Transacciones Pesimistas

DC - FCEyN - UBA

BBDD - 1C - 2017

Enunciado

Dadas las transacciones:
$T_1 = r_1(A); w_1(A); r_1(B); w_1(B); c_1$
$T_2 = r_2(A); w_2(A); c_2$
$T_3 = r(B)_3; r_3(A); w_3(A); w_3(B); c_3$
Y la historia:
$H_1 = r_1(A); w_1(A); r_2(A); r_1(B); w_1(B);$
$c_1; r(B)_3; w_2(A); r_3(A); c_2; w_3(A); w_3(B); c_3$
Se pide:

a. Construir el SG(H_1) (grafo de precedencia).

b. Indicar si H_1 es SR (serializable) y en caso afirmativo indicar las historias seriales equivalentes.

c. Dar una historia H_2 equivalente a la ejecución serial $T_2\,T_1\,T_3$, que sea SR y RC (recuperable) pero no ACA (ACA: evita aborts en cascada).

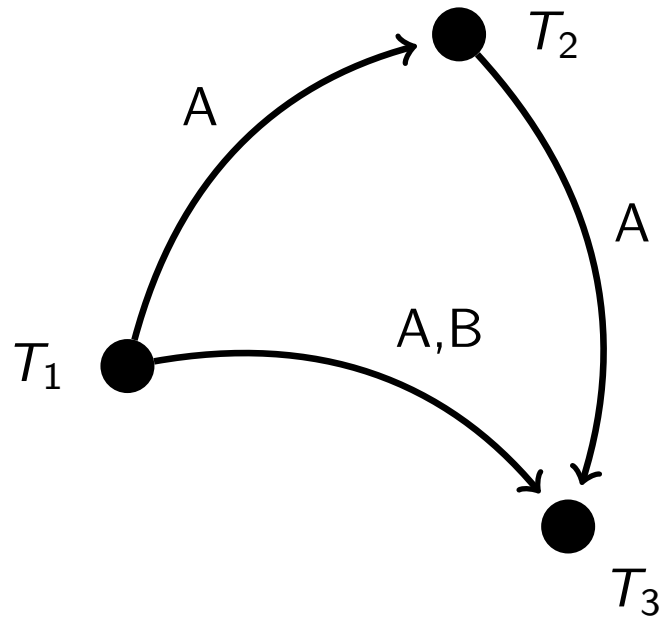
Grafo de precedencia

Pero primero... La forma Tabular!

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

Grafo de Precedencia

- Hacer un nodo por cada T_i.
- Si alguna operación de T_i precede y conflictúa con alguna operación de T_j $(i \neq j)$ en H, hacer un arco $T_i \rightarrow T_j$.
- Dos operaciones son conflictivas si operan sobre el mismo ítem y al menos una de ellas es una escritura.



H es SR iif. SG(H) es un DAG.

Ordenes seriales

Si el SG (H) es acíclico entonces los órdenes seriales equivalentes son los diferentes ordenes topológicos del grafo.

Algorítmo de Kahn

$L \leftarrow$ Empty list that will contain the sorted elements
$S \leftarrow$ Set of all nodes with no incoming edges
while *S es no vacío* **do**
    REMOVE(n, S)
    APPEND(n, L)
    foreach *nodo m con un eje e desde n hacia m* **do**
        REMOVE(e, G) **if** *m no tiene otros ejes entrantes* **then**
            APPEND(m, S)
        end
    end
end
if *G tiene ejes* **then**
    return *error, el grafo no es acíclico*
end
else
    return *L, un orden topológico*
end

Dar una historia H_2 equivalente a la ejecución serial $T_2 T_1 T_3$, que sea SR y RC pero no ACA.

- Una historia H es RC si siempre que una transacción T_i lee de T_j con $i \neq j$ en H y $c_i \in H$ entonces $c_j < c_i$.
  - (ie: Si T_i lee A de T_j entonces antes del commit de i hubo un commit de j).
- Una historia H es Avoids Cascading Aborts (ACA) si siempre que una transacción T_i lee X de T_j con $i \neq j$ en H entonces $c_j < r_i(X)$. Lee sólo valores de transacciones que ya hicieron commit.
  - (ie: Si T_i lee A de T_j entonces antes del commit de i hubo un commit de j).

$$H_3 = H_2 = r_2(A); w_2(A); c_2;$$
$$r_1(A); w_1(A); r_1(B); w_1(B); c_1;$$
$$r(B)_3; r_3(A); w_3(A); w_3(B); c_3$$

Dada la siguiente historia H_3 en el modelo ReadLock / WriteLock / UnLock (ternario).

$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A);$
$rl_3(A); c_1; u_3(A); wl_3(B); c_2; u_3(B); c_3$

- ¿Es H_3 legal?
- ¿Es T_3 2PL (two phase locking)?
- Dibujar el Grafo de Precedencia.

Legalidad

H es legal si

1. Una Ti no puede leer ni escribir un ítem X hasta tanto no haya hecho un lock de X.
2. Una Ti que desea obtener un lock sobre X que ha sido lockeado por Tj en un modo que conflictúa, debe esperar hasta que Tj haga unlock de X.

¿Es $H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A); rl_3(A); c_1; u_3(A); wl_3(B); c_2; u_3(B); c_3$ legal?

Two Phase Locking

Idea: Todas las operaciones de lock preceden la primer operación de unlock.

¿Es $H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A);$
$rl_3(A); c_1; u_3(A); wl_3(B); c_2; u_3(B); c_3$ 2PL?

$H_3 = rl_1(A); rl_2(B); u_2(B); u_1(A); wl_2(A); u_2(A);$
$rl_3(A); c_1; u_3(A); wl_3(B); c_2; u_3(B); c_3$ **No es 2PL**

Grafo de Precedencia en modelo de lock ternario

1. Hacer un nodo por cada T_i
2. Si T_i hace un $rl_i(X)$ o $wl_i(X)$ y luego T_j con $j \neq i$ hace un $wl_j(X)$ en H hacer un arco $T_i \rightarrow T_j$
3. Si T_i hace un $wl_i(X)$ y T_j con $j \neq i$ hace un $rl_j(X)$ en H entonces hacer un arco $T_i \rightarrow T_j$

Básicamente dice que si dos transacciones realizan un *lock* sobre el mismo ítem y al menos uno de ellas es un *write lock* se debe dibujar un eje desde la primera a la segunda.