CSE 444: Database Internals

Section 5:
Transactions

Today

- Serializability and Conflict Serializability
  - Precedence graph
- Two-Phase Locking
  - Strict two phase locking
- Timestamp-based Concurrency Control
- Multiversion Concurrency Control

Problem 1: Serializability and Locking

- Is this schedule conflict serializab

What is
- Serializability
- Conflict Serializability?

T_0	T_1
$R_0(A)$	
$W_0(A)$	
	$R_1(A)$
	$R_1(B)$
	C_1
$R_0(B)$	
$W_0(B)$	
C_0	

Review: (Conflict) Serializable Schedule

- A schedule is ***serializable*** if it is equivalent to a serial schedule
- A schedule is ***conflict serializable*** if it can be transformed into a serial schedule by a series of swappings of adjacent non-conflicting actions

Example:

$r_1(A); w_1(A); r_2(A); w_2(A); r_1(B); w_1(B); r_2(B); w_2(B)$

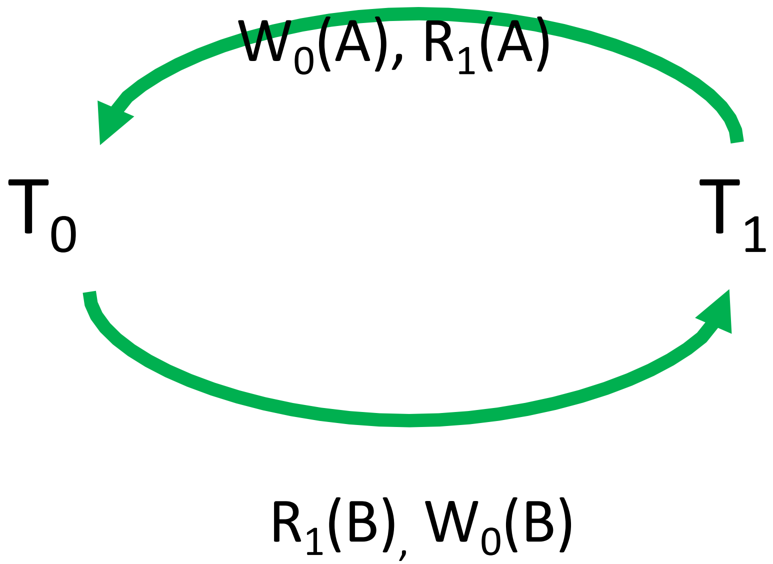
$r_1(A); w_1(A); r_1(B); w_1(B); r_2(A); w_2(A); r_2(B); w_2(B)$

Problem 1: Serializability and Locking

- Is this schedule conflict serializable?

T_0	T_1
$R_0(A)$	
$W_0(A)$	
	$R_1(A)$
	$R_1(B)$
	C_1
$R_0(B)$	
$W_0(B)$	
C_0	

- No.
- The **precedence graph** contains a cycle



- So, use 2PL ...
  - Original schedule below

T_0	T_1
$R_0(A)$	
$W_0(A)$	
	$R_1(A)$
	$R_1(B)$
	C_1
$R_0(B)$	
$W_0(B)$	
C_0	

- So, use 2PL …
  - ❑ Original schedule below

What is
- Two Phase Locking
- Strict Two Phase Locking?

T_0	T_1
$R_0(A)$	
$W_0(A)$	
	$R_1(A)$
	$R_1(B)$
	C_1
$R_0(B)$	
$W_0(B)$	
C_0	

Review: (Strict) Two Phase Locking (2PL)

The 2PL rule:

In every transaction, all lock requests must precede all unlock requests

Strict 2PL:

All locks held by a transaction are released when the transaction is completed

- Ensures that schedules are recoverable
  - Transactions commit only after all transactions whose changes they read also commit
- Avoids cascading rollbacks

- How can 2PL can ensure a conflict-serializable schedule?
  - Original schedule below

T_0	T_1
$R_0(A)$	
$W_0(A)$	
	$R_1(A)$
	$R_1(B)$
	C_1
$R_0(B)$	
$W_0(B)$	
C_0	

T_0	T_1
$L_0(A)$	
$R_0(A)$	
$W_0(A)$	
	$L_1(A)$: Block
$L_0(B)$	
$R_0(B)$	
$W_0(B)$	
$U_0(A)$	
$U_0(B)$	
C_0	
	$L_1(A)$: Granted
	$R_1(A)$
	$L_1(B)$
	$R_1(B)$
	$U_1(A)$
	$U_1(B)$
	C_1

Is this strict 2PL?

No, release locks after commit

Problem 2: Timestamp-based Concurrency Control

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

Timestamp-based Concurrency Control - Four Rules

- **Rule 1**: **Read** request on X by T
  - TS(T) < WT(X), **abort**, (read too late)
  - TS(T) >= WT(X), physically realizable
    - If C = 1, **accept**, 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) >= RT(X), physically realizable
    - TS(T) >= WT(X)
      - **accept**, update WT(X), set C = 0 (as it's not committed yet)
    - TS(T) < WT(X)
      - If C = 1, **ignore** (**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** T
  - Check if the waiting transactions can proceed now.

Timestamp-based Concurrency Control

What will happen at the last request?

- ST_1 -> ST_2 -> $R_1(A)$ -> $R_2(A)$ -> $W_1(B)$ -> $\mathbf{W_2(B)}$
- ST_1 -> ST_2 -> $R_2(A)$ -> C_2 -> $R_1(A)$ -> $\mathbf{W_1(A)}$

Timestamp-based Concurrency Control

What will happen at the last request?

- ST_1 -> ST_2 -> $R_1(A)$ -> $R_2(A)$ -> $W_1(B)$ -> $\mathbf{W_2(B)}$
  - **ACCEPTED** [no need to check C(B)]
- ST_1 -> ST_2 -> $R_2(A)$ -> C_2 -> $R_1(A)$ -> $\mathbf{W_1(A)}$
  - **ROLLED BACK** [$R_2(A)$ precedes]

Problem 2: Timestamp-based Concurrency Control

- Explain what happens when a time-stamp based concurrency control is used.
- TS_1 -> TS_2 -> TS_3 -> TS_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

- Remember!
  - Note changes to RT, WT, A and C bit for each element
  - Apply four rules

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$						

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$					

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$					

1. Physically realizable:
$TS(T_1) >= WT(X)$

2. C = 1: grant request

3. Update RT : $TS(T_1) > RT(X)$

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$					

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$					

1. Physically realizable:
$TS(T_2) >= WT(X)$

2. C = 1: grant request

3. Update WT

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$			WT=2, C=0		
$W_1(X)$						

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$			WT=2, C=0		
$W_1(X)$: **abort**						

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$			WT=2, C=0		
$W_1(X)$: **abort**						

1. **NOT** Physically realizable:
$TS(T_1) < RT(X)$

Abort/rollback

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				RT=1		
	$R_2(X)$			RT=2		
	$W_2(X)$			WT=2, C=0		
$W_1(X)$: **abort**						
		$W_3(Y)$				

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				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	
	$W_2(Y)$					

1. Physically realizable:
$TS(T_3)$ >= $RT(X)$ and $TS(T_3) >= WT(X)$

2. Update WT and C (not committed yet)

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				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**					

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				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:
$TS(T_3)$ >= $RT(X)$ although $TS(T_2) < WT(X)$
2. We could not apply Thomas' write rule **(ignore $W_2(Y)$)** since C=0

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				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				

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				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	

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				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!

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
$R_1(X)$				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 $W_2(Y)$ and proceed					

$ST_1 \rightarrow ST_2 \rightarrow ST_3 \rightarrow ST_4 \rightarrow R_1(X) \rightarrow R_2(X) \rightarrow W_2(X) \rightarrow W_1(X) \rightarrow W_3(Y) \rightarrow W_2(Y) \rightarrow C_3 \rightarrow W_4(Z) \rightarrow C_4 \rightarrow R_2(Z)$

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**					
			$W_4(Z)$			

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 ->

1. Physically realizable:
TS(T_4) >= RT(X) and TS(T_4) >= WT(X)

2. Update WT and C (not committed yet)

					Y	Z
					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

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ -> C_4 -> $R_2(Z)$

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
			C_4			C=1

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ ->C_4 ->$R_2(Z)$

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
			C_4			C=1
	$R_2(Z)$					

ST_1 -> ST_2 -> ST_3 -> ST_4 -> $R_1(X)$ -> $R_2(X)$ -> $W_2(X)$ -> $W_1(X)$ -> $W_3(Y)$ -> $W_2(Y)$ -> C_3 -> $W_4(Z)$ ->C_4 -
->$R_2(Z)$

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
	d **proceed**					
			$W_4(Z)$			WT=4, C = 0
			C_4			C=1
	$R_2(Z)$: **abort**					

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

T_1	T_2	T_3	T_4	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

T_1	T_2	T_3	T_4	A_0	A_{150}	A_{200}
150	200	175	225			
$r_1(A)$				Read		
$w_1(A)$					Create	
	$r_2(A)$				Read	
	$w_2(A)$					Create
		$r_3(A)$			Read	
			$r_4(A)$			Read