Labo 24

Revisa el contenido de la tabla clientes\_banca desde la ventana que inicializaste como la segunda sesión:  
  
SELECT \* FROM CLIENTES\_BANCA  
  
**¿Que pasa cuando deseas realizar esta consulta?**

**Se despliegan los 3 registros de clientes banca que fueron insertados en la sesión 1**

**¿Qué pasa cuando deseas realizar esta consulta?**

**La consulta se queda esperando respuesta**

**Intenta con la siguiente consulta desde la segunda sesión:**  
SELECT \* FROM CLIENTES\_BANCA where NoCuenta='001'

**La consulta se queda esperando respuesta**

**Explica por qué ocurre dicho evento.**

**Esto ocurre debido a que la primer transacción sigue pendiente de completarse. Se ha iniciado una protección de la transacción pero no se ha concluido, por lo que en este caso la transacción está pendiente y por lo tanto SqlServer no puede entregar resultados sobre esta tabla.**

Revisa nuevamente el contenido de la tabla clientes\_banca desde la ventana que inicializaste como la segunda sesión:  
  
SELECT \* FROM CLIENTES\_BANCA  
  
**¿Qué ocurrió y por qué?**

**Me regresa los 3 registros que se dieron de alta debido a que la transacción PRUEBA2 recibio un rollback. Dejando sin efecto el insert de esos tres registros.**

Posteriormente revisa si las tablas de clientes\_banca y movimientos sufrieron algún cambio, es decir, si dio de alta el registro que se describe en la transacción y su actualización.

Todo se llevo a cabo de manera correcta.

**¿Para qué sirve el comando @@ERROR revisa la ayuda en línea?**  
es una variable de transact-sql que regresa el numero de error de la ultima transacción ejecutada.

**¿Qué hace la transacción?**  
Inserta tres registros en la tabla Clientes\_Banca

Valida con la variable @@ERROR si non hubo algun error, si no hubo error, esto se comprueba con el valor 0, entonces hace efectiva la transacción con un COMMIT TRANSACTION. En caso de encontrar error, envia un mensaje y reversa la transacción con la instrucción ROLLBACK TRANSACTION.

**¿Hubo alguna modificación en la tabla? Explica qué pasó y por qué.**

Se dieron de alta tres nuevos clientes en la tabla clientes-banca

La transacción inserta tres nuevos clientes en la tabla debido a que no encuentra error al hacer la inserción de los tres nuevos registros.

**Por analogía crea las siguientes transacciones dentro de un stored procedure:**  
  
Una transacción puede estar definida dentro de un stored procedure, respetando la misma sintaxis y con la ventaja de recibir parámetros.

Una transacción que registre el retiro de una cajero. nombre del store procedure REGISTRAR\_RETIRO\_CAJERO que recibe 2 parámetros en NoCuenta y el monto a retirar.  
  
Una transacción que registre el deposito en ventanilla. Nombre del store procedure REGISTRAR\_DEPOSITO\_VENTANILLA que recibe 2 parámetros en NoCuenta y el monto a depositar.  
  
**Las transacciones deben de ser diseñadas para que en caso de que ocurra algún error la base de datos mantenga su consistencia.**

create procedure REGISTRAR\_RETIRO\_CAJERO (@cuenta varchar(5), @monto numeric(10,2) )

as

begin

SET NOCOUNT ON

BEGIN TRANSACTION retiro

INSERT INTO Movimientos VALUES(@cuenta, 'B', GETDATE(),@monto )

UPDATE CLIENTES\_BANCA SET SALDO = SALDO - @monto

WHERE NoCuenta=@cuenta

IF @@ERROR = 0

COMMIT TRANSACTION retiro

ELSE

BEGIN

PRINT 'Se presento un error en el retiro, se reversa';

ROLLBACK TRANSACTION retiro

end

end

create procedure REGISTRAR\_DEPOSITO\_VENTANILLA (@cuenta varchar(5), @monto numeric(10,2) )

as

begin

SET NOCOUNT ON

BEGIN TRANSACTION deposito

INSERT INTO Movimientos VALUES(@cuenta, 'A', GETDATE(),@monto )

UPDATE CLIENTES\_BANCA SET SALDO = SALDO + @monto

WHERE NoCuenta=@cuenta

IF @@ERROR = 0

COMMIT TRANSACTION deposito

ELSE

BEGIN

PRINT 'Se presento un error en el deposito, se reversa';

ROLLBACK TRANSACTION deposito

end

end