Bases de Datos
Práctico 8: Buenas Prácticas para la Serializabilidad

G. Saiz - N. Wolovick

6 de noviembre de 2000

1. Sean las transacciones

$$T_1 : r_1[x]w_1[x]c_1 \qquad T_2 : w_2[x]r_2[z]w_2[z]c_2 \qquad T_3 : r_3[y]r_3[z]w_3[y]r_3[x]w_3[z]w_3[x]c_3$$

   Dar planificaciones *2PL* no necesariamente estrictas, mostrando locks y unlocks, para lograr:

   (a) Una planificación SR.

   (b) Un deadlock.

2. Vimos como a través de las propiedades P_1, P_2, P_3 se caracterizaban las transacciones e historias *2PL*. Una transacción se dice *bien formada* cuando cumple con P_1, y es *dos fases* si cumple con P_3, mientras que una historia que cumple con P_2 se dice que *respeta locks*.

   (a) Dar una transacción bien formada pero no en dos fases y otra que no esté bien formada, pero si sea de dos fases.

   (b) Demuestre que si H no está *bien formada* o no es *dos fases*, entonces es posible escribir una transacción bien formada y en dos fases que produzca una planificación no serializable. Siempre supondremos que se cumple con la propiedad P_2.

3. Para las dos transacciones que se muestran a continuación[1]:

$$T_1 = r_1[x]w_1[x]r_1[y]w_1[y] \qquad T_2 = r_2[y]w_2[y]r_2[x]w_2[x]$$

   ¿Cuántos historias *2PL* totalmente ordenadas hay?

4. Demuestre *2PL* $\subsetneq CSR$

5. En tipos de lock $\{r, w, ir, iw, riw\}$ del MGL se dice que q "*es más fuerte que*" p cuando:

$$p \ll q \equiv (\forall o \cdot \mathit{conflict}(o, q) \Rightarrow \mathit{conflict}(o, p))$$

   La relación $\ll$ define un orden parcial entre los tipos de locks. Obtenga el reticulado asociado a $\ll$ y *la tabla de conversión de locks* de doble entrada, donde cada celda es el valor de la función $sup(p, q) = p \vee q$.

6. Dado el protocolo MGL para grafos de instancia que son árboles, suponga que ahora se permite a una transacción liberar un lock sobre un item de datos x antes de liberar un lock sobre un hijo de x. Asumiendo que el planificador es *2PL*, obtenga una ejecución incorrecta que pueda surgir a partir de esta modificación.

[1] Prescindimos de los c_i de estas transacciones.

7. Sea $S \subseteq P$ un subconjunto de procesos y $\rightarrow$ la relación *espera-a*, entonces decimos que S está en *deadlock* cuando $(\forall x \cdot x \in S \cdot (\exists y \cdot y \in S \wedge x \neq y \cdot x \rightarrow y))$. Bajo esta definición, demostrar que $\{T_1, \ldots, T_n\}$ está en deadlock sii *WFG* contiene un ciclo.

8. Demostrar que en D2PL un *deadlock fantasma* sólo puede ocurrir debido a abortos espontáneos[2].

9. En el esquema de prevención de deadlocks para *D2PL* cambiamos la definición de *Wait-Die* a:
if $ts(T_i) > ts(T_j)$ **then** T_i espera **else** T_i aborta
¿Este método sufre de deadlock? ¿Y de livelock? Respalde su respuesta con pruebas o contraejemplos.

10. Para la historia
$$H : r_2[x]r_3[x]w_2[y]w_3[x]r_1[y]r_4[y]r_1[x]w_1[z]w_4[x]$$
decidir cuando y cuales de las cuatro transacciones aborta (si es que alguna lo hace) teniendo en cuenta que el planificador es TO y las estampillas de tiempo son:

    (a) $ts(T_1) = 300, ts(T_2) = 310, ts(T_3) = 320, ts(T_4) = 330$
    (b) $ts(T_1) = 250, ts(T_2) = 200, ts(T_3) = 210, ts(T_4) = 275$

    Indique en cada caso el valor final del arrgelo `maxSch` para todos los items de datos y operaciones.

11. Demuestre que TO y $2PL$ son *incomparables* ($TO \nsubseteq 2PL \wedge 2PL \nsubseteq TO$). Para esto encuentre historias H_1 y H_2 tales que $H_1 \in TO \wedge H_1 \notin 2PL$ y $H_2 \in 2PL \wedge H_2 \notin TO$.

12. Demuestre que TO es una propiedad *pcc*.

13. Para la planificación H:

T_1	T_2	T_3
read(A)		
	write(A)	
write(A)		
		write(A)

    decidir si $H \in 2PL$, $H \in TO$, $H \in TO + TWR$

14. Proponga una modificación en el algoritmo del planificador TO básico para lograr que las historias resultantes sean ACA.

15. [3] *Caracterización de las historias 2PL y TO respecto a sus equivalentes seriales.*

    (a) En una historia *2PL*, se define el *punto de lock* de T_i cuando se han tomado todos los locks y no se ha soltado ninguno. Demuestre que si $H \in 2PL$ entonces $H \equiv_C H_S$ donde el orden de H_S es el de los *locked points*.
    (b) Demuestre que $H \in TO \Rightarrow H \equiv_C H_S$ donde el orden de H_S está dado por $ts(T_i)$.

16. [4]Sea $T_1 \rightarrow \ldots \rightarrow T_n \rightarrow T_1$ un ciclo en un *WFG*. Una arista $T_i \rightarrow T_j$ es una *cuerda* de un ciclo si T_i, T_j están en el ciclo pero $T_i \rightarrow T_j$ no es arista del ciclo. Un ciclo es *elemental* si no tiene cuerdas.

    Suponga que usamos un detector de deadlocks que encuentra todos los ciclos elementales del *WFG* y rompe cada uno de esos deadlocks parciales abortando una víctima en el ciclo. Demuestre que esto rompe todos los ciclos del *WFG*.

[2]Llegue a un absurdo suponiendo un ciclo en el grafo de espera del *GDD*, la ausencia de deadlock real y que ninguna transacción abortó.
[3]*/2
[4]*/3