Ejemplos de problemas del acceso concurrente

Si tenemos en la tabla **RECAMBIOS** una columna denominada **Stock** cuyo valor inicial es 12 unidades**,** supongamos que dos transacciones concurrentes actualizan el valor del stock de la siguiente forma:

Transacción 1: **compra** de 100 unidades de ese recambio. Transacción 2: **venta** de 5 unidades del recambio.

Actualizaciones perdidas (*Dirty Read*):

Se produce cuando antes de finalizar una transacción, otra accede a los mismos datos, por tanto la segunda transacción actualiza el dato inicial y la primera modificación se pierde
La ejecución normal de las transacciones debería ser:

TRANSACCION	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 1	Stock=12 + 100	112
Transacción 2	Leer Stock	112
Transacción 2	Stock=112-5	107

Cuando una transacción accede a un valor antes de que la transacción anterior haya finalizado:

Actualización perdida

TRANSACCIÓN	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 2	Leer Stock	12
Transacción 1	Stock =12 +100	112
Transacción 2	Stock=12-5	7

Datos no comprometidos (*Non Repeatable*):

Se produce, por ejemplo, cuando una transacción se deshace después de que la segunda ya ha accedido a los datos.
Con los datos iniciales supongamos que la primera transacción se deshace. La ejecución correcta debería ser:

TRANSACCIÓN	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 1	Stock =12 +100	112
Transacción 1	ROLLBACK	12
Transacción 2	Leer Stock	12
Transacción 2	Stock =12 -5	7

Cuando una transacción se deshace después de que otra haya accedido a los datos:

Datos no comprometidos

TRANSACCIÓN	OPERACIÓN	VALOR Stock
Transacción 1	Leer Stock	12
Transacción 1	Stock =12 +100	112
Transacción 2	Leer Stock	112
Transacción 2	Stock =112 -5	107
Transacción 1	ROLLBACK	12

Recuperaciones inconsistentes (*Phantom*):

Ocurre, por ejemplo, cuando una transacción incluye operaciones de agregado (sumar, contar, etc.) sobre unos datos, mientras otra transacción los actualiza. Si la función lee unos datos antes de actualizar y otros después el resultado es inconsistente.

Transacción 1: Calcula la cantidad total disponible de Stock. Transacción 2: Actualiza en 30 unidades la cantidad disponible de Stock de dos de esos productos: Z y V.

La ejecución normal de estas transacciones es:

IdRecambio	VALOR Stock	VALOR AGREGADO
X	12	12
Y	15	15
Z	20	20+30=50
V	35	35-30=5
W	50	50
U	40	40
TOTAL	**172**	**172**

Cuando la transacción lee unos datos antes de ser modificados y otros después:

Lectura posterior a la modificación

Lectura anterior a la modificación

TRANSACCIÓN	OPERACIÓN	VALOR Stock	VALOR AGREGADO
Transacción 1	Leer Stock para el IdRecambio X	12	12
Transacción 1	Leer Stock para el IdRecambio Y	15	27
Transacción 2	Leer Stock para el IdRecambio Z	20	
Transacción 2	Modifica el Stock =20+30		
Transacción 1	Leer Stock para el IdRecambio Z	50	77 (X+Y+Z después de modificar)
Transacción 1	Leer Stock para el IdRecambio V	35	112 (X+Y+Z+V antes de modificar)
Transacción 2	Leer Stock para el IdRecambio V	35	
Transacción 2	Modifica el Stock=35-30	5	
Transacción 1	Leer Stock para el IdRecambio W	50	162
Transacción 1	Leer Stock para el IdRecambio U	40	**202**

Como vemos, el resultado final no es correcto porque el valor agregado excede en 30 unidades el resultado calculado anteriormente.