PARCIALITO 5: Concurrencia y Recuperación

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

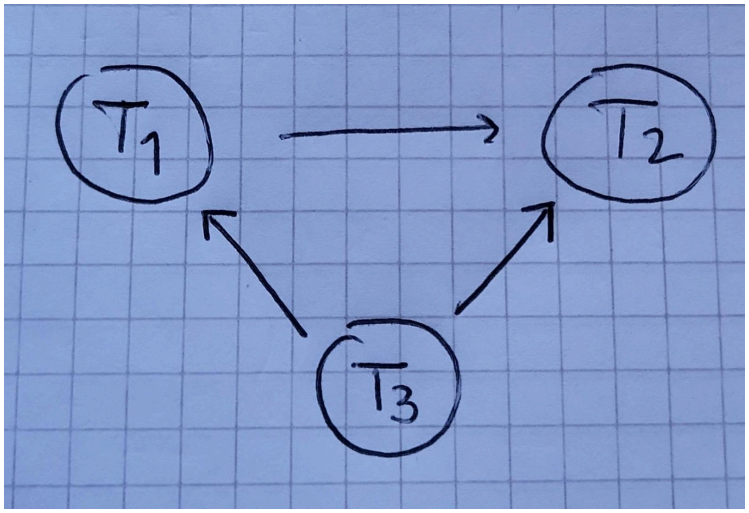
a. bT1; bT2; bT3; RT1(X); RT2(Z); RT1(Z); RT3(X); RT3(Y); WT1(X); WT3(Y); RT2(Y); WT2(Z); WT2(Y); cT1; cT2; cT3;

Vemos que las secuencias quedan de la forma:

X: RT1(X) RT3(X) WT1(X)
Y: RT3(Y) WT3(Y) RT2(Y) WT3(Y)
Z: RT2(Z) RT1(Z) WT2(Z)

<u>Entonces los conflictos son:</u>
X: T3 => T1 de tipo RW, T1 escribe un valor que lee T3
Y: T3 => T2 de tipos RW, WR, WW
Z: T1 => T2 de tipo RW

Y el grafo queda =>



Y la secuencia es serializable porque no hay ciclos.

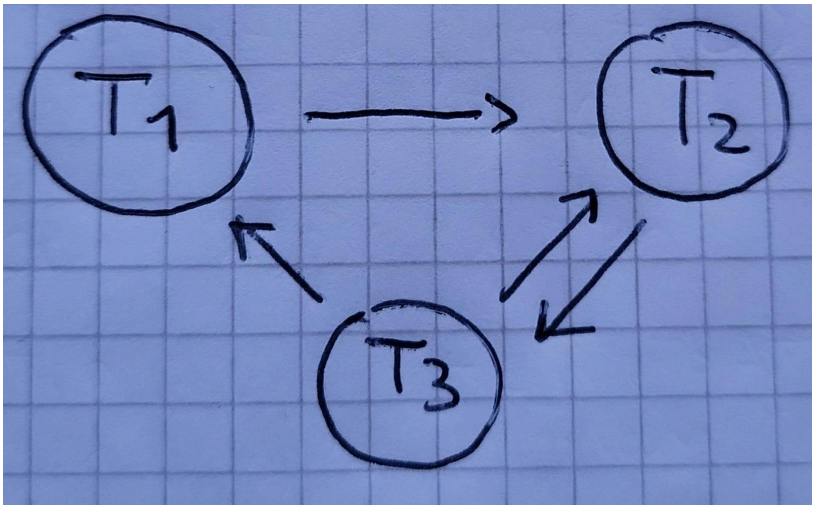
b. bT1; bT2; bT3; RT1(X); RT2(Z); RT3(X); RT1(Z); RT2(Y); RT3(Y); WT1(X); WT2(Z); WT3(Y); WT2(Y); cT1; cT2; cT3;

Vemos que las secuencias quedan de la forma:

X: RT1(X) RT3(X) WT1(X)
Y: RT2(Y) RT3(Y) WT3(Y) WT2(Y)
Z: RT2(Z) RT1(Z) WT2(Z)

<u>Los conflictos son:</u>
X: T3 => T1 de tipo RW
Y: T2 => T3 de tipo RW y T3 => T2 de tipos RW y WW
Z: T1 => T2 de tipo RW

Y el grafo queda =>



Y la secuencia no es serializable ya que hay ciclos.

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;

Vemos que las secuencias quedan de la forma:

A: RT1(A) RT4(A) WT1(A) RT2(A) WT4(A)
B: RT2(B) WT2(B)
C: RT3(C) WT3(C) RT4(C) RT1(C) WT4(C)

<u>Los conflictos son:</u>
A:

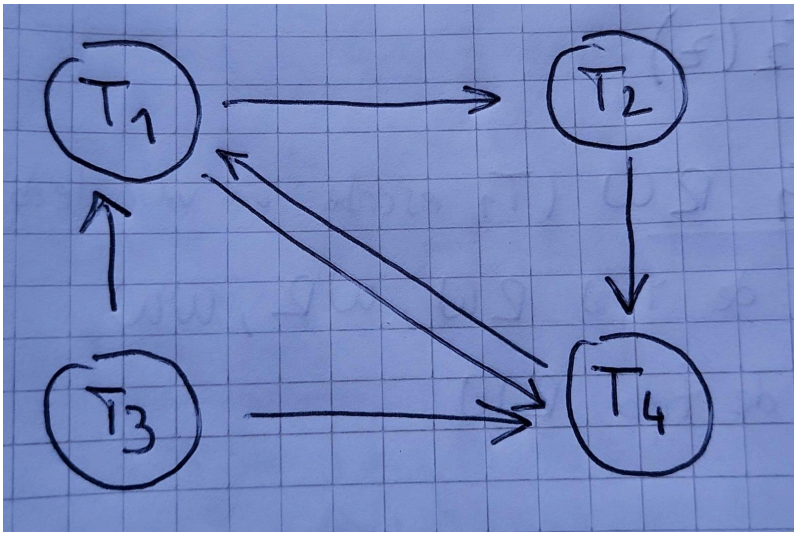
- T4 => T1 de tipo RW
- T1 => T4 de tipos RW, WW
- T1 => T2 de tipo WR
- T2 => T4 de tipo RW

B: no hay

C:

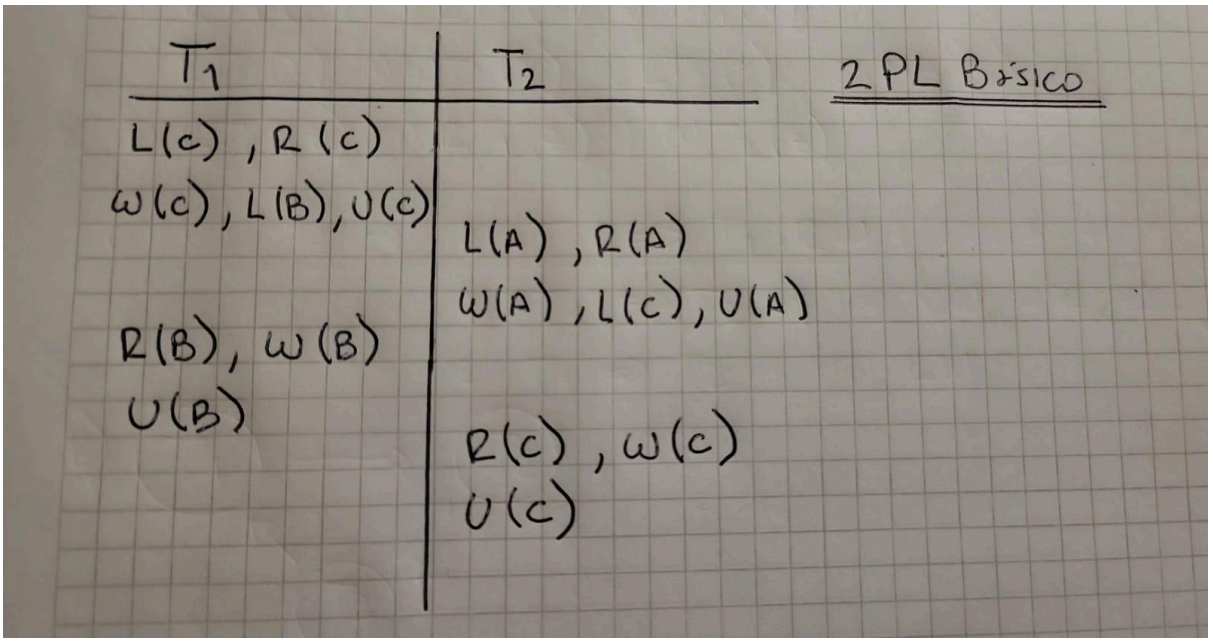
- T3 => T4 de tipos RW, WR y WW
- T1 => T4 de tipo RW
- T3 => T1 de tipo WR

Y el grafo queda =>



Y no es serializable porque tiene ciclos.

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.



Y el resultado es el mismo que una ejecución serial.

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

Un solapamiento es recuperable si ninguna transacción T realiza el commit hasta que todas las transacciones que escribieron datos que T leyó hayan commiteado. Para ver si este solapamiento es recuperable hay que ver si alguna transacción hizo commit antes de que la otra de la que leyó haga commit.
En nuestro ejercicio hay una disputa por la lectura y escritura sobre la variable C, ya que T1 y T2 quieren leer y escribir sobre ella, primero lo hace T1 mientras que T2 lee y escribe sobre A, luego T1 suelta el lock y libera C, para luego leer y escribir sobre B y por último commitear. Luego C es tomada por T2 que lee C después de que T1 haya commiteado. Vemos entonces que el solapamiento es recuperable porque las dos transacciones cumplen con la condición de que ninguna tx T realiza el commit hasta que todas las txs que escribieron datos que T leyó hayan commiteado.

3. Supongamos el siguiente log de un sistema que usa undo/redo logging. ¿Cual 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 línea 19?

Vemos los cambios que realizan cada transacción:
T6:

- Start en línea 1
- Cambia X de 20 a 80 (línea 2)
- Cambia Y de 20 a 150 (línea 4)
- Cambia X de 80 a 90 (línea 5)
- Commit (línea 8)
- Entonces en disco queda: X=90, Y=150

T7:

- Start en línea 3
- Cambia Z de 30 a 50 (línea 6)
- Cambia W de 40 a 70 (línea 7)
- Cambia W de 70 a 100 (línea 11)
- Cambia Z de 50 a 110 (línea 13)
- Commit (línea 14)
- Entonces en disco queda: Z=110, W=100

T8:

- Start en línea 9
- Cambia U de 50 a 120 (línea 10)
- Commit (línea 18)
- En disco queda: U=120

T9:

- Start en línea 15
- Cambia V de 60 a 140 (línea 16)
- Cambia T de 70 a 20 (línea 17)
- Cambia V de 140 a 200 (línea 19, la falla ocurre justo antes de esta línea)

No hay commit de la transacción T9 antes de la línea 19, por lo tanto los cambios no se reflejan en disco todavía, V queda en 60 y T en 70.

Entonces queda:

X: 90

Y: 150

Z: 110

W: 100

U: 120

V: 60

T: 70

b. Justo antes de la línea 24?

Entre las líneas 19 y 24 no hay commits, por lo que las variables quedan de la misma forma que en el punto anterior. Igualmente entre estas líneas hay algunos cambios en las transacciones:

T9:

- Cambia V de 200 a 170 (línea 23)

T10:

- Start en línea 20
- Cambia Z de 110 a 180 (línea 21)

Pero nunca hay commit de la T10 así que estos cambios nunca se van a ver reflejados.

c. Después de la línea 24?

Si se produce una falla luego de la línea 24, ahora si vemos cambios en las variables, porque en la línea 24 se hace el commit de la T9, entonces las variables quedan:

X: 90

Y: 150

Z: 110

W: 100

U: 120

V: 200

T: 20

Los cambios de la transacción 10 nunca se ven reflejados porque no vemos el commit.