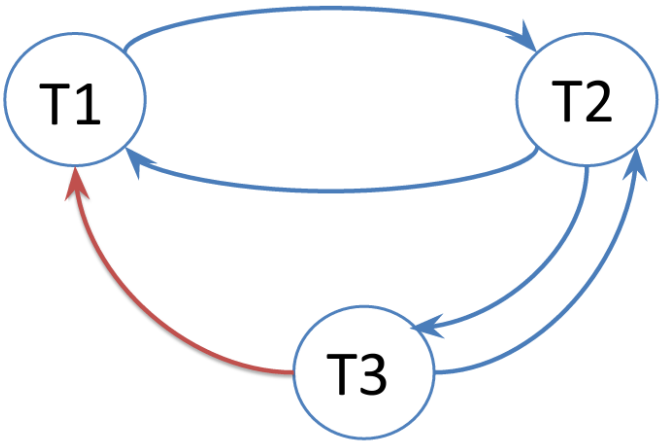
CONCURRENCIA Y TRANSACCIONES

Ejercicio: comprobar si es serializable

S_1: $r_1(X)$; $r_2(Z)$; $r_1(Z)$; $r_2(X)$; $w_2(X)$; $w_1(Z)$; $r_3(Y)$; $w_2(Y)$; $w_3(Y)$; $r_1(Y)$;



65

Protocolos de control de concurrencia

2PL básico: *Tenemos dos fases, en la primera adquirimos locks, en la segunda los liberamos.*

2PL Conservador: *Bloquea con antelación todos los recursos.*

Estricto S2PL: *Una transacción no puede adquirir un lock luego de haber liberado uno que había adquirido, y los locks de escritura sólo pueden ser liberados después de haber commiteado la transacción.*

Riguroso R2PL: *no diferencia los tipos de locks. Los locks sólo pueden ser liberados después del commit.*

Organice las siguientes transacciones usando bloqueo de dos fases básico (2PL) y conservador, con bloqueos que pueden ser exclusivos (escritura) o compartidos (lectura), minimizando el tiempo de atención (o inicio) promedio de cada transacción. ¿Qué pasaría en cada caso?

1) T1=(r(A),w(A),r(B),w(B),r(C),w(C))
T2=(r(B),w(B),r(C),w(C),r(A),w(A))
T3=(r(C),w(C),r(D),w(D),r(B),w(B))

T1	T2	T3
L(A), L(B), L(C)		
R(A), W(A)		
R(B), W(B)		

U(A), U(B)		
	L(B)	
	R(B), W(B)	
R(C), W(C)		
U(C)		
	L(A), L(C)	
	R(C), W(C)	
	U(C), U(B)	
		L(C), L(D), L(B)
		R(C), W(C)
		R(D), (W(D)
		R(B), W(B)
		U(C), U(D), U(B)
	R(A), W(A)	
	U(A)	

2) T1=(r(A),r(B),w(A),r(C),w(C))
T2=(r(B),r(A),w(A),w(B))

T1	T2
L(A), L(B), L(C)	
R(A)	
R(B)	
U(A), U(B)	
	L(B), L(A)
R(C), W(C)	R(B)
U(C)	R(A), W(A)
	W(B)
	U(A), U(B)

3) T1=(r(A),r(B),w(B),w(C))
T2=(r(D),r(E),w(E),w(D))
T3=(r(B),r(D),w(B),w(D))
T4=(w(A),w(B),w(C),w(D))

T1	T2	T3	T4
L(A), L(B), L(C)			
R(A),	L(D), L(E)		
R(B); W(B)	R(D)		
R(C)	R(E), W(E)		
U(A), U(B), U(C)	W(D)		
	U(D), U(E)		
		L(B), L(D)	L(A), L(C)
		R(B)	W(A)
		R(D)	
		W(B)	
		W(D)	
		U(B), U(D)	
			L(B), L(D)
			W(B)

			W(C)
			W(D)
			U(A), U(B), U(C), U(D)

Ejercicio de parcial

1. Dado el siguiente solapamiento de transacciones:

bT1 ; bT2 ; bT3 ; WT3(Y); WT3(X); WT1(Y); cT3 ; RT2(Y); RT2(X); cT2 ; WT1(X); cT1

a) Dibuje el grafo de precedencias del solapamiento.

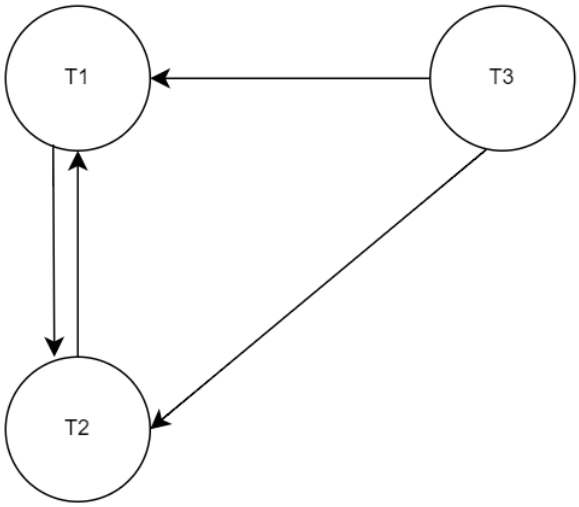
b) Indique si el solapamiento es serializable. Justifique su respuesta.

c) Indique si el solapamiento es recuperable. Justifique su respuesta.

a. bT1; bT2; bT3; WT3(Y); WT3(X); WT1(Y); cT3; RT2(Y), RT2(X); cT2; WT1(X); cT1

conflictos

- WT3(Y), WT1(Y)
- WT3(Y), RT2(Y)
- WT3(X), RT2(X)
- WT3(X), WT1(X)
- WT1(Y), RT2(Y)



b. El solapamiento no es serializable pues el grafo de precedencias es cíclico
c. NO es recuperable porque se realizan lecturas sin que antes una transacción que haya escrito el elemento leído haya comitteado.

La transacción T1 escribe Y, y la T2 lee Y antes de que T1 se comitte.

2. Considere el siguiente solapamiento de 3 transacciones:

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

Se pide:

1) Explique si es posible que este solapamiento ocurra utilizando el protocolo de lock de dos fases (2PL). Para ello, intente colocar locks L() y unlocks U() respetando el protocolo, y analice si ello es factible.

2) Indique si el solapamiento es serializable, justificando su respuesta.

3) Indique si el solapamiento es recuperable, justificando su respuesta.

4) Indique si es posible que este solapamiento se haya producido utilizando el mecanismo de control de concurrencia Snapshot Isolation. Si es posible, entonces indique cuál sería un orden serial equivalente a esta ejecución.

1.

T1	T2	T3
L(x), L(Y)		
R(X)		
R(Y)		
U(Y)		
	L(Y), L(Z)	
	R(Y)	
	R(Z)	
W(X)		
U(X)		
	U(Z), U(Y)	
		L(Z), L(X)
		R(Z)
		R(X)
	W(Z)	

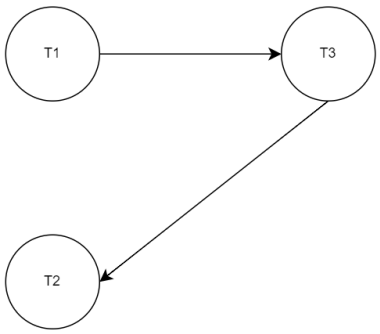
ERROR: yo ya no puedo escribir Z desde la transacción 2 porque libere el lock, para que pueda ser utilizada por la transacción 3, y dado el protocolo 2PL, yo no puedo volver a adquirir el lock sobre Z para T2 porque ya pase la fase de liberación de locks.

Por lo tanto, no es posible que este solapamiento ocurra usando el protocolo de lock de dos fases.

2. Para analizar la seriabilidad, busco los conflictos y armo el grafo de precedencias.

Conflictos:

- WT1(X), RT3(X)
- RT3(Z), WT2(Z)



el solapamiento es serializable, pues el grafo de precedencias es acíclico.

3. El solapamiento NO es recuperable porque la transacción 3 lee el item X, que fue previamente escrito por la T1, sin que T1 commitee antes de la lectura de T3

3. Se dice que una transacción T1 realiza una Lectura no Repetible ó Unrepeatable Read cuando la misma lee un ítem X, luego otra transacción T2 escribe ese mismo ítem, y posteriormente T1 vuelve a leer el ítem, encontrando un valor distinto al anteriormente leído. Indique si las siguientes afirmaciones sobre la Lectura no Repetible son verdaderas ó falsas. Justifique cada una de sus respuestas.

1) En un solapamiento recuperable puede ocurrir una Lectura no Repetible.

2) En un solapamiento que evita rollbacks en cascada puede ocurrir una Lectura no Repetible.

3) Aplicando el Protocolo de Lock de Dos Fases (2PL) puede ocurrir una Lectura no Repetible.

4) Bajo el nivel de aislamiento Read Committed definido en el estándar SQL puede ocurrir una Lectura no Repetible.

1. VERDADERO: porque el solapamiento recuperable lo que garantiza es que si T2 lee un dato escrito por T1, T2 no se committee antes que T1. No cubre estrictamente que T1 escriba un item, T2 lo lea y modifique, y luego T1 lo vuelva a leer.

2. VERADERO: Para evitar hacer rollbacks en cascada, es necesario que una transacción no lea valores que aún no fueron commiteados. Esto evita los conflictos de tipo (WTi (X); RTj (X)) pero no cubre la anomalia de actualizacion perdida y por lo tanto tampoco la de lectura no repetible

3. FALSO: si T2 adquirio el lock para escribir el item X, Significa que T1 ya había liberado el item con un U(X), y por el protocolo 2PL, no puede volver a adquirir el lock sobre X para volver a leer el item una vez que fue liberado.

4. VERADERO: Los cambios confirmados son visibles dentro de otra transacción, esto significa que dos consultas dentro de una misma transacción pueden retornar diferentes resultados.