

# Concurrencia y Recuperabilidad

## Paradigma Optimista

Lic. Andrea Manna



DEPARTAMENTO  
DE COMPUTACION

2025

# Introducción

- Bibliografía: Database Systems. The Complete Book. Second Edition. Hector García-Molina, J.D. Ullman y Jennifer Widom (Capítulo 18)

- Bibliografía: Database Systems. The Complete Book. Second Edition. Hector García-Molina, J.D. Ullman y Jennifer Widom (Capítulo 18)
- Estos métodos asumen que no ocurrirá un comportamiento no serializable y actúan para reparar el problema sólo cuando ocurre una violación aparente.

- Bibliografía: Database Systems. The Complete Book. Second Edition. Hector García-Molina, J.D. Ullman y Jennifer Widom (Capítulo 18)
- Estos métodos asumen que no ocurrirá un comportamiento no serializable y actúan para reparar el problema sólo cuando ocurre una violación aparente.
- Métodos:
  - TimeStamping
  - TimeStamping Multiversion
  - Validación

# Timestamping

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

- Para generar los timestamps se puede:
  - 1 Usar el reloj del sistema.
  - 2 El *scheduler* o planificador mantiene un contador: Una transacción nueva que comienza siempre tiene un número mayor que una que comenzó antes.



- Para generar los timestamps se puede:
  - 1 Usar el reloj del sistema.
  - 2 El *scheduler* o planificador mantiene un contador: Una transacción nueva que comienza siempre tiene un número mayor que una que comenzó antes.
- El planificador debe mantener una tabla de las transacciones y sus *timestamps*.

- Para generar los timestamps se puede:
  - 1 Usar el reloj del sistema.
  - 2 El *scheduler* o planificador mantiene un contador: Una transacción nueva que comienza siempre tiene un número mayor que una que comenzó antes.
- El planificador debe mantener una tabla de las transacciones y sus *timestamps*.
- El planificador maneja la ejecución concurrente de tal manera que los timestamps determinan el **orden de serialización**

- **RT(X)**: tiempo de lectura, el timestamp más alto de una transacción que ha leído X

- **RT(X)**: tiempo de lectura, el timestamp más alto de una transacción que ha leído X
- **WT(X)**: tiempo de escritura, el timestamp más alto de una transacción que ha escrito X

- **RT(X)**: tiempo de lectura, el timestamp más alto de una transacción que ha leído X
- **WT(X)**: tiempo de escritura, el timestamp más alto de una transacción ha escrito X
- **C(X)**: bit de commit para X, es verdadero si y sólo si la transacción más reciente que escribió X ha realizado commit

- El planificador asume que el orden de llegada de las transacciones es el orden serial en que deberían parecer que se ejecutan.
- El planificador además de asignar timestamps y actualizar **RT**, **WT** y **C** para cada elemento de una transacción, debe verificar que cuando ocurre una lectura o escritura también podría haber ocurrido si cada transacción se hubiera realizado instantáneamente al momento del timestamp.

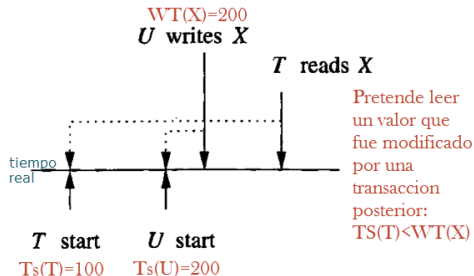
Si eso no ocurre entonces el comportamiento se denomina: **físicamente irrealizable**.

## Read too Late

- $TS(T) < WT(X)$
- Una transacción T intenta leer X pero el valor de escritura indica que X fue escrito después de que teóricamente debería haberlo leído T.

## Read too Late

- $TS(T) < WT(X)$
- Una transacción T intenta leer X pero el valor de escritura indica que X fue escrito después de que teóricamente debería haberlo leído T.



## T debe abortar



- $WT(X) < TS(T) < RT(X)$ .
- T intenta escribir pero el tiempo de lectura de X indica que alguna otra transacción debería haber leído el valor escrito por T (lee otro valor en su lugar).

## T debe abortar

**Definición:** Una lectura sucia ocurre cuando se le permite a una transacción la lectura de un elemento que ha sido modificado por otra transacción concurrente pero que todavía no ha sido cometida (commit).

U writes X

T reads X

Se deshace la escritura por el abort de U pero T ya escribió

tiempo real

U start

T start

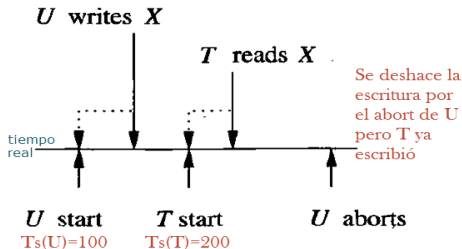
U aborts

$T_s(U)=100$

$T_s(T)=200$

## Dirty data (Lectura Sucia)

**Definición:** Una lectura sucia ocurre cuando se le permite a una transacción la lectura de un elemento que ha sido modificado por otra transacción concurrente pero que todavía no ha sido cometida (commit).



### Dirty Data

Por lo tanto, aunque no hay nada **físicamente irrealizable** sobre la lectura de X por parte de T, es mejor **retrasar** la lectura hasta que U realice el commit o aborte

# Dirty Data: Ejemplo

Transacción 1

```
/* Query 1 */  
SELECT edad FROM usuarios WHERE id = 1;  
/* Leerá 20 */
```

```
/* Query 1 */  
SELECT edad FROM usuarios WHERE id = 1;  
/* Leerá 21 */
```

Transacción 2

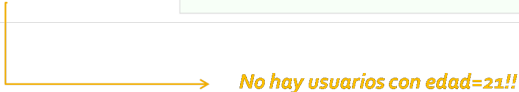
```
/* Consulta 2 */  
UPDATE usuarios SET edad = 21 WHERE id = 1;  
/* No se hace commit */
```

```
ROLLBACK;
```

usuarios

nombre	edad
José	20
Juana	25





**No hay usuarios con edad=21!!**



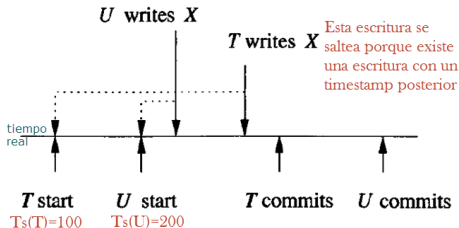
Es decir cuando  $WT(X) > TS(T)$

# Regla de escritura de Thomas

## Thomas write rule

La escritura puede “saltarse” cuando ya existe una escritura de una transacción con un timestamp de mayor valor.

Es decir cuando  $WT(X) > TS(T)$

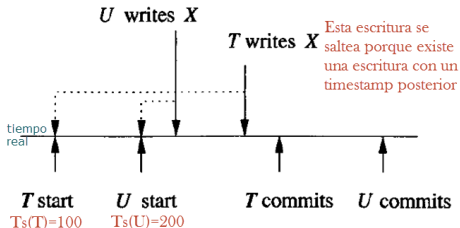


# Regla de escritura de Thomas

## Thomas write rule

La escritura puede “saltarse” cuando ya existe una escritura de una transacción con un timestamp de mayor valor.

Es decir cuando  $WT(X) > TS(T)$



¿Que sucede si U realiza *abort* en vez de *commit*?

Cuando una transacción U escribe un elemento X, la escritura es **tentativa** y **puede ser deshecha si U aborta**. C(X) se pone falso y el planificador hace una copia de los valores de X y de WT(X) previos.

Timeline diagram illustrating a write skew scenario:

- T start** ( $T_S(T)=100$ )
- U start** ( $T_S(U)=200$ )
- T writes X**
- U writes X** (Note: Esta escritura se efectiviza porque U aborta)
- T commits**
- U aborts**

Ante la solicitud de una transacción T para una lectura o escritura, el planificador puede:

### 1 Conceder la solicitud

- 1 Conceder la solicitud
- 2 Abortar y reiniciar T con un nuevo timestamp (rollback)



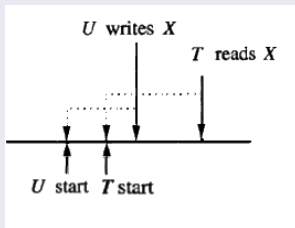
- 1 Conceder la solicitud
- 2 Abortar y reiniciar T con un nuevo timestamp (rollback)
- 3 Demorar T y decidir luego si abortar o conceder la solicitud (si el requerimiento es una lectura que podría ser sucia).

**Caso 1:** Si  $TS(T) \geq WT(X)$  - es **físicamente realizable** es decir, no sucede **read too late**

# Reglas para el planificador

El planificador recibe una solicitud de **lectura**  $r_t(X)$ .

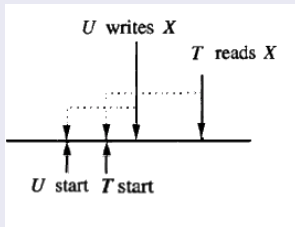
**Caso 1:** Si  $TS(T) \geq WT(X)$  - es **físicamente realizable** es decir, no sucede **read too late**



# Reglas para el planificador

El planificador recibe una solicitud de **lectura**  $r_t(X)$ .

**Caso 1:** Si  $TS(T) \geq WT(X)$  - es **físicamente realizable** es decir, no sucede **read too late**

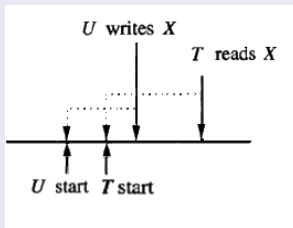


- 1 Si **C(X)** es **True**, conceder la solicitud. Si  $TS(T) > RT(X)$  hacer  $RT(X) = TS(T)$ , de otro modo no cambiar  $RT(X)$ .

# Reglas para el planificador

El planificador recibe una solicitud de **lectura**  $r_t(X)$ .

**Caso 1:** Si  $TS(T) \geq WT(X)$  - es **físicamente realizable** es decir, no sucede **read too late**



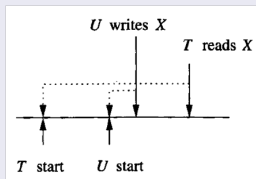
- 1 Si **C(X) es True**, conceder la solicitud. Si  $TS(T) > RT(X)$  hacer  $RT(X) = TS(T)$ , de otro modo no cambiar  $RT(X)$ .
- 2 Si **C(X) es False** demorar T hasta que C(X) sea verdadero o la transacción que escribió a X aborta.

**Caso 2:** Si  $TS(T) < WT(X)$  - es **físicamente irrealizable** (*read too late*)

# Reglas para el planificador

El planificador recibe una solicitud de **lectura**  $r_t(X)$ .

**Caso 2:** Si  $TS(T) < WT(X)$  - es **físicamente irrealizable** (*read too late*)



Se hace **Rollback T** (abortar y reiniciar con un nuevo timestamp).



**Caso 1:** Si  $TS(T) \geq RT(X)$  y  $TS(T) \geq WT(X)$  - es **físicamente realizable**, es decir, no sucede **write too late**

Timeline diagram illustrating a write skew anomaly:

- U start**: Process U begins execution.
- T start**: Process T begins execution.
- Y start**: Process Y begins execution (highlighted with a purple oval).
- U reads X**: Process U reads the initial value of X.
- T writes X**: Process T writes a new value to X.
- Y writes X**: Process Y writes a new value to X.

Dashed lines indicate the sequence of operations on the shared variable X:

- U reads X (before T writes X).
- T writes X.
- Y reads X (before T writes X, due to write skew).
- Y writes X (after T writes X).

- 1 Escribir el nuevo valor para X
- 2  $WT(X) := TS(T)$ , o sea asignar nuevo WT a X.
- 3  $C(X) := \text{false}$ , o sea poner en falso el bit de commit.

**Caso 2:** Si  $TS(T) \geq RT(X)$  pero  $TS(T) < WT(X)$  - es **físicamente realizable**, pero ya hay un valor posterior en X.



- 1 Si  $C(X)$  es true, ignora la escritura.
- 2 Si  $C(X)$  es falso demorar  $T$  hasta que  $C(X)$  sea verdadero o la transacción que escribió a  $X$  aborte

**Caso 3:** Si  $TS(T) < RT(X)$  - es **físicamente irrealizable**, es decir, **write too late**.

Se hace **Rollback T** (abortar y reiniciar con un nuevo timestamp).



Para cada uno de los elementos  $X$  escritos por  $T$  se hace:

- $C(X) := \text{true}$ .
- Se permite proseguir a las transacciones que esperan a que  $X$  sea committed

# Reglas para el planificador

El planificador recibe una solicitud de abort o rollback  $A(T)$  o  $R(T)$ .

Cada transacción que estaba esperando por un elemento  $X$  que  $T$  escribió debe repetir el intento de lectura o escritura y verificar si ahora el intento es legal

## Reglas para el planificador: Ejemplo

Supongamos transacciones  $T_1$ ,  $T_2$  y  $T_3$  con los Timestamp como muestra la figura y suponer que el *commit* se realiza inmediatamente después del ultimo *write*.

$T_1$	$T_2$	$T_3$	$A$	$B$	$C$
200	150	175			
$r_1(B);$					
	$r_2(A);$				
		$r_3(C);$			
$w_1(B);$					
$w_1(A);$					
	$w_2(C);$				
		$w_3(A);$			

# Reglas para el planificador: Ejemplo

## Atención

Suponemos que inmediatamente luego de la ultima operación, se realiza el commit

$T_1$	$T_2$	$T_3$	$A$	$B$	$C$
200	150	175	RT=0	RT=0	RT=0
			WT=0	WT=0	WT=0
$r_1(B);$					
	$r_2(A);$				
		$r_3(C);$			
$w_1(B);$					
$w_1(A);$					
	$w_2(C);$				
		$w_3(A);$			

# Reglas para el planificador: Ejemplo

## Atención

Suponemos que inmediatamente luego de la ultima operación, se realiza el commit

$T_1$	$T_2$	$T_3$	$A$	$B$	$C$
200	150	175	<del>RT=0</del> WT=0	<del>RT=0</del> WT=0	<del>RT=0</del> WT=0
$r_1(B);$				RT=200	
	$r_2(A);$		RT=150		
		$r_3(C);$			RT=175
$w_1(B);$					
$w_1(A);$					
	$w_2(C);$				
		$w_3(A);$			

$TS(T) \geq WT(X)? Si! \Rightarrow \text{como}$   
 $TS(T) > RT(X) \Rightarrow RT(X) = TS(T)$

# Reglas para el planificador: Ejemplo

## Atención

Suponemos que inmediatamente luego de la ultima operación, se realiza el commit

$T_1$	$T_2$	$T_3$	$A$	$B$	$C$
200	150	175	<del>RT=0</del> <del>WT=0</del>	<del>RT=0</del> <del>WT=0</del>	<del>RT=0</del> WT=0
$r_1(B);$				RT=200	
	$r_2(A);$		RT=150		
		$r_3(C);$			RT=175
$w_1(B);$				WT=200	
$w_1(A);$			WT=200		
	$w_2(C);$				
		$w_3(A);$			

$TS(T) \geq WT(X)? Si! \Rightarrow \text{como}$   
 $TS(T) > RT(X) \Rightarrow RT(X) = TS(T)$   
 $TS(T) \geq RT(X) \text{ y } TS(T) \geq WT(X)$   
 $\Rightarrow WT(X) = TS(T)$   
 $C(A)=T, C(B)=T$

# Reglas para el planificador: Ejemplo

## Atención

Suponemos que inmediatamente luego de la ultima operación, se realiza el commit

$T_1$	$T_2$	$T_3$	$A$	$B$	$C$
200	150	175	<del>RT=0</del> <del>WT=0</del>	<del>RT=0</del> <del>WT=0</del>	<del>RT=0</del> WT=0
$r_1(B);$				RT=200	
	$r_2(A);$		RT=150 0		
		$r_3(C);$		WT=200	RT=175
$w_1(B);$			WT=200		
$w_1(A);$					
	$w_2(C);$				
	<b>Rollback!</b>				
		$w_3(A);$			

$T_s(T) \geq WT(X)? Si! \Rightarrow como$   
 $T_s(T) > RT(X) \Rightarrow RT(X) = T_s(T)$   
 $T_s(T) \geq RT(X) \text{ y } T_s(T) \geq WT(X)$   
 $\Rightarrow WT(X) = T_s(T)$   
 $C(A)=T, C(B)=T$   
 $T_s(T) < RT(X) \Rightarrow \text{es físicamente irrealizable, por lo tanto, Rollback!}$



# Reglas para el planificador: Ejemplo

## Atención

Suponemos que inmediatamente luego de la ultima operación, se realiza el commit

$T_1$	$T_2$	$T_3$	$A$	$B$	$C$
200	150	175	<del>RT=0</del> <del>WT=0</del>	<del>RT=0</del> <del>WT=0</del>	<del>RT=0</del> WT=0
$r_1(B);$				RT=200	
	$r_2(A);$		RT=0		RT=175
$w_1(B);$		$r_3(C);$		WT=200	
$w_1(A);$			WT=200		
	$w_2(C);$				
	<b>Rollback!</b>	$w_3(A);$			

$Ts(T) \geq WT(X)? Si! \Rightarrow$  como  $Ts(T) > RT(X) \Rightarrow RT(X) = Ts(T)$   
 $Ts(T) \geq RT(X)$  y  $Ts(T) \geq WT(X) \Rightarrow WT(X) = Ts(T)$   
 $C(A)=T, C(B)=T$   
 $Ts(T) < RT(X) \Rightarrow$  es físicamente irrealizable, por lo tanto, Rollback!  
 $Ts(T) \geq RT(X)$  pero  $Ts(T) < WT(X) \Rightarrow$  A se escribió en un timestamp posterior, por lo tanto, no hay Rollback pero A no se escribe.

## Ejemplo 2: Enunciado

Dada la siguiente historia en un planificador con timestamp:

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

en donde los valores iniciales de X, Y, Z son 0 y sucede que:

t1 escribe  $Y = 1$

t2 escribe  $X=2$

t3 escribe  $X=3, Y=30$

t4 escribe  $Z = 4$

- Decir que pasa en cada acción y que valores quedan en X,Y,Z si el planificador no utiliza la técnica multiversión

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4
1	st1=100			
2		st2=200		
3		r2(X)		
4			st3=300	
5				st4=400
6	r1(Y)			
7				r4(Z)
8			w3(X)	
9			w3(Y)	
10				w4(Z)
11		w2(X)		
12	w1(Y)			
13			r3(Z)	

Asumimos los bits de commit para cada elemento inicialmente en true.

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0	RT(X)=0, RT(Y)=0, RT(Z)=0	WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100						
2		st2=200					
3		r2(X)					
4			st3=300				
5				st4=400			
6	r1(Y)						
7				r4(Z)			
8			w3(X)				
9			w3(Y)				
10				w4(Z)			
11		w2(X)					
12	w1(Y)						
13			r3(Z)				

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0	RT(X)=0, RT(Y)=0, RT(Z)=0	WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100		
2		st2=200			Se lanza T2 con timestamp=200		
3		r2(X)					
4			st3=300				
5				st4=400			
6	r1(Y)						
7				r4(Z)			
8			w3(X)				
9			w3(Y)				
10				w4(Z)			
11		w2(X)					
12	w1(Y)						
13			r3(Z)				

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0	RT(X)=0 , RT(Y)=0, RT(Z)=0	WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100		
2		st2=200			Se lanza T2 con timestamp=200		
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) =>		
4			st3=300				
5				st4=400			
6	r1(Y)						
7				r4(Z)			
8			w3(X)				
9			w3(Y)				
10				w4(Z)			
11		w2(X)					
12	w1(Y)						
13			r3(Z)				

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0      RT(X)=200, RT(Y)=0, RT(Z)=0      WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			$T_s(T2) \geq WT(X) (200 \geq 0)$ y como $T_s(T2) \geq RT(X) (200 \geq 0) \Rightarrow RT(X)=200$
4			st3=300		
5				st4=400	
6	r1(Y)				
7				r4(Z)	
8			w3(X)		
9			w3(Y)		
10				w4(Z)	
11		w2(X)			
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0      RT(X)=200 , RT(Y)=0, RT(Z)=0      WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			$Ts(T2) \geq WT(X) (200 \geq 0)$ y como $Ts(T2) \geq RT(X) (200 \geq 0) \Rightarrow RT(X)=200$
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				
7				r4(Z)	
8			w3(X)		
9			w3(Y)		
10				w4(Z)	
11		w2(X)			
12	w1(Y)				
13			r3(Z)		



## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0      RT(X)=200, RT(Y)=0, RT(Z)=0      WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			$Ts(T2) \geq WT(X)$ ( $200 \geq 0$ ) y como $Ts(T2) \geq RT(X)$ ( $200 \geq 0$ ) $\Rightarrow$ <b>RT(X)=200</b>
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 $\Rightarrow$ ( $100 \geq 0$ )
7				r4(Z)	Idem tiempo 3 $\Rightarrow$ ( $400 \geq 0$ )
8			w3(X)		
9			w3(Y)		
10				w4(Z)	
11		w2(X)			
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0      RT(X)=200, RT(Y)=100, RT(Z)=400      WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			$T_s(T2) \geq WT(X) (200 \geq 0)$ y como $T_s(T2) \geq RT(X) (200 \geq 0) \Rightarrow RT(X)=200$
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 $\Rightarrow (100 \geq 0) \Rightarrow RT(Y)=100$
7				r4(Z)	Idem tiempo 3 $\Rightarrow (400 \geq 0) \Rightarrow RT(Z)=400$
8			w3(X)		
9			w3(Y)		
10				w4(Z)	
11		w2(X)			
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=0, Y=0, Z=0      RT(X)=200, RT(Y)=100, RT(Z)=400      WT(X)=0, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) => RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 => (100 >= 0) => RT(Y)=100
7				r4(Z)	Idem tiempo 3 => (400 >= 0) => RT(Z)=400
8			w3(X)		Ts (T3) >= RT(X) (300>=200) y Ts (T3) >= WT(X) (300>=0) =>
9			w3(Y)		
10				w4(Z)	
11		w2(X)			
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=3, Y=0, Z=0      RT(X)=200, RT(Y)=100, RT(Z)=400      WT(X)=300, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) => RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 => (100 >= 0) => RT(Y)=100
7				r4(Z)	Idem tiempo 3 => (400 >= 0) => RT(Z)=400
8			w3(X)		Ts (T3) >= RT(X) (300>=200) y Ts (T3) >= WT(X) (300>=0) => X:=3, WT(X)=300, C(X)=false
9			w3(Y)		
10				w4(Z)	
11		w2(X)			
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=3, Y=0, Z=0      RT(X)=200, RT(Y)=100, RT(Z)=400      WT(X)= 300, WT(Y)=0, WT(Z)=0
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) => RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 => (100 >= 0) => RT(Y)=100
7				r4(Z)	Idem tiempo 3 => (400 >= 0) => RT(Z)=400
8			w3(X)		Ts (T3) >= RT(X) (300>=200) y Ts (T3) >= WT(X) (300>=0) => X:= 3, WT(X)= 300, C(X)=false
9			w3(Y)		Idem tiempo 8 => Y:=30, WT(Y)=300, C(Y)=false
10				w4(Z)	Idem tiempo 8 => Z:= 4, WT(Z)= 400, C(Z)=true
11		w2(X)			
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=3, Y=30, Z=4 RT(X)=200, RT(Y)=100, RT(Z)=400 WT(X)= 300, WT(Y)=300 , WT(Z)= 400
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) => RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 => (100 >= 0) => RT(Y)=100
7				r4(Z)	Idem tiempo 3 => (400 >= 0) => RT(Z)=400
8			w3(X)		Ts (T3) >= RT(X) (300>=200) y Ts (T3) >= WT (X) (300>=0) => X:= 3, WT(X)= 300, C(X)=false
9			w3(Y)		Idem tiempo 8 => Y:=30, WT(Y)=300, C(Y)=false
10				w4(Z)	Idem tiempo 8 => Z:= 4, WT(Z)= 400, C(Z)=true
11		w2(X)			
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=3, Y=30, Z=4 RT(X)=200, RT(Y)=100, RT(Z)=400 WT(X)= 300, WT(Y)=300 , WT(Z)= 400
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) => RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 => (100 >= 0) => RT(Y)=100
7				r4(Z)	Idem tiempo 3 => (400 >= 0) => RT(Z)=400
8			w3(X)		Ts (T3) >= RT(X) (300>=200) y Ts (T3) >= WT (X) (300>=0) => X:= 3, WT(X)= 300, C(X)=false
9			w3(Y)		Idem tiempo 8 => Y:=30, WT(Y)=300, C(Y)=false
10				w4(Z)	Idem tiempo 8 => Z:= 4, WT(Z)= 400, C(Z)=true
11		w2(X)			Ts (T2) >= RT(X) (200>=200) pero Ts (T2) < WT (X) (200<300) =>
12	w1(Y)				
13			r3(Z)		

## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=3, Y=30, Z=4 RT(X)=200, RT(Y)=100, RT(Z)=400 WT(X)= 300, WT(Y)=300 , WT(Z)= 400
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) => RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 => (100 >= 0) => RT(Y)=100
7				r4(Z)	Idem tiempo 3 => (400 >= 0) => RT(Z)=400
8			w3(X)		Ts (T3) >= RT(X) (300>=200) y Ts (T3) >= WT(X) (300>=0) => X:= 3, WT(X)= 300, C(X)=false
9			w3(Y)		Idem tiempo 8 => Y:=30, WT(Y)=300, C(Y)=false
10				w4(Z)	Idem tiempo 8 => Z:= 4, WT(Z)= 400, C(Z)=true
11		w2(X)			Ts (T2) >= RT(X) (200>=200) pero Ts (T2) < WT(X) (200<300) => Se demora la escritura hasta que T3 finalice o aborte
12	w1(Y)				
13			r3(Z)		





## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=3, Y=30, Z=4 RT(X)=200, RT(Y)=100, RT(Z)=400 WT(X)=300, WT(Y)=300, WT(Z)=400
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			Ts (T2) >= WT(X) (200>=0) y como Ts (T2) >= RT(X) (200>=0) => RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 => (100 >= 0) => RT(Y)=100
7				r4(Z)	Idem tiempo 3 => (400 >= 0) => RT(Z)=400
8			w3(X)		Ts (T3) >= RT(X) (300>=200) y Ts (T3) >= WT(X) (300>=0) => X:=3, WT(X)=300, C(X)=false
9			w3(Y)		Idem tiempo 8 => Y:=30, WT(Y)=300, C(Y)=false
10				w4(Z)	Idem tiempo 8 => Z:=4, WT(Z)=400, C(Z)=true
11		w2(X)			Ts (T2) >= RT(X) (200>=200) pero Ts (T2) < WT(X) (200<300) => Se demora la escritura hasta que T3 finalice o aborte
12	w1(Y)				Ts (T1) >= RT(Y) (100>=100) pero Ts (T1) < WT(Y) (100<300) => Se demora la escritura hasta que T3 finalice o aborte
13			r3(Z)		TS(T3) < WT(Z) (300<400) =>

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

También termina T4



## Ejemplo 2: Resolución

st1, st2, r2(X), st3, st4, r1(Y), r4(Z), w3(X), w3(Y), w4(Z), w2(X), w1(Y), r3(Z)

Tiempo	T1	T2	T3	T4	X=2, Y=1, Z=4 RT(X)=200, RT(Y)=100, RT(Z)=400 WT(X)= 200, WT(Y)=100 , WT(Z)= 400
1	st1=100				Se lanza T1 con timestamp=100
2		st2=200			Se lanza T2 con timestamp=200
3		r2(X)			$Ts(T2) \geq WT(X)$ ( $200 \geq 0$ ) y como $Ts(T2) \geq RT(X)$ ( $200 \geq 0$ ) $\Rightarrow$ RT(X)=200
4			st3=300		Se lanza T3 con timestamp=300
5				st4=400	Se lanza T4 con timestamp=400
6	r1(Y)				Idem tiempo 3 $\Rightarrow$ ( $100 \geq 0$ ) $\Rightarrow$ RT(Y)=100
7				r4(Z)	Idem tiempo 3 $\Rightarrow$ ( $400 \geq 0$ ) $\Rightarrow$ RT(Z)=400
8			w3(X)		$Ts(T3) \geq RT(X)$ ( $300 \geq 200$ ) y $Ts(T3) \geq WT(X)$ ( $300 \geq 0$ ) $\Rightarrow$ X:= 3, WT(X)= 300, C(X)=false
9			w3(Y)		Idem tiempo 8 $\Rightarrow$ Y:=30, WT(Y)=300, C(Y)=false
10				w4(Z)	Idem tiempo 8 $\Rightarrow$ Z:= 4, WT(Z)= 400, C(Z)=true
11		w2(X)			$Ts(T2) \geq RT(X)$ ( $200 \geq 200$ ) pero $Ts(T2) < WT(X)$ ( $200 < 300$ ) $\Rightarrow$ Se demora la escritura hasta que T3 finalice o aborte
12	w1(Y)				$Ts(T1) \geq RT(Y)$ ( $100 \geq 100$ ) pero $Ts(T1) < WT(Y)$ ( $100 < 300$ ) $\Rightarrow$ Se demora la escritura hasta que T3 finalice o aborte
13			r3(Z)		$TS(T3) < WT(Z)$ ( $300 < 400$ ) $\Rightarrow$

Aborta T3, se produce un Rollback

Al abortar T3, las escrituras en el tiempo 11 y 12

que estaban demoradas continúan y T1 y T2 terminan normalmente

También termina T4

Valores finales: X=2, Y=1, Z=4

# Optimista vs. Pesimista

## Compromiso

- Generalmente el timestamping es mejor cuando la mayoría de las transacciones son de lectura o es raro que haya transacciones que traten de leer y escribir el mismo elemento.
- En situaciones de mucho conflicto, locking suele comportarse mejor.
- Se establece entonces un compromiso utilizado en los sistemas comerciales:
  - 1 Transacciones read-only vs. Transacciones read-write.
  - 2 Transacciones read-write se manejan con locking pero crean versiones de los elementos.
  - 3 Transacciones read-only se manejan con versiones creadas por transacciones read-write