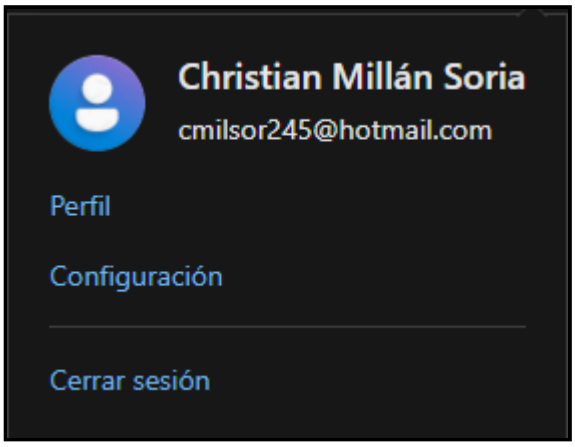
TERCER CURSO

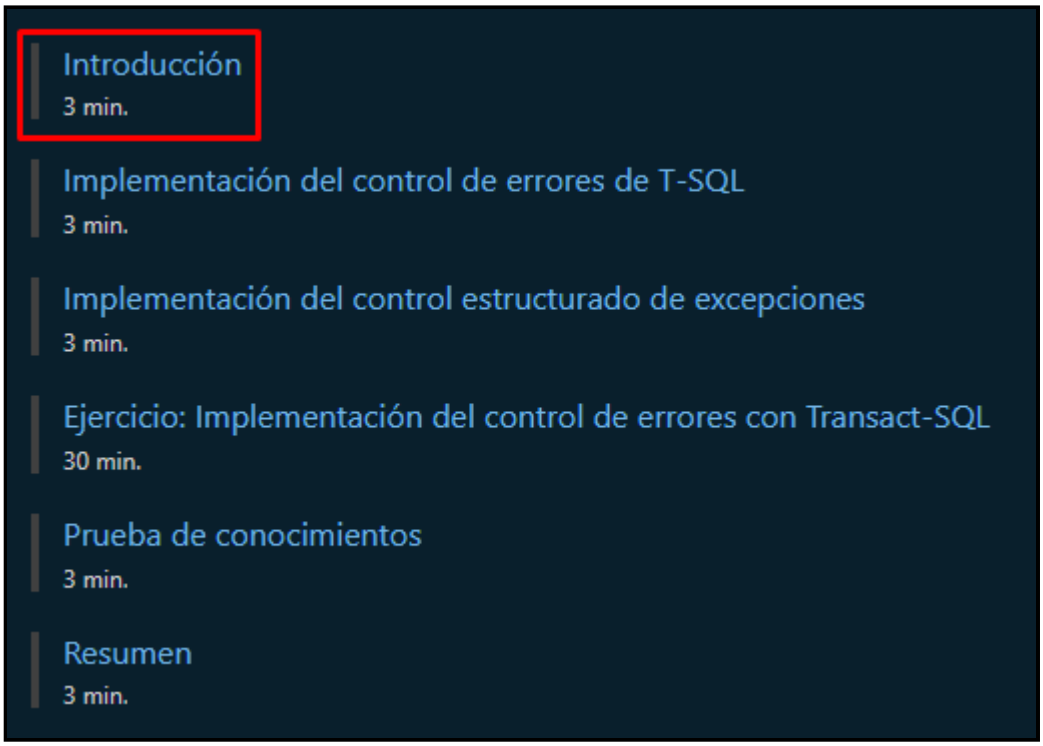
CHRISTIAN MILLÁN SORIA

Comenzamos entrando en este enlace.

Iniciamos sesión con una cuenta de Hotmail.



Una vez tenemos una cuenta, bajamos y encontramos una serie de apartados. Entramos en el primero.



INTRODUCCIÓN

```
Transact-SQL (T-SQL) es un poderoso lenguaje declarativo que permite explorar y
manipular la base de datos. A medida que aumenta la complejidad de los programas,
también lo hace el riesgo de errores, por ejemplo, por falta de coincidencia de
los tipos de datos o variables que no contienen los valores esperados. Si no se
```

administran correctamente, estos errores pueden hacer que los programas dejen de ejecutarse o que se produzcan comportamientos inesperados.

Aquí se explica el control básico de errores de T-SQL, incluida la forma en que se pueden generar intencionadamente, y cómo configurar la activación de alertas cuando se produzcan errores.

Después de completar esta unidad, podrá:

· Generar errores mediante la instrucción RAISERROR.
· Generar errores mediante la instrucción THROW.
· Use la variable del sistema @@ERROR.
· Crear errores personalizados.
· Crear alertas que se activen cuando se produzcan errores.

IMPLEMENTACIÓN DEL CONTROL DE ERRORES DE T-SQL

Un error indica un problema o una incidencia importante que surge durante una operación de base de datos. Los errores puede generarlos el Motor de base de datos de Microsoft SQL Server en respuesta a un evento o un error a nivel del sistema; o usted puede generar errores de aplicación en el código de Transact-SQL.

ELEMENTOS DE LOS ERRORES DEL MOTOR DE BASE DE DATOS

Sea cual sea la causa, cada error se compone de los siguientes elementos:

· Número de error: número único que identifica el error específico.
· Mensaje de error: texto que describe el error.
· Gravedad: indicación numérica de la gravedad de 1 a 25.
· Estado: código de estado interno de la condición del motor de base de datos.
· Procedimiento: nombre del procedimiento almacenado o desencadenador en que se produjo el error.
· Número de línea: la instrucción del lote o procedimiento que generó el error.

ERRORES DEL SISTEMA

Los errores del sistema están predefinidos y se pueden ver en la vista del sistema sys.messages. Cuando se produce un error del sistema, SQL Server puede tomar medidas correctivas automáticas, dependiendo de la gravedad del error. Por ejemplo, cuando se produce un error de gravedad alta, SQL Server puede desconectar una base de datos o incluso detener el servicio del motor de base de datos.

ERRORES PERSONALIZADOS

Puede generar errores en el código de Transact-SQL para responder a condiciones específicas de la aplicación o para personalizar la información enviada a las aplicaciones cliente en respuesta a errores del sistema. Estos errores de aplicación se pueden definir en línea donde se generan, o puede predefinirlos en la tabla sys.messages junto con los errores proporcionados por el sistema. Los números de error utilizados para los errores personalizados deben ser a partir de 50001, inclusive.

Para agregar un mensaje de error personalizado a sys.messages, use sp_addmessage. El usuario del mensaje debe ser miembro de los roles fijos de servidor sysadmin o serveradmin.

Esta es la sintaxis de sp_addmessage:

```
sp_addmessage [ @msgnum= ] msg_id , [ @severity= ] severity , [ @msgtext= ] 'msg'
     [ , [ @lang= ] 'language' ]
     [ , [ @with_log= ] { 'TRUE' | 'FALSE' } ]
     [ , [ @replace= ] 'replace' ]
```

Este es un ejemplo de un mensaje de error personalizado que usa esta sintaxis:

```
sp_addmessage 50001, 10, N’Unexpected value entered’;
```

Además, puede definir mensajes de error personalizados; los miembros del rol de servidor sysadmin también pueden usar un parámetro adicional, @with_log. Cuando se establece en TRUE, el error también se registrará en el registro de aplicaciones de Windows. Los mensajes escritos en el registro de aplicaciones de Windows también se escriben en el registro de errores de SQL Server. Sea precavido con el uso de la opción @with_log, ya que a los administradores de red y del sistema no les suelen gustar las aplicaciones "charlatanas" en los registros del sistema. Sin embargo, si el error debe capturarse con una alerta, primero debe escribirse en el registro de aplicaciones de Windows.

Los mensajes se pueden reemplazar sin eliminarlos primero mediante la opción @replace= "replace".

Los mensajes son personalizables y se pueden agregar otros diferentes para el mismo número de error en varios idiomas, según el valor de language_id.

GENERACIÓN DE ERRORES MEDIANTE RAISERROR

Tanto PRINT como RAISERROR se pueden usar para devolver información o mensajes de advertencia a las aplicaciones. RAISERROR permite que las aplicaciones generen un error que luego puede capturar el proceso de llamada.

RAISERROR

La posibilidad de generar errores en T-SQL facilita el control de errores en la aplicación, ya que se envía como cualquier otro error del sistema. RAISERROR se usa para:

· Ayudar a solucionar problemas con el código de T-SQL.
· Comprobar los valores de los datos.
· Devolver mensajes que contengan texto variable.

Este es un ejemplo de un mensaje de error personalizado que usa RAISERROR.

```
RAISERROR (N'%s %d', -- Message text,
    10, -- Severity,
    1, -- State,
    N'Custom error message number',
    2)
```

Cuando se desencadena, devuelve:

```
Custom error message number 2
```

En el ejemplo anterior, %d es un marcador de posición de un número y %s es un marcador de posición de una cadena. Además, debe advertir que no se ha mencionado un número de mensaje. Cuando se producen errores con las cadenas de mensaje que usan esta sintaxis, el número de error es siempre 50000.

GENERACIÓN DE ERRORES MEDIANTE THROW

La instrucción THROW ofrece un método más sencillo de generar errores en el código. Los errores deben tener un número de error mínimo de 50000.

THROW

THROW se diferencia de RAISERROR de varias maneras:

· Los errores generados por THROW siempre son de gravedad 16.
· Los mensajes devueltos por THROW no están relacionados con ninguna entrada de sys.sysmessages.
· Los errores generados por THROW solo provocan la anulación de transacciones cuando se usan junto con SET XACT_ABORT ON y la sesión se termina.

```
THROW 50001, 'An Error Occured',0
```

CAPTURA DE CÓDIGOS DE ERROR CON @@ERROR

El código de control de errores más tradicional de las aplicaciones de SQL Server se ha creado mediante @@ERROR. El control estructurado de excepciones se introdujo en SQL Server 2005 y proporciona una muy buena alternativa al uso de @@ERROR. Este tema se tratará en la lección siguiente. Una gran cantidad del código de control de errores de SQL Server existente se basa en @@ERROR, por lo que es importante comprender cómo usarlo.

@@ERROR

@@ERROR es una variable del sistema que contiene el número del último error que se ha producido. Un problema importante con @@ERROR es que el valor que contiene se restablece rápidamente a medida que se ejecuta cada instrucción adicional.

Por ejemplo, considere el siguiente código:

```
RAISERROR(N'Message', 16, 1);
IF @@ERROR <> 0
PRINT 'Error=' + CAST(@@ERROR AS VARCHAR(8));
GO
```

Cabría esperar que, cuando se ejecute el código, el número de error se devuelva en una cadena impresa. Sin embargo, cuando se ejecuta el código, se devuelve:

```
Msg 50000, Level 16, State 1, Line 1
Message
Error=0
```

El error se ha generado, pero le mensaje impreso ha sido "Error=0". En la primera línea de la salida, puede ver que, como se esperaba, el error era realmente 50000, con un mensaje pasado a RAISERROR. Esto se debe a que la instrucción IF que sigue a la instrucción RAISERROR se ha ejecutado correctamente y ha provocado el restablecimiento del valor @@ERROR. Por este motivo, al trabajar con @@ERROR, es importante capturar el número de error en una variable tan pronto como se genere y, luego, continuar el procesamiento con la variable.

El código siguiente muestra este escenario:

```
DECLARE @ErrorValue int;
RAISERROR(N'Message', 16, 1);
SET @ErrorValue = @@ERROR;
IF @ErrorValue <> 0
PRINT 'Error=' + CAST(@ErrorValue AS VARCHAR(8));
```

Cuando se ejecuta este código, se devuelve la salida siguiente:

```
Msg 50000, Level 16, State 1, Line 2
Message
Error=50000
```

El número de error se notifica ahora correctamente.