Bases de Datos

Control de Concurrencia: Ejercicios

Andrea Manna

DEPARTAMENTO
DE COMPUTACION

2025

Enunciado 1: Control de Concurrencia Pesimista

Parte 1 (sin lock):

Dadas las siguientes transacciones:

$T_1 = r_1(A); w_1(A); r_1(B); w_1(B) ; c_1$

$T_2 = r_2(A); w_2(A); c_2$

$T_3 = r_3(B) ; r_3(A) ; w_3(A); w_3(B) ; c_3$

Y la historia:

$H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A) ; c_2; w_3(A); w_3(B) ; c_3$

Se pide:

- Construir el SG(H_1) (grafo de precedencia).
- Indicar si H_1 es SR (serializable) y en caso afirmativo indicar las historias seriales equivalentes.
- Dar una historia H_2 equivalente a la ejecución serial T_2, T_1, T_3, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

Enunciado 1: Solución

- Dibujarlo en forma columna para visualizar mejor.

Enunciado 1: Solución

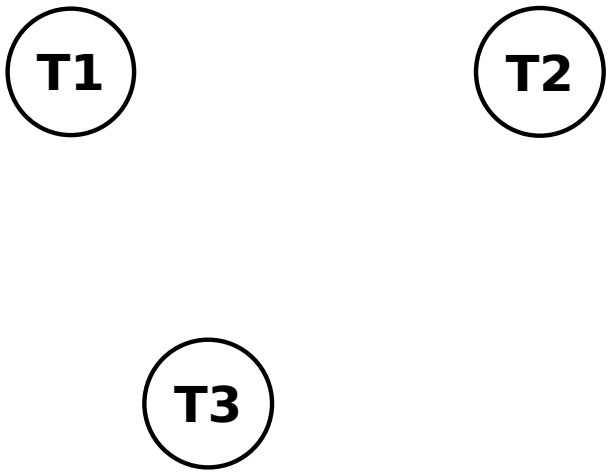
- $H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A); c_2; w_3(A); w_3(B) ; c_3$

T_1	T_2	T_3
$r_1(A)$		
$w_1(A)$		
	$r_2(A)$	
$r_1(B)$		
$w_1(B)$		
c_1		
		$r_3(B)$
	$w_2(A)$	
		$r_3(A)$
	c_2	
		$w_3(A)$
		$w_3(B)$
		c_3

Enunciado 1: Solución

- $H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A); c_2; w_3(A); w_3(B) ; c_3$

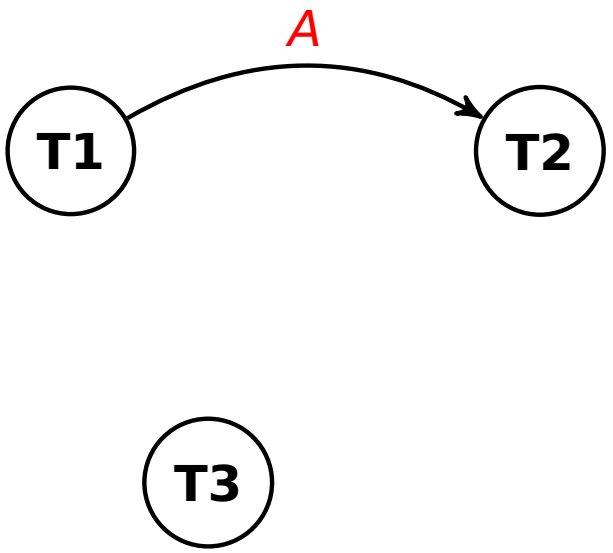
T_1	T_2	T_3
$r_1(A)$		
$w_1(A)$		
	$r_2(A)$	
$r_1(B)$		
$w_1(B)$		
c_1		
		$r_3(B)$
	$w_2(A)$	
		$r_3(A)$
	c_2	
		$w_3(A)$
		$w_3(B)$
		c_3



Enunciado 1: Solución

- $H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A); c_2; w_3(A); w_3(B) ; c_3$

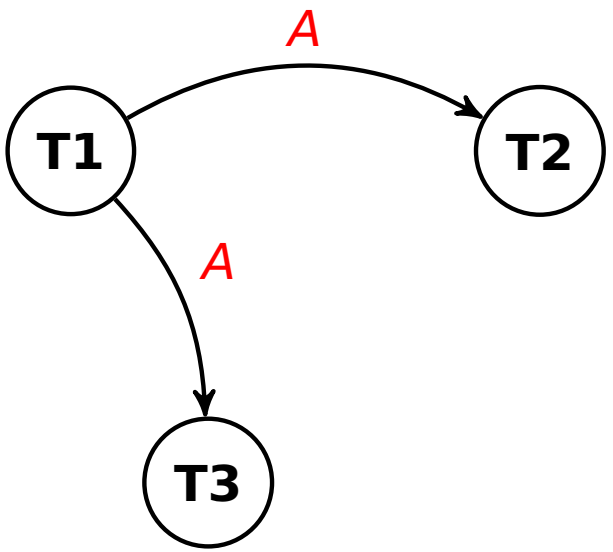
T_1	T_2	T_3
$r_1(A)$		
$w_1(A)$		
	$r_2(A)$	
$r_1(B)$		
$w_1(B)$		
c_1		
		$r_3(B)$
	$w_2(A)$	
		$r_3(A)$
	c_2	
		$w_3(A)$
		$w_3(B)$
		c_3



Enunciado 1: Solución

- $H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A); c_2; w_3(A); w_3(B) ; c_3$

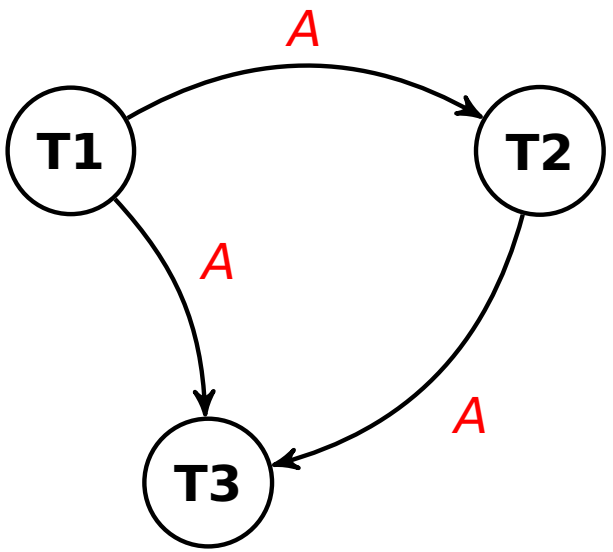
T_1	T_2	T_3
$r_1(A)$		
$w_1(A)$		
	$r_2(A)$	
$r_1(B)$		
$w_1(B)$		
c_1		
		$r_3(B)$
	$w_2(A)$	
		$r_3(A)$
	c_2	
		$w_3(A)$
		$w_3(B)$
		c_3



Enunciado 1: Solución

- $H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A); c_2; w_3(A); w_3(B) ; c_3$

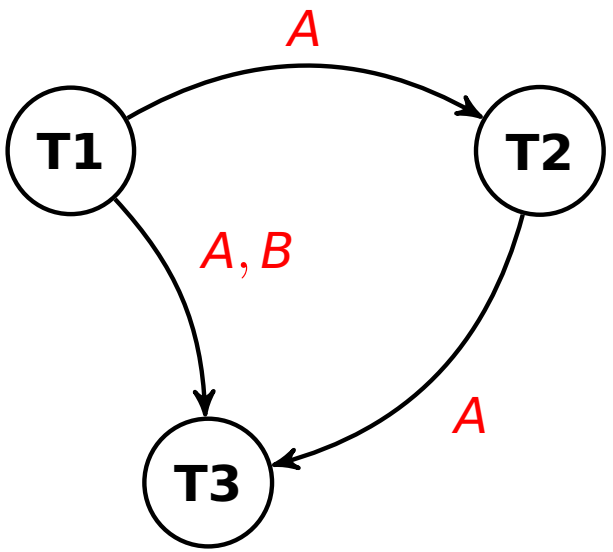
T_1	T_2	T_3
$r_1(A)$		
$w_1(A)$		
	$r_2(A)$	
$r_1(B)$		
$w_1(B)$		
c_1		
		$r_3(B)$
	$w_2(A)$	
		$r_3(A)$
	c_2	
		$w_3(A)$
		$w_3(B)$
		c_3



Enunciado 1: Solución

- $H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A); c_2; w_3(A); w_3(B) ; c_3$

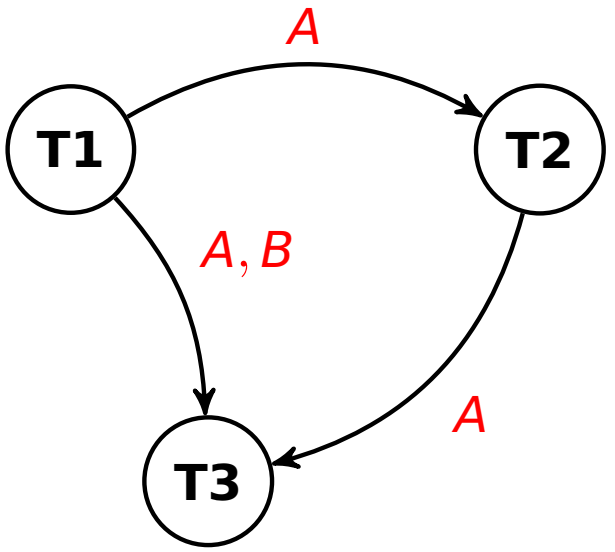
T_1	T_2	T_3
$r_1(A)$		
$w_1(A)$		
	$r_2(A)$	
$r_1(B)$		
$w_1(B)$		
c_1		
		$r_3(B)$
	$w_2(A)$	
		$r_3(A)$
	c_2	
		$w_3(A)$
		$w_3(B)$
		c_3



Enunciado 1: Solución

- $H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B) ; c_1; r_3(B); w_2(A); r_3(A); c_2; w_3(A); w_3(B) ; c_3$

T_1	T_2	T_3
$r_1(A)$		
$w_1(A)$		
	$r_2(A)$	
$r_1(B)$		
$w_1(B)$		
c_1		
		$r_3(B)$
	$w_2(A)$	
		$r_3(A)$
	c_2	
		$w_3(A)$
		$w_3(B)$
		c_3



Es un grafo sin ciclos => H es serializable: T_1, T_2, T_3

Enunciado 1: Solución

Dar una historia H_2 equivalente a la ejecución serial T_2, T_1, T_3, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

Enunciado 1: Solución

Dar una historia H_2 equivalente a la ejecución serial T_2, T_1, T_3, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

Empecemos por escribir la historia serial:

Enunciado 1: Solución

Dar una historia H_2 equivalente a la ejecución serial T_2, T_1, T_3, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

Empecemos por escribir la historia serial:

$H_2 = r_2(A); w_2(A); c_2; r_1(A); w_1(A); r_1(B); w_1(B); c_1; r_3(B); r_3(A); w_3(A); w_3(B); c_3$

Enunciado 1: Solución

Dar una historia H_2 equivalente a la ejecución serial T_2, T_1, T_3, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

Empecemos por escribir la historia serial:

$H_2 = r_2(A); w_2(A); c_2; r_1(A); w_1(A); r_1(B); w_1(B); c_1; r_3(B); r_3(A); w_3(A); w_3(B); c_3$

Esta historia es SR, RC y ACA. Como hacemos para "*romper*" la propiedad ACA?

Enunciado 1: Solución

Dar una historia H_2 equivalente a la ejecución serial T_2, T_1, T_3, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

Empecemos por escribir la historia serial:

$H_2 = r_2(A); w_2(A); c_2; r_1(A); w_1(A); r_1(B); w_1(B); c_1; r_3(B); r_3(A); w_3(A); w_3(B); c_3$

Esta historia es SR, RC y ACA. Como hacemos para "*romper*" la propiedad ACA?

Alternativa 1: $H_2 = r_2(A); w_2(A); c_2; r_1(A); w_1(A); r_1(B); w_1(B); r_3(B); c_1; r_3(A); w_3(A); w_3(B); c_3$

Enunciado 1: Solución

Dar una historia H_2 equivalente a la ejecución serial T_2, T_1, T_3, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

Empecemos por escribir la historia serial:

$H_2 = r_2(A); w_2(A); c_2; r_1(A); w_1(A); r_1(B); w_1(B); c_1; r_3(B); r_3(A); w_3(A); w_3(B); c_3$

Esta historia es SR, RC y ACA. Como hacemos para "*romper*" la propiedad ACA?

Alternativa 1: $H_2 = r_2(A); w_2(A); c_2; r_1(A); w_1(A); r_1(B); w_1(B); r_3(B); \mathbf{c_1;} r_3(A); w_3(A); w_3(B); c_3$

Alternativa 2: $H_2 = r_2(A); w_2(A); r_1(A); w_1(A); r_1(B); w_1(B); r_3(B); r_3(A); w_3(A); w_3(B); \mathbf{c_2; c_1;} c_3$

Enunciado 2: Control de Concurrencia Pesimista

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

H_3=rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3

- ¿H_3 es legal?
- ¿Se respeta el protocolo 2PL (two phase locking)?
- Dibujar el Grafo de Precedencia y dar todas las historias seriales equivalentes

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3$

- ¿H_3 es legal?

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3$

- ¿H_3 es legal?

Si! Una H ilegal podría ser:

$H'_3 = rl_1(A); rl_2(B); u_2(B); wl_2$**(A);** u_1**(A);** $u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3$

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3$

- ¿Se respeta el protocolo 2PL (two phase locking)?

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

H_3=rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3

- ¿Se respeta el protocolo 2PL (two phase locking)?

No se respeta el protocolo, por ejemplo:

....rl_2(B); u_2(B); u_1(A); wl_2(A)....

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

$H_3 = rl_1(A);\ rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A);\ u_2(A);\ rl_3(A)\ ;\ c_1;\ u_3(A);\ wl_3(B);\ c_2\ ;\ u_3(B);\ c_3$

- ¿Se respeta el protocolo 2PL (two phase locking)?

No se respeta el protocolo, por ejemplo:

$....rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A)....$

- ¿Se podrá lograr una historia equivalente que sea 2PL?

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

H_3=rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3

- ¿Se respeta el protocolo 2PL (two phase locking)?

No se respeta el protocolo, por ejemplo:

....rl_2(B); u_2(B); u_1(A); wl_2(A)....

- ¿Se podrá lograr una historia equivalente que sea 2PL?

Una manera de verlo, es desglosar cada transaccion:

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

$H_3 = rl_1(A);\ rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A);\ u_2(A);\ rl_3(A)\ ;\ c_1;\ u_3(A);\ wl_3(B);\ c_2\ ;\ u_3(B);\ c_3$

- ¿Se respeta el protocolo 2PL (two phase locking)?

No se respeta el protocolo, por ejemplo:

$....rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A)....$

- ¿Se podrá lograr una historia equivalente que sea 2PL?

Una manera de verlo, es desglosar cada transaccion:

$T_1 = rl_1(A);\ u_1(A);\ c_1$

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

$H_3 = rl_1(A);\ rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A);\ u_2(A);\ rl_3(A)\ ;\ c_1;\ u_3(A);\ wl_3(B);\ c_2\ ;\ u_3(B);\ c_3$

- ¿Se respeta el protocolo 2PL (two phase locking)?

No se respeta el protocolo, por ejemplo:

$....rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A)....$

- ¿Se podrá lograr una historia equivalente que sea 2PL?

Una manera de verlo, es desglosar cada transaccion:

$T_1 = rl_1(A);\ u_1(A);\ c_1$

$T_2 = rl_2(B);\ u_2(B);\ wl_2(A);\ u_2(A);\ c_2$

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

$H_3 = rl_1(A);\ rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A);\ u_2(A);\ rl_3(A)\ ;\ c_1;\ u_3(A);\ wl_3(B);\ c_2\ ;\ u_3(B);\ c_3$

- ¿Se respeta el protocolo 2PL (two phase locking)?

No se respeta el protocolo, por ejemplo:

$....rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A)....$

- ¿Se podrá lograr una historia equivalente que sea 2PL?

Una manera de verlo, es desglosar cada transaccion:

$T_1 = rl_1(A);\ u_1(A);\ c_1$

$T_2 = rl_2(B);\ u_2(B);\ wl_2(A);\ u_2(A);\ c_2$

$T_3 = rl_3(A)\ ;\ u_3(A);\ wl_3(B);\ u_3(B)\ ;\ c_3$

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

H_3=rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3

- ¿Se respeta el protocolo 2PL (two phase locking)?

No se respeta el protocolo, por ejemplo:

....rl_2(B); u_2(B); u_1(A); wl_2(A)....

- ¿Se podrá lograr una historia equivalente que sea 2PL?

Una manera de verlo, es desglosar cada transaccion:

T_1= rl_1(A); u_1(A); c_1

T_2= rl_2(B); u_2(B); wl_2(A); u_2(A); c_2

T_3= rl_3(A) ; u_3(A); wl_3(B); u_3(B) ; c_3

Conclusión: T_1 es 2PL, pero ni T_2, ni T_3 lo son, por lo tanto no se podrá lograr una historia 2PL

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

H_3=rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3

- Dibujar el Grafo de Precedencia y dar todas las historias seriales equivalentes

Enunciado 2: Solución

Parte 2 (con lock):

Dada la siguiente historia H_3 en el modelo ReadLock /WriteLock / UnLock (ternario):

H_3=rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3

- Dibujar el Grafo de Precedencia y dar todas las historias seriales equivalentes

Enunciado 2: Solución

H_3=rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3

Enunciado 2: Solución

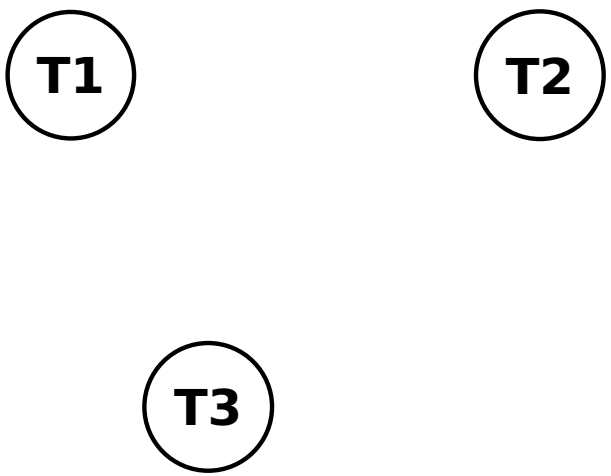
$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3$

T_1	T_2	T_3
$rl_1(A)$		
	$rl_2(B)$	
	$u_2(B)$	
$u_1(A)$		
	$wl_2(A)$	
	$u_2(A)$	
		$rl_3(A)$
c_1		
		$u_3(A)$
		$wl_3(B)$
	c_2	
		$u_3(B)$
		c_3

Enunciado 2: Solución

$H_3 = rl_1(A);\ rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A);\ u_2(A);\ rl_3(A)\ ;\ c_1;\ u_3(A);\ wl_3(B);\ c_2\ ;\ u_3(B);\ c_3$

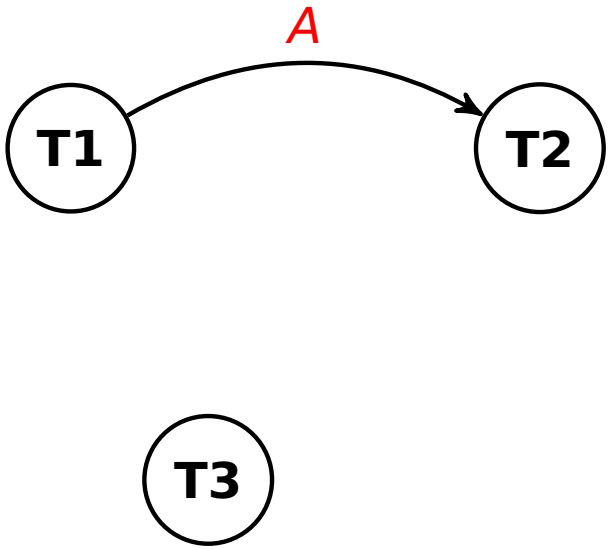
T_1	T_2	T_3
$rl_1(A)$		
	$rl_2(B)$	
	$u_2(B)$	
$u_1(A)$		
	$wl_2(A)$	
	$u_2(A)$	
		$rl_3(A)$
c_1		
		$u_3(A)$
		$wl_3(B)$
	c_2	
		$u_3(B)$
		c_3



Enunciado 2: Solución

$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A); c_1; u_3(A); wl_3(B); c_2; u_3(B); c_3$

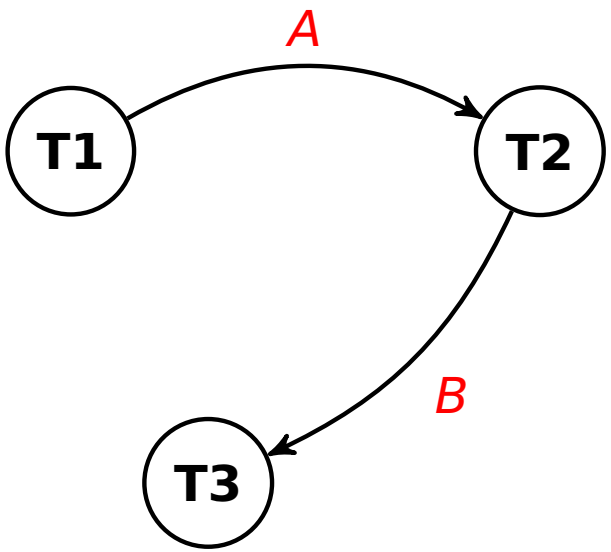
T_1	T_2	T_3
$rl_1(A)$		
	$rl_2(B)$	
	$u_2(B)$	
$u_1(A)$		
	$wl_2(A)$	
	$u_2(A)$	
		$rl_3(A)$
c_1		
		$u_3(A)$
		$wl_3(B)$
	c_2	
		$u_3(B)$
		c_3



Enunciado 2: Solución

$H_3 = rl_1(A);\ rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A);\ u_2(A);\ rl_3(A)\ ;\ c_1;\ u_3(A);\ wl_3(B);\ c_2\ ;\ u_3(B);\ c_3$

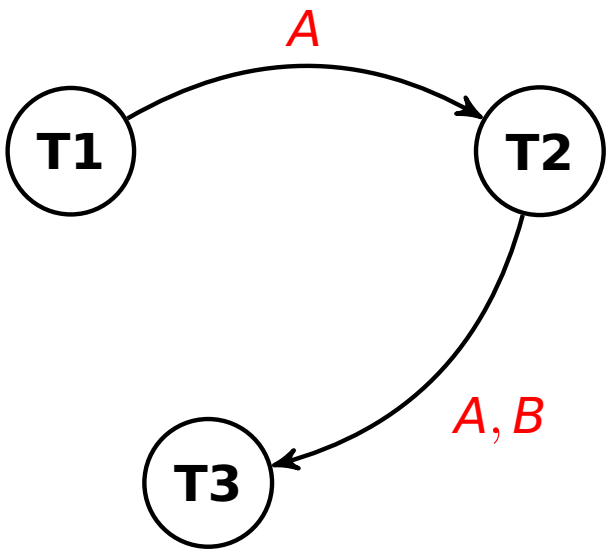
T_1	T_2	T_3
$rl_1(A)$		
	$rl_2(B)$	
	$u_2(B)$	
$u_1(A)$		
	$wl_2(A)$	
	$u_2(A)$	
		$rl_3(A)$
c_1		
		$u_3(A)$
		$wl_3(B)$
	c_2	
		$u_3(B)$
		c_3



Enunciado 2: Solución

$H_3 = rl_1(A);\ rl_2(B);\ u_2(B);\ u_1(A);\ wl_2(A);\ u_2(A);\ rl_3(A)\ ;\ c_1;\ u_3(A);\ wl_3(B);\ c_2\ ;\ u_3(B);\ c_3$

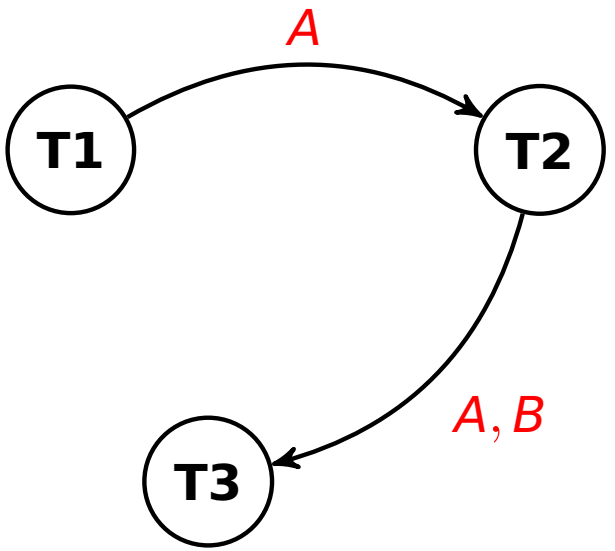
T_1	T_2	T_3
$rl_1(A)$		
	$rl_2(B)$	
	$u_2(B)$	
$u_1(A)$		
	$wl_2(A)$	
	$u_2(A)$	
		$rl_3(A)$
c_1		
		$u_3(A)$
		$wl_3(B)$
	c_2	
		$u_3(B)$
		c_3



Enunciado 2: Solución

$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A) ; c_1; u_3(A); wl_3(B); c_2 ; u_3(B); c_3$

T_1	T_2	T_3
$rl_1(A)$		
	$rl_2(B)$	
	$u_2(B)$	
$u_1(A)$		
	$wl_2(A)$	
	$u_2(A)$	
		$rl_3(A)$
c_1		
		$u_3(A)$
		$wl_3(B)$
	c_2	
		$u_3(B)$
		c_3



Es un grafo sin ciclos => H es serializable: T_1, T_2, T_3

Enunciado 2: Bonus Track!

- Se podrán realizar una mímina cantidad de cambios en las transacciones para convertirlas a 2PL?

Enunciado 2: Bonus Track!

- Se podrán realizar una mímina cantidad de cambios en las transacciones para convertirlas a 2PL?

Teniamos las transacciones:

$T_1 = rl_1(A); u_1(A); c_1$

$T_2 = rl_2(B); u_2(B); wl_2(A); u_2(A); c_2$

$T_3 = rl_3(A) ; u_3(A); wl_3(B); u_3(B) ; c_3$

Enunciado 2: Bonus Track!

- Se podrán realizar una mímina cantidad de cambios en las transacciones para convertirlas a 2PL?

Teniamos las transacciones:
$T_1 = rl_1(A); u_1(A); c_1$
$T_2 = rl_2(B); u_2(B); wl_2(A); u_2(A); c_2$
$T_3 = rl_3(A) ; u_3(A); wl_3(B); u_3(B) ; c_3$
Convierto T_2 y T_3 a 2PL. **Ojo!! Estamos cambiando las operaciones!!!!**

Enunciado 2: Bonus Track!

- Se podrán realizar una mímina cantidad de cambios en las transacciones para convertirlas a 2PL?

Teniamos las transacciones:
$T_1 = rl_1(A); u_1(A); c_1$
$T_2 = rl_2(B); u_2(B); wl_2(A); u_2(A); c_2$
$T_3 = rl_3(A) ; u_3(A); wl_3(B); u_3(B) ; c_3$
Convierto T_2 y T_3 a 2PL. **Ojo!! Estamos cambiando las operaciones!!!!**
$T_1 = rl_1(A); u_1(A); c_1$

Enunciado 2: Bonus Track!

- Se podrán realizar una mímina cantidad de cambios en las transacciones para convertirlas a 2PL?

Teniamos las transacciones:

$T_1 = rl_1(A); u_1(A); c_1$

$T_2 = rl_2(B); u_2(B); wl_2(A); u_2(A); c_2$

$T_3 = rl_3(A) ; u_3(A); wl_3(B); u_3(B) ; c_3$

Convierto T_2 y T_3 a 2PL. **Ojo!! Estamos cambiando las operaciones!!!!**

$T_1 = rl_1(A); u_1(A); c_1$

$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$

Enunciado 2: Bonus Track!

- Se podrán realizar una mímina cantidad de cambios en las transacciones para convertirlas a 2PL?

Teniamos las transacciones:

$T_1 = rl_1(A); u_1(A); c_1$

$T_2 = rl_2(B); u_2(B); wl_2(A); u_2(A); c_2$

$T_3 = rl_3(A) ; u_3(A); wl_3(B); u_3(B) ; c_3$

Convierto T_2 y T_3 a 2PL. **Ojo!! Estamos cambiando las operaciones!!!!**

$T_1 = rl_1(A); u_1(A); c_1$

$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$

$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$

Enunciado 2: Bonus Track!

- Se podrán realizar una mímina cantidad de cambios en las transacciones para convertirlas a 2PL?

Teniamos las transacciones:

$T_1 = rl_1(A); u_1(A); c_1$

$T_2 = rl_2(B); u_2(B); wl_2(A); u_2(A); c_2$

$T_3 = rl_3(A) ; u_3(A); wl_3(B); u_3(B) ; c_3$

Convierto T_2 y T_3 a 2PL. **Ojo!! Estamos cambiando las operaciones!!!!**

$T_1 = rl_1(A); u_1(A); c_1$

$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$

$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

$T_1 = rl_1(\text{A}); u_1(\text{A}); c_1$
$T_2' = rl_2(\text{B}); wl_2(\text{A}); u_2(\text{B}); u_2(\text{A}); c_2$
$T_3' = rl_3(\text{A}) ; wl_3(\text{B}); u_3(\text{A}); u_3(\text{B}) ; c_3$

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

$T_1 = rl_1(A); u_1(A); c_1$
$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$
$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$
$H_4 = rl_1(A); rl_2(B); u_1(A); wl_2(A); u_2(B); u_2(A); rl_3(A) ; c_1; wl_3(B); u_3(A); c_2 ; u_3(B); c_3$

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

$T_1 = rl_1(A); u_1(A); c_1$
$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$
$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$
$H_4 = rl_1(A); rl_2(B); u_1(A); wl_2(A); u_2(B); u_2(A); rl_3(A) ; c_1; wl_3(B); u_3(A); c_2 ; u_3(B); c_3$
H_4 ahora es 2PL!!

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

$T_1 = rl_1(A); u_1(A); c_1$
$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$
$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$
$H_4 = rl_1(A); rl_2(B); u_1(A); wl_2(A); u_2(B); u_2(A); rl_3(A) ; c_1; wl_3(B); u_3(A); c_2 ; u_3(B); c_3$
H_4 ahora es 2PL!!
H_4 es 2PL estricto?

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

$T_1 = rl_1(A); u_1(A); c_1$
$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$
$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$
$H_4 = rl_1(A); rl_2(B); u_1(A); wl_2(A); u_2(B); u_2(A); rl_3(A) ; c_1; wl_3(B); u_3(A); c_2 ; u_3(B); c_3$
H_4 ahora es 2PL!!
H_4 es 2PL estricto?
No!!!

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

$T_1 = rl_1(A); u_1(A); c_1$
$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$
$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$
$H_4 = rl_1(A); rl_2(B); u_1(A); wl_2(A); u_2(B); u_2(A); rl_3(A) ; c_1; wl_3(B); u_3(A); c_2 ; u_3(B); c_3$
H_4 ahora es 2PL!!
H_4 es 2PL estricto?
No!!!
H_4 es 2PL riguroso?

Enunciado 2: Bonus Track!

- Con el nuevo conjunto de transacciones podría generar una historia 2PL

$T_1 = rl_1(A); u_1(A); c_1$
$T_2' = rl_2(B); wl_2(A); u_2(B); u_2(A); c_2$
$T_3' = rl_3(A) ; wl_3(B); u_3(A); u_3(B) ; c_3$
$H_4 = rl_1(A); rl_2(B); u_1(A); wl_2(A); u_2(B); u_2(A); rl_3(A) ; c_1; wl_3(B); u_3(A); c_2 ;$
$u_3(B); c_3$
H_4 ahora es 2PL!!
H_4 es 2PL estricto?
No!!!
H_4 es 2PL riguroso?
No!!!