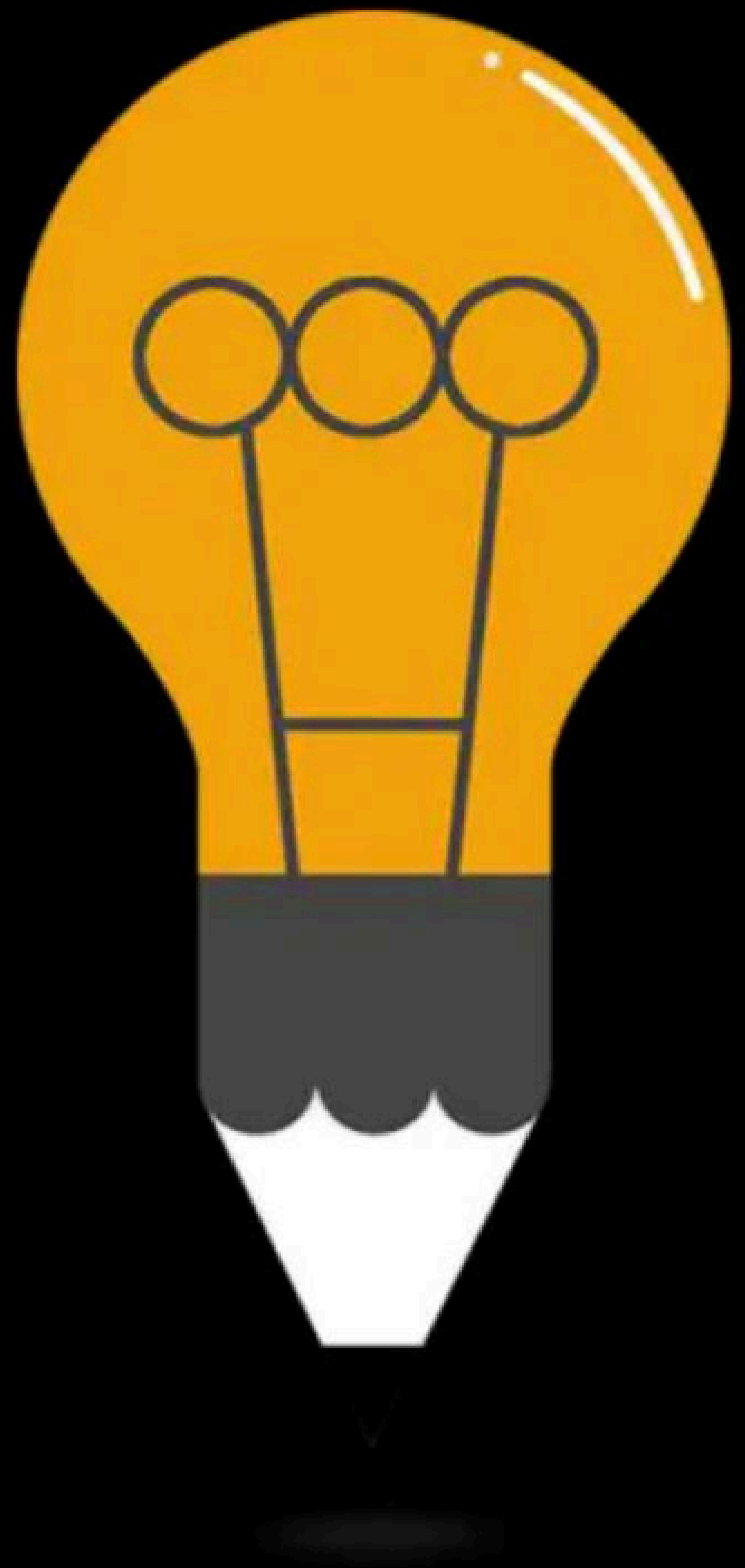
Doubt Clearing Session

Complete Course on Database Management System

Vishvadeep Gothi • Lesson 28 • July 25, 2023



DBMS

Doubts & Timestamp based protocols

By: Vishvadeep Gothi

View Serializability

T1	T2	T3
R(X)		
	R(Y)	
		W(Y)
W(Y)		
	W(X)	

View Serializability

T1	T2	T3
R(X)		
	W(X)	
W(X)		
		W(X)

view serializable

T1 → T2 → T3

View Serializability

T1	T2	T3
R(X)		
	R(Y)	
W(Z)		
		W(Z)
		R(Y)
	W(Y)	

$T_1 \rightarrow T_3$

$T_3 \rightarrow T_2$

view serializable:-

$T_1 \rightarrow T_3 \rightarrow T_2$

View Serializability

T1	T2	T3
W(X)		
	R(X)	
		W(X)
		W(Y)
	R(Y)	
W(Y)		
	W(X)	

not view serializable

Timestamp

Younger vs Older transaction

TS(T1) < TS(T2)

T2 younger

T1 older

Deadlock Prevention

1. Wait_Die
2. Wait_Wound

unacademy

Deadlock Prevention

Assume 2 transactions T_i and T_j. T_i tries to acquire lock on a database item x, which is already locked by T_j.

T_i	T_j
	lock(x)
lock(x)	

$T_i \xrightarrow{\text{Tries to acquire}} x \xrightarrow{\text{acquired}} T_j$

Deadlock Prevention

Assume 2 transactions T_i and T_j. T_i tries to acquire lock on a database item x, which is already locked by T_j.

Wait_Die: An older transaction is allowed to wait for a younger transaction, whereas a younger transaction requesting an item held by an older transaction is aborted and restarted with same timestamp.

old ⇒ wait

young ⇒ die

$TS(T_i) < TS(T_j) \Rightarrow T_i$ waits

$TS(T_i) > TS(T_j) \Rightarrow T_i$ is aborts & restarted with same t.s.

Deadlock Prevention

Assume 2 transactions T_i and T_j. T_i tries to acquire lock on a database item x, which is already locked by T_j.

Wait_Wound: A younger transaction is allowed to wait for an older one, whereas if an older transaction requests an item held by the younger transaction, we preempt the younger transaction by aborting it.

younger ⇒ wait

older ⇒ wound

$TS(T_i) < TS(T_j) \Rightarrow$ abort T_j, T_i acquires lock

$TS(T_i) > TS(T_j) \Rightarrow$ T_i waits

Starvation

wait - die

⇓

older transactions may starve

wait - wound

⇓

no starvation

for older but younger may starve if old transactions keeps locks for indefinite time

Question

Assume that T1 requests a lock held by t2. Consider the following table which shows the actions taken for wait_die and wait_wount schemes:

	Wait_Die	Wait_Wound
T1 is younger than T2	W	X
T1 is older than T2	Y	Z

What will be the correct status of T1 andT2 at W, X, Y, and Z respectively?

T1

W :- T1 aborted (die) T2 uses lock	X :- T1 waits T2 uses lock
Y :- T1 waits T2 uses lock	Z :- T2 aborted (wound) T1 uses lock

Question GATE-2017

In a database system, unique timestamps are assigned to each transaction using Lamport's logical clock. Let $TS(T_1)$ and $TS(T_2)$ be the timestamps of transactions T_1 and T_2 respectively. Besides, T_1 holds a lock on the resource R, and T_2 has requested a conflicting lock on the same resource R. The following algorithm is used to prevent deadlocks in the database system assuming that a killed transaction is restarted with the same timestamp.

if $TS(T_2) < TS(T_1)$ then

T_1 is killed

else T_2 waits.

TS (T1) and TS (T2)

T1 $\leftarrow$ lock $\leftarrow$ T2

Assume any transaction that is not killed terminates eventually. Which of the following is TRUE about the database system that uses the above algorithm to prevent deadlocks?

A. The database system is both deadlock-free and starvation-free.
B. The database system is deadlock-free, but not starvation-free.
C. The database system is starvation-free, but not deadlock-free.
D. The database system is neither deadlock-free nor starvation-free.

unacademy

Timestamp

Read Timestamp(A): Youngest transaction who read A

Write Timestamp(A): Youngest transaction who write A

Initially read & write timestamps are zero.

$TS(T_1) = 1$ | $TS(T_2) = 2$

T_1	T_2
$R(x)$	
	$R(x)$
	$W(x)$
$W(x)$	
$W(y)$	
	$R(y)$

$R\_TS(x) = \cancel{0}\ \cancel{1}\ 2$

$W\_TS(x) = \cancel{0}\ 2$

$R\_TS(y) = 2$

$W\_TS(y) = 1$

Basic Timestamp Algorithm

Whenever a Transaction T issues a W_item(X) operation, check the following conditions:

- If R_TS(X) > TS(T) or if W_TS(X) > TS(T), then abort and rollback T and reject the operation.
- Else execute W_item(X) operation of T and set W_TS(X) to TS(T).

Whenever a Transaction T issues a R_item(X) operation, check the following conditions:

- If W_TS(X) > TS(T), then abort and rollback T and reject the operation, else
- If W_TS(X) <= TS(T), then execute the R_item(X) operation of T and set R_TS(X) to the larger of TS(T) and current R_TS(X).

$$R\_TS(x) = \max\left(R\_TS(x), TS(T)\right)$$

Basic Timestamp Algorithm

Restarted transaction gets a younger timestamp

In whichever order transactions arrive, that order only all transactions should follow to run.

Assume 3 transactions T1, T2, T3

$TS(T1) < TS(T2) < TS(T3)$

$T1 \longrightarrow T2 \longrightarrow T3$ allowed

Basic Timestamp Algorithm

unacademy

$TS(T1) = 1, \; TS(T2) = 2$

T1	T2
	R(X) or W(X)
W(X)	

abort & rollback

$R\_TS(x) = \cancel{0}\ 2$

$W\_TS(x) = \cancel{0}\ 2$

T1	T2
	W(X)
R(X)	

Reject

Rollback

$R\_TS(x) = 0$

$W\_TS(x) = \cancel{0}\ 2$

T1	T2
R(x) or W(x)	
	W(x)

allowed

$R\_TS(x) = 1$

$W\_TS(x) = \cancel{1}\ 2$

T1	T2
W(x)	
	R(x)

allowed

$R\_TS(x) = \cancel{0}\ 2$

$W\_TS(x) = \cancel{0}\ 1$

unacademy

ex:-

T1	T2	T3
W(x)		
	R(x)	
		R(x)
	R(x)	

$R\_TS(x) = \cancel{\emptyset}\ \cancel{2}\ 3$

$W\_TS(x) = \cancel{\emptyset}\ 1$

all allowed

unacademy

Basic Timestamp Algorithm

T1	T2	T3
		R(X)
	R(X)	
R(X)		
W(X)		

reject

abort & rollback

$R\_TS(x) = \cancel{0}\ 3$

$W\_TS(x) = 0$

$TS(T1) = 1$

$TS(T2) = 2$

$TS(T3) = 3$

T2, T3 completed

T1 ⇒ aborted

unacademy

Question

abort

T1	T2	T3	T4
			R(X)
		R(X)	
	R(X)		
	W(X)		
		W(X)	
			W(X)
R(X)			

abort

abort

$R\_TS(x) = \cancel{\phi}\ 4$

$W-TS(x) = \cancel{\phi}\ 4$

only T4 completed

Basic Timestamp Algorithm

T1	T2	T3
R(X)		
	W(X)	
W(X)		
		W(X)

abort & rollback

$R\text{-}TS(x) = \cancel{0}\ \cancel{1}\ 0$

$W\text{-}TS(x) = \cancel{0}\ \cancel{2}\ 3$

$x = \cancel{15}\ \cancel{20}\ \cancel{25}\ 30$

T1	T2	T3
R(x)		
w(x)		
	w(x)	
		w(x)

$x = \cancel{15}\ \cancel{20}\ \cancel{25}\ 30$

T1	T2
$R(x)$	
	$W(x) = 15$
$W(x) = 10$	

$x = \cancel{2}\ 15$

serial
for sequence $T1 \rightarrow T2$
final value of x must be the value written by T_2.

Thomas Write Rule

- Read is same as basic timestamp rules
- Write(A) in transaction T:
  - If R_TS(A)>TS(T) then abort T, rollback and restart T
  - Else If W_TS(A)> then skip write operation
  - Else perform Write(A) of T and update W_TS(A)=TS(T)

unacademy

T_1	T_2	T_3
✓ $R(x)$		
	✓ $W(x)$	
$W(x)$		
		✓ $W(x)$

skip

$R\_TS(x) = \phi$ $\underline{1}$

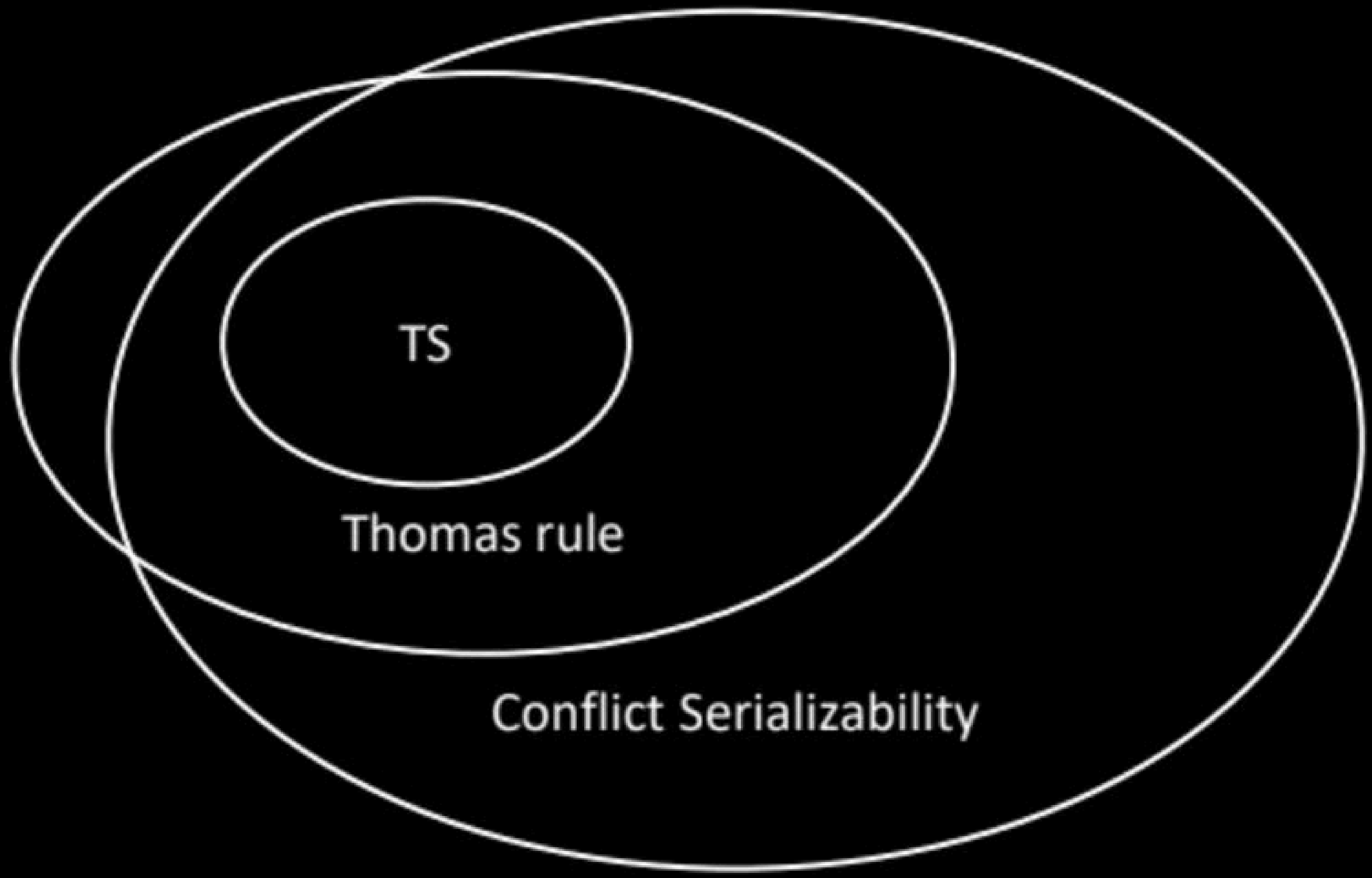
$W\_TS(x) = \phi$ $\cancel{2}$ 3

Question

T1	T2	T3	T4
	R(X)		
		R(X)	
			W(X)
		W(X)	
W(X)			

Timestamp

- Basic TS allows conflict serializable schedules
- Thomas rule allows more than conflict serializable schedules



No Need to Study

- Multiversion Protocol
- Multigranularity Protocol

Happy Learning.!