Parcialito 5

1. Para las siguientes planificaciones:
   - Dibujar los grafos de precedencia
   - Listar los conflictos
   - Determinar cuáles son serializables

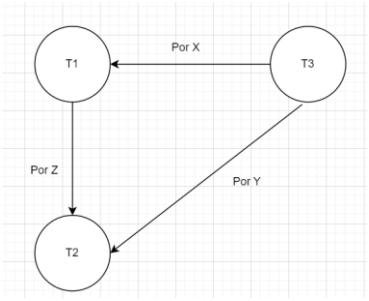
a) `bT1; bT2; bT3; RT1(X); RT2(Z); RT1(Z); RT3(X); RT3(Y); WT1(X); WT3(Y); RT2(Y); WT2(Z); WT2(Y); cT1; cT2; cT3;`

b) `bT1; bT2; bT3; RT1(X); RT2(Z); RT3(X); RT1(Z); RT2(Y); RT3(Y); WT1(X); WT2(Z); WT3(Y); WTY2(Y); cT1; cT2; cT3;`

c) `bT1; bT2; bT3; bT4; RT1(A); RT2(B); RT3(C); RT4(A); WT1(A); RT2(A); WT3(C); RT4(C); WT2(B); WT4(A); RT1(C); WT4(C); cT1; cT2; cT3; cT4;`

a)

bT1; bT2; bT3; RT1(X); RT2(Z); RT1(Z); RT3(X); RT3(Y); WT1(X); WT3(Y); RT2(Y); WT2(Z); WT2(Y); cT1; cT2; cT3



El grafo es acíclico, así que esta planificación ***es serializable***.

Ejecución serial equivalente:

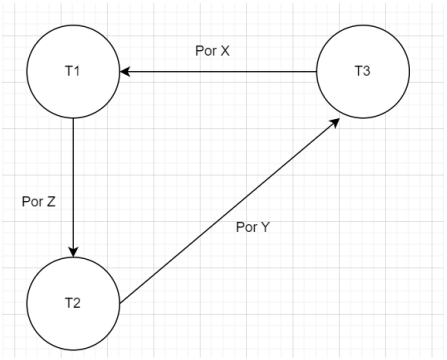
T3, T1, T2

Conflictos:

- ✓ $R_{T3}(X)$ y$W_{T1}(X)$.
- ✓ $R_{T1}(Z)$ y $W_{T2}(Z)$.
- ✓ $W_{T3}(Y)$ y $R_{T2}(Y)$.
- ✓ $R_{T3}(X)$ y $W_{T1}(X)$.

b)

bT1; bT2; bT3; RT1(X); RT2(Z); RT3(X); RT1(Z); RT2(Y); RT3(Y); WT1(X); WT2(Z); WT3(Y); WT2(Y); cT1; cT2; cT3;



no es serializable, hay un ciclo en el grafo.

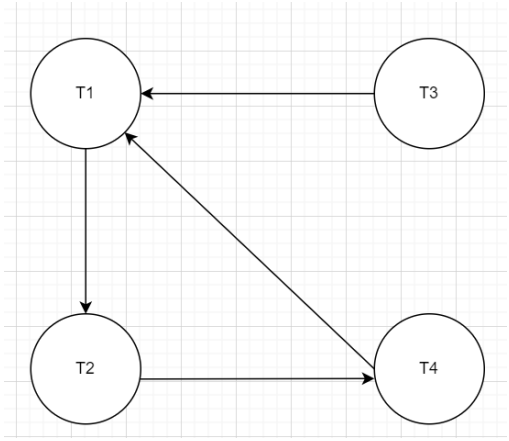
conflictos

- ✓ RT3(X) y WT1(X)
- ✓ RT1(Z) y WT2(Z)

- ✓ WT3(Y) y WT2(Y)

c)

bT1; bT2; bT3; bT4; RT1(A); RT2(B); RT3(C); RT4(A); WT1(A); RT2(A); WT3(C); RT4(C); WT2(B); WT4(A); RT1(C); WT4(C); cT1; cT2; cT3; cT4;



No es serializable porque el grafo es ciclico.

Conflictos

- ✓ RT4(A) y WT1(A)
- ✓ WT1(A) y RT2(A)
- ✓ RT2(A) y WT4(A)
- ✓ WT3(C) y RT4(C)
- ✓ RT4(C) y RT1(C)
- ✓ RT1(C) y WT4(C)

2. Dado el siguiente par de transacciones:

```
T1 : bT1 ; RT1(C); WT1(C); RT1(B); WT1(B); cT1

T2 : bT2 ; RT2(A); WT2(A); RT2(C); WT2(C); cT2
```

Se pide:

a) Coloque locks y unlocks a ambas transacciones de manera de respetar el Protocolo de Lock de 2 Fases, intentando a la vez minimizar el tiempo que las transacciones mantienen los locks sobre los recursos.

b) Defina que es ser recuperable. Indique si el solapamiento es recuperable, justificando su respuesta.

a)

T1	T2
Begin	Begin
L(C); R(C)	
W(C); L(B); U(C)	
	L(A); R(A)
	W(A); L(C); U(A)
R(B)	
W(B); U(B)	
	R(C)
	W(C); U(C)
Commit	Commit

b) Un solapamiento es recuperable si y solo si ninguna transacción T realiza el commit hasta tanto todas las transacciones que escribieron datos antes de que T los leyera hayan commiteado. Se busca que cuando se haga el commit, si leyó datos de otra transacción, esa otra haya commiteado antes de la lectura.

Para el ejemplo, dado, el solapammiento **no es recuperable** porque antes de que T2 leyera, T1 no hizo commit de los datos escritos en el item C.

3. Supongamos el siguiente log de un sistema que usa undo/redo logging. ¿Cuál es el valor de los ítems X, Y, Z, W, U, V y T en disco después de la recuperación si la falla se produce:

   a) Justo antes de la linea 19?
   b) Justo antes de la linea 24?
   c) Después de la linea 24?

Nro linea	log
1	<START T6>
2	<T6, X, 20, 80>
3	<START T7>
4	<T6, Y, 20, 150>
5	<T6, X, 80, 90>
6	<T7, Z, 30, 50>
7	<T7, W, 40, 70>
8	<COMMIT T6>
9	<START T8>
10	<T8, U, 50, 120>
11	<T7, W, 70, 100>
12	<START CKPT(T7,T8)>
13	<T7, Z, 50, 110>
14	<COMMIT T7>
15	<START T9>
16	<T9, V, 60, 140>
17	<T9, T, 70, 20>
18	<COMMIT T8>
19	<T9, V, 140, 200>
20	<START T10>
21	<T10, Z, 110, 180>
22	<END CKPT>
23	<T9, V, 200, 170>
24	<COMMIT T9>

3.1 Justo antes de la línea 19:

- X = 90
- Y = 150
- Z = 110
- W = 100
- U = 120
- V = 60
- T = 70

3.2 Justo antes de la línea 24:

- X = 90
- Y = 150
- Z = 110
- W = 100
- U = 120
- V = 60
- T = 70

3.3 Justo después de la línea 24:

- X = 90
- Y = 150
- Z = 110

- W = 100
- U = 120
- V = 170
- T = 20