Ejercicios 5 de Programación en Transact-SQL

Seguiremos con la base Gestion8 y practicaremos ahora el bloque TRY y CATCH.

```
USE Gestion8
```

1.- Vamos a escribir un procedimiento que inserte en la tabla Clientes un cliente cuyos datos se habrán pasado como parámetros. El procedimiento debe controlar si ocurren errores y en caso de producirse mandará un mensaje al usuario indicando el error (código, severidad y mensaje).

```
SET NOCOUNT ON -- Como la prueba del procedimiento va a tener muchos mensajes,
quitamos el mensaje que el sistema nos envía por defecto indicando el nº de filas
afectadas.
IF object_id('InsertaCliente5_1', 'P') IS NOT NULL DROP PROC InsertaCliente5_1;
GO
CREATE PROCEDURE InsertaCliente5_1 @clie INT, @nombre CHAR(20), @repclie int, @limite
money
AS
      BEGIN TRY
            INSERT INTO Clientes (numclie, nombre, repclie, limitecredito)
                        VALUES (@clie,@nombre, @repclie,  @limite)
            PRINT 'Cliente insertado con éxito' -- si ocurre algún error el control
                              pasará al bloque CATCH y no pasará por este PRINT.
      END TRY
      BEGIN CATCH
            PRINT 'CATCH interno' -- Como después probaremos con un bloque CATCH a
                                    nivel de script ponemos una indicación para saber
                                    en qué bloque nos encontramos.
            PRINT error_number()
            PRINT error_severity()
            PRINT error_message()
      END CATCH
GO
-- Empieza la Prueba de InsertaClientes5_1. Como vamos a ejecutar varias veces el
procedimiento y en algunas provocando un error, para seguir mejor la prueba vamos a
insertar mensajes para saber por donde vamos en cada momento.
PRINT '========= Empieza la prueba sin bloque CATCH a nivel interno ========'
PRINT '== Prueba de insertar un cliente que no existe - inserción correcta - =='
EXEC InsertaCliente5_1 6000,'Luis',NULL,0
PRINT '== Prueba de insertar un cliente que ya existe =='
EXEC InsertaCliente5_1 2101,'Luis',NULL,0
PRINT '== Prueba de insertar un cliente que no existe con un valor cuyo tipo no
corresponde =='
exec InsertaCliente5_1 6001,'Luis',NULL,'ll'
PRINT '== Prueba de insertar un cliente que no existe con un repclie incorrecto =='
EXEC InsertaCliente5_1 6001,'Luis',500,0
```

Ejecutamos el script escrito hasta ahora y obtenemos el siguiente resultado:

```
========= Empieza la prueba sin bloque CATCH a nivel interno ========
== Prueba de insertar un cliente que no existe - inserción correcta - ==
Cliente insertado con éxito   ← Este es el PRINT enviado por el procedimiento después del INSERT
== Prueba de insertar un cliente que ya existe ==
CATCH interno ← Hemos entrado al CATCH del procedimiento pq se ha producido un error.
2627          ← Este es el nº del error
14            ← Nos indica la gravedad del error
Infracción de la restricción PRIMARY KEY 'PK_clientes'. No se puede insertar una clave duplicada en el
objeto 'dbo.Clientes'.       ← Este es el mensaje del error. ¿Te suena? Hemos intentado insertar
un cliente que ya existe en la tabla.
Después se sale del procedimiento y se vuelve al script, a la instrucción posterior al EXEC.
== Prueba de insertar un cliente que no existe con un valor cuyo tipo no corresponde ==
Mens 8114, Nivel 16, Estado 1, Procedimiento InsertaCliente5_1, Línea 0
Error al convertir el tipo de datos varchar a money.  ← En este caso se ha producido un error pero
no se ha entrado en el bloque CATCH porque este es un error no capturado a este nivel.
Ha saltado el error, el sistema nos ha enviado su mensaje pero después se sigue con el script.
== Prueba de insertar un cliente que no existe con un repclie incorrecto ==
```

```
CATCH interno  ← Hemos entrado al CATCH del procedimiento pq se ha producido un error.
547            ← Este es el nº del error
16             ← Nos indica la gravedad del error
Instrucción INSERT en conflicto con la restricción FOREIGN KEY "FK_clientes_repclie". El conflicto ha
aparecido en la base de datos "Gestion11", tabla "dbo.empleados", column 'numemp'.      ← Este es el
mensaje del error. ¿Te suena? Si no repasa el concepto de integridad referencial.
```

Fíjate que este error tiene el mismo nivel de gravedad (16) que el anterior pero este tipo de error ha sido detectado por el control de excepciones mientras que el otro no. Bueno el anterior también ha sido controlado pero el control, en ese caso, no se pasa al CATCH del mismo nivel sino al del nivel superior. Lo vamos a comprobar de la siguiente forma, añade al script el siguiente código:

```
-- Ahora probaremos lo mismo pero incluyendo los EXEC dentro de un bloque TRY para
comprobar qué pasa con los errores no detectados a nivel de procedimiento.

PRINT '========= Empieza la prueba con bloque CATCH a nivel externo ========'
BEGIN TRY
      PRINT '== Prueba de insertar un cliente que no existe - inserción correcta -
==='
      EXEC InsertaCliente5_1 7000,'Luis',NULL,0 -- hemos cambiado a 7000 porque el
6000 ya existe, lo acabamos de insertar.
      PRINT '== Prueba de insertar un cliente que ya existe =='
      EXEC InsertaCliente5_1 2101,'Luis',NULL,0
      PRINT '== Prueba de insertar un cliente que no existe con un valor cuyo tipo no
corresponde =='
      EXEC InsertaCliente5_1 6001,'Luis',NULL,'ll'
      PRINT 'Prueba de insertar un cliente que no existe con un repclie incorrecto'
      EXEC InsertaCliente5_1 6001,'Luis',500,0
END TRY
BEGIN CATCH
      PRINT 'CATCH externo'  -- Para saber que hemos entrado en este CATCH
      PRINT error_number()
      PRINT error_severity()
      PRINT error_message()
END CATCH
```

Veamos el resultado obtenido, esto es lo que debe de aparecer en la pestaña mensajes si ejecutamos el script completo:

```
========= Empieza la prueba sin bloque CATCH a nivel interno ========
== Prueba de insertar un cliente que no existe - inserción correcta - ==
Cliente insertado con éxito
== Prueba de insertar un cliente que ya existe ==
CATCH interno
2627
14
Infracción de la restricción PRIMARY KEY 'PK_clientes'. No se puede insertar una clave duplicada en el
objeto 'dbo.Clientes'.
== Prueba de insertar un cliente que no existe con un valor cuyo tipo no corresponde ==
Mens 8114, Nivel 16, Estado 1, Procedimiento InsertaCliente5_1, Línea 0
Error al convertir el tipo de datos varchar a money.
== Prueba de insertar un cliente que no existe con un repclie incorrecto ==
CATCH interno
547
16
Instrucción INSERT en conflicto con la restricción FOREIGN KEY "FK_clientes_repclie". El conflicto ha
aparecido en la base de datos "Gestion11", tabla "dbo.empleados", column 'numemp'.
========= Empieza la prueba con bloque CATCH a nivel externo ========
== Prueba de insertar un cliente que no existe - inserción correcta - ==
Cliente insertado con éxito
== Prueba de insertar un cliente que ya existe ==
CATCH interno          ← Ha entrado en el CATCH del procedimiento. Cuando termine con el CATCH
                       (después del PRINT del mensaje)saldrá del procedimiento.
2627
14
Infracción de la restricción PRIMARY KEY 'PK_clientes'. No se puede insertar una clave duplicada en el
objeto 'dbo.Clientes'.
== Prueba de insertar un cliente que no existe con un valor cuyo tipo no corresponde ==
CATCH externo          ← No ha entrado en el CATCH del procedimiento, pero ha enviado el error al
nivel superior (el script en este caso) y como el script tiene CATCH, entra en el CATCH del
```

```
script. Cuando termine con el CATCH (después del PRINT del mensaje)saldrá del script. Y
entonces no se ejecutarán las instrucciones posteriores al EXEC que ha provocado el error.
16
Error al convertir el tipo de datos varchar a money.
```

El último EXEC no se ejecuta.

2.- El problema que nos podemos encontrar cuando se hacen pruebas con instrucciones que cambian los datos de la base de datos es que no se pueden reutilizar los valores en pruebas sucesivas. Por ejemplo la primera prueba la hemos realizado insertando el cliente 6000 que no existe, pero si volvemos a ejecutar el script el cliente 6000 ya existirá y entonces tendremos que cambiar el valor 6000 por otro nuevo. Esto puede llegar a ser muy engorroso cuando estamos poniendo a punto un programa. Una forma de solucionarlo de forma sencilla es definiendo una transacción.
Una transacción está formada por un conjuntos de instrucciones que se tienen que ejecutar o todas o ninguna, el lenguaje nos permite delimitar una trasacción y determinar si la damos por buena o si queremos deshacerla con lo cual se volverá al estado inicial de la base de datos antes de iniciar la transacción y todas las instrucciones ejecutadas dentro de la transacción se desharán.
Una transacción se inicia con la instrucción BEGIN TRANSACTION . Si queremos finalizar una transacción dando por buenos los cambios realizados utilizamos COMMIT TRANSACTION , si queremos abortar la transacción y que lo cambios realizados queden anulados utilizamos ROLLBACK TRANSACTION.
La palabra TRANSACTION en todos los casos se puede sustituir por su abreviatura TRAN.

Pruébalo añadiendo un BEGIN TRANSACTION después del comentario: -- Empieza la Prueba de InsertaClientes5_1 (antes del primer EXEC), y escribe un ROLLBACK TRANSACTION al final del script, para que todos los cambios realizados no surtan efecto.
Ejecuta el script completo y comprueba en la tabla Clientes como los registros insertados con éxito no están en la tabla.

Quien quiera profundizar en el tema puede seguir este enlace:
http://msdn.microsoft.com/en-us/library/ms188929(v=sql.90).aspx

3.- Ahora vamos a mejorar nuestra rutina de error para enviar al usuario mensajes más fáciles de interpretar.

```
USE Gestion8
IF object_id('InsertaCliente5_2', 'P') IS NOT NULL DROP PROC InsertaCliente5_2;
GO
CREATE PROCEDURE InsertaCliente5_2 @clie INT, @nombre CHAR(20), @repclie int, @limite
money
AS
      BEGIN TRY
            INSERT INTO Clientes (numclie, nombre, repclie, limitecredito) VALUES
(@clie,@nombre, @repclie,  @limite)
            PRINT 'Cliente insertado con éxito'
      END TRY
      BEGIN CATCH
             PRINT CASE ERROR_NUMBER()
                  WHEN 2627 THEN 'ERROR: El cliente ya existe'
                  WHEN 547  THEN  'ERROR: El representante asignado no es un
empleado'
                  WHEN 8114  THEN  'ERROR: El tipo de datos asignado no es correcto'
                  ELSE  'Se ha producido el siguiente error: ' + str(ERROR_NUMBER())
+ ERROR_MESSAGE()
                  END
      END CATCH
GO
-- Empieza la Prueba de InsertaClientes5_2
BEGIN TRANSACTION
SET NOCOUNT ON
PRINT '===== Empieza la prueba sin CATCH EXTERNO ====='
PRINT '== Prueba de insertar un cliente que no existe - inserción correcta - =='
EXEC InsertaCliente5_2 8000,'Luis',NULL,0
PRINT '== Prueba de insertar un cliente que ya existe =='
```

```
EXEC InsertaCliente5_2 2101,'Luis',NULL,0
PRINT '== Prueba de insertar un cliente que no existe con un valor cuyo tipo no corresponde =='
EXEC InsertaCliente5_2 6001,'Luis',NULL,'ll'
PRINT '== Prueba de insertar un cliente que no existe con un repclie incorrecto =='
EXEC InsertaCliente5_2 6001,'Luis',500,0
-- Empieza la Prueba con CATCH externo
      PRINT '===== Empieza la segunda prueba ahora con los EXEC dentro de un TRY ====='
BEGIN TRY
      PRINT '== Prueba de insertar un cliente que no existe - inserción correcta - =='
      EXEC InsertaCliente5_2 8001,'Luis',NULL,0
      PRINT '== Prueba de insertar un cliente que ya existe =='
      EXEC InsertaCliente5_2 2101,'Luis',NULL,0
      PRINT '== Prueba de insertar un cliente que no existe con un valor cuyo tipo no corresponde =='
      EXEC InsertaCliente5_2 6001,'Luis',NULL,'ll'
      PRINT '== Prueba de insertar un cliente que no existe con un repclie incorrecto =='
      EXEC InsertaCliente5_2 6001,'Luis',500,0
END TRY
BEGIN CATCH
      IF ERROR_NUMBER() = 2627  PRINT 'ERROR: El cliente ya existe'
      ELSE IF ERROR_NUMBER() = 547  PRINT  'ERROR: El representante asignado no es un empleado'
            ELSE IF ERROR_NUMBER() = 8114  PRINT  'ERROR: El tipo de datos asignado no es correcto'
                  ELSE PRINT 'Se ha producido el siguiente error: ' + str(ERROR_NUMBER()) + ERROR_MESSAGE()
END CATCH
ROLLBACK TRAN
```

El resultado es:

```
===== Empieza la prueba sin CATCH EXTERNO =====
== Prueba de insertar un cliente que no existe - inserción correcta - ==
Cliente insertado con éxito
== Prueba de insertar un cliente que ya existe ==
ERROR: El cliente ya existe
== Prueba de insertar un cliente que no existe con un valor cuyo tipo no corresponde ==
Mens 8114, Nivel 16, Estado 1, Procedimiento InsertaCliente5_1, Línea 0
Error al convertir el tipo de datos varchar a money.
== Prueba de insertar un cliente que no existe con un repclie incorrecto ==
ERROR: El representante asignado no es un empleado
===== Empieza la segunda prueba ahora con los EXEC dentro de un TRY =====
== Prueba de insertar un cliente que no existe - inserción correcta - ==
Cliente insertado con éxito
== Prueba de insertar un cliente que ya existe ==
ERROR: El cliente ya existe
== Prueba de insertar un cliente que no existe con un valor cuyo tipo no corresponde ==
ERROR: El tipo de datos asignado no es correcto
```

Nota.

El código incluido en cada uno de los bloques CATCH es diferente para que veas diferentes formas de resolver lo mismo. En los dos casos enviamos el mismo mensaje dependiendo del código del error ocurrido.

En el CATCH interno se ha utilizado el CASE mientras que en el CATCH externo lo hemos resuelto con IF.

En el caso del CASE, el CASE que tenemos en Transact-SQL actúa como una función, devuelve un valor dependiendo de una condición (o varias), por eso se ha puesto después de PRINT.

La interpretación sería PRINT “un mensaje” pero el mensaje será lo devuelto por el CASE por eso escribimos PRINT CASE .... END.

En el caso del IF indicamos:

Si el error ha sido tal número envía tal mensaje,

si no, si el error ha sido tal otro número entonces envía tal mensaje,….