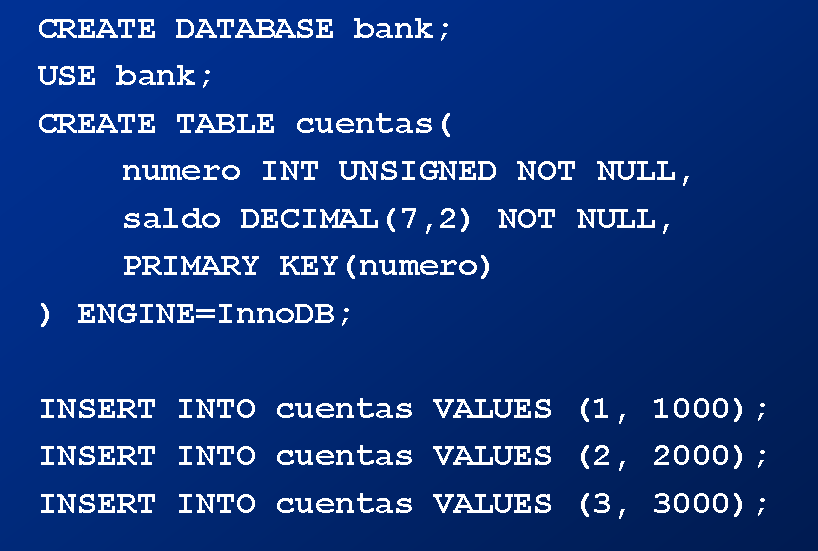
**SP, Triggers y Transacciones**

Dada la siguiente BDD:



Creando un Stored Procedure:

use bank; #Decimo que utilizaremos la bdd banco

delimiter ! # define “!” como delimitador

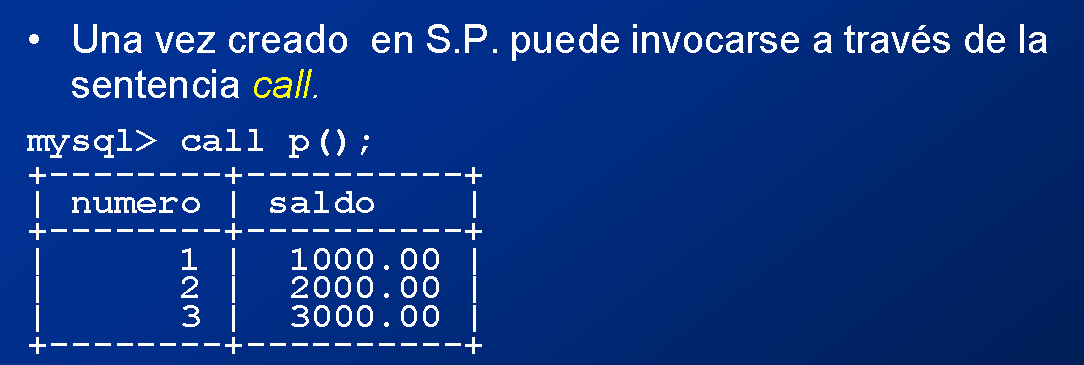
create procedure p()

begin

SELECT \* FROM cuentas;

end; !

delimiter ; # reestablece el “;” como delimitador

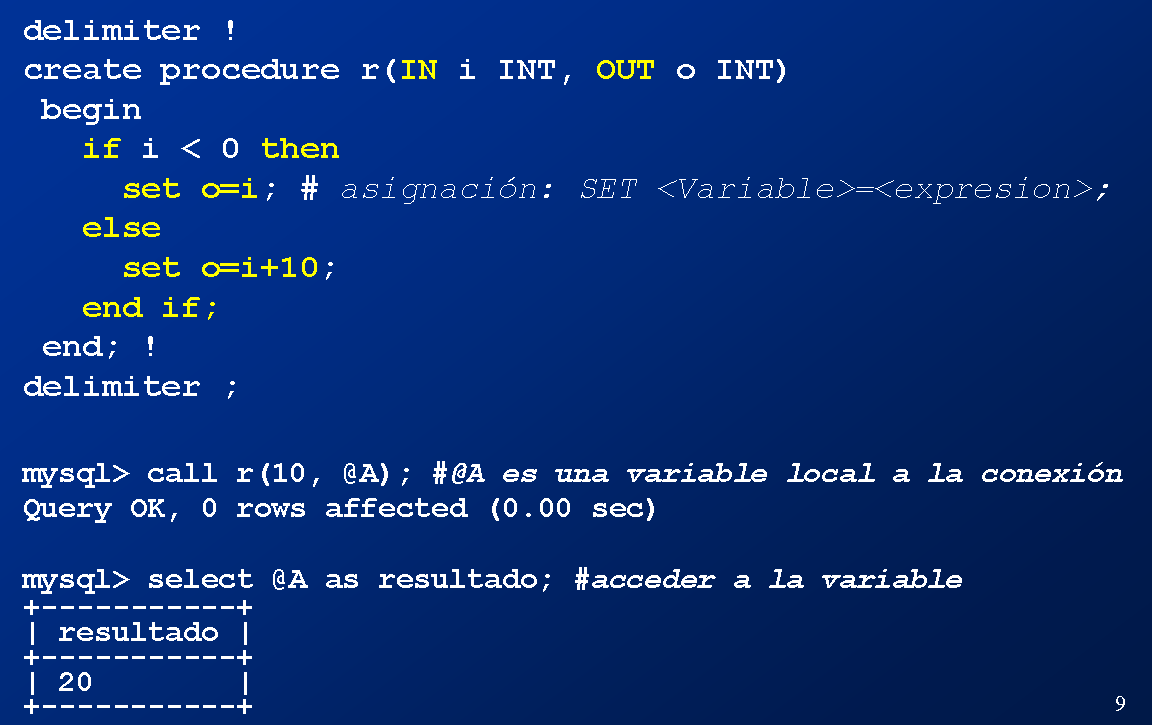


Los **usuarios** **deben** **tener** un **privilegio** **especial** para **poder** **ejecutar** un **S.P.**

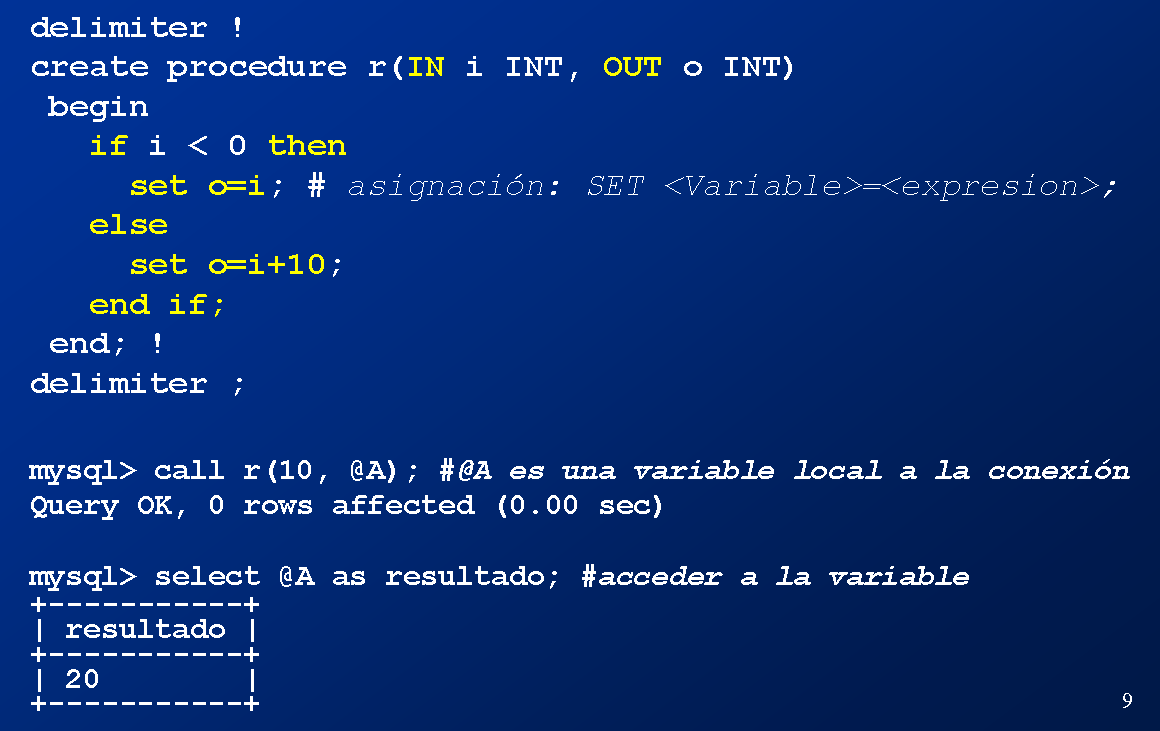
**grant execute on procedure bank.p to <usuario>;**

Para eliminar un S.P. utilizamos la sentencia drop procedure. mysql> drop procedure p;

**Ejemplo** con parámetros de entrada:

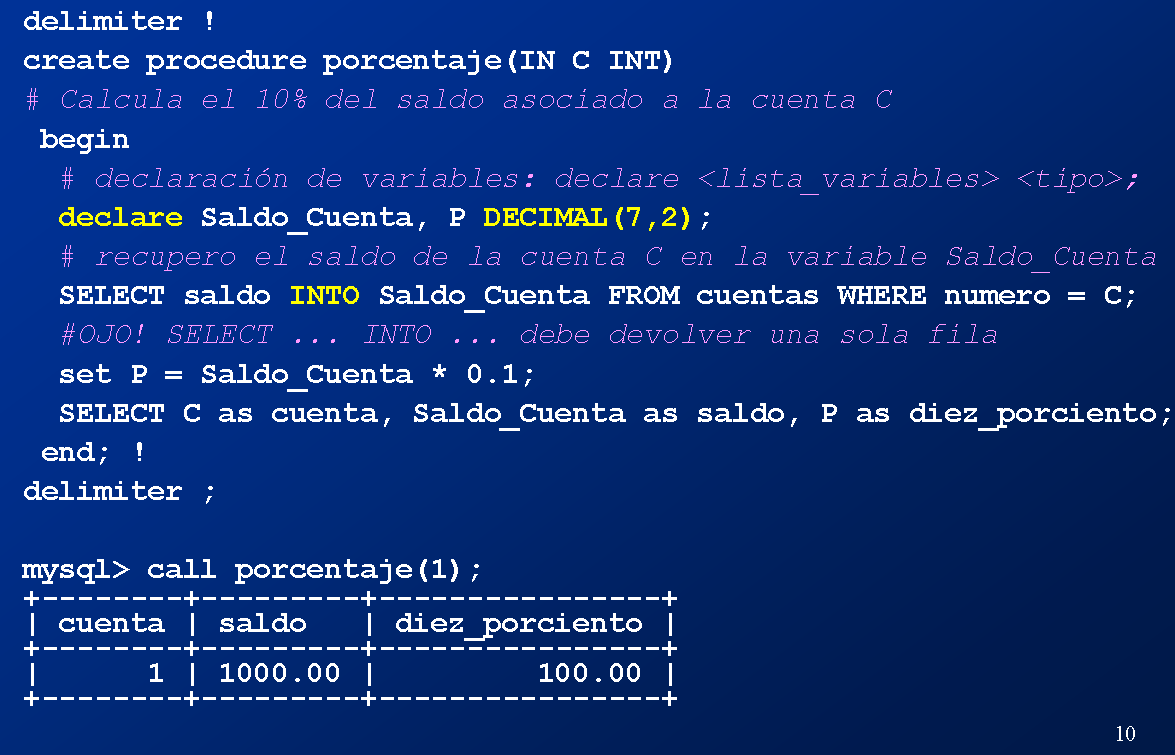


**Ejemplo 2:**



También se podría haber escrito “select @A;”

**Ejemplo con declaración y uso de variables:**



declare Saldo\_Cuenta, P DECIMAL (7,2);

Declara las variables “Saldo\_Cuenta” y “P” de tipo DECIMAL (7,2)

Lo que sigue debajo:

SELECT saldo INTO Saldo\_Cuenta FROM cuentas WHERE numero = C;

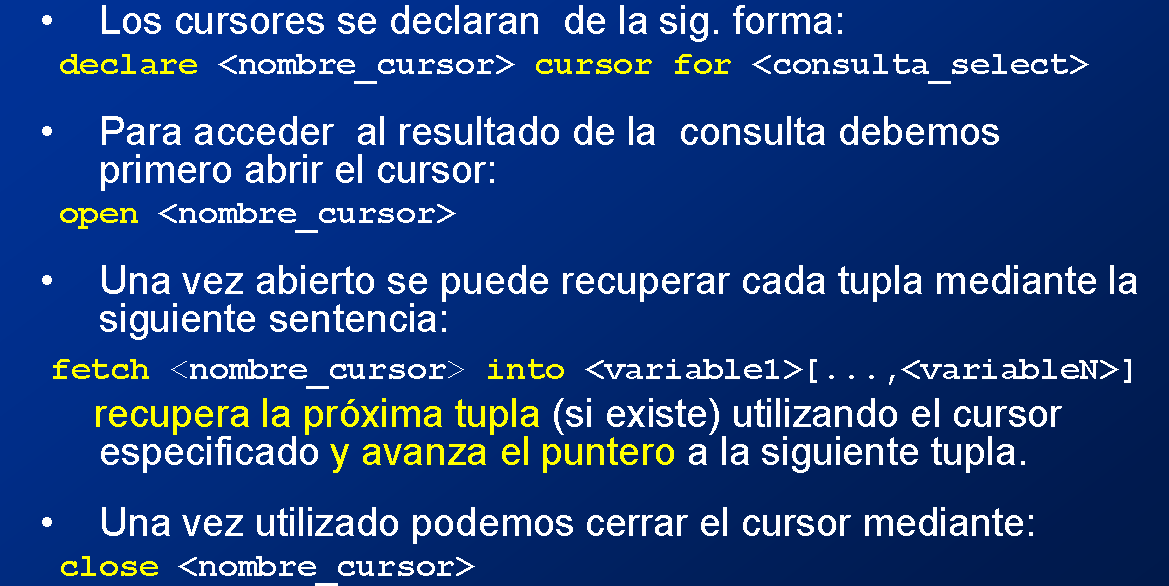
**Lo que hace es seleccionar el saldo de las cuentas cuyo número sea igual a C y ese saldo guárdalo en la variable Saldo\_Cuenta. Es importante que solo haya un resultado para guardar en la variable Saldo\_Cuenta asi que tiene que estar bien armada la sentencia.**

set P = Saldo\_Cuenta \* 0.1;

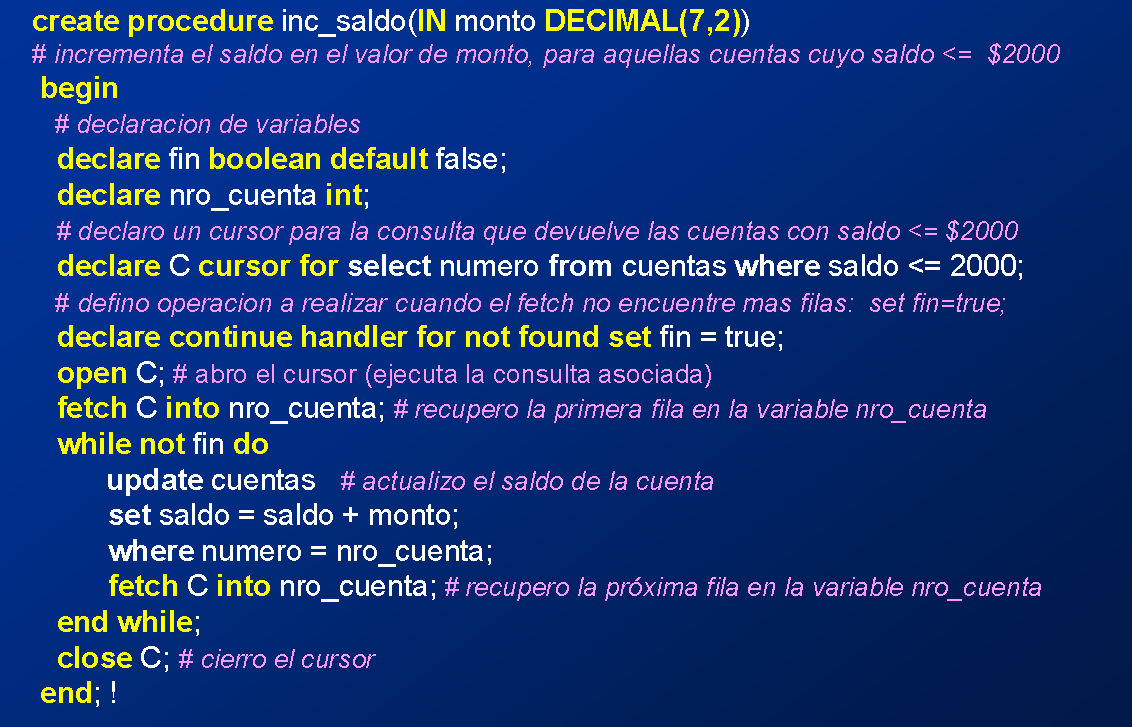
Setea P con el 10% del saldo de la cuenta.

**Cursores**

Si quiero hacer una consulta e iterar sobre todas las filas de la consulta entonces puedo utilizar cursores para realizar operaciones en las filas que desee.



**Ejemplo:**

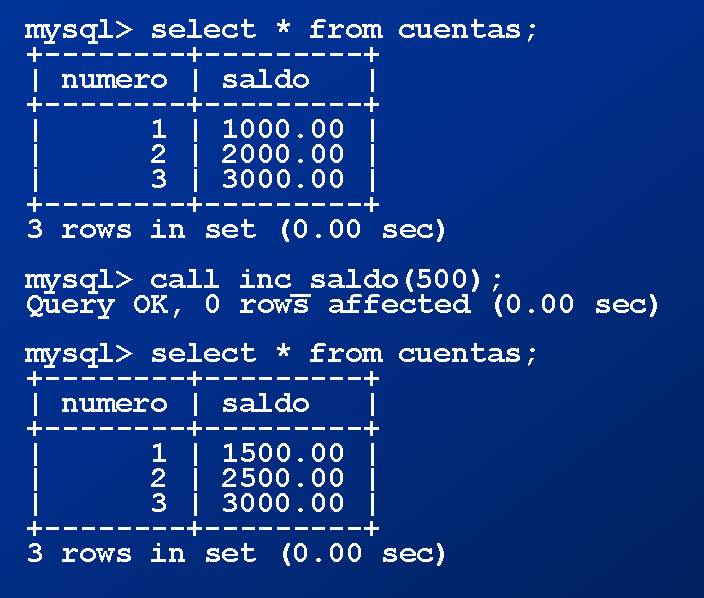


La var fin será la bandera que utilizaremos para finalizar de iterar.

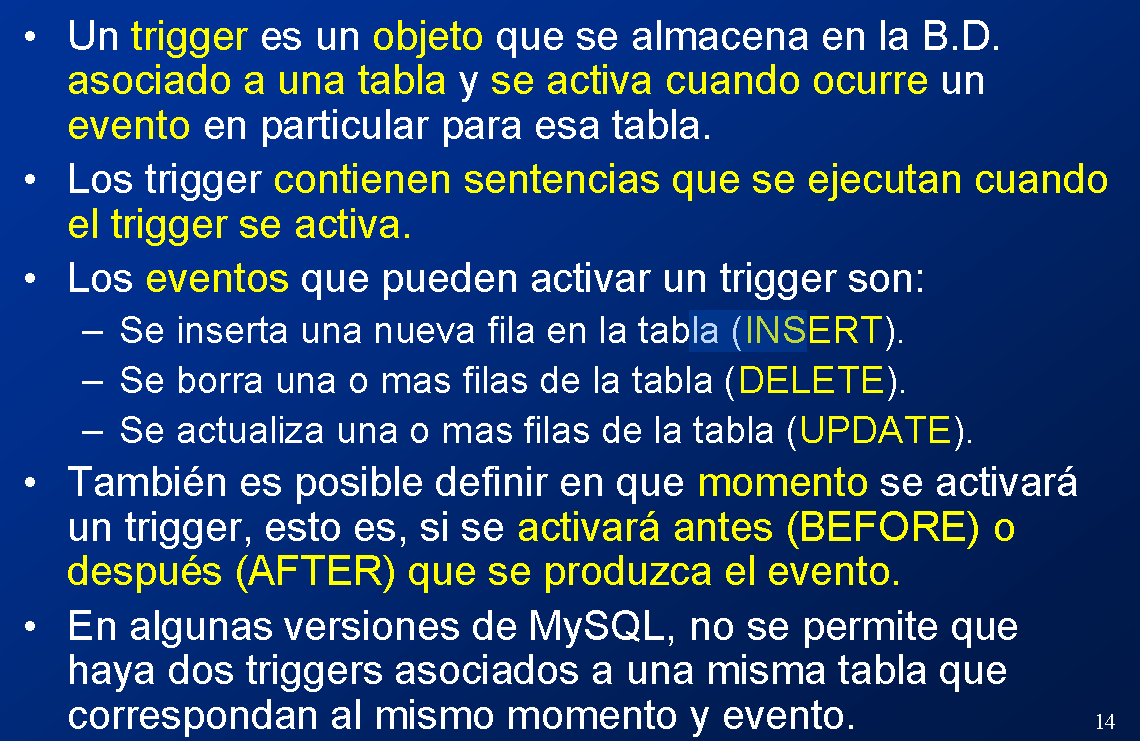
**En la iteración**: si nro\_cuenta tiene algo entonces actualizamos el saldo de la cuenta que tenga el mismo numero que estoy leyendo en este momento por eso escribe:

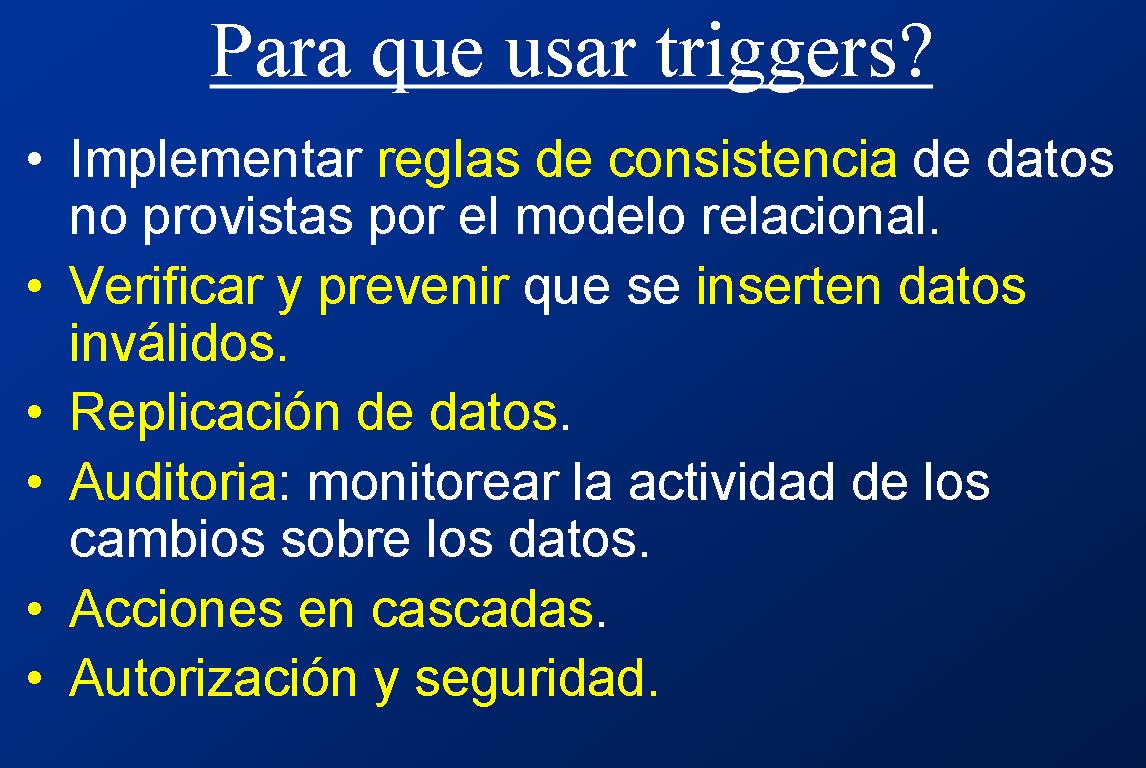
where número = nro\_cuenta

Veamos el antes y después de ejecutar el SP:



**Triggers**



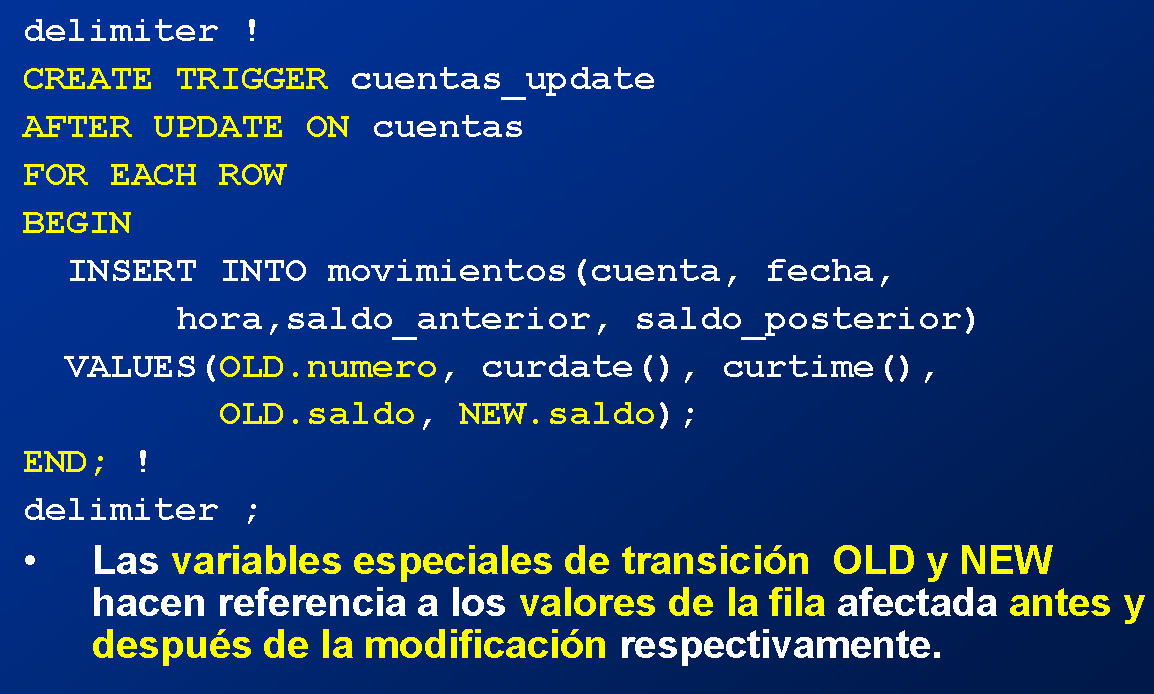




**Observación**: hay que darle el privilegio a los usuarios para que puedan crear Triggers.

{Before/After}Se especifica cuando se ejecuta el Trigger, antes o después de “x” evento.

**Ejemplo:**

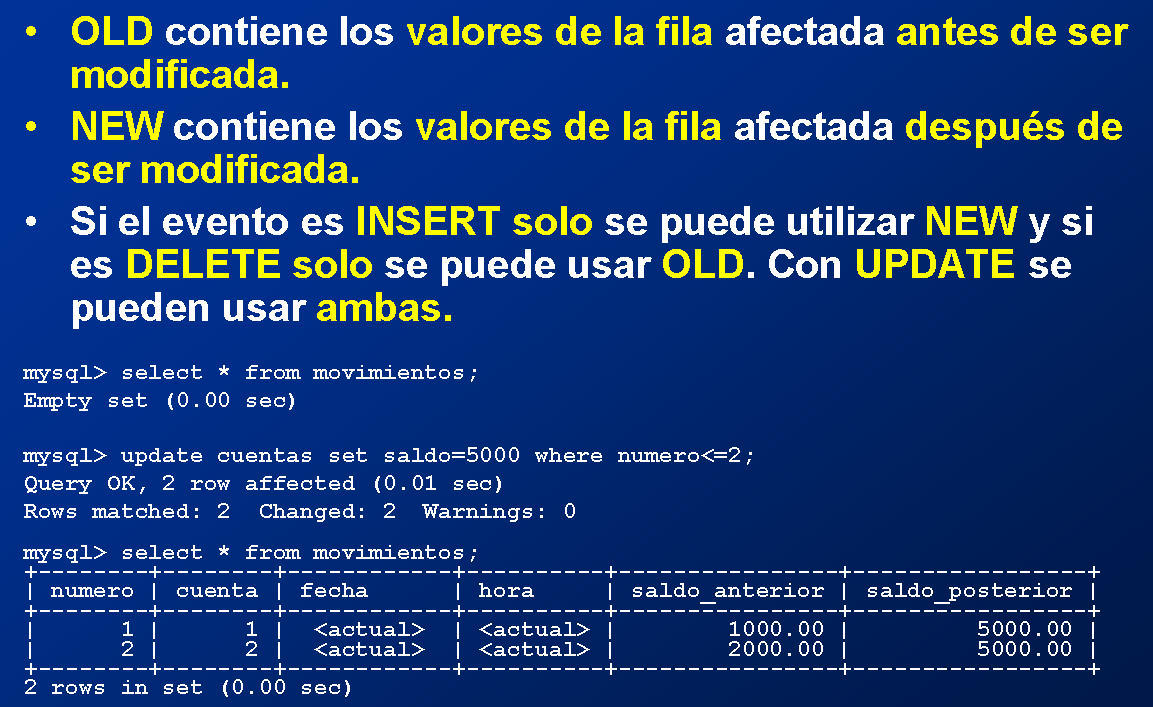
****

**El Trigger se llama cuentas\_update y se dispara después de cada “update” que se haga sobre la tabla cuentas. Entonces por cada fila en la que se produzca un update sobre cuentas ejecutamos la sentencia que inserta en la tabla “movimientos” los siguientes valores:**

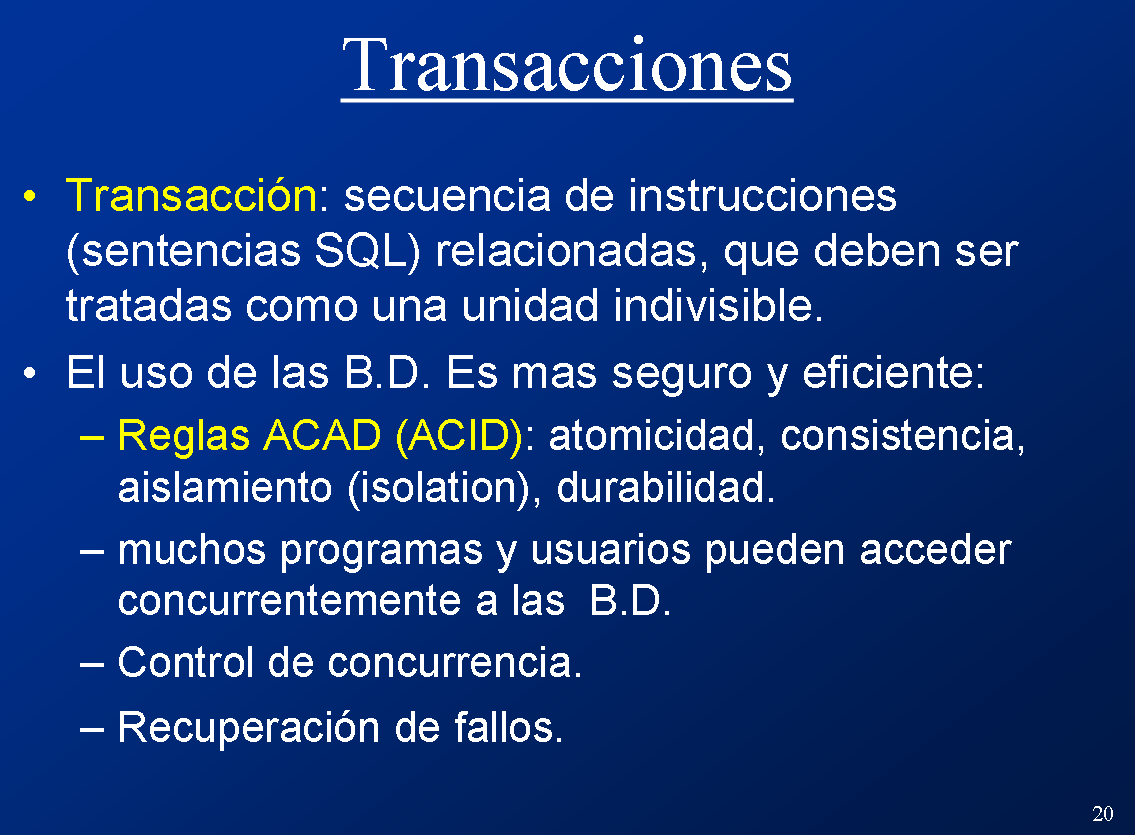
**En cuenta inserta OLD.numero, “OLD” viene a ser el valor anterior al update en la variable número.**

**También inserta curdate(), curtime(), OLD.saldo y NEW.saldo, “NEW” viene a ser el valor siguiente al update en la variable saldo.**

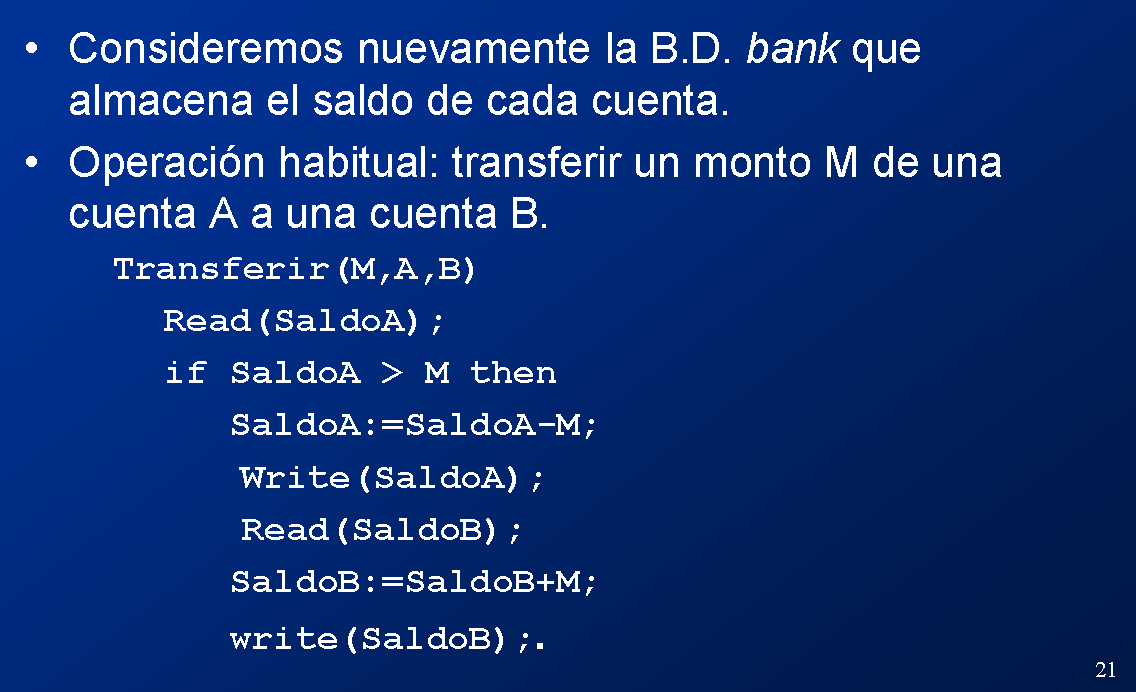
**Ejemplo:**

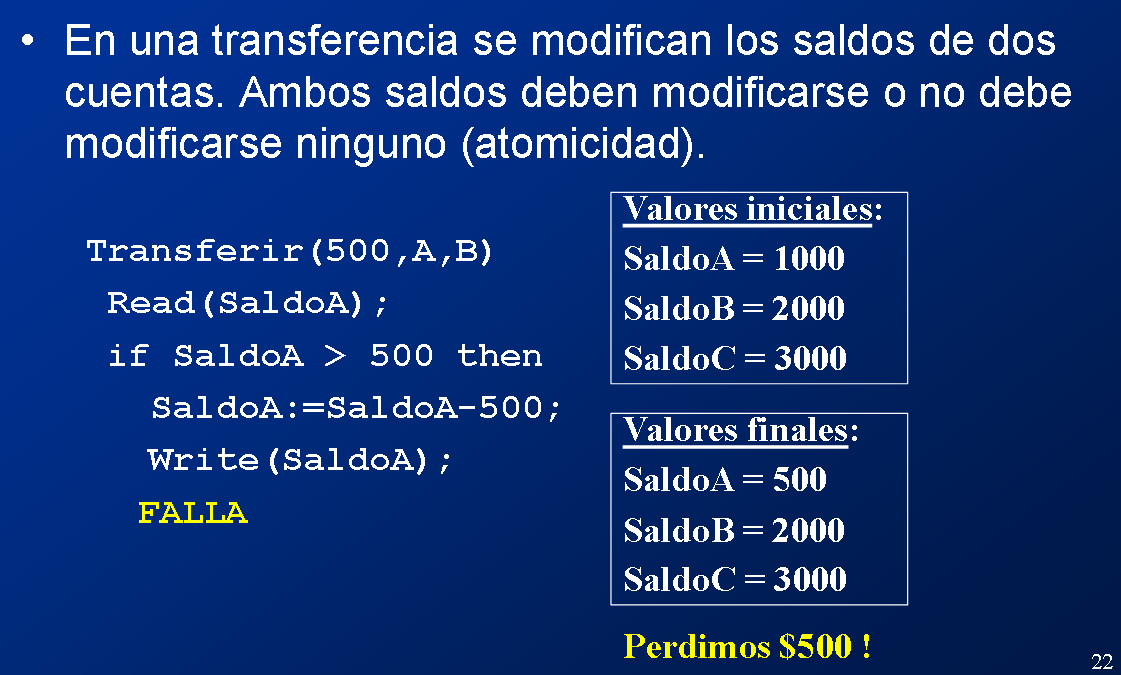
****

**Transacciones**

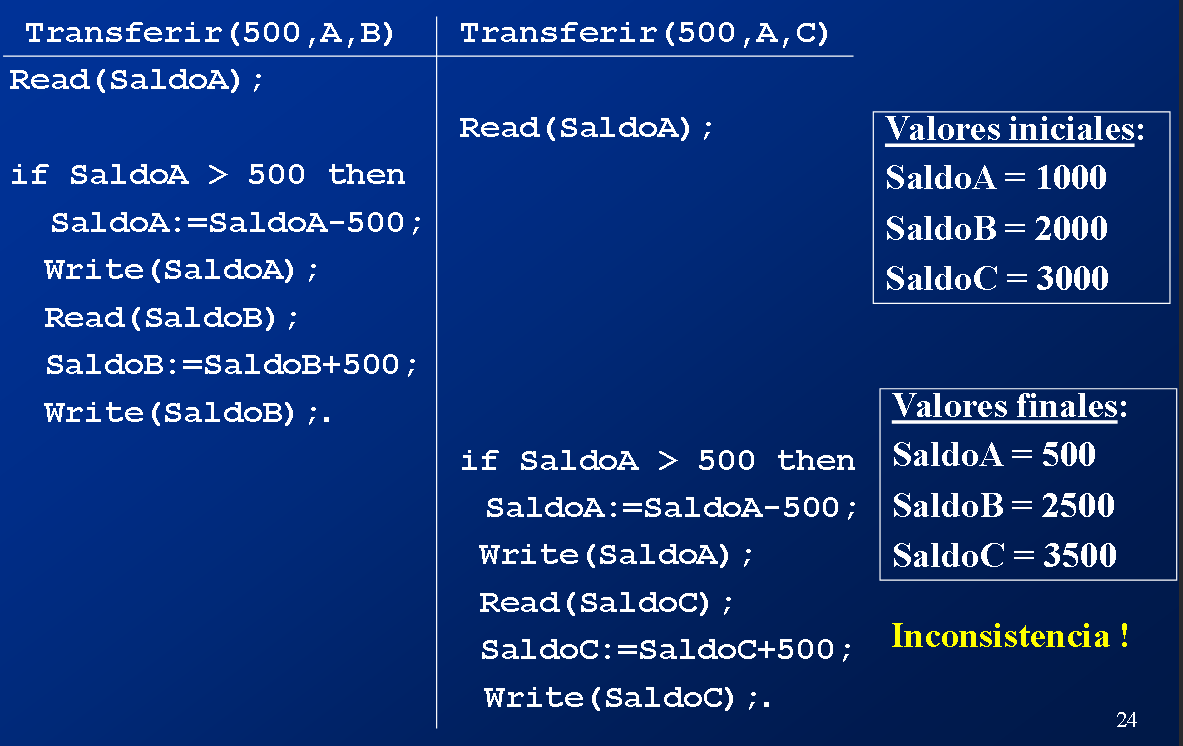


**Ejemplo:**



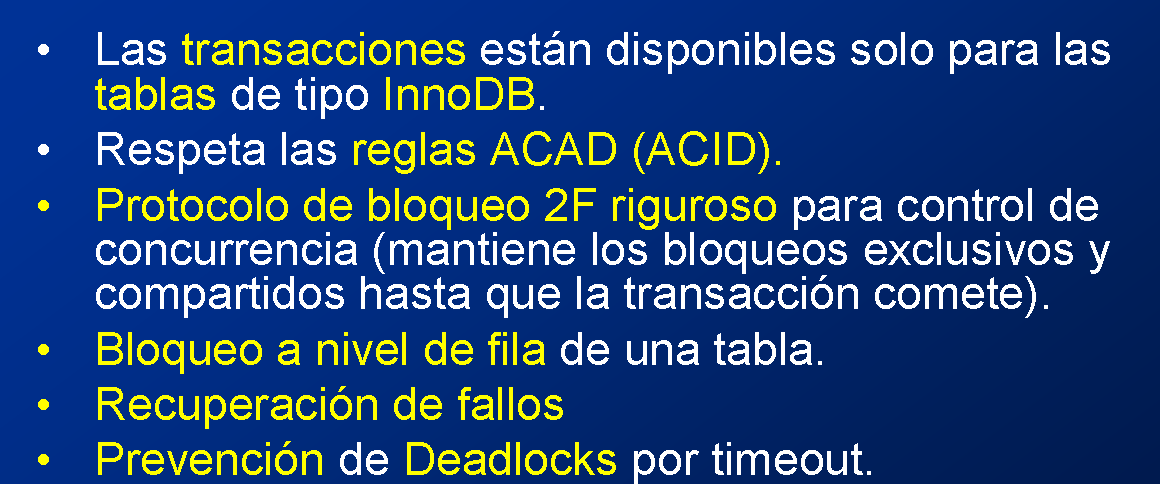


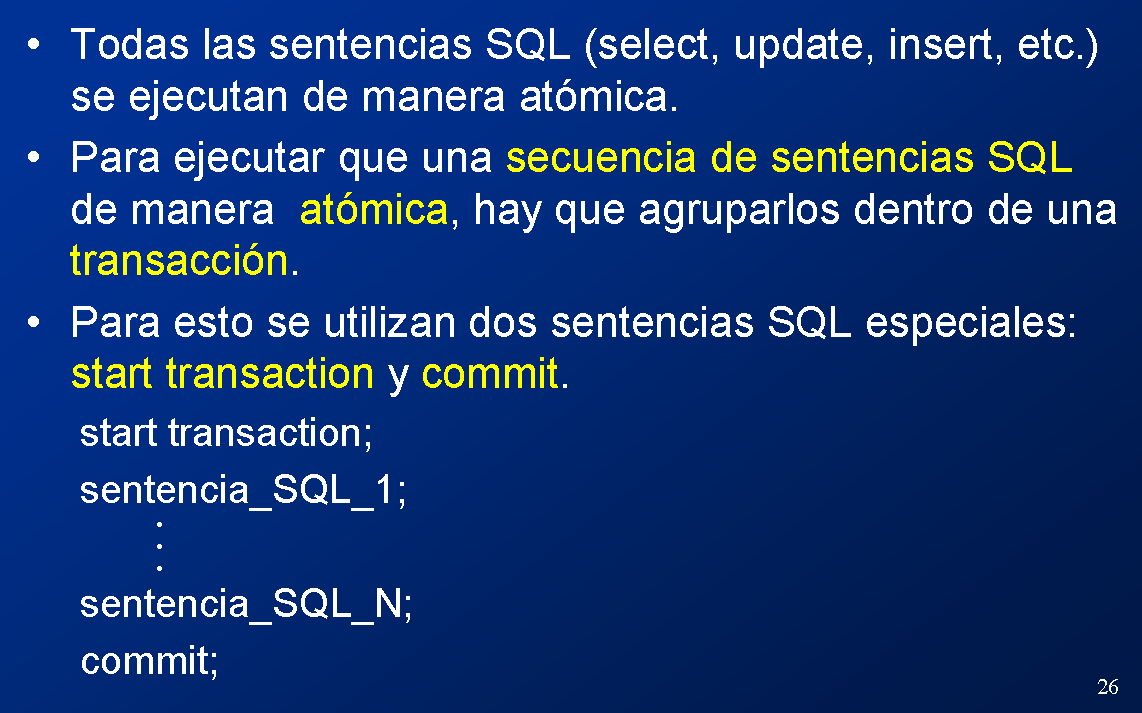
**Ejemplo2:**



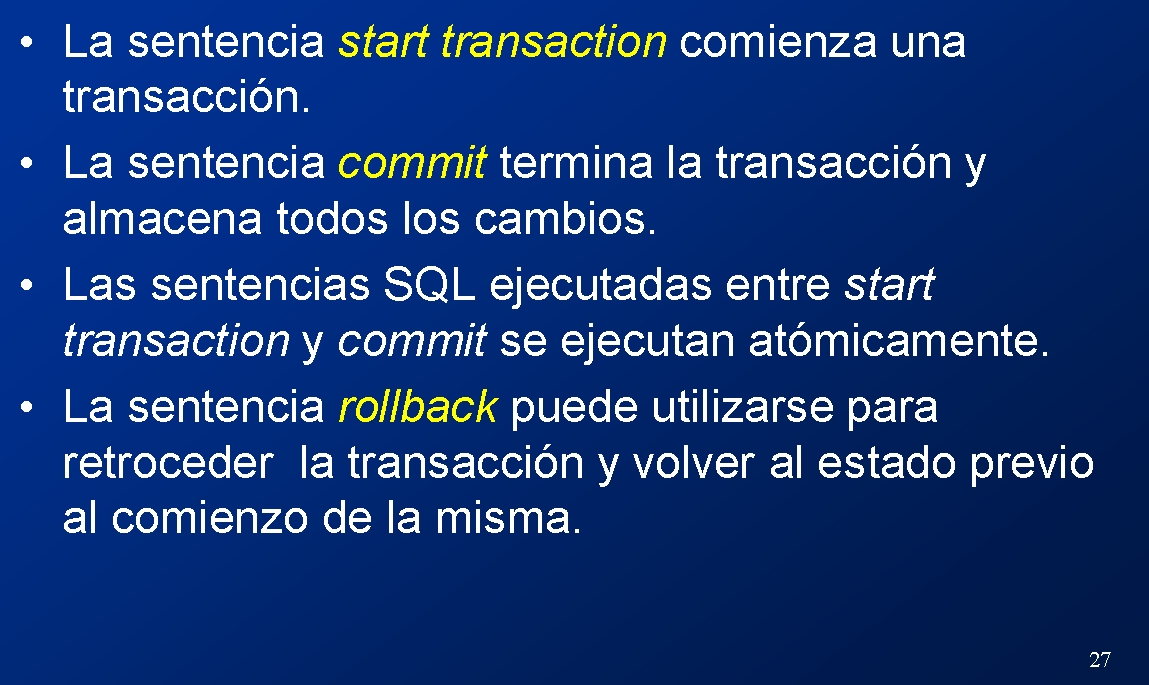
Esta incosistencia ocurre porque ambas transferencias leen el mismo valor inicial de A y cada una actualizo suponiendo que A seguía teniendo 1000 pesos (notar que en A se escribe 2 veces 500)

Solucion: Deberia haber alguna forma de garantizar que si una transaccion leyó un valor y la otra también lee el mismo entonces que no se pueda actualizar el valor o si lo leo lo dejo bloqueado para escritura.

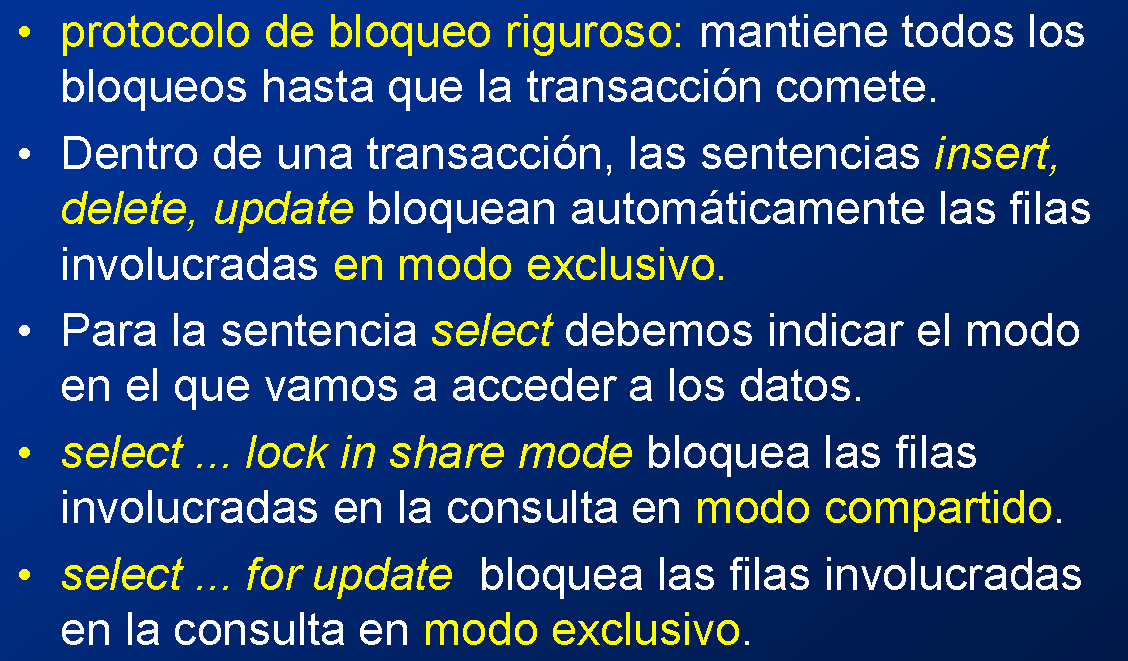




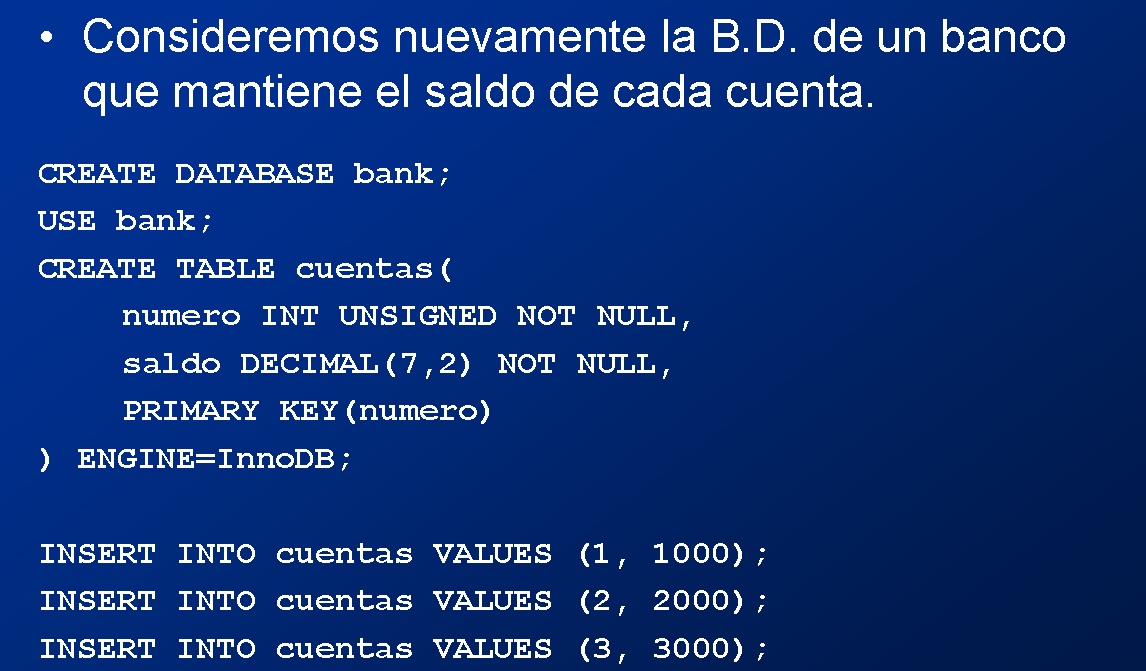
Sentencias de Transacciones

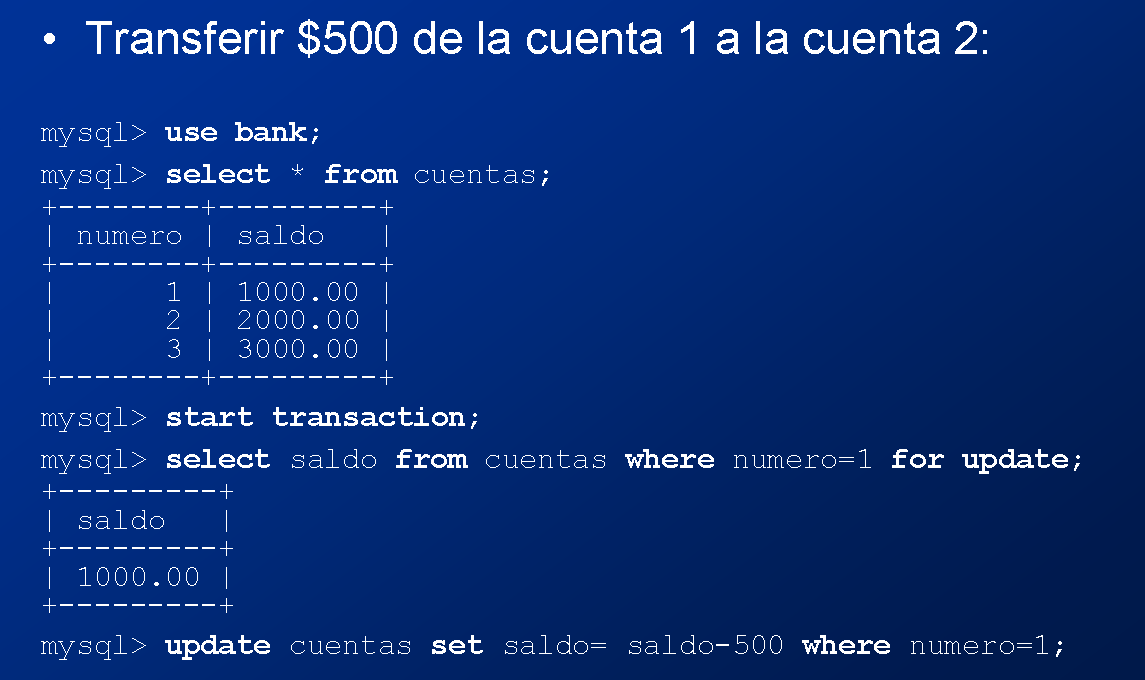


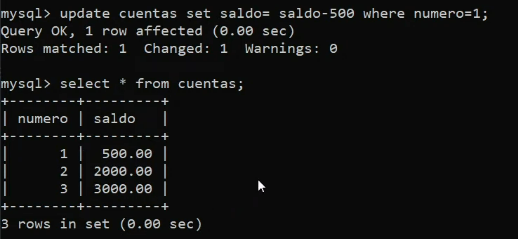
Bloqueos en las transacciones



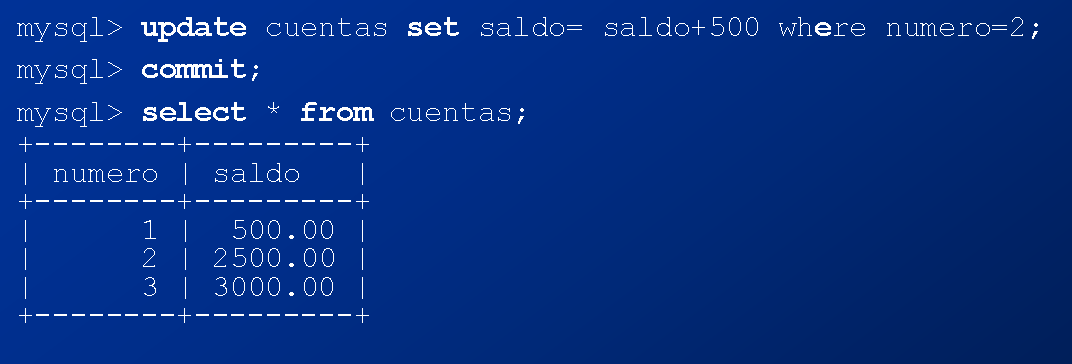
**Ejemplo:**

****

****

****

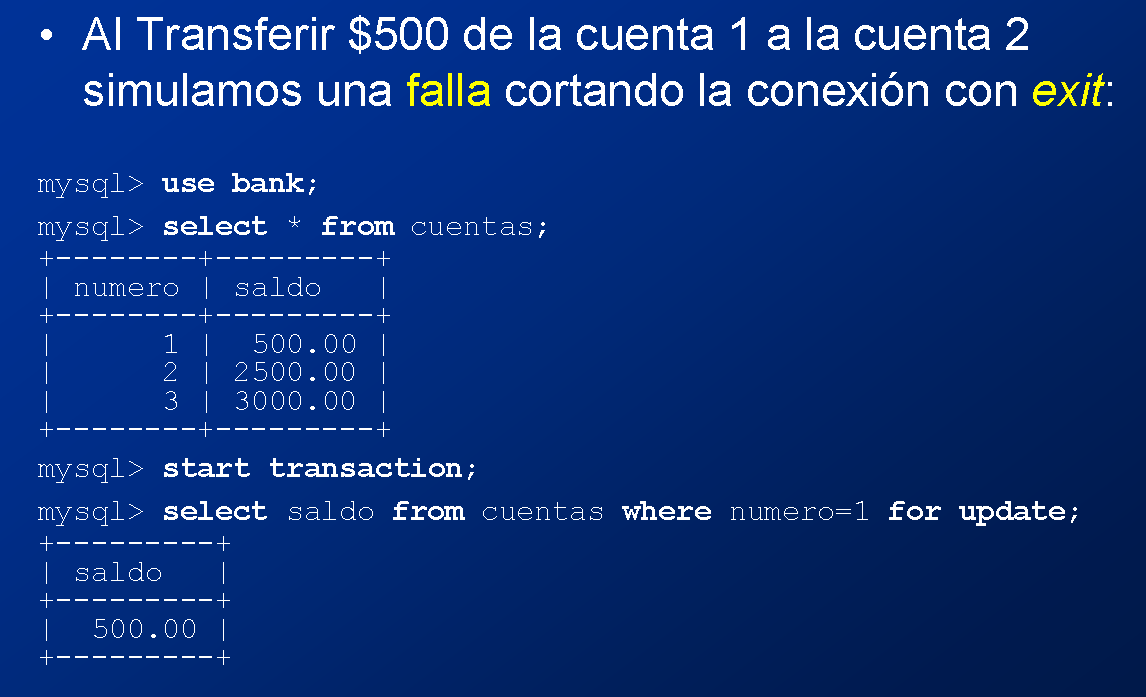
**Aca bloqueamos la consulta en modo exclusivo y actualizamos el saldo de la cuenta con numero 1 restandole 500**

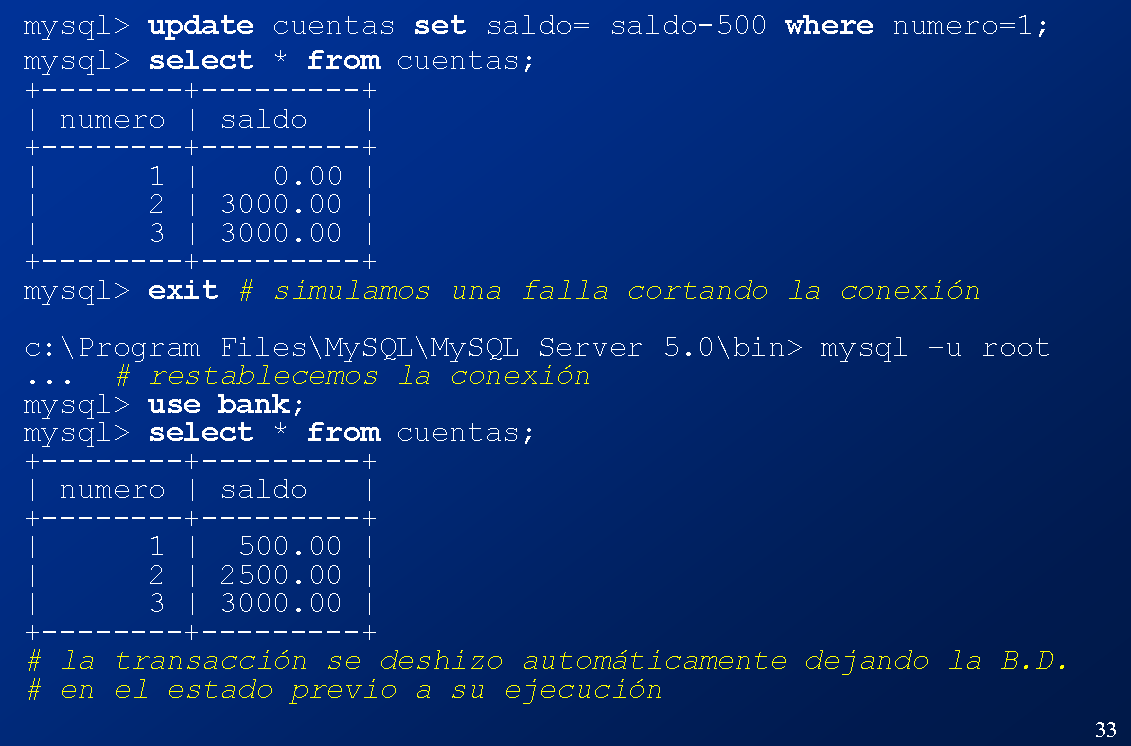
****

**Luego le sumamos 500 a la cuenta con número 2 y comiteamos la transaccion para terminarla.**

Esto sería una transferencia exitosa.

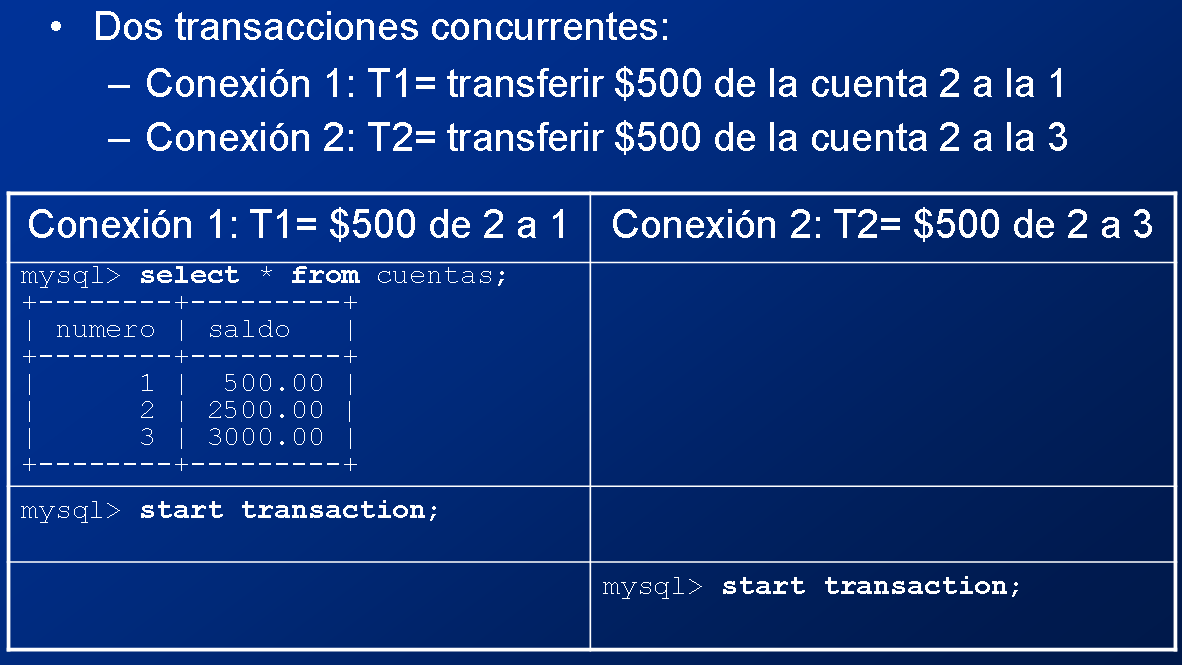
**Ejemplo 2:**

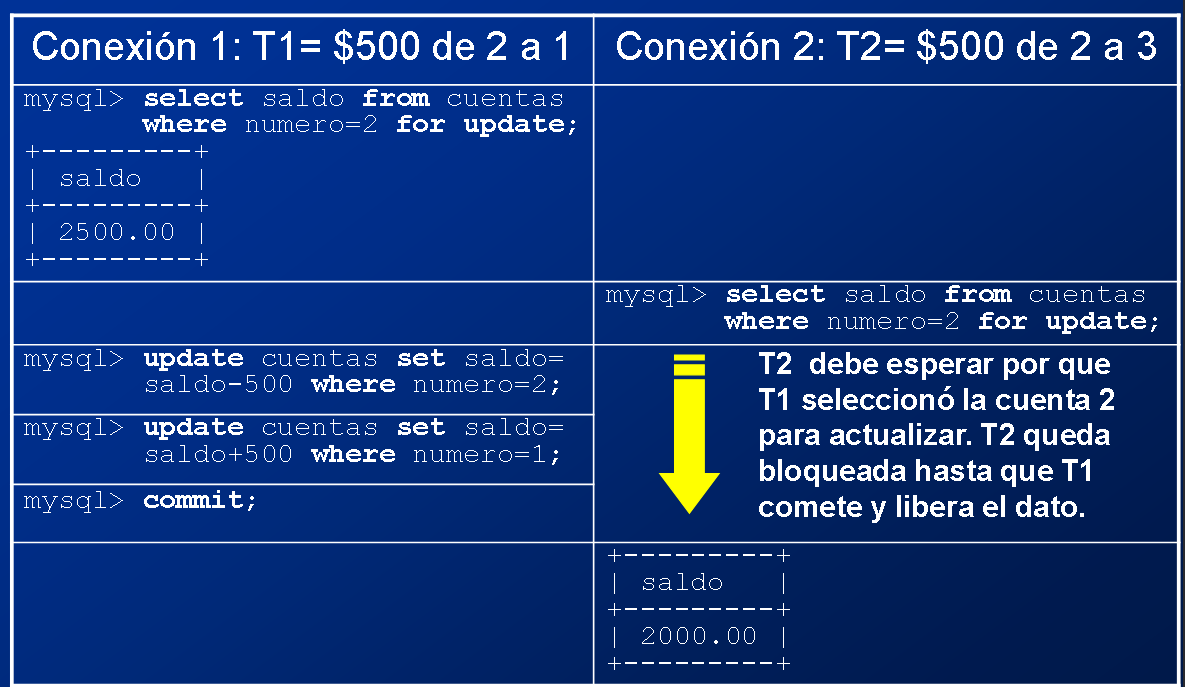
****

****

**En este caso no se había alcanzado a realizar un commit.**

**Ejemplo 3: Transacciones que ocurren concurrentemente**

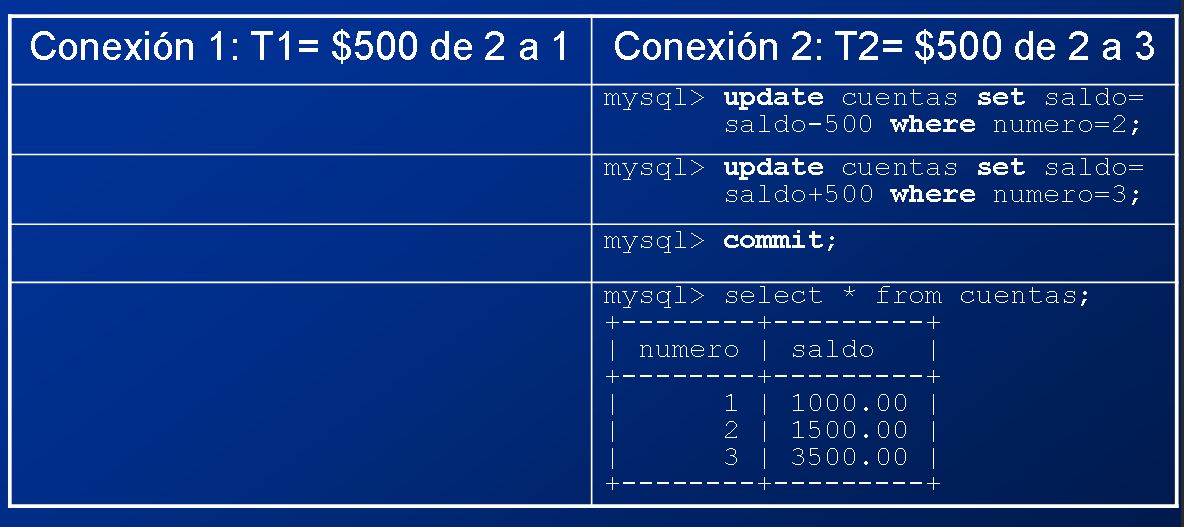
****

****

**T1 pide el bloqueo exclusivo del saldo de la cuenta con número 2, luego T2 pide exactamente lo mismo, pero como T1 lo pidió primero entonces se queda esperando a que T1 termine.**

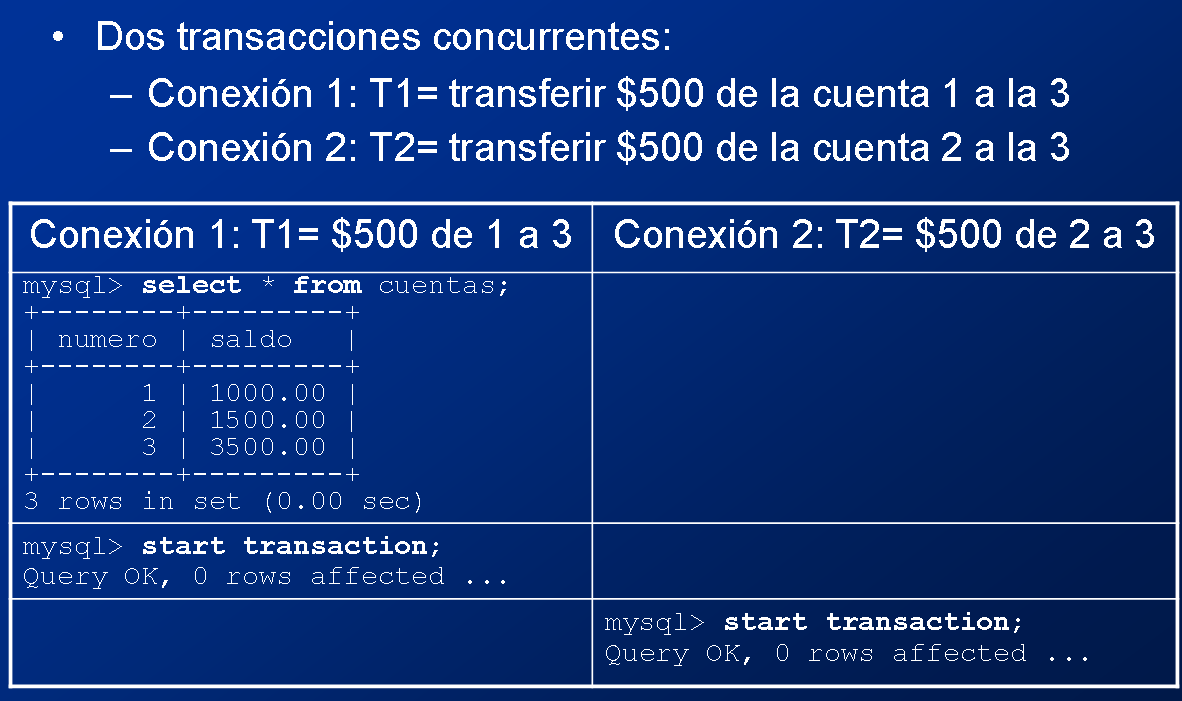
**Una vez que T1 commitea, se guardan los datos y entonces T2 inmediatamente ejecuta el bloqueo para el dato que había pedido antes.**

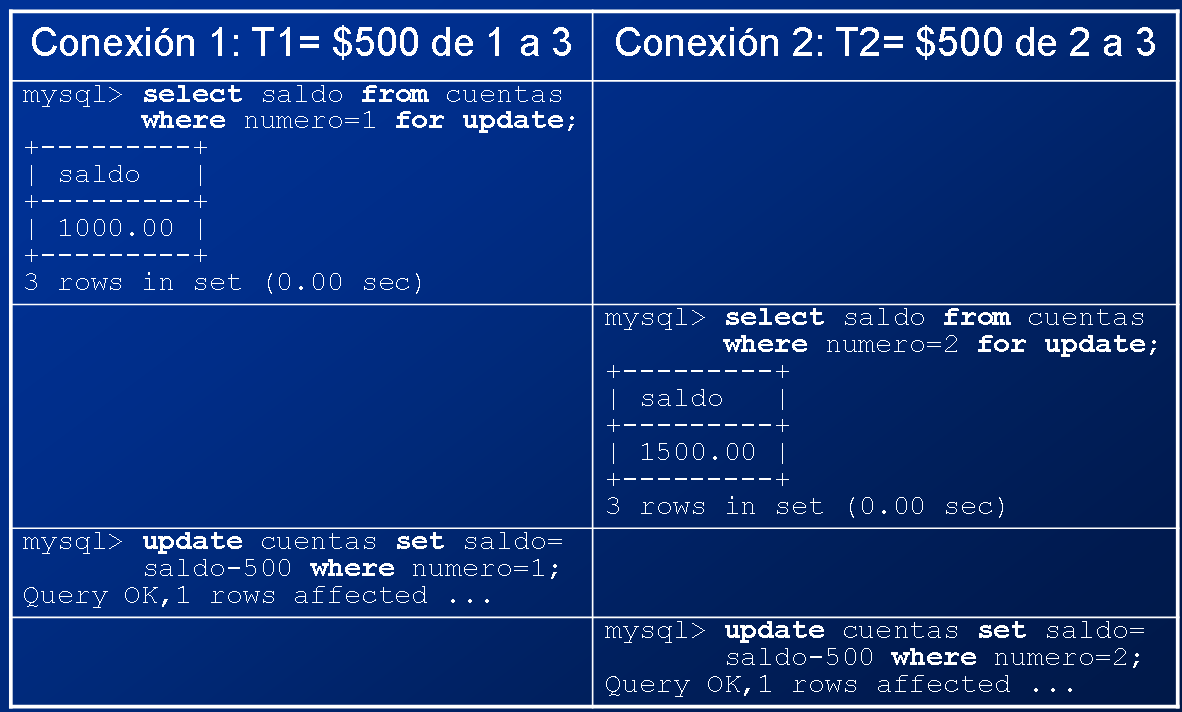
**Luego T2 realiza la misma transacción, pero con otra cuenta.**

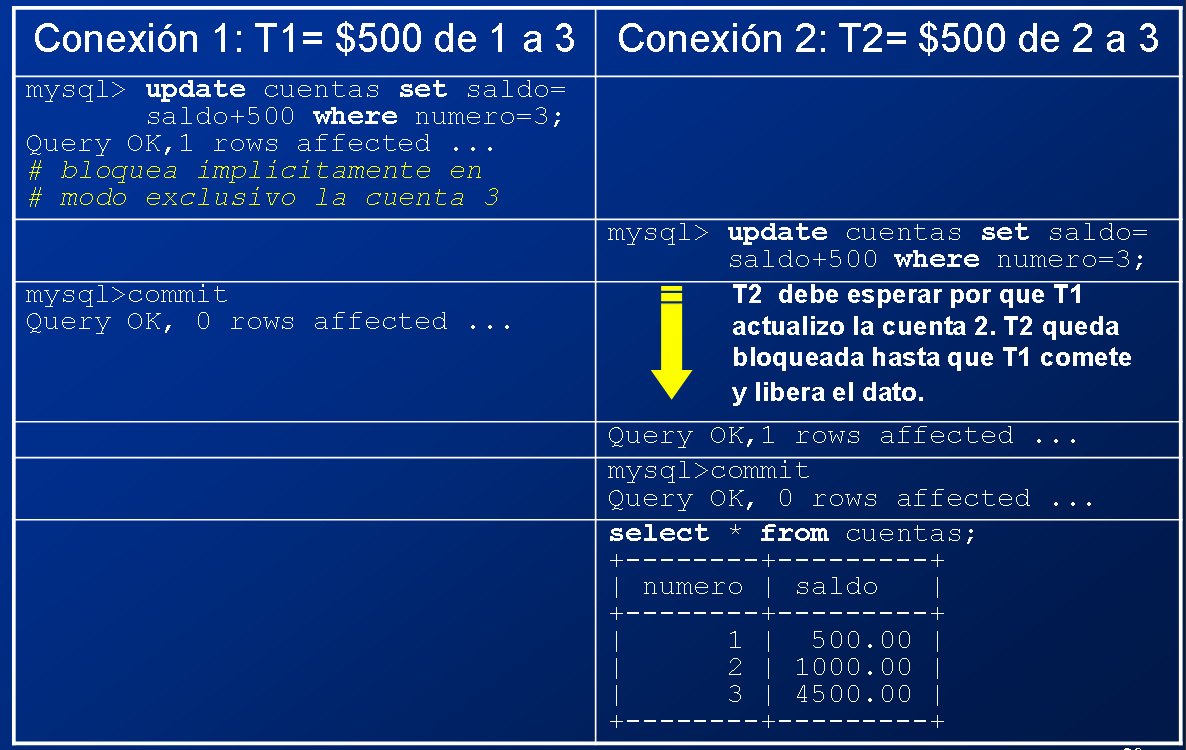
****

**2:03:30 pg37**

**Ejemplo 4:**

****

****

****