Ejemplos de transacciones
(Bases de datos)

Sean dos transacciones:

T1: W(A), W(B), R(C)
T2: W(B), W(C), R(A)

En base a T1 y T2 genere:

- Dos planes seriales distintos.
- Un plan serializable pero no serial.
- Un plan con hambre según MT básico.
- Un plan con deadlock según 2PL.
- Un plan serializable por conflicto.

Solución

Planes seriales

Un plan serial es aquel que ocurre en serie... Los dos planes posibles son:

Plan 1

T1	T2
W(A)	
W(B)	
R(C)	
	W(B)
	W(C)
	R(A)

Plan 2

T1	T2
	W(B)
	W(C)
	R(A)
W(A)	
W(B)	
R(C)	

Sólo hay dos planes posibles porque sólo hay dos transacciones. Los órdenes fueron (T1,T2) y (T2,T1). En general, N transacciones tienen N! maneras seriales de ser ordenados.

Serializable y no serial

Debe haber equivalencia con un plan serial y no ser tal. O sea, T1 y T2 deben estar intercalados.

¿Qué tal si tomamos el Plan 1 serial e intercalamos T1:R(C) con T2:W(B)? Ojo que T1 ya realizó todo cambio posible. Y R(C) no afecta a T2 en términos de información aunque presente riesgo en un conflicto lectura-escritura.

Plan propuesto

T1	T2
W(A)	
W(B)	
	W(B)
R(C)	
	W(C)
	R(A)

Este plan es equivalente al plan serial 1; no hay interferencia entre T1 y T2.

La noción de equivalencia hace depender qué se entiende por planes equivalentes. Por ejemplo, equivalencia por **resultados** y equivalencia por **orden**. Resultados hace alusión a la noción clásica.

Marcas de tiempo

Los algoritmos de marcas de tiempo asignan valores a las transacciones, según los cuales deciden qué transacción/operación ejecutar.

*No se ha dicho nada acerca de cuándo ocurriría cada operación. Para demostrar el algoritmo básico de marcas de tiempo (BTS), ordenaremos de tal manera de realizar una operación de **abort**. El abort debe ocurrir repetidamente para la hambruna. T2 muere. Luego T1 muere. T2. T1. Etc.*

El plan sería:

T1	T2
W(A)	
	W(B)
W(B)	
Abort	
	W(C)
W(A)	
	R(A)
	Abort
	W(B)
W(B)	
Abort	
	W(C)
W(A)	
...	...

Esta hambruna ocurre como un deadlock. No es cierto que las dos transacciones estén en espera de la otra, pero sí que no pueden terminar su ejecución.

Para ayudar a la comprensión del problema, enumere las transacciones como naturales: 1, 2, 3... Dentro del ejemplo, debiera llegar hasta 5.

¿Cómo mejorar marcas de tiempo básico? Hay varias maneras. Una es postergar la transacción que produce conflicto (*rec. la auxiliar*). Otro es usar multiversiones. Esas son las mejoras básicas... pero hay muchas variantes. Por ejemplo, dar prioridad a las transacciones que fueron abortadas.

Deadlock con 2PL

La versión básica de bloqueo de dos fases es fácil de hacer llevar a deadlock. Basta realizar bloqueos cruzados. La única restricción de 2PL es que primero se deben tomar los bloqueos. Luego se comienza a desbloquear.

T1	T2
L(A)	
W(A)	
	L(B)
	W(B)
L(B)	
espera T2	L(A)
	espera t1

Las esperas no acabarán. Ninguna transacción perderá los bloqueos que ha logrado.

El uso de *timeouts* evita muchos casos en que 2PL cae en deadlock. Sin embargo, hay técnicas de bloqueo que impiden los deadlock por completo.

Serializable por conflicto

Como sabemos, 2PL asegura serializabilidad por conflicto, así también los seriales.

Veamos un 2PL corriente (2PL estricto resulta equivalente a un plan serial) para variar:

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