



ELEMENTOS DE BASES DE DATOS  
Segundo Cuatrimestre de 2013  
Trabajo Práctico N° 8  
**Protocolos para Control de Concurrencia**

## Ejercicios

- ¿Por qué los SDBD con manejo de transacciones requieren de Protocolos de Control de Concurrencia?
- Definir
  - Bloqueo (*lock*).
  - Bloqueo compartido (*lock-S*).
  - Bloqueo exclusivo (*lock-X*).
  - Compatibilidad entre tipos de bloqueo
- Dadas las siguientes transacciones:

i)

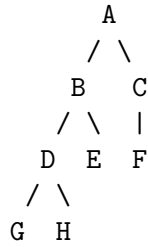
T <sub>1</sub>	T <sub>2</sub>
Read(A)	Read(B)
Write(A)	Read(A)
Read(B)	Write(B)

ii)

T <sub>3</sub>	T <sub>4</sub>	T <sub>5</sub>
Read(A)	Read(A)	Read(B)
Write(A)	Read(B)	Write(B)
Read(B)		

Para los incisos i) y ii) se pide:

- Encontrar, si es posible, una planificación concurrente resultante de aplicar el protocolo de dos fases con sólo locks exclusivos.
  - Encontrar, si es posible, una planificación concurrente resultante de aplicar el protocolo de dos fases con posibilidades de locks exclusivos y compartidos.
  - Encontrar, si es posible, una planificación concurrente resultante de aplicar el protocolo de dos fases con posibilidades de locks compartidos, upgrade y downgrade.
  - Encontrar una planificación concurrente resultante de aplicar alguno de los protocolos de dos fases que resulte en deadlock.
  - Encontrar una planificación concurrente resultante de aplicar a los requerimientos de entrada un protocolo de locking sin imponer dos fases. Es serializable? Es posible llegar a una planificación no serializable?
  - Analizar las diferentes alternativas de bloqueo para los protocolos de dos fases en cuanto a serializabilidad de las planificaciones, nivel de concurrencia y posibilidad de deadlock.
- ¿Qué ventajas y desventajas proporciona la alternativa de bloqueo de dos fases estricto?
  - ¿El protocolo de bloqueo garantiza planificaciones serializables? Justificar.
  - Dada la siguiente transacción: T<sub>0</sub> = req(C); req(D); req(G) (por *req(X)* entiéndase, requiere X) y el siguiente grafo de precedencia:



Mostrar la secuencia de locks necesarios para poder ejecutar  $T_0$ .

7. Analice los protocolos basados en árbol en cuanto los siguientes puntos. Justifique.
  - Serializabilidad de las planificaciones.
  - Nivel de concurrencia.
  - Posibilidad de deadlock.
  - Posibilidad de inanición.
8. ¿Qué representa la estampilla de una transacción?.
9. ¿Los protocolos de control de concurrencia basados en estampillas de tiempo resuelven el problema de *deadlock*? ¿Tienen nuevos problemas que no tengan los protocolos basados en bloqueos?.
10. Dadas las siguientes transacciones:

$T_0 = \text{start}; \text{read}(B); \text{write}(B); \text{read}(A); \text{write}(A); \text{finish}$

$T_1 = \text{start}; \text{read}(B); \text{read}(C); \text{write}(A); \text{write}(C); \text{finish}$

$T_2 = \text{start}; \text{read}(A); \text{write}(A); \text{write}(B); \text{finish}$

Con  $0 < \text{ts}(T_0) < \text{ts}(T_1) < \text{ts}(T_2)$  y  $R\text{-ts}(A) = R\text{-ts}(B) = R\text{-ts}(C) = W\text{-ts}(A) = W\text{-ts}(B) = W\text{-ts}(C) = 0$

- a) Encontrar, si es posible, una planificación concurrente donde al menos una de las transacciones retroceda aplicando el algoritmo de estampilla de tiempos.
- b) Encontrar, si es posible, una planificación concurrente donde ninguna transacción retroceda aplicando el algoritmo de estampilla de tiempos.
- c) Encontrar, si es posible, una planificación concurrente donde deba retroceder en cascada más de una transacción aplicando el algoritmo de estampilla de tiempos.
- d) Encontrar, si es posible, una planificación concurrente donde la falla de una transacción resulte en una planificación no recuperable.
- e) Encontrar, si es posible, una planificación donde se pueda aplicar la regla de escritura de Thomas.

Para mostrar las planificaciones encontradas utilice, por ejemplo, el siguiente formato:

	$T_0$	$T_1$	$T_2$	Estampillas
1.	start			
2.		start		
3.		Read(B)		$\langle B, R\text{-ts}=\text{ts}(T_1), W\text{-ts}=0 \rangle$
4.	...	...	...	...

11. Para la siguiente planificación de entrada:

	$T_0$	$T_1$	$T_2$
1.	Read(A)		
2.		Write(A)	
3.	Write(A)		
4.			Write(A)

Con  $0 < ts(T_0) < ts(T_1) < ts(T_2)$  y  $R-ts(A)=R-ts(B)=R-ts(C)=W-ts(A)=W-ts(B)=W-ts(C)=0$ .

- a) Verificar si es serializable en conflictos y en vistas.
- b) ¿Puede resultar esta planificación de aplicar el protocolo de dos fases?
- c) ¿Puede resultar esta planificación de aplicar el protocolo de estampillas de tiempo tradicional?
- d) ¿Puede resultar esta planificación de aplicar el protocolo de estampillas con la regla de escritura de Thomas?

**Nota:** Para los protocolos de estampillas, indique para cada instante de tiempo como se actualizan las estampillas correspondientes. Por ejemplo: 1.  $\langle A, R-ts=ts(T_0), W-ts=0 \rangle$ .

12. Considere la siguiente planificación para las transacciones  $T_1, T_2$  y  $T_3$ , con  $ts(T_1) = 1, ts(T_2) = 2$  y  $ts(T_3) = 3$ , y suponga que inicialmente existen las siguientes versiones:  
 $\langle A_0, 11, R-ts=0, W-ts=0 \rangle, \langle B_0, 12, R-ts=0, W-ts=0 \rangle$

	$T_1$	$T_2$	$T_3$
1.	read(A)		
2.	read(B)		
3.		read(A)	
4.		$A:=A+10$	
5.		write(A)	
6.			read(A)
7.	$B:=A+B$		
8.	write(B)		
9.		$B:=A$	
10.		write(B)	
11.			read(B)

- a) Analizar el resultado de aplicar el protocolo de multiversión.
- b) Encontrar, si el posible, otra planificación que al aplicar el protocolo de multiversión resulte en algún retroceso.

**Nota:** Indique para cada instante de tiempo como se actualizan las estampillas y los valores de las versiones correspondientes. Por ejemplo: 1.  $\langle A_0, 11, R-ts=ts(T_0), W-ts=0 \rangle$

13. Para las transacciones  $T_0$ ,  $T_1$  y  $T_2$  y la siguiente planificación, analizar el resultado de aplicar el protocolo de validación. **Nota:** Las operación Write modifica los datos locales de la transacción y la operación *output* vuelca los valores de estos datos en la base de datos.

$T_0$	$T_1$	$T_2$
<i>start</i>		
Read(A)		
Read(B)		
	<i>start</i>	
	Read(A)	
Write(B)		
	Read(C)	
		<i>start</i>
		Read(A)
	Write(B)	
		Read(B)
<i>valid</i>		
<i>output(B)</i>		
<i>finish</i>		
	<i>valid</i>	
	<i>output(B)</i>	
	<i>finish</i>	
		<i>valid</i>
		<i>finish</i>

14. Protocolo de concurrencia por validación. Suponga que existen las transacciones  $T_1$ ,  $T_2$  y  $T_3$  y los elementos de base de datos A, B y C. Los conjuntos de lectura (RS) y escritura (WS) para las transacciones son los siguiente:

$$\begin{aligned}
 RS(T_1) &= \{A, B\} & WS(T_1) &= \{B\} \\
 RS(T_2) &= \{A, C\} & WS(T_2) &= \{A\} \\
 RS(T_3) &= \{C\} & WS(T_3) &= \{B, C\}
 \end{aligned}$$

Utilizaremos  $S_i, V_i$  y  $F_i$  para representar cuando una transacción  $T_i$  comienza, intenta validar y termina respectivamente. Dada la siguiente secuencia de eventos:  $S_1 S_2 V_1 S_3 V_3 F_1 V_2 F_2 F_3$ , Determine cuales transacciones validan y cuales retroceden, justificando en cada caso.

15. Para las transacciones:

$T_0 = \text{read}(A); \text{read}(B); A := A+10; \text{write}(A); B := B+20; \text{write}(B)$

$T_1 = B := 22; \text{write}(B); C := 23; \text{write}(C)$

$T_2 = \text{read}(C); \text{read}(B); C := C + 10; \text{write}(C); B := B+20; \text{write}(B)$

Encontrar una planificación concurrente (no en serie) resultante de aplicar:

- el protocolo de bloqueo de dos fases.
- el protocolo de estampillas.
- el protocolo de estampillas con regla de escritura de Thomas.
- el protocolo de árbol. Defina un árbol de precedencia a su elección.
- el protocolo de multiversión. Suponga que inicialmente existen las siguientes versiones:  $\langle A_0, 11, R\text{-ts}=0, W\text{-ts}=0 \rangle$ ,  $\langle B_0, 12, R\text{-ts}=0, W\text{-ts}=0 \rangle$ ,  $\langle C_0, 13, R\text{-ts}=0, W\text{-ts}=0 \rangle$
- el protocolo de validación.

16. Para las transacciones  $T_1$  y  $T_2$  que se muestran a continuación, con valor inicial de  $A = 100$ ,  $B = 200$ ,  $C = 300$  y  $D = 400$ .

$T_1 = \text{Read}(B); B = B * 4; \text{Write}(B); \text{Read}(C); \text{Read}(D); C = B + D; \text{Write}(C)$

$T_2 = \text{Read}(A); A = A + 100; \text{Write}(A); \text{Read}(B); \text{Read}(C); B = C; \text{Write}(B)$

- a) Dar una planificación concurrente serializable usando el protocolo de bloqueos de dos fases con locks compartidos y exclusivos que no caiga en deadlock. ¿Cuál es la serie equivalente? ¿Cuáles son los valores finales de A, B, C y D?
- b) Dar una planificación concurrente serializable usando el protocolo de bloqueos de dos fases con locks compartidos y upgrade que caiga en deadlock. Indicar claramente el punto donde se produce el deadlock.
- c) ¿Es posible encontrar una planificación concurrente serializable usando el protocolo de dos fases estricto?