UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

/ UGR / decsai

Relación de problemas IV: *Seguridad y fiabilidad en los datos de un SGBD*

ADMINISTRACIÓN DE BASES DE DATOS / CURSO 2019/20

Ejercicio 1

Una posibilidad para la tabla de modificaciones sería:

T_id	Estado	Operación	Dato	Valor antiguo	Valor nuevo
T3	Start				
T3		Update	B	50	?
T3		Update	C	200	150
T3	Commit				
T2	Start				
T2		Update	B	?	100
T2	Commit				
T1	Start				
T1		Update	A	100	150
T1	Commit				

Ejercicio 2

Si consideramos el instante en que se ejecuta *Escribe(T3,C)*, en ese momento T3 no ha ejecutado su commit luego sufrirá un UNDO. Las transacciones T1 y T2 ni siquiera han empezado, luego no hacemos nada con ellas.

> En este caso no ha lugar ninguna consideración sobre COMMIT "cercanos", pues el fallo (según se ha dicho en el enunciado) se produce antes de *Escribe(T3,C)*, y el commit se ejecuta después. Si se diera el caso hipotético de tener un commit y después el fallo, la transacción T3 sufriría un REDO.

Ejercicio 3

Ahora se plantea que el fallo se produce después de *Escribe*(T2,B): este caso es más complejo, pues hay que hacer varias consideraciones:

- La transacción T3 ya ha ejecutado el *commit* **antes** de Escribe(T2,B) y por tanto se va a rehacer **en cualquier circunstancia** dentro de la casuística posible para este lugar de fallo.
- si se produce el fallo justo antes del commit de T2, es decir, entre el Escribe(T2,B) y su *commit*:
  - *No* se habrá completado la transacción T2 y habrá que **deshacerla (UNDO).**
  - T1 aún **no ha empezado**, luego no se contempla ninguna operación con ella.
- Si se produce el fallo justo después del commit de T2, esta transacción se sumaría a T3 para ser rehecha (REDO).
- Si se produce el fallo en cualquier momento tras el *start* de T1, pero antes de su *commit*, T1 también se **deshacerá** (UNDO).
- Si se produce el fallo en cualquier momento tras el commit de T1, todas las transacciones sufrirán un REDO, pues de todas se ha recibido el start y el commit (antes del fallo).

Ejercicio 4

La tabla de modificaciones completa resulta:

T_{id}	*Estado*	*Operación*	*Dato*	*Valor antiguo*	*Valor nuevo*
T1	*start*				
T2	*start*				
T1		*update*	A	10	20
T1	*commit*				
T3	*start*				
T2		*update*	A	20	30
	savepoint				
T3		*update*	B	0	15
T3		*update*	D	8	25
T3	*commit*				
T2		*update*	E	35	35
T2	*commit*				

Ejercicio 5

El fallo se ha producido justo antes de que se realice la opeción UPDATE de T2 (según la solución aportada en el Ejercicio 4). De esta forma, se ha perdido dicho UPDATE y también el COMMIT de T2. Esto significa que, en base al lugar del checkpoint y su relación con los start/commit de cada transacción:

- La transacción T1 se "***ignorará***", ya que su commit está antes del checkpoint.
- La transacción T3 tendrá que **rehacerse** (REDO) porque su commit está después del checkpoint.
- La transacción T2 tendrá que **deshacerse** porque no tiene un commit tras el checkpoint.

En cuanto a los valores que resultan, tendremos en cuenta las conclusiones recién extraídas para cada Ti, tal que esos **valores finales** serán:

A = 20, E = 35 (T2 deshecha), B = 15, D = 25.

Ejercicio 6

En este caso, el fallo se produce (de nuevo en base a mi respuesta al Ejercicio 4) antes del commit de T3. Esto significa que:

- T1 de nuevo se ignora que su start y su commit están antes del checkpoint.
- T2 y T3 tendrá que **deshacerse,** ya que no tienen commit alguno tras el último (y único) checkpoint de la tabla.

Lo anterior, de nuevo supone un reajuste de los valores. Se quedan como sigue:

A= 20, B = 0, D = 8, E = 35.