Funciones del Sistemas

@@ERROR (Transact-SQL)

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

[Sintaxis](javascript:void(0))

@@ERROR

[Tipos de valor devuelto](javascript:void(0))

integer

[Comentarios](javascript:void(0))

Devuelve 0 si la instrucción Transact-SQL anterior no encontró errores.

Devuelve un número de error si la instrucción anterior encontró un error. Si el error era uno de los errores de la vista de catálogo sys.messages, entonces @@ERROR contendrá el valor de la columna sys.messages.message\_id column para dicho error. Puede ver el texto asociado con el número de error @@ERROR en sys.messages.

Como @@ERROR se borra y restablece con cada instrucción ejecutada, debe comprobarlo inmediatamente después de la instrucción que se está comprobando o guardarlo en una variable local para examinarlo posteriormente.

Use la construcción TRY...CATCH para controlar errores. La construcción TRY...CATCH también admite funciones de sistema adicionales (ERROR\_LINE, ERROR\_MESSAGE, ERROR\_PROCEDURE, ERROR\_SEVERITY y ERROR\_STATE) que devuelven más información sobre errores que @@ERROR. TRY...CATCH también admite una función ERROR\_NUMBER que no se limita a devolver el número de error en la instrucción inmediatamente después de la instrucción que generó el error. Para obtener más información, vea [TRY...CATCH (Transact-SQL)](https://msdn.microsoft.com/es-es/library/ms175976.aspx).

[Ejemplos](javascript:void(0))

A.Utilizar @@ERROR para detectar un error específico

En el ejemplo siguiente se utiliza @@ERROR para comprobar si se infringe una restricción CHECK (error nº 547) en una instrucción UPDATE.

USE AdventureWorks2012;

GO

UPDATE HumanResources.EmployeePayHistory

SET PayFrequency = 4

WHERE BusinessEntityID = 1;

IF @@ERROR = 547

PRINT N'A check constraint violation occurred.';

GO

B.Utilizar @@ERROR para salir condicionalmente de un procedimiento

En los ejemplos siguientes se utilizan las instrucciones IF...ELSE para probar @@ERROR después de una instrucción INSERT en un procedimiento almacenado. El valor de la variable @@ERROR determina el código devuelto enviado al programa que llamó, lo que indica si el procedimiento se realizó correcta o incorrectamente.

USE AdventureWorks2012;

GO

-- Drop the procedure if it already exists.

IF OBJECT\_ID(N'HumanResources.usp\_DeleteCandidate', N'P') IS NOT NULL

DROP PROCEDURE HumanResources.usp\_DeleteCandidate;

GO

-- Create the procedure.

CREATE PROCEDURE HumanResources.usp\_DeleteCandidate

(

@CandidateID INT

)

AS

-- Execute the DELETE statement.

DELETE FROM HumanResources.JobCandidate

    WHERE JobCandidateID = @CandidateID;

-- Test the error value.

IF @@ERROR <> 0

BEGIN

-- Return 99 to the calling program to indicate failure.

PRINT N'An error occurred deleting the candidate information.';

RETURN 99;

END

ELSE

BEGIN

-- Return 0 to the calling program to indicate success.

PRINT N'The job candidate has been deleted.';

RETURN 0;

END;

GO

C.Utilizar @@ERROR con @@ROWCOUNT

En el ejemplo siguiente se utiliza @@ERROR con @@ROWCOUNT para validar la operación de una instrucción UPDATE. Se comprueba el valor de @@ERROR para ver si hay un error y se utiliza @@ROWCOUNT para asegurarse de que la actualización se aplica correctamente a una fila de la tabla.

USE AdventureWorks2012;

GO

IF OBJECT\_ID(N'Purchasing.usp\_ChangePurchaseOrderHeader',N'P')IS NOT NULL

DROP PROCEDURE Purchasing.usp\_ChangePurchaseOrderHeader;

GO

CREATE PROCEDURE Purchasing.usp\_ChangePurchaseOrderHeader

(

@PurchaseOrderID INT

,@BusinessEntityID INT

)

AS

-- Declare variables used in error checking.

DECLARE @ErrorVar INT;

DECLARE @RowCountVar INT;

-- Execute the UPDATE statement.

UPDATE PurchaseOrderHeader

SET BusinessEntityID = @BusinessEntityID

WHERE PurchaseOrderID = @PurchaseOrderID;

-- Save the @@ERROR and @@ROWCOUNT values in local

-- variables before they are cleared.

SELECT @ErrorVar = @@ERROR

,@RowCountVar = @@ROWCOUNT;

-- Check for errors. If an invalid @BusinessEntityID was specified,

-- the UPDATE statement returns a foreign key violation error #547.

IF @ErrorVar <> 0

BEGIN

IF @ErrorVar = 547

BEGIN

PRINT N'ERROR: Invalid ID specified for new employee.';

 RETURN 1;

END

ELSE

BEGIN

PRINT N'ERROR: error '

+ RTRIM(CAST(@ErrorVar AS NVARCHAR(10)))

+ N' occurred.';

RETURN 2;

END

END

-- Check the row count. @RowCountVar is set to 0

-- if an invalid @PurchaseOrderID was specified.

IF @RowCountVar = 0

BEGIN

PRINT 'Warning: The BusinessEntityID specified is not valid';

RETURN 1;

END

ELSE

BEGIN

PRINT 'Purchase order updated with the new employee';

RETURN 0;

END;

GO

@@IDENTITY (Transact-SQL)

Se trata de una función del sistema que devuelve el último valor de identidad insertado.

Icono de vínculo a temas

[Sintaxis](javascript:void(0))

@@IDENTITY

[Tipos de valor devuelto](javascript:void(0))

**numeric(38,0)**

[Comentarios](javascript:void(0))

Cuando se completa una instrucción INSERT, SELECT INTO o de copia masiva, @@IDENTITY contiene el último valor de identidad generado por la instrucción. Si la instrucción no ha afectado a ninguna tabla con columnas de identidad, @@IDENTITY devuelve NULL. Si se insertan varias filas, lo que genera varios valores de identidad, @@IDENTITY devuelve el último valor de identidad generado. Si la instrucción activa uno o más desencadenadores que realizan inserciones que, a su vez, generan valores de identidad, al llamar a @@IDENTITY inmediatamente después de la instrucción, se obtiene el último valor de identidad generado por los desencadenadores. Si un desencadenador se activa tras una acción de inserción en una tabla que tiene una columna de identidad y se inserta en otra tabla que no tiene una columna de identidad, @@IDENTITY devuelve el valor de identidad de la primera inserción. Si se produce un error en la instrucción INSERT o SELECT INTO o en la copia masiva o se revierte la transacción, el valor de @@IDENTITY no revierte a un valor anterior.

Las instrucciones y las transacciones erróneas pueden modificar la identidad actual de una tabla y crear espacios en los valores de las columnas de identidad. El valor de identidad jamás se revierte, aun cuando no se haya confirmado la transacción que intentó insertar el valor en la tabla. Por ejemplo, si se produce un error en una instrucción INSERT debido a una infracción de IGNORE\_DUP\_KEY, el valor de identidad actual de la tabla se sigue incrementando.

Las funciones @@IDENTITY, SCOPE\_IDENTITY e IDENT\_CURRENT se parecen, puesto que devuelven el último valor insertado en la columna IDENTITY de una tabla.

@@IDENTITY y SCOPE\_IDENTITY devuelven el último valor de identidad generado en una tabla en la sesión actual. No obstante, SCOPE\_IDENTITY solo devuelve el valor en el ámbito actual; @@IDENTITY no se limita a un ámbito específico.

IDENT\_CURRENT no está limitado por el ámbito y la sesión; se limita a una tabla especificada. IDENT\_CURRENT devuelve el valor de identidad generado para una tabla específica en cualquier sesión y cualquier ámbito. Para obtener más información, vea [IDENT\_CURRENT (Transact-SQL)](https://msdn.microsoft.com/es-es/library/ms175098.aspx).

El ámbito de la función @@IDENTITY es la sesión actual en el servidor local en el que se ejecuta. Esta función no se puede aplicar a servidores remotos o vinculados. Para obtener un valor de identidad de un servidor diferente, ejecute un procedimiento almacenado en ese servidor remoto o vinculado y haga que dicho procedimiento (que se está ejecutando en el contexto del servidor remoto o vinculado) recopile el valor de identidad y lo devuelva a la conexión que llama del servidor local.

La replicación puede afectar al valor @@IDENTITY, ya que se usa en los desencadenadores de replicación y en los procedimientos almacenados. @@IDENTITY no es un indicador confiable de la identidad más reciente creada por el usuario si la columna forma parte de un artículo de replicación. Puede usar la sintaxis de función SCOPE\_IDENTITY() en lugar de @@IDENTITY. Para obtener más información, vea [SCOPE\_IDENTITY (Transact-SQL)](https://msdn.microsoft.com/es-es/library/ms190315.aspx).

|  |
| --- |
| **Nota Nota** |
| El procedimiento almacenado o la instrucción Transact-SQL que realizan la llamada se deben volver a escribir para usar la función SCOPE\_IDENTITY(), que devolverá la última identidad usada en el ámbito de esa instrucción de usuario en lugar de la identidad en el ámbito del desencadenador anidado usado por la replicación. |

[Ejemplos](javascript:void(0))

En el siguiente ejemplo se inserta una fila en una tabla con una columna de identidad (LocationID) y se utiliza @@IDENTITY para mostrar el valor de identidad empleado en la nueva fila.

USE AdventureWorks2012;

GO

--Display the value of LocationID in the last row in the table.

SELECT MAX(LocationID) FROM Production.Location;

GO

INSERT INTO Production.Location (Name, CostRate, Availability, ModifiedDate)

VALUES ('Damaged Goods', 5, 2.5, GETDATE());

GO

SELECT @@IDENTITY AS 'Identity';

GO

--Display the value of LocationID of the newly inserted row.

SELECT MAX(LocationID) FROM Production.Location;

GO

# @@ROWCOUNT (Transact-SQL)

Devuelve el número de filas afectadas por la última instrucción. Si el número de filas es mayor de dos mil millones, use [ROWCOUNT\_BIG](https://msdn.microsoft.com/es-es/library/ms181406.aspx).

## [Sintaxis](javascript:void(0))

@@ROWCOUNT

## [Tipos de valor devuelto](javascript:void(0))

**int**

## [Comentarios](javascript:void(0))

Las instrucciones Transact-SQL pueden establecer el valor de @@ROWCOUNT de las siguientes maneras:

* Establecer @@ROWCOUNT en el número de filas afectadas o leídas. Las filas pueden o no enviarse al cliente.
* Conservar @@ROWCOUNT de la anterior ejecución de una instrucción.
* Restablecer @@ROWCOUNT en 0 y no devolver el valor al cliente.

Las instrucciones que realizan una asignación simple siempre establecen el valor @@ROWCOUNT en 1. No se envían filas al cliente. Ejemplos de estas instrucciones son: SET @*local\_variable*, RETURN, READTEXT y Select sin instrucciones de consulta como SELECT GETDATE() o SELECT **'***Generic Text***'**.

Las instrucciones que realizan una asignación en una consulta o utilizan RETURN en una consulta establecen el valor @@ROWCOUNT en el número de filas afectadas o leídas por la consulta, por ejemplo: SELECT @*local\_variable* = c1 FROM t1.

Las instrucciones de lenguaje de manipulación de datos (DML) establecen el valor de @@ROWCOUNT en el número de filas afectadas por la consulta y devuelven ese valor al cliente. Las instrucciones DML pueden no enviar ninguna fila al cliente.

DECLARE CURSOR y FETCH establecen el valor de @@ROWCOUNT en 1.

Las instrucciones EXECUTE conservan el valor anterior de @@ROWCOUNT.

Las instrucciones como USE, SET <opción>, DEALLOCATE CURSOR, CLOSE CURSOR, BEGIN TRANSACTION o COMMIT TRANSACTION restablecen el valor de ROWCOUNT en 0.

Los procedimientos almacenados compilados de forma nativa mantienen el @@ROWCOUNT anterior. Las instrucciones de Transact-SQL dentro de procedimientos almacenados compilados de forma nativa no establecen @@ROWCOUNT

## [Ejemplos](javascript:void(0)" \o "Haga clic para contraer. Haga doble clic para contraer todo.)

En el siguiente ejemplo se ejecuta una instrucción UPDATE y se utiliza @@ROWCOUNT para detectar si ha cambiado alguna fila.

USE AdventureWorks2012;

GO

UPDATE HumanResources.Employee

SET JobTitle = N'Executive'

WHERE NationalIDNumber = 123456789

IF @@ROWCOUNT = 0

PRINT 'Warning: No rows were updated';

GO