Práctico 10

Concurrencia y Recuperación

Ejercicio 1. (*)

Sea el siguiente programa de transferencia en cuentas bancarias:

```
program transfer;
    begin
        START
        input (cta_origen, cta_destino, monto);
        tmp := read(cta_origen);  /* lee saldo en cta_origen */
        if (tmp < monto )  {
            output ("saldo insuficiente");
            ABORT
        } else {
            write (cta_origen, tmp-mont);  /* actualizo saldo */
            tmp := read (cta_destino);  /* leo saldo en cta_destino */
            write (cta_destino, tmp+monto);  /* actualizo saldo */
            COMMIT
            output ("transacción completa");
        }
end program.
```

y el siguiente programa que imprime la suma de los saldos de dos cuentas:

```
program suma2ctas;
    begin
        input (cta1, cta2);
        tmp1 := read (cta1);
        tmp2 := read (cta2);
        output (tmp1+tmp2);
    end program.
```

(a) Dar un ejemplo de una ejecución concurrente de los programas anteriores tal que el resultado de suma2ctas sea incorrecto.

(b) Dar un ejemplo de una ejecución concurrente de dos transferencias tal que:

I. sea recuperable pero no serializable.

II. sea serializable pero no recuperable.

III. no sea serializable ni recuperable.

Ejercicio 2. (*)

Sean las transacciones:

$T_1 : w_1(x), r_1(y), C_1$

$T_2 : r_2(x), w_2(y), C_2$

(a) Dar una historia o explicar porqué no existe una historia (que contenga los COMMIT), que sea recuperable y que presente las operaciones en el siguiente orden:
$w_1(x), r_2(x), w_2(y), r_1(y)$

(b) Dar una ejecución concurrente de T_1 y T_2 tal que la historia que representa a esa ejecución sea recuperable. Decir si es serializable.

Ejercicio 3. (*)

Sean las siguientes transacciones:

T_0:

```
read(a);
read(b);
if a = 0 then b :=  b+1;
write(b).
```

T_1:

```
read(b);
read(a);
if b = 0 then a := a+1;
write(a).
```

Sea la siguiente restricción de integridad:

a = 0 o b = 0, con a = b = 0 los valores iniciales.

(a) Mostrar que toda ejecución serial que involucra a T_0 y T_1 preserva la consistencia de la base de datos.

(b) Dar una ejecución concurrente de T_0 y T_1 que no sea serializable.

(c) ¿Existe una ejecución concurrente de T_0 y T_1 que sea serializable?

Ejercicio 4.

(a) Probar que si dos historias son equivalentes entonces sus grafos de seriabilidad son idénticos.

(b) Probar o dar un contraejemplo para el recíproco.

(c) Probar o dar un contraejemplo que si dos historias H_1 y H_2 son sobre el mismo conjunto de transacciones, tienen las mismas operaciones y el mismo grafo de seriabilidad, entonces son equivalentes.

Ejercicio 5. (*)

(a) Dadas las siguientes transacciones:

T_1:

```
leer_elemento(X);
X := X*N;
escribir_elemento(X);
leer_elemento(Y);
Y := Y + N;
escribir_elemento(Y);
```

T_2:

```
leer_elemento(X);
X := X + M;
escribir_elemento(X);
```

I. Dar una historia de T_1 y T_2 recuperable y otra no recuperable. Justificar.

II. Dadas las siguientes historias de T_1 y T_2:
$H_1: r_1(X), r_2(X), w_2(X), c_2, w_1(X), r_1(Y), w_1(Y), c_1$
$H_2: r_1(X), w_1(X), r_1(Y), r_2(X), w_1(Y), w_2(X), c_2, c_1$

1. Decir si son serializables y si son recuperables, justificando las respuestas. En los casos en que sea serializable dar la historia serial equivalente.
2. ¿H_1 y H_2 son historias equivalentes? ¿Por qué?

(b) Dadas

$T_1: r_1(X), r_1(Y), w_1(Y)$ $T_2: r_2(X), w_2(X), r_2(X), w_2(X)$

que bloquean y desbloquean los objetos de la siguiente forma:

$T_1: rl_1(X), r_1(X), rl_1(Y), r_1(Y), wl_1(Y), u_1(X), w_1(Y), u_1(Y)$

$T_2: rl_2(X), r_2(X), wl_2(X), w_2(X), rl_2(X), r_2(X), wl_2(X), w_2(X), u_2(X)$

siendo: $rl =$ read_lock, $wl =$ write_lock, $u =$ unlock

I. Decir si T_1 y T_2 siguen el protocolo 2PL. Justificar.

II. Dar una historia que no sea serializable a causa de las operaciones que violan 2PL.

Ejercicio 6. (*)

Sean las transacciones:

$T_1: r_1(x),\ w_1(y), w_1(u), c_1$

$T_2: r_2(y), w_2(z), r_2(w), w_2(w), c_2$

$T_3: r_3(z), w_3(x), c_3$

y la siguiente historia que finaliza con la caída del sistema:

$H: r_1(x), w_1(y),\ r_2(y), w_2(z), r_3(z), w_3(x), r_2(w), w_2(w),$<Falla sistema >

Se pide:

(a) Decir que transacciones deberían revertirse y por qué.

(b) Dar una historia de T_1, T_2 y T_3 en la cual haya que revertir solo T_2. Explicarlo.

(c) Sea la historia:

$H: r_1(x), w_1(y),\ r_2(y), w_2(z), r_3(z), w_3(x), c_3, r_2(w), w_2(w),$ <Falla sistema>

I. Decir que transacciones deberían revertirse y por qué.

II. ¿ Qué fenómeno se da en este caso y por qué?

III. Decir si la base de datos corre riesgo de quedar en un estado inconsistente.

Ejercicio 7.

Dadas las transacciones:

$T_1: r_1(A), r_1(D), w_1(D)$

$T_2: r_2(B), w_2(B), r_2(D), w_2(D)$

$T_3: r_3(C), w_3(B), r_3(A), w_3(A)$

y la siguiente historia

$H: r_3(C), w_3(B), r_2(B), w_2(B), r_1(A), r_1(D), w_1(D), r_2(D), w_2(D), r_3(A), a_3$

Se pide:

(a) Decir qué operaciones deberían revertirse y por qué.

(b) ¿ Qué fenómeno se da en este caso? Explicarlo.

Ejercicio 8.

Dadas las transacciones T_1 y T_2:

$T_1 : r_1(X) r_1(Y) w_1(Y)$

$T_2 : r_2(Z) w_2(Z) r_2(Y) w_2(Y)$

y el siguiente escenario de falla en un sistema de bases de datos:

$H : r_1(X) r_1(Y) r_2(Z) w_1(Y) c_1 w_2(Z)$<falla sistema>

suponiendo que el último checkpoint se realiza antes de comenzar H, Se pide:

(a) Dar el algoritmo de recuperación que se ejecutara para los casos de:
   I. Actualización diferida
   II. Actualización inmediata (los 2 tipos posibles)

(b) ¿Qué sucede si el escenario de falla es el siguiente?

$H : r_1(X) r_1(Y) r_2(Z) w_1(Y) w_2(Z) r_2(Y) w_2(Y) c2$ <falla sistema>