

SISTEMAS INFORMÁTICOS II

SISTEMAS INFORMÁTICOS II

Asignatura: Grupo:
 Apellidos: Nombre:
 Ejercicio del día: 3 de septiembre de 2004. Examen final.



2.- PROBLEMA (10 puntos). Para realizar cargos en tarjetas de crédito (Visa, MasterCard, etc.) se emplea un procesador de transacciones. El sistema tiene conectados un conjunto de datáfonos que transmiten los datos de la tarjeta y los datos de la operación (importe, concepto, comercio). Para comprobar el crédito disponible y reducido en el valor correspondiente al importe de la operación se propone usar el siguiente programa transaccional (escrito en pseudo-código):

```
BEGIN_TX
READ tarjeta
WRITE contador
if (tarjeta=null || tarjeta.anulada) {
    // Incrementa un contador de tarjetas procesadas
    // No existe la tarjeta o está anulada.
    ROLLBACK
}
else {
    if (tarjeta.saldo >= cargo.tarjeta) {
        // se realiza el cargo de la operación
        tarjeta.saldo -= cargo.tarjeta // se decrementa el saldo
        WRITE tarjeta
        COMMIT
    }
    else
        ROLLBACK
}
```

Un segundo programa transaccional se encarga de anular una tarjeta en caso de robo:

```
BEGIN_TX
READ tarjeta
if (tarjeta!=null) {
    tarjeta.anulada=TRUE
    WRITE tarjeta
}
COMMIT
```

Nota: Las actualizaciones del objeto tarjeta reemplazan todos los campos del mismo.

2.1.- (2 puntos) Escribir el modelo equivalente simple de la transacción resultante de ejecutar el primer programa transaccional en caso de que el cliente tenga crédito suficiente para realizar la operación, que llamaremos T1; otra en caso contrario, que llamaremos T2, y otra de la ejecución del segundo programa transaccional, que llamaremos T3. En todos los casos, se supone que la tarjeta existe y no está anulada.

T1	T2	T3
T1.1 READ tarjeta T1.2 WRITE contador (... operación...) T1.3 WRITE tarjeta	T2.1 READ tarjeta T2.2 WRITE contador (...) T2.3 WRITE (undo) contador	T3.1 READ tarjeta (...) T3.2 WRITE tarjeta

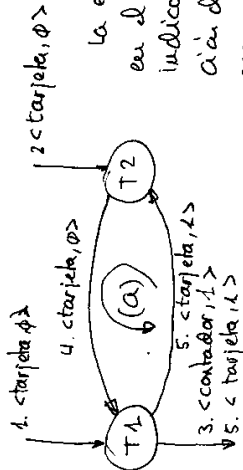


Asignatura: Grupo:
 Apellidos: Nombre:
 Ejercicio del día: 3 de septiembre de 2004. Examen final.

2.2.- (5 puntos) Un defecto de este sistema es que, según la secuencia de acciones que se produzcan, podrían perderse anulaciones de tarjetas. Escribir una posible historia de ejecución de las transacciones T1 y T3 en la cual se produzca esta situación. Indicar la violación o violaciones de aislamiento que se producen en la historia elegida, mediante la realización del diagrama de dependencias, y justificarlas con el mismo.

- 1 T1.1 READ <tarjeta, 0>
- 2 T3.1 READ <tarjeta, 0>
- 3 T1.2 WRITE <contador, 1>
- 4 T3.2 WRITE <tarjeta, 1>
- 5 T1.3 WRITE <tarjeta, 2>

Diagrama de dependencias:



La existencia del bucle (a) en el diagrama de dependencias indica la existencia de la violación del aislamiento entre las pasas 4 y 5 de la historia mostrada.

Es una violación de tipo WRITE-WRITE, o Actualización Perdida.

2.3.- (3 puntos) Incluir las acciones necesarias en los programas transaccionales de modo que las transacciones que produzcan sean siempre en dos fases y bien formadas. Escribir los modelos equivalentes simples de las transacciones que generan en los mismos casos descritos en el apartado primero, que denominaremos T1a, T2a y T3a, respectivamente, en una tabla similar a la presentada en el apartado 2.1. Realizar una historia de la ejecución de T1a y T3a en este caso, y dibujar su diagrama de dependencias.

(Contestar a la vuelta de la hoja)

2.3.1. Programar transacciones

```

BEGIN- TX
XLOCK tarjeta
READ tarjeta
if (tarjeta != null) {
    tarjeta. anulado = TRUE
    WRITE tarjeta
}
COMMIT

if (tarjeta == null || tarjeta. anulado) {
    ROLBACK
}
else {
    if (tarjeta. saldo >= cargo- tarjeta) {
        tarjeta. saldo -= cargo- tarjeta
        WRITE tarjeta
        COMMIT
    }
    else {
        ROLBACK
    }
}

```

2.3.2. Modelos equivalentes simplificados de las transacciones pendientes

	T1a	T2a	T3a
T1a.1	XLOCK tarjeta	T2a.1 XLOCK tarjeta	T3a.1 XLOCK tarjeta
T1a.2	READ tarjeta	T2a.2 READ tarjeta	T3a.2 READ tarjeta
T1a.3	XLOCK contador	T2a.3 XLOCK contador	T3a.3 WRITE tarjeta
T1a.4	WRITE contador	T2a.4 WRITE contador	T3a.4 UNLOCK tarjeta
T1a.5	WRITE tarjeta	T2a.5 WRITE (undo) contador	
T1a.6	UNLOCK contador	T2a.6 UNLOCK contador	
T1a.7	UNLOCK tarjeta	T2a.7 UNLOCK tarjeta	

2.3.3. Ejemplo de historia de T1a y T3a y diagrama de dependencias.
 La única historia legal posible es la historia serie, comenzando con T1a:

- 1 T1a.1 XLOCK tarjeta, 0
- 2 T1a.2 READ tarjeta, 0
- 3 T1a.3 XLOCK contador, 0
- 4 T1a.4 WRITE contador, 1
- 5 T1a.5 WRITE tarjeta, 1
- 6 T1a.6 UNLOCK contador, 0
- 7 T1a.7 UNLOCK tarjeta, 1
- 8 T3a.1 XLOCK tarjeta, 1
- 9 T3a.2 READ tarjeta, 1
- 10 T3a.3 WRITE tarjeta, 2
- 11 T3a.4 UNLOCK tarjeta, 2

