SQL Programming

Módulo 7 - Tratamiento de errores

Tratamiento de errores

@@ERROR

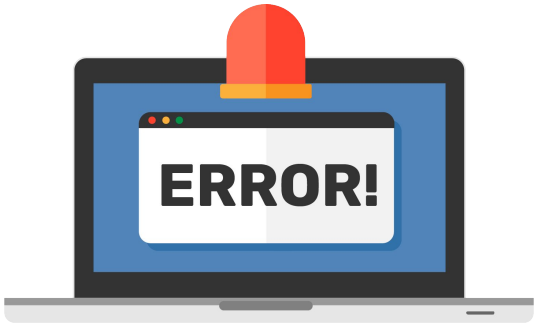
Devuelve el número de error de la última instrucción Transact-SQL ejecutada.

Sintaxis

```
BEGIN TRANSACTION

      SELECT 1/0;
      IF @@ERROR<>0
          BEGIN
              ROLLBACK TRANSACTION;
              RETURN;
      END

      COMMIT TRANSACTION
GO
```



Vistas de catálogo sys.messages

Contiene una fila para cada message_id o language_id los mensajes de error en el sistema, para los mensajes definidos por el usuario y definido por el sistema.

Sintaxis

```
SELECT * FROM sys.messages;
```

TRY... CATCH

Implementa un mecanismo de control de errores para Transact-SQL que es similar al control de excepciones en los lenguajes Microsoft Visual C# y Microsoft Visual C++.

Se puede incluir un grupo de instrucciones Transact-SQL en un bloque TRY.

Si se produce un error en el bloque TRY, el control se transfiere a otro grupo de instrucciones que está incluido en un bloque CATCH.

Sintaxis

```
-- Entidades necesarias
CREATE TABLE Errores(ID INT, Valor MONEY);

BEGIN TRY
    BEGIN TRANSACTION
        INSERT Errores (id) VALUES (1)
        UPDATE Errores SET Valor = 1/0 WHERE ID = 1;
        COMMIT TRANSACTION
END TRY
BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
END CATCH

SELECT * FROM Errores;
```

Notas

Una construcción TRY...CATCH detecta todos los errores de ejecución que tienen una gravedad mayor de 10 y que no cierran la conexión de la base de datos.

Un bloque TRY debe ir seguido inmediatamente por un bloque CATCH asociado. Si se incluye cualquier otra instrucción entre las instrucciones END TRY y BEGIN CATCH se genera un error de sintaxis.

Una construcción TRY...CATCH no puede abarcar varios lotes. Una construcción TRY...CATCH no puede abarcar varios bloques de instrucciones Transact-SQL. Por ejemplo, una construcción TRY...CATCH no puede abarcar dos bloques BEGIN...END de instrucciones Transact-SQL y no puede abarcar una construcción IF...ELSE.

Si no hay errores en el código incluido en un bloque TRY, cuando la última instrucción de este bloque ha terminado de ejecutarse, el control se transfiere a la instrucción inmediatamente posterior a la instrucción END CATCH asociada. Si hay un error en el código incluido en un bloque TRY, el control se transfiere a la primera instrucción del bloque CATCH asociado. Si la instrucción END CATCH es la última instrucción de un procedimiento almacenado o desencadenador, el control se devuelve a la instrucción que llamó al procedimiento almacenado o activó el desencadenador.

Notas

Cuando finaliza el código del bloque CATCH, el control se transfiere a la instrucción inmediatamente posterior a la instrucción END CATCH. Los errores capturados por un bloque CATCH no se devuelven a la aplicación que realiza la llamada. Si es necesario devolver cualquier parte de la información sobre el error a la aplicación, debe hacerlo el código del bloque CATCH a través de mecanismos como los conjuntos de resultados SELECT o las instrucciones RAISERROR y PRINT.

Si no hay errores en el código incluido en un bloque TRY, cuando la última instrucción de este bloque ha terminado de ejecutarse, el control se transfiere a la instrucción inmediatamente posterior a la instrucción END CATCH asociada. Si hay un error en el código incluido en un bloque TRY, el control se transfiere a la primera instrucción del bloque CATCH asociado. Si la instrucción END CATCH es la última instrucción de un procedimiento almacenado o desencadenador, el control se devuelve a la instrucción que llamó al procedimiento almacenado o activó el desencadenador.

Las construcciones TRY...CATCH pueden estar anidadas. Un bloque TRY o un bloque CATCH puede contener construcciones TRY...CATCH anidadas. Por ejemplo, un bloque CATCH puede contener una construcción TRY...CATCH incrustada para controlar los errores detectados por el código de CATCH.

Los errores que se encuentren en un bloque CATCH se tratan como los errores generados en otros lugares. Si el bloque CATCH contiene una construcción TRY...CATCH anidada, los errores del bloque TRY anidado transferirán el control al bloque CATCH anidado. Si no hay ninguna construcción TRY...CATCH anidada, el error se devuelve al autor de la llamada.

Notas

Las construcciones TRY...CATCH capturan los errores no controlados de los procedimientos almacenados o desencadenadores ejecutados por el código del bloque TRY.

Alternativamente, los procedimientos almacenados o desencadenadores pueden contener sus propias construcciones TRY...CATCH para controlar los errores generados por su código.

Por ejemplo, cuando un bloque TRY ejecuta un procedimiento almacenado y se produce un error en éste, el error se puede controlar de las formas siguientes:

- Si el procedimiento almacenado no contiene su propia construcción TRY...CATCH, el error devuelve el control al bloque CATCH asociado al bloque TRY que contiene la instrucción EXECUTE.
- Si el procedimiento almacenado contiene una construcción TRY...CATCH, el error transfiere el control al bloque CATCH del procedimiento almacenado. Cuando finaliza el código del bloque CATCH, el control se devuelve a la instrucción inmediatamente posterior a la instrucción EXECUTE que llamó al procedimiento almacenado.

No se pueden utilizar instrucciones GOTO para entrar en un bloque TRY o CATCH. Estas instrucciones se pueden utilizar para saltar a una etiqueta dentro del mismo bloque TRY o CATCH, o bien para salir de un bloque TRY o CATCH.

La construcción TRY...CATCH no se puede utilizar en una función definida por el usuario.

Recuperar información sobre errores

En el ámbito de un bloque CATCH, se pueden utilizar las siguientes funciones del sistema para obtener información acerca del error que provocó la ejecución del bloque CATCH:

- **ERROR_NUMBER()** devuelve el número del error.
- **ERROR_SEVERITY()** devuelve la gravedad.
- **ERROR_STATE()** devuelve el número de estado del error.
- **ERROR_PROCEDURE()** devuelve el nombre del procedimiento almacenado o desencadenador donde se produjo el error.
- **ERROR_LINE()** devuelve el número de línea de la rutina que provocó el error.
- **ERROR_MESSAGE()** devuelve el texto completo del mensaje de error. El texto incluye los valores proporcionados para los parámetros sustituibles, como las longitudes, nombres de objeto o tiempos.

Estas funciones devuelven NULL si se las llama desde fuera del ámbito del bloque CATCH. Con ellas se puede recuperar información sobre los errores desde cualquier lugar dentro del ámbito del bloque CATCH.

Por ejemplo, en el siguiente script se muestra un procedimiento almacenado que contiene funciones de control de errores. Se llama al procedimiento almacenado en el bloque CATCH de una construcción TRY...CATCH y se devuelve información sobre el error.

Recuperar información sobre errores

Sintaxis

```
BEGIN TRY
    SELECT 1/0;
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS NumerodeError;
END CATCH;

-----
8134
```

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 de una construcción TRY...CATCH.

Las nuevas aplicaciones deben usar THROW en su lugar.

RAISERROR

Sintaxis

```
BEGIN TRY
            BEGIN TRANSACTION
                        UPDATE [dbo].[Errores] SET [Valor] = 1/0
WHERE ID = 1;
            COMMIT TRANSACTION
END TRY
BEGIN CATCH
            ROLLBACK TRANSACTION;
            RAISERROR (
                         'Algo malo ha ocurrido ', -- Mensaje
personalizado.
                          16, -- Severidad.
                          1 -- Estado.
                       );
END CATCH
---------------------------------------
Msg 50000, Level 16, State 1, Line 17
Algo malo ha ocurrido
```

THROW

Produce una excepción y transfiere la ejecución a un bloque CATCH de una construcción TRY...CATCH.

Notas

La instrucción anterior a la instrucción THROW debe ir seguida del terminador de instrucción punto y coma (;).

Si una construcción TRY...CATCH no está disponible, el lote de instrucciones se finaliza. Se establecen el número de línea y el procedimiento donde se produce la excepción. La gravedad se establece en 16.

Si la instrucción THROW se especifica sin parámetros, debe aparecer dentro de un bloque CATCH. Esto hace que se produzca la excepción detectada. Cualquier error que se produzca en una instrucción THROW hace que el lote de instrucciones se finalice.

% es un carácter reservado en el texto del mensaje de una instrucción THROW y debe ser de escape. Duplique el carácter % para devolver % como parte del texto del mensaje, por ejemplo "el aumento supera un 15 %% el valor original".

THROW

En el siguiente ejemplo se muestra cómo utilizar la instrucción THROW para producir una excepción.

Sintaxis

```
BEGIN TRY
            BEGIN TRANSACTION
                        UPDATE [dbo].[Errores] SET [Valor] = 1/0
WHERE ID = 1;
            COMMIT TRANSACTION
END TRY
BEGIN CATCH
            ROLLBACK TRANSACTION;
            THROW;
END CATCH
---------------------------------------
(0 rows affected)
Msg 8134, Level 16, State 1, Line 6
Error de división entre cero.
```

Diferencias entre RAISERROR y THROW

A continuación se enumeran algunas diferencias entre las instrucciones THROW y RAISERROR.

RAISERROR	THROW
Si se pasa **msg_id** a RAISERROR, el identificador se debe definir en sys.messages.	El parámetro **error_number** no tiene que definirse en sys.messages.
El parámetro **msg_str** puede contener estilos de formato de printf.	El parámetro **message** no acepta el formato de estilo de printf.
El parámetro **severity** especifica la gravedad de la excepción.	No hay ningún parámetro **severity.** La gravedad de la excepción siempre está establecida en 16.

¡Muchas gracias!

¡Sigamos trabajando!