 10)

ⓒ T.S.(T_1, T_2, T_3) = (10, 20, 30)

ⓓ. T.S.(T_1, T_2, T_3) = (10, 30, 20)

Q:- Consider the following schedule

(b)

30 T_1	20 T_2	10 T_3
R(A)		
	R(B)	
W(C) WTS(C)=30		
		R(B)
		R(C) ∴ Not allowed Rollback T_3 WTS(C)=30 > TS(T_3)
	W(B)	
		W(A)

Which of the following time stamp ordering of the transaction will allow the schedule to be executed by B.T.S.O.P.

(a) T.S. $(T_1, T_2, T_3) = (30, 10, 20)$

(b) T.S. $(T_1, T_2, T_3) = (30, 20, 10)$

(c) T.S. $(T_1, T_2, T_3) = (10, 20, 30)$

(d). T.S. $(T_1, T_2, T_3) = (10, 30, 20)$

Q:- Consider the following schedule

ⓒ

10	20	30
T_1	T_2	T_3
✓ R(A)		
	R(B) ✓	
✓ W(C) WTS(C)=10		
		R(B) ✓ WTS(C)=10<TS(T_3)
		R(C) ✓ ∴ allowed RTS(C) = 30
	RTS(B)=30>TS(T_2) ↑W(B) ∴ Not allowed Rollback T_2	
		W(A)

Which of the following time stamp ordering of the transaction will allow the schedule to be executed by B.T.S.O.P.

ⓐ T.S.(T_1, T_2, T_3) = (30, 10, 20)

ⓑ T.S.(T_1, T_2, T_3) = (30, 20, 10)

ⓒ T.S.(T_1, T_2, T_3) = (10, 20, 30)

ⓓ. T.S.(T_1, T_2, T_3) = (10, 30, 20)

Q:- Consider the following schedule

ⓐ 10	30	20
T_1	T_2	T_3
✓R(A)		
	R(B)✓	
✓W(C)		
		R(B)✓
		WTS(C)=10<TS(T_3) R(C)✓∴ Allowed

	WTS(B)=0
	RTS(B)=30≯TS(T_2)
	W(B)
		∴ Allowed

		WTS(A)=0 RTS(A)=10<TS(T_3) ✓W(A) -∴ Allowed

Which of the following time stamp ordering of the transaction will allow the schedule to be executed by B.T.S.O.P.

ⓐ T.S.(T_1, T_2, T_3) = (30, 10, 20)

ⓑ T.S.(T_1, T_2, T_3) = (30, 20, 10)

ⓒ T.S.(T_1, T_2, T_3) = (10, 20, 30)

ⓓ. T.S.(T_1, T_2, T_3) = (10, 30, 20)

Note :- ① A schedule S is allowed to execute using B.T.S.O.P. if and only if schedule S is a Conflict serializable schedule and Conflict equivalent serial schedule is according to the Time stamp ordering of the transactions

② If schedule is not a Conflict serializable schedule then it will never be allowed to execute using B.T.S.O.P.

③ If schedule is a C.S.S. but conflict equivalent serial schedule is not as per time stamp ordering of the transactions then schedule is not allowed by B.T.S.O.P.

- B.T.S.O.P Covers only conflict serializable schedules, but it does not cover the schedules that are view serializable but not conflict serializable,

∴ We define time stamp ordering protocol with "Thomas write rule"

Topic : Time stamp ordering protocol with Thomas write rule

Let T_1 & T_2 are two transactions such that $TS(T_1) < TS(T_2)$

Case ① When transaction T_1 issue a Read(A) opn

(i)

T_1	T_2
	R(A)
R(A)	

(ii)

T_1	T_2
	W(A)
R(A)	

These three Case will be exactly same as B.T.S.O.P.

Case② When transaction T_1 issue a Write (A) opn

(i)

T_1	T_2
	R(A)
W(A)	

(ii)

T_1	T_2
	W(A)
W(A)	

This case will be different from B.T.S.O.P.

Time stamp ordering protocol with Thomas write Rule :-

Let T_1 & T_2 are two transactions such that $TS(T_1) < TS(T_2)$

① When transaction T_1 issue a Read(A) op^n

(i)

T_1	T_2
	R(A)
R(A)	

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp Ordering is
$T_1 \rightarrow T_2$

If T_1 is allowed to Peform this R(A) op^n then also the behaviour of the Schedule will be view equivalent to the serial schedule based on time stamp ordering of transactions. {i.e. view equivalent to $T_1 \rightarrow T_2$}

∴ T_1 is allowed to Perform this R(A) op^n

(ii)

T_1	T_2
	W(A)
R(A)	

Rollback

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp Ordering is
$T_1 \rightarrow T_2$

If T_1 is allowed to Peform this R(A) op^n then the behaviour of the Schedule will not be view equivalent to the serial schedule based on time stamp ordering of transactions. {i.e. not view equivalent to $T_1 \rightarrow T_2$}

∴ T_1 is not allowed to Peform this R(A) op^n {Rollback T_1}

If T_1 issue a R(A) op^n and $WTS(A) > TS(T_1)$, then Rollback T_1

Time stamp ordering protocol with Thomas write Rule :-

Let T_1 & T_2 are two transactions such that $TS(T_1) < TS(T_2)$

② When transaction T_1 issue a Write(A) op^n

(i)

T_1	T_2
	R(A)
W(A)	

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time stamp ordering is $T_1 \rightarrow T_2$

If transaction T_1 is allowed to peform this W(A) op^n, then behaviour of schedule will not be view equivalent to serial schedule $T_1 \rightarrow T_2$,

∴ T_1 is not allowed to perform this W(A) op^n and rollback T_1

* If T_1 issue A W(A) op^n and $RTS(A) > TS(T_1)$, then Rollback T_1

Time stamp ordering protocol with Thomas write Rule :-

Let T_1 & T_2 are two transactions such that $TS(T_1) < TS(T_2)$

② When transaction T_1 issue a Write(A) op^n

(i)

T_1	T_2
	R(A)
W(A)	

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time stamp ordering is $T_1 \rightarrow T_2$

{ Available at Previous slide }

(ii)

T_1	T_2
	W(A)
W(A)	

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time stamp ordering is $T_1 \rightarrow T_2$

⟹ If we ignore(skip) the W(A) op^n of transaction T_1 then schedule becomes

T_1	T_2
	W(A)

Serial schedule based on time stamp ordering of transactions is

$T_1 \rightarrow T_2$

T_1	T_2
W(A)	
	W(A)

If transaction T_1 is allowed to Perform this W(A) op^n then behaviour of schedule will not be view equivalent to serial schedule as per time stamp ordering (i.e. $T_1 \rightarrow T_2$)

But if we ignore (skip) this W(A) op^n of transaction T_1 then schedule produced will be view equivalent to the serial schedule as per time stamp ordering of transaction.

∴ In time stamp ordering Protocol with thomas write Rule
If transaction T_1 issue a W(A) op^n and
$WTS(A) > TS(T_1)$, then ignore (skip) this W(A) op^n of transaction T_1
and Continue with other operation of transaction T_1

Time . stamp ordering Protocol with Thomas Write Rule :-

Let T_1 & T_2 are two transactions s.t. $TS(T_1) < TS(T_2)$

ie, T_1 is older than T_2

① Let T_1 issue Read(A) op^n :-

ⓐ If $WTS(A) > TS(T_1)$, then Rollback T_1

ⓑ otherwise T_1 will peform that R(A) op^n, and set $RTS(A) = Max(RTS(A), TS(T_1))$

② Let T_1 issue Write(A) op^n :-

ⓐ If $RTS(A) > TS(T_1)$, then Rollback T_1

And ⓑ If $WTS(A) > TS(T_1)$, then skip(ignore) this W(A) op^n of transaction T_1 and Continue with remaining operation

ⓒ Otherwise, Peform this W(A) op^n of transaction T_1 And set. $WTS(A) = TS(T_1)$

Q. Check whether the Schedule is allowed to Execute using BTSOP or not

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

⇒ Not a C.S.S.

∴ Not allowed by BTSOP.

Q. Check whether the schedule is allowed to Execute using Time stamp ordering protocol with Thomas write Rule or not If yes then what will be the schedule produced

30 T1	10 T2	20 T3
	R(A)	
		W(A)
	W(A)	
W(A)		
		R(B)

⇒

30 T1	10 T2	20 T3
	R(A) ✓	
		W(A) ✓
	~~W(A)~~ ← skip	
✓ W(A)		
		R(B) ✓

Allowed by Thomas write Rule

⇒

Schedule produced is

T1	T2	T3
	R(A)	
		W(A)
W(A)		
		R(B)

Schedule Produced by Thomas write Rule is view Equivalent to Serial schedule as per time stamp ordering

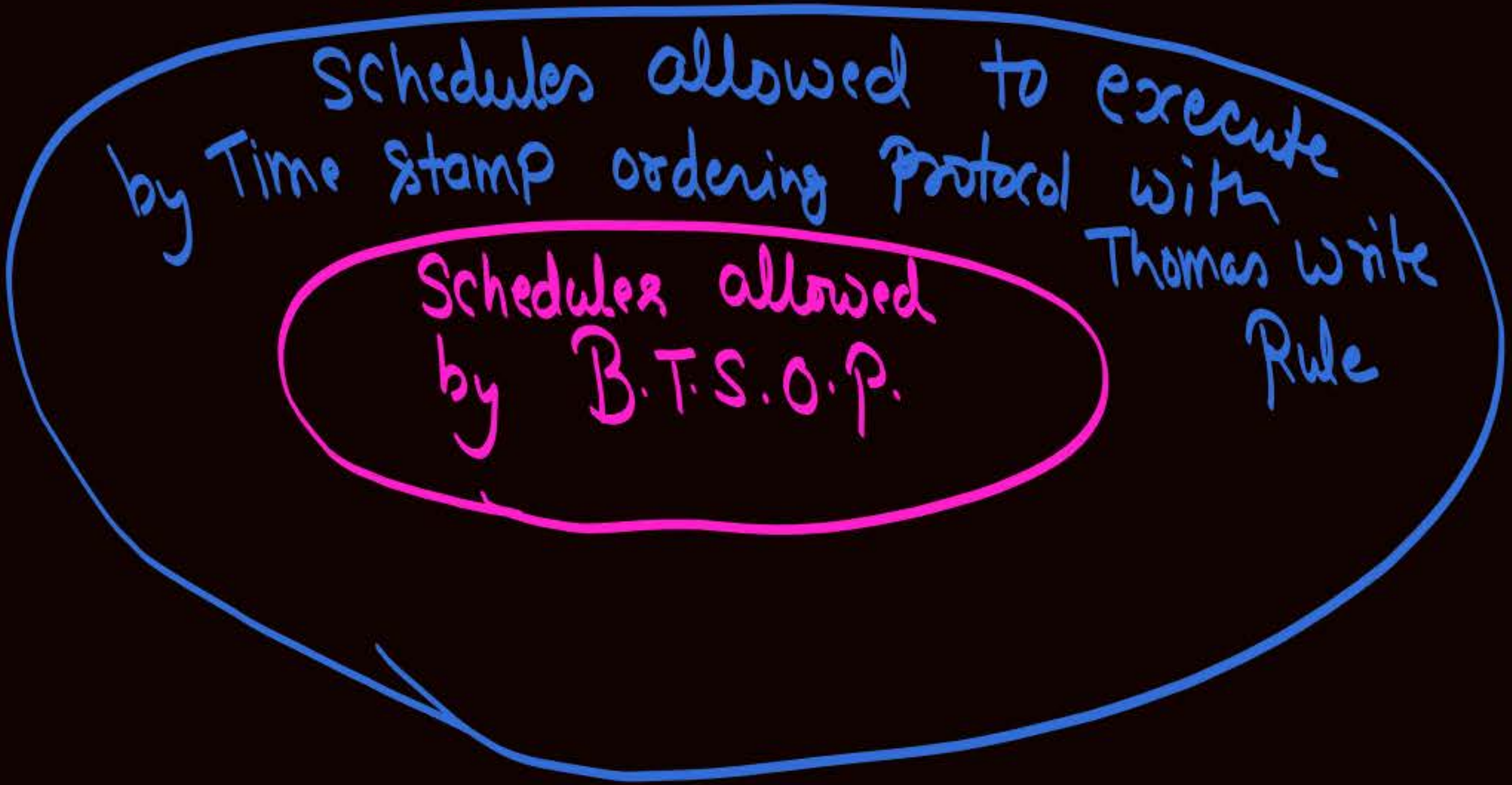
✓

Serial schedule as per time stamp ordering
T2 → T3 → T1

T1	T2	T3
	R(A)	
	W(A)	
		W(A)
		R(B)
W(A		

Notes :-

① Every schedule which is allowed by B.T.S.O.P. is allowed to execute using Time stamp ordering Protocol with Thomas write Rule, But Converse of the statement need not be true


Schedules allowed to execute by Time stamp ordering Protocol with Thomas write Rule
Schedules allowed by B.T.S.O.P.

Note:- ②

If A schedule is allowed to execute using Time Stamp ordering protocol with Thomas write Rule, then schedule produced by Time stamp ordering protocol with Thomas write Rule will be view equivalent to the serial schedule based on time stamp ordering of the transactions.

Note :- ③ Both B.T.S.O.P & Time stamp ordering protocol with Thomas write rule are free from deadlock, but starvation is possible

④ Schedules allowed by B.T.S.O.P and/or Time stamp ordering protocol with Thomas write rule may suffer from irrecoverability, cascading rollback and lost-update problem, To solve that problem we can implement Strict time stamp ordering protocols.

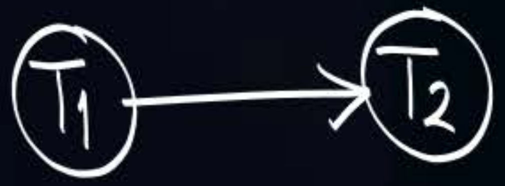
Topic : Deadlock prevention algorithms

{For lock based Protocols}

→ We can use the Concept of time stamps assigned by DBMS to prevent the deadlock in lock based Concurreny Control Protocols.

If dependency graph is Cyclic, then deadlock may be possible ⟹

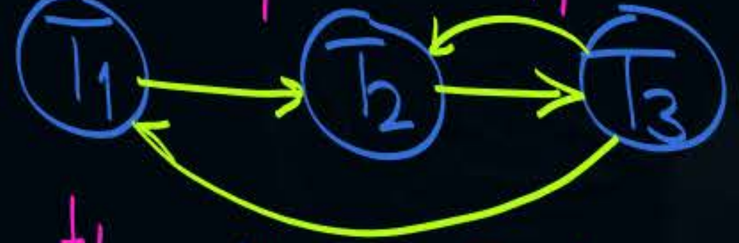
Dependeny graph ↓



it represent that T_1 requires the resource held by T_2

⟹

If dependeny graph is of the type



then it is Cyclic & Deadlock may be Possible

⟹ If we Can Prevent from the formation of Cycle in the dependency graph, then We can prevent from deadlock.

Topic : Deadlock prevention algorithms

There are two algorithms (Protocols) that can be used to prevent the the formation of Cycles in the dependency graphs., and hence can be used to prevent the deadlock in lock based protocols.

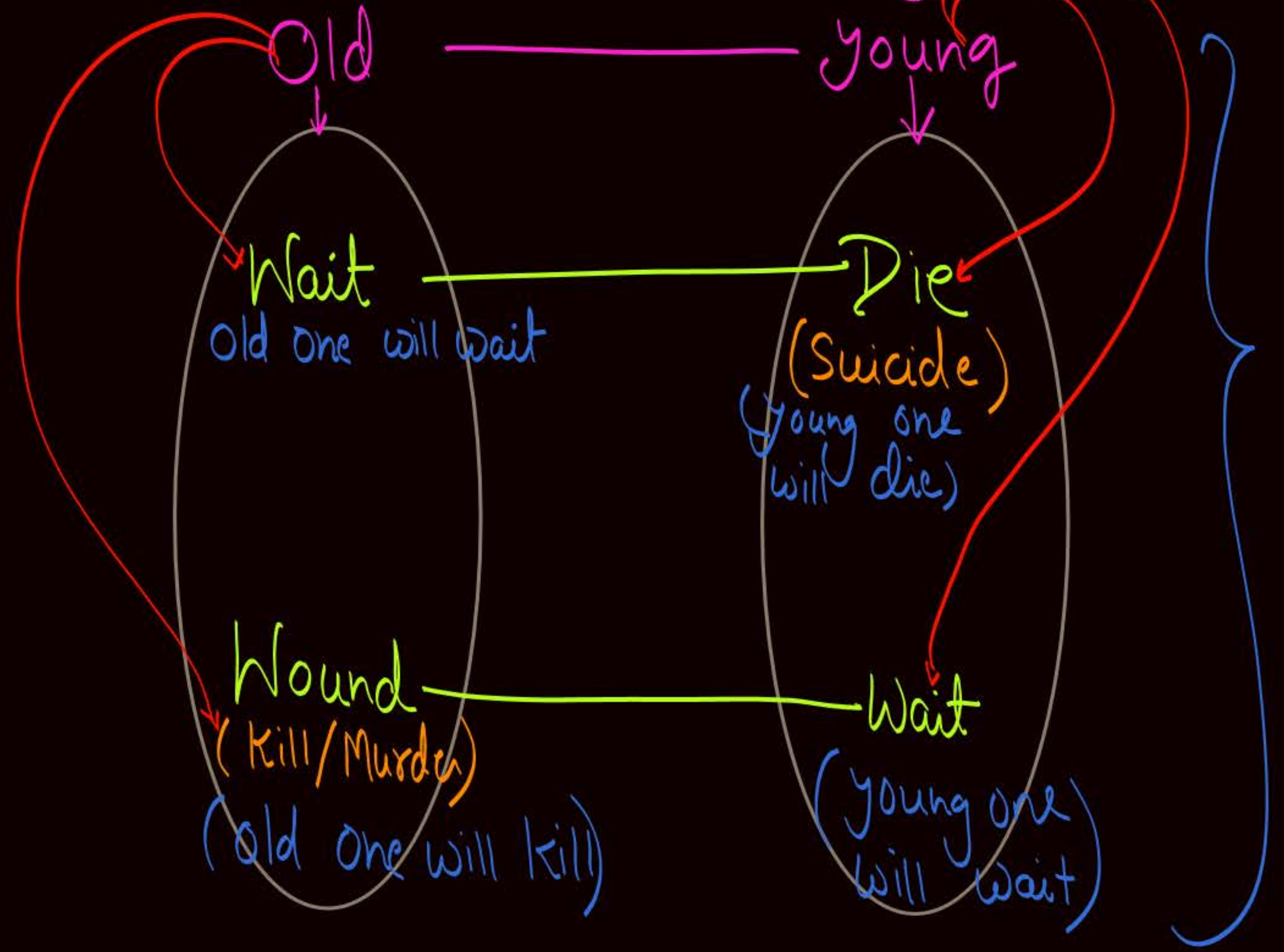
Those algorithm are

① Wait - die protocol

② Wound- wait protocol.

These names are w.r.t. Old-young Pair.

1



Old

Young

Wait

Old one will wait

Die

(Suicide)

(Young one will die)

Wound

(Kill/Murder)

(Old one will kill)

Wait

(young one will wait)

Rollback will always be of "young" transaction

Topic : Wait-die algorithms

(old) (young)

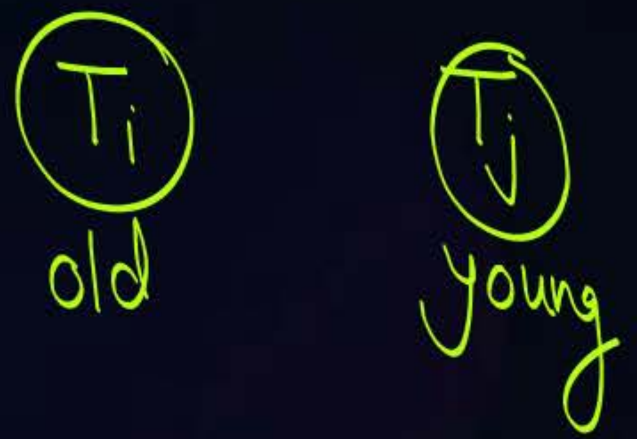
old one will wait — young one will die

If old transaction requires the resource held by young transaction then old transaction is allowed to wait.

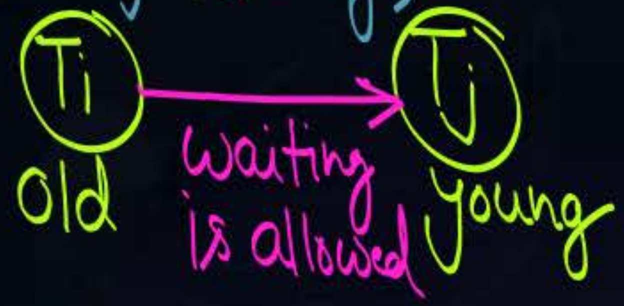
If young transaction requires the resource held by old transaction then young transaction will rollback (die)

Consider two transaction T_i & T_j such that $TS(T_i) < TS(T_j)$

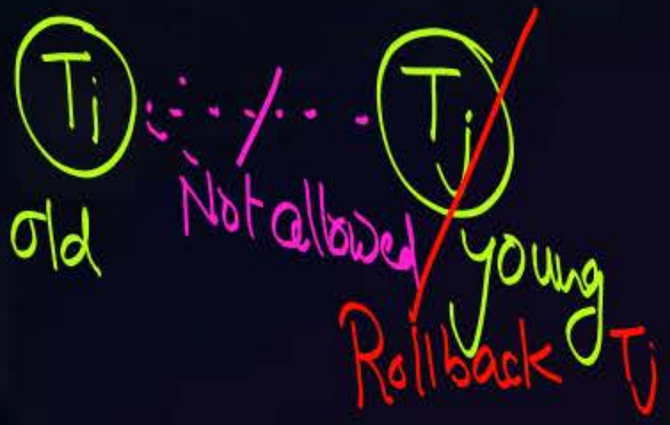
i.e.



① If T_i (old) requires the resource held by T_j (young) then T_i is allowed to wait for T_j i.e.



② If T_j (young) requires the resource held by T_i (old), then rollback T_j (young) i.e.



Topic : Wound-wait algorithms

(old) (young)

old one will wound (kill) the young one

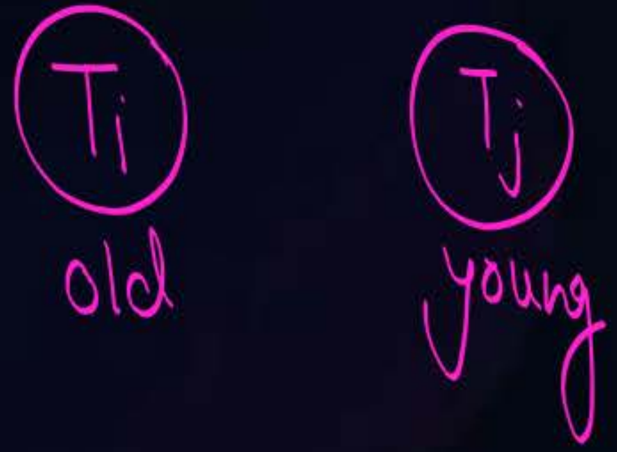
young one will wait

If old transaction requires the resource held by young transaction then old one will wound (kill) the young transaction

Consider two transactions T_i & T_j such that

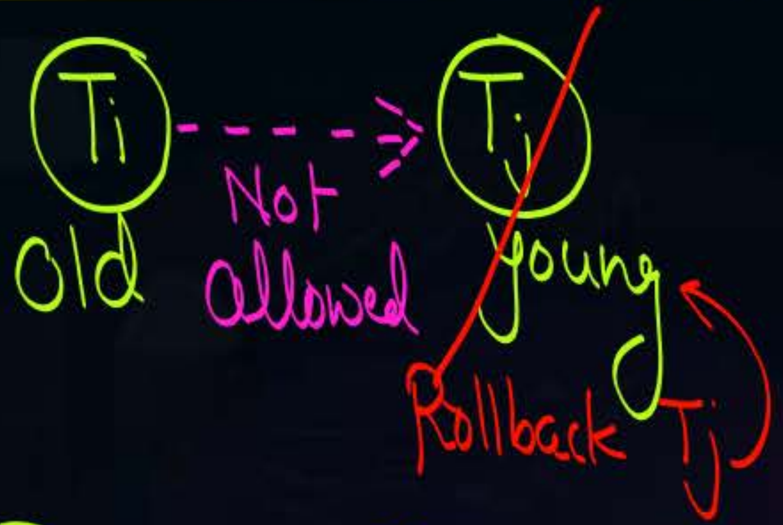
$$TS(T_i) < TS(T_j)$$

i.e.



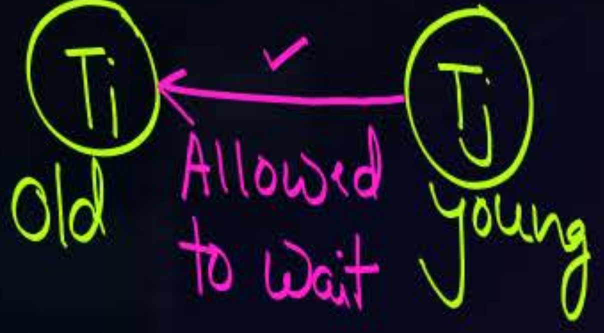
① If T_i(old) requires the resource held by T_j(young) then T_i will Kill (wound) T_j, (Rollback T_j)

i.e.



② If T_j(young) requires the resource held by T_i(old) then T_j(young one) is allowed to wait

i.e,



Note: Using wait-die & Wound-wait protocols We can prevent from deadlock in lock based Concurrency Control protocols., But starvation is still possible.

2 mins Summary

- ✓ Topic — Basic time stamp ordering protocol
- ✓ Topic — Time stamp ordering protocol with Thomas write rule
- ✓ Topic — Wait-die protocol
- ✓ Topic — Wound-wait protocol

Slide

THANK - YOU

Slide