

CEPSUNI

www.ceps.uni.edu.pe

PROGRAMACIÓN



Microsoft®
SQL Server®

UNIDAD 04

CONTROL DE ERRORES

GUSTAVO CORONEL

www.youtube.com/DesarrollaSoftware

gcoronel@uni.edu.pe

Temas

1	CONTROL DE ERRORES	3
1.1	VARIABLE: @@ROWCOUNT	3
1.2	FUNCIÓN: ROWCOUNT_BIG ()	3
1.3	VARIABLE: @@ERROR	3
1.4	FUNCIÓN: RAISERROR ()	4
2	MANEJO DE EXCEPCIONES	4
2.1	ESTRUCTURA TRY/CATCH	4
2.2	SENTENCIA: THROW	8
3	REQUERIMIENTOS A RESOLVER	9
3.1	REQUERIMIENTO 1.....	9
3.2	REQUERIMIENTO 2.....	9
3.3	REQUERIMIENTO 2.....	10
4	CURSOS RELACIONADOS	11

1 CONTROL DE ERRORES

1.1 Variable: @@ROWCOUNT

Devuelve el número de filas afectadas por la última instrucción. Si el número de filas es mayor de 2 mil millones, use ROWCOUNT_BIG.

```
@@ROWCOUNT
```

Más información en: <http://technet.microsoft.com/es-pe/library/ms187316.aspx>

1.2 Función: ROWCOUNT_BIG ()

Devuelve el número de filas afectadas por la última instrucción ejecutada. Esta función actúa como @@ROWCOUNT, pero el tipo de valor devuelto de ROWCOUNT_BIG es bigint.

```
ROWCOUNT_BIG ( )
```

Más información en: <http://technet.microsoft.com/es-es/library/ms181406.aspx>

1.3 Variable: @@ERROR

Devuelve el número de error de la última instrucción Transact-SQL ejecutada.

```
@@ERROR
```

Más información en: <http://technet.microsoft.com/es-es/library/ms188790.aspx>

1.4 Función: RAISERROR ()

Genera un mensaje de error e inicia el procesamiento de errores de la sesión. RAISERROR puede hacer referencia a un mensaje definido por el usuario almacenado en la vista de catálogo **sys.messages** o puede generar un mensaje dinámicamente. El mensaje se devuelve como un mensaje de error de servidor a la aplicación que realiza la llamada o a un bloque CATCH asociado a una estructura TRY...CATCH.

Las nuevas aplicaciones deben utilizar THROW en su lugar.

```
RAISERROR ( { msg_id | msg_str | @local_variable }  
           { ,severity ,state }  
           [ ,argument [ ,...n ] ] );
```

Más información en: <http://msdn.microsoft.com/es-es/library/ms178592.aspx>

2 MANEJO DE EXCEPCIONES

2.1 Estructura TRY/CATCH

Una mejora importante que tenemos en SQL Server es el manejo de errores que ahora es posible en T-SQL con los bloques TRY/CATCH, sin olvidar la sintaxis que utilizamos para las transacciones.

Sintaxis:

```
BEGIN TRY  
    BEGIN TRANSACTION;  
  
    -- Bloque de código SQL a proteger  
  
    COMMIT TRANSACTION;  
END TRY  
BEGIN CATCH  
    ROLLBACK TRANSACTION;  
    -- Código para mostrar el mensaje de la excepción  
END CATCH;
```

Si ocurre un error dentro de la transacción en un bloque TRY inmediatamente se dirige al bloque CATCH.

Para poder acceder a la información del error tenemos las siguientes funciones:

ERROR_NUMBER()	: Devuelve el número de error.
ERROR_MESSAGE()	: Devuelve el texto completo del mensaje de error. El texto incluye los valores suministrados para los parámetros sustituibles, como longitudes, nombres de objeto u horas.
ERROR_SEVERITY()	: Devuelve la gravedad del error.
ERROR_STATE()	: Devuelve el número de estado de error.
ERROR_LINE()	: Devuelve el número de línea donde se produjo el error.
ERROR_PROCEDURE()	: Devuelve el nombre del procedimiento donde se produjo el error.

Más información en: <http://technet.microsoft.com/es-pe/library/ms175976.aspx>

Pasemos a crear un ejemplo bastante sencillo en donde vamos a provocar una excepción en una división por cero para observar el resultado de estas funciones.

Ejemplo 1: Captura de Error

```
BEGIN TRY
    DECLARE @TOTAL INT;
    SET @TOTAL = 20;
    SELECT @TOTAL/0
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS Numero_de_Error,
        ERROR_SEVERITY() AS Gravedad_del_Error,
        ERROR_STATE() AS Estado_del_Error,
        ERROR_PROCEDURE() AS Procedimiento_del_Error,
        ERROR_LINE() AS Linea_de_Error,
        ERROR_MESSAGE() AS Mensaje_de_Error;
END CATCH;
GO
```

Un bloque CATCH no controla los siguientes tipos de errores cuando se producen en el mismo nivel de ejecución que la construcción TRY...CATCH:

- Errores de compilación, como errores de sintaxis, que impiden la ejecución de un lote.
- Errores que se producen durante la recompilación de instrucciones, como errores de resolución de nombres de objeto que se producen después de la compilación debido a una resolución de nombres diferida.

Estos errores se devuelven al nivel de ejecución del lote, procedimiento almacenado o desencadenador.

En el Ejemplo 2 se muestra cómo la estructura TRY...CATCH no captura un error de resolución de nombre de objeto generado por una instrucción SELECT, sin embargo, el bloque CATCH si captura el error cuando la misma instrucción SELECT se ejecuta dentro de un procedimiento almacenado, tal como se puede apreciar en el Ejemplo 3.

Ejemplo 2: Error no capturado

```
BEGIN TRY
    SELECT * FROM TablaNoExiste;
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS ErrorNumber,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

Mens. 208, Nivel 16, Estado 1, Línea 2
El nombre de objeto 'TablaNoExiste' no es válido.

El error no se captura y el control se transfiere hacia fuera de la estructura TRY...CATCH, al siguiente nivel superior.

Al ejecutar la instrucción SELECT dentro de un procedimiento almacenado, el error se produce en un nivel inferior al bloque TRY. La estructura TRY...CATCH controlará el error.

Ejemplo 3: Captura de error de un nivel inferior

```
USE EDUCA;
GO

IF OBJECT_ID ( N'usp_ProcEjemplo', N'P' ) IS NOT NULL
    DROP PROCEDURE usp_ProcEjemplo;
GO

CREATE PROCEDURE usp_ProcEjemplo
AS
    SELECT * FROM TablaNoExiste;
GO

BEGIN TRY
    EXECUTE usp_ProcEjemplo;
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS ErrorNumber,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO

ErrorNumber ErrorMessage
-----
208          El nombre de objeto 'TablaNoExiste' no es válido.

(1 filas afectadas)
```

Más información en: <http://technet.microsoft.com/es-pe/library/ms175976.aspx>

2.2 Sentencia: THROW

Genera una excepción y transfiere el control a un bloque CATCH de una estructura TRY...CATCH.

Sintaxis:

```
THROW [ { error_number | @local_variable },  
        { message | @local_variable },  
        { state | @local_variable }  
] [ ; ]
```

Ejemplo 4: Generar una nueva excepción

```
THROW 51000, 'El registro no existe.', 1;
```

Ejemplo 5: Propagar la última excepción

```
BEGIN TRY  
  
    -- Bloque a controlar  
  
END TRY  
BEGIN CATCH  
  
    -- Proceso de control  
  
    THROW; -- Propaga la última excepción  
END CATCH;
```

Más información en: <http://technet.microsoft.com/es-pe/library/ee677615.aspx>

3 REQUERIMIENTOS A RESOLVER

3.1 Requerimiento 1

Base de Datos: **EDUTEC**

Se necesita un procedimiento que permita generar un nuevo ciclo, las condiciones son las siguientes:

1. Por cada año existen doce ciclos, uno por mes, por ejemplo: 2020-01, 2020-02, 2020-03, etc.
2. El procedimiento debe leer el último ciclo y generar el que continua.
3. La fecha de inicio es el primer día del mes.
4. La fecha de fin es el último día del mes.
5. El nuevo ciclo se debe grabar en la tabla CICLO.

3.2 Requerimiento 2

Base de Datos: **EDUTEC**

Se necesita un procedimiento para registrar una nueva matricula, las condiciones son las siguientes:

1. Se debe verificar que el curso programado exista y se encuentre vigente.
2. Se debe verificar que el alumno exista.
3. Se debe verificar que el alumno no se encuentre matriculado.
4. Se debe verificar que existan vacantes disponibles.
5. La matrícula se registra en la tabla MATRICULA.
6. Se deben actualizar las columnas **vacantes** y **matriculados** en la tabla CURSOPROGRAMADO.

3.3 Requerimiento 2

Base de datos: **EDUCA**

Se necesita un procedimiento para registrar un nuevo pago, las condiciones son las siguientes:

1. Se debe verificar que exista la matricula.
2. Se debe verificar que exista pago pendiente.
3. Si el pago es en una cuota, debe ser por el total del costo del curso.
4. Si el pago es la primera cuota, debe ser mínimo por el 30% del costo del curso.
5. Si se trata de la última cuota, debe ser por el saldo pendiente.
6. Si el pago es por el saldo pendiente, y es necesario corregir el número de cuotas se debe realizar.
7. En otros casos se debe aceptar el pago de la cuota.

4 CURSOS RELACIONADOS

<https://www.ceps.uni.edu.pe/>



