

CSE 444: Database Internals

Section 7: Concurrency

Today

- Timestamp-based Concurrency Control
- Multiversion Concurrency Control

Problem 1: Timestamp-based Concurrency Control

Timestamp-based Concurrency Control

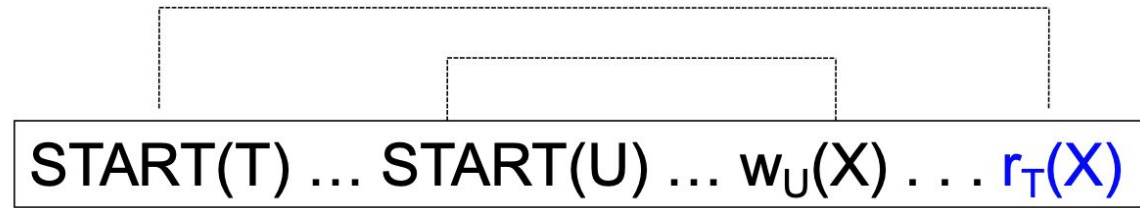
- Some transaction, T .
- Some element (tuple/page), X .
- $TS(T)$ - timestamp for transaction T
 - Stays constant for all of T 's operations
- $WT(X)$ – latest write timestamp for X
 - Set $WT(X) = TS(T)$
- $RT(X)$ – latest read timestamp for X
 - Set $RT(X) = TS(T)$
- $C(X)$ – X 's value has been committed
 - 1 if true, 0 if not

Timestamp-based Concurrency Control

- **Actions for transaction T**
 - **Grant** a read/write request for a transaction
 - **Abort** (in case T violates physical reality – late actions)
 - **Delay** (make the Grant or Abort decision later)
 - When writing, the change is always tentative until we decide to commit. For this, we use a commit bit C to keep track if the transaction that last wrote X has committed
 - **Ignore *Thomas Write Rule*** – ignore outdated writes

Timestamp-based Concurrency Control - Four Rules

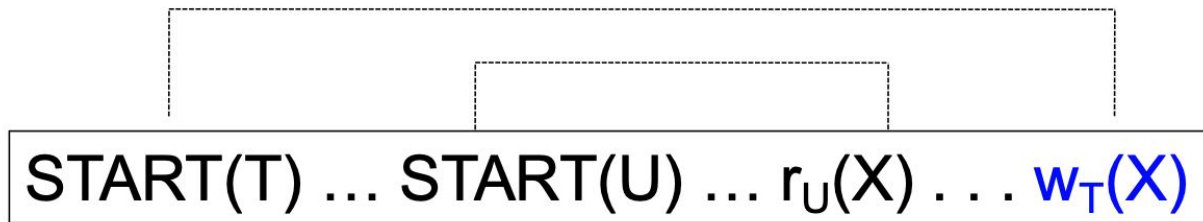
- **Rule 1:** **Read** request on **X** by **T**



- $TS(T) < WT(X)$, **abort**, (read too late)
- $TS(T) \geq WT(X)$, physically realizable
 - If $C = 1$, **grant**, update $RT(X)$
 - If $C = 0$, **delay** T

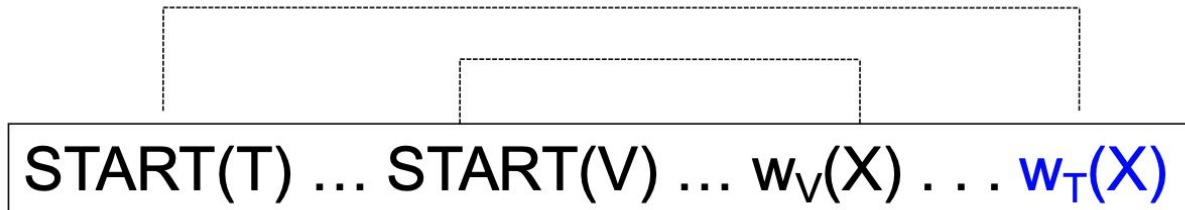
Timestamp-based Concurrency Control - Four Rules

- Rule 2: **Write** request on **X** by **T**



- $TS(T) < RT(X)$ (write too late)
 - **Abort**

- $TS(T) \geq RT(X)$, physically realizable
 - $TS(T) \geq WT(X)$
 - then **grant**, update $WT(X)$, set $C = 0$ (as it's not committed yet)



- $TS(T) < WT(X)$
 - If $C = 1$, **don't write X at all!** (*Thomas Write Rule* – ignore outdated writes)
 - If $C = 0$, **delay**

Timestamp-based Concurrency Control - Four Rules

- **Rule 3: Commit** request by **T**
 - Set $C = 1$ for all **X** written by **T**
 - Allow waiting transactions to proceed
- **Rule 4: Abort** transaction **T**
 - Check if the waiting transactions can proceed now.

Summary

Transaction wants to READ element X

If $WT(X) > TS(T)$ then ROLLBACK

Else If $C(X) = \text{false}$, then WAIT

Else READ and update $RT(X)$ to larger of $TS(T)$ or $RT(X)$

Transaction wants to WRITE element X

If $RT(X) > TS(T)$ then ROLLBACK

Else if $WT(X) > TS(T)$

Then If $C(X) = \text{false}$ then WAIT

else IGNORE write (**Thomas Write Rule**)

Otherwise, WRITE, and update $WT(X)=TS(T)$, $C(X)=\text{false}$

Timestamp-based Concurrency Control

Two transactions get started.

- $\text{Start}(T_1) \rightarrow \text{Start}(T_2)$

Timestamp-based Concurrency Control

What will happen at the last request?

- $\text{Start}(T_1) \rightarrow \text{Start}(T_2) \rightarrow R_{T_1}(A) \rightarrow R_{T_2}(A) \rightarrow W_{T_1}(B) \rightarrow \mathbf{W_{T_2}(B)}$

Timestamp-based Concurrency Control

What will happen at the last request?

- $\text{Start}(T_1) \rightarrow \text{Start}(T_2) \rightarrow R_{T_1}(A) \rightarrow R_{T_2}(A) \rightarrow W_{T_1}(B) \rightarrow \mathbf{W_{T_2}(B)}$
– **ACCEPTED**

Timestamp-based Concurrency Control

What will happen at the last request?

- $\text{Start}(T_1) \rightarrow \text{Start}(T_2) \rightarrow R_{T_1}(A) \rightarrow R_{T_2}(A) \rightarrow W_{T_1}(B) \rightarrow \mathbf{W_{T_2}(B)}$
– **ACCEPTED**
- $\text{Start}(T_1) \rightarrow \text{Start}(T_2) \rightarrow R_{T_2}(A) \rightarrow \text{Commit}_{T_2} \rightarrow R_{T_1}(A) \rightarrow \mathbf{W_{T_1}(A)}$

Timestamp-based Concurrency Control

What will happen at the last request?

- Start(T_1) \rightarrow Start(T_2) \rightarrow $R_{T_1}(A)$ \rightarrow $R_{T_2}(A)$ \rightarrow $W_{T_1}(B)$ \rightarrow **$W_{T_2}(B)$**
– **ACCEPTED**

- Start(T_1) \rightarrow Start(T_2) \rightarrow $R_{T_2}(A)$ \rightarrow Commit $_{T_2}$ \rightarrow $R_{T_1}(A)$ \rightarrow **$W_{T_1}(A)$**
– **ABORT T_1** because $R_{T_2}(A)$ precedes

Problem 1: Timestamp-based Concurrency Control

T1	T2	T3	T4	X	Y	Z
1	2	3	4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$			WT=2, C=0		
$W_1(X)$: abort						
		$W_3(Y)$			WT=3, C=0	
	$W_2(Y)$: delay					

1. Physically realizable:

$TS(T_2) \geq RT(Y)$ although $TS(T_2) < WT(Y)$

2. We could not apply Thomas' write rule (**ignore $W_2(Y)$**) since $C=0$

T1	T2	T3	T4	X	Y	Z
1	2	3	4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$			WT=2, C=0		
$W_1(X)$: abort						
		$W_3(Y)$			WT=3, C=0	
	$W_2(Y)$: delay					
		C_3			C=1	

A later write by T_3 has been committed!

T1	T2	T3	T4	X	Y	Z
1	2	3	4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
R ₁ (X)				RT=1		
	R ₂ (X)			RT=2		
	W ₂ (X)			WT=2, C=0		
W ₁ (X): abort						
		W ₃ (Y)			WT=3, C=0	
	W ₂ (Y): delay					
		C ₃			C=1	
	Ignore W₂(Y) and proceed					

T1	T2	T3	T4	X	Y	Z
1	2	3	4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
	Ignore $W_2(Y)$ and proceed					
			$W_4(Z)$			

T1	T2	T3	T4	X	Y	Z
1	2	3	4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
	Ignore $W_2(Y)$ and proceed					
			$W_4(Z)$			WT=4, C = 0

1. Physically realizable:

$$TS(T_4) \geq RT(Z) \text{ and } TS(T_4) \geq WT(Z)$$

2. Update WT and C (not committed yet)

T1	T2	T3	T4	X	Y	Z
1	2	3	4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
	Ignore $W_2(Y)$ and proceed					
			$W_4(Z)$			WT=4, C = 0
			C_4			C=1

T1	T2	T3	T4	X	Y	Z
1	2	3	4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
	Ignore $W_2(Y)$ and proceed					
			$W_4(Z)$			WT=4, C = 0
			C_4			C=1
	$R_2(Z)$					

1. **NOT** Physically realizable:

$$TS(T_2) < WT(Z)$$

Abort/rollback

T4	X	Y	Z
4	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1	RT = 0, WT = 0, C = 1
$W_4(Z)$			WT=4, C = 0
C_4			C=1

and proceed

$R_2(Z)$: abort

Timestamp-based Concurrency Control

Questions?

Multiversion Concurrency Control

- Maintains **old** versions of database elements in addition the current version in the database itself.
- The idea is to allow reads that would otherwise result in an abort (as the current version was written by future transaction)

Problem with Timestamp-Based Scheduling

T1	T2	T3	T4	A
150	200	175	225	RT = 0 WT = 0
$R_1(A)$				RT = 150
$W_1(A)$				WT = 150
	$R_2(A)$			RT = 200
	$W_2(A)$			WT = 200
		$R_3(A)$		
		Abort		
			$R_4(A)$	RT = 225

Had to abort because
WT(A) is greater than
my own timestamp

Would have been useful if I
had access to an old version
of A (from 150)...

Multiversion Timestamps

T1	T2	T3	T4	A ₀	A ₁₅₀	A ₂₂₅
150	200	175	225	RT = 0 WT = 0		
R ₁ (A)				RT = 150		
W ₁ (A)					Create	
	R ₂ (A)				RT=200	
	W ₂ (A)					Create
		R ₃ (A)			RT=175	
			R ₄ (A)			RT=225

Don't have to abort

Just read a previous value of A