

1 Serializabilidad / Recuperabilidad de Historias

1.1 Dada la siguiente historia (schedule) que es conflicto-serIALIZABLE transformarla en una historia serial mediante una secuencia de intercambios no conflictivos de acciones adyacentes.

H1 = r1(A); w1(A); r2(A); w2(A); r1(B); w1(B); r2(B); w2(B);

T1	T2
r1(A)	
w1(A)	
	r2(A)
	w2(A)
r1(B)	
w1(B)	
	r2(B)
	w2(B)

Orden serial: T1 → T2 **es esto lo que pide??**

1.2. Dadas las siguientes historias (schedules):

H1 = r1(X); r4(Y); w1(X); w4(X); r3(X); **c1**; w4(Y); w3(Y); w4(Z); **c4**; w3(X); **c3**.

H2 = r1(X); w2(X); r1(Y); w1(X); w1(Y); **c1**; r2(Z); w2(Y); **c2**.

H3 = w1(X); w2(X); w2(Y); **c2**; w1(Y); **c1**; w3(X); w3(Y); **c3**.

H4 = w2(X); r1(X); r2(Z); r1(Y); w2(Y); w1(Y); **c2**; w1(X); **c1**.

H5 = r1(X); r4(Y); r3(X); w1(X); w4(Y); w3(X); **c1**; w3(Y); w4(Z); **c3**; W4(X); **c4**.

(a) Indicar para cada una si es NoRC, RC, ACA o ST.

(b) Indicar cuales podrían producir un dirty read o un lost update.

Historia Recuperable **RC**

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 $c_j < c_i$.

Intuitivamente una historia es recuperable si una transacción realiza commit sólo después de que hicieron commit todas las transacciones de las cuales lee.

Avoids Cascading Aborts **ACA**

Una historia H es **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*

Stricta **ST**

Una historia H es **ST** si siempre que $w_j(X) < o_i(X)$ con $i \neq j$ entonces $a_j < o_i(X)$ o $c_j < o_i(X)$ siendo $o_i(X)$ igual a $r_i(X)$ o a $w_i(X)$

Es decir no se puede leer ni escribir un ítem hasta que la transacción que lo escribió previamente haya hecho *commit* o *abort*.

$H1 = r1(X); r4(Y); w1(X); w4(X); r3(X); c1; w4(Y); w3(Y); w4(Z); c4; w3(X); c3.$

No es Stricta, ej: $w1(X) < w4(X)$, pero antes de $w4$ no hay $c1$ ni $a1$.

No es ACA, ej: $w4(X) < r3(X)$, pero antes de $r3$ no hay $c4$

Parece que es RC, $r3(X)$ se lee de 1 y 4, y tenemos $c1 < c4 < c3$. Los otros reads no tienen una ti previa.

Podría producirse lost update por la secuencia $r1(X), w1(X), w4(X)$.

$H2 = r1(X); w2(X); r1(Y); w1(X); w1(Y); c1; r2(Z); w2(Y); c2.$

No es stricta, ej: $w2(x) < w1(x)$, pero antes de $w1(X)$ no hay $c2$.

Parece que es ACA, las lecturas no tienen una transacción previa de la cual leer.

Podría producirse lost update por la secuencia $r1(X), w2(X), w1(X)$

$H3 = w1(X); w2(X); w2(Y); c2; w1(Y); c1; w3(X); w3(Y); c3.$

No es ST, se tiene $w1(X) < w2(X)$, sin un $c1$ ni $a1$ entre ambas.

Parece que es ACA, no hay lecturas,

Podría producirse lost update por la secuencia de writes sin lecturas (??).

H4 = w2(X); r1(X); r2(Z); r1(Y); w2(Y); w1(Y); c2; w1(X); c1.

No es ST, se tiene $w2(X) < r1(X)$, sin c2 ni a2 en el medio.

No es ACA, se tiene $w2(X) < r1(X)$, sin c2 ni a2 en el medio.

Parece ser RC, $c2 < c1$ y la lectura $r1(Y)$ no tiene transacción previa

Podría producirse lost update por la secuencia $r1(Y)$, $w2(Y)$, $w1(Y)$

H5 = r1(X); r4(Y); r3(X); w1(X); w4(Y); w3(X); c1; w3(Y); w4(Z); c3; W4(X); c4.

No es ST: $w1(X) < w3(X)$

Parece ser ACA.

Podría producirse lost update por la secuencia $r1(X)$, $r3(X)$, $w1(X)$, $w3(X)$

Y CUALES PUEDEN TENER DIRTY READ? ES SI FALLA EL COMMIT?

1.3. Clasificar según recuperabilidad las siguientes historias:

H1 = r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); c1; w3(Y); c3; r2(Y); w2(Z); w2(Y); c2.

H1 es stricta?

H2 = r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); w3(Y); r2(Y); w2(Z); w2(Y); c1; c2; c3.

H2 no es stricta.

H3 = r1(X); r2(Z); r3(X); r1(Z); r2(Y); r3(Y); w1(X); c1; w2(Z); w3(Y); w2(Y); c3; c2.

TERMINAR

1.4. Dadas las siguientes transacciones:

T1 = w1(B); r1(C); w(C);

T2 = r2(D) r2(B); w2(C);

(a) Dar una historia H1 que no sea RC.

(b) Dar una historia H2 que sea ACA pero no ST.

(c) Dar una historia H3 que sea ST.

2 Introducción Protocolos de Bloqueo o Locking

2.1. Dada las siguientes transacciones escritas como Read/Write:

T1 = r1(A); w1(A); r1(B); w1(B)

T2 = r2(A); w2(A); r2(B); w2(B)

Nota: Si bien ambas parecen iguales en realidad pueden realizar operaciones diferentes sobre los ítems leídos.

(a) Incorporar locks para que llevarlas a un modelo con locking binario

(b) Realizar una historia legal pero no serializable