BASE DE DATOS II

ACTIVIDAD 13

Integrantes:

Cristopher García (6298)

Alex Andino (6313)

RAISERROR VS THROW EN SQL SERVER

	RAISERROR	THROW
USOS	RAISERROR genera un mensaje de error e inicia el procesamiento de errores para la sesión. RAISERROR puede hacer referencia a un mensaje definido por el usuario almacenado en la vista de catálogo SYS.messages o crear un mensaje de forma dinámica. El mensaje se devuelve como un mensaje de error del servidor a la aplicación que realiza la llamada o a un bloque CATCH asociado de una construcción TRY/CATCH.	THROW genera una excepción y transfiere la ejecución a un bloque CATCH de una construcción TRY/CATCH en SQL Server 2012 y versiones superiores.
VERSION	RAISERROR fue introducido en SQL Server 7.0. Soporta hasta SQL Server 2012. RAISERROR no se puede utilizar en los Procedimientos almacenados compilados de forma nativa de SQL Server 2014.	THROW fue introducido en SQL Server versión 2012. THROW se puede utilizar en SQL Server 2014 y versiones anteriores. Microsoft sugiere usar THROW en lugar de usar RAISERROR.
SINTÁXIS	RAISERROR ({ msg_id | msg_str | @local_variable } { ,severity ,state } [,argument [,...n]]) [WITH option [,...n]] **NOTA:** RAISERROR con NOWAITstatement también se puede utilizar para vaciar todos los mensajes de instrucciones PRINT / SELECT almacenados en búfer dentro de un lote.	THROW [{ error_number | @local_variable }, { message | @local_variable }, { state | @local_variable }] [;] **NOTA:** La instrucción SQL antes de la instrucción THROW debe ir seguida por el terminador de la coma (;), pero la instrucción throw puede o no terminar con (;).

DETALLES ORIGINALES DE LA EXCEPCIÓN	No. RAISERROR siempre genera una nueva excepción y da como resultado la pérdida de los detalles de la excepción original especificados con los parámetros apropiados.
	
	EJEMPLO:
	`DECLARE @result INT`
	`BEGIN TRY`
	`SET @result = 1/0 -- divide-by-zero error`
	`END TRY`
	`BEGIN CATCH`
	`DECLARE`
	`@ErrorNumber INT,`
		SÍ. La excepción original THROW atrapada en el bloque TRY; simplemente podemos especificar la sentencia THROW sin ningún parámetro en el bloque CATCH.
		
		EJEMPLO:
		`BEGIN TRY`
		`DECLARE @RESULT INT = 55/0`
		`END TRY`
		`BEGIN CATCH`
		`PRINT 'BEFORE THROW';`
		`THROW;`
		`PRINT 'AFTER THROW'`
		`END CATCH`

	@ErrorSeverity INT, @ErrorState INT, @ErrorProcedure NVARCHAR(2048), @ErrorMessage NVARCHAR(2048) SELECT --get the error details @ErrorNumber = ERROR_NUMBER(), @ErrorSeverity =ERROR_SEVERITY(), @ErrorState = ERROR_STATE(), @ErrorProcedure=ERROR_PROCEDURE(), @ErrorMessage = ERROR_MESSAGE() PRINT 'BEFORE RAISERROR'; RAISERROR (@ErrorMessage, @Error Severity, @ErrorState) --WITH NOWAIT PRINT 'AFTER RAISERROR'; END CATCH PRINT 'AFTER CATCH'; **OUTPUT:** BEFORE RAISERROR Msg 50000, Level 16, State 1, Line 22 Divide by zero error encountered. AFTER RAISERROR AFTER CATCH	PRINT 'AFTER CATCH' **OUTPUT:** BEFORE THROW Msg 8134, Level 16, State 1, Line 2 Divide by zero error encountered.
TERMINACIÓN PARA EJECUTAR	En la salida de ejemplo anterior puede ver la instrucción PRINT ejecutada después de la instrucción RAISERROR.	En el ejemplo de salida anterior puede ver que la instrucción PRINT no se ejecuta después de la instrucción THROW.
AJUSTE DEL NIVEL DE SEGURIDAD	El parámetro de seguridad debe establecer en RAISERROR la seguridad de la excepción.	No hay ningún parámetro de seguridad establecido en THROW. El nivel en la excepción siempre se establece en 16. (a menos que se vuelva a lanzar en un bloque CATCH)
MENSAJE DE ERROR	Con RAISERROR podemos elevar los mensajes de error del sistema que se almacenaron en la tabla SYS.Messages. SELECT * FROM sys.messages **WHERE** message_id <=50000 - System Messages message_id >50000 - User Defined Messages **EJEMPLO:** RAISERROR (17051,16,1) **OUTPUT:** Msg 17051, Level 16, State 1, Line 1 SQL Server evaluation period has expired.	Con THROW no podemos elevar la excepción del sistema. **EJEMPLO:** THROW 17051, 'SQL Server evaluation period has expired.',1; **OUTPUT:** Msg 35100, Level 16, State 10, Line 7 Error number 17051 in the THROW statement is outside the valid range. Specify an error number in the valid range of 50000 to 2147483647.
SP_ADD MESSAGE	sp_addmessage se usa para agregar mensajes definidos por el usuario a la tabla SYS.Messages. **EJEMPLO:** -- Create a user-defined message in U.S. English EXEC sp_addmessage @msgnum = 60000, @severity = 16, @msgtext = N'The item named %s already exists in %s.', @lang = 'us_english';	

<table>
<tr><td></td><td>

```
-- Create a user-defined message in French
EXEC sp_addmessage
    @msgnum = 60000,
    @severity = 16,
    @msgtext = N'L''élément nommé %1! existe déjà dans %2!',
    @lang = 'French';
GO

SELECT @@LANGUAGE  -- Get to know the system language


SET LANGUAGE 'us_english' -- Change language to us_english
SET LANGUAGE 'French'   -- Change language to French
SET LANGUAGE 'us_english'-- Change language back to us_english
```

</td></tr>
<tr><td>SP_DROP MESSAGE</td><td>

sp_dropmessage se usa para eliminar mensajes definidos por el usuario de la tabla SYS.Messages.

EJEMPLO:

```
-- This statement will fail as long as the localized version
-- of the message exists.
EXEC sp_dropmessage 60000;
```

OUTPUT:

```
Msg 15280, Level 16, State 1, Procedure sp_dropmessage, Line 64
All localized versions of this message must be dropped before the us_english version can
be dropped.
```

```
-- This statement will drop the message.
EXEC sp_dropmessage
    @msgnum = 60000,
    @lang = 'all';
```

OUTPUT:

```
Command(s) completed successfully.
```

```
-- This statement will remove only the localized version of the
-- message.
EXEC sp_dropmessage
    @msgnum = 60000,
    @lang = 'French';
```

OUTPUT:

```
Command(s) completed successfully.
```

</td></tr>
<tr><td>AUMENTAR EL ERROR DEL MENSAJE DEFINIDO POR EL USUARIO</td><td>

Con RAISERROR podemos elevar los mensajes de error definidos por el usuario que se almacenan en la tabla SYS.Messages.

```
SELECT * FROM sys.messages

WHERE
message_id <=50000 - System Messages
message_id >50000 - User Defined
Messages
```

EJEMPLO:

```
RAISERROR (60000,16,1, 'salary', 'emp
table')
```

OUTPUT:

```
Msg 60000, Level 16, State 1, Line 4
The item named salary already exists in emp
table.
```

Con THROW podemos elevar los mensajes de error definidos por el usuario que se almacenaron en la tabla SYS.Messages.

EJEMPLO:

```
DECLARE @msg NVARCHAR(2048) =FORMATMESSAGE(60
000, N'salary', N'emp table');
THROW 60000, @msg, 1;
```

OUTPUT:

```
Msg 60000, Level 16, State 1, Line 15
The item named salary already exists in emp
table.
```

</td></tr>
</table>

RESUMEN	Si se pasa un msg_id a RAISERROR, la ID debe definirse en sys.messages. El parámetro msg_str puede contener estilos de formato de printf. El parámetro severity especifica la severidad de la excepción.	El parámetro error_number no tiene que estar definido en sys.messages. El parámetro de mensaje no acepta el formato de estilo printf. No hay ningún parámetro de severidad. El nivel de seguridad siempre se establece en 16.