**SCOPE\_IDENTITY (Transact-SQL)**

Returns the last identity value inserted into an identity column in the same scope. A scope is a module: a stored procedure, trigger, function, or batch. Therefore, two statements are in the same scope if they are in the same stored procedure, function, or batch.

Topic link icon[Transact-SQL Syntax Conventions](http://msdn.microsoft.com/en-us/library/ms177563.aspx)

http://i.msdn.microsoft.com/Global/Images/clear.gif Syntax

SCOPE\_IDENTITY()

http://i.msdn.microsoft.com/Global/Images/clear.gif Return Types

**numeric(38,0)**

http://i.msdn.microsoft.com/Global/Images/clear.gif Remarks

SCOPE\_IDENTITY, IDENT\_CURRENT, and @@IDENTITY are similar functions because they return values that are inserted into identity columns.

IDENT\_CURRENT is not limited by scope and session; it is limited to a specified table. IDENT\_CURRENT returns the value generated for a specific table in any session and any scope. For more information, see [IDENT\_CURRENT (Transact-SQL)](http://msdn.microsoft.com/en-us/library/ms175098.aspx).

SCOPE\_IDENTITY and @@IDENTITY return the last identity values that are generated in any table in the current session. However, SCOPE\_IDENTITY returns values inserted only within the current scope; @@IDENTITY is not limited to a specific scope.

For example, there are two tables, T1 and T2, and an INSERT trigger is defined on T1. When a row is inserted to T1, the trigger fires and inserts a row in T2. This scenario illustrates two scopes: the insert on T1, and the insert on T2 by the trigger.

Assuming that both T1 and T2 have identity columns, @@IDENTITY and SCOPE\_IDENTITY will return different values at the end of an INSERT statement on T1. @@IDENTITY will return the last identity column value inserted across any scope in the current session. This is the value inserted in T2. SCOPE\_IDENTITY() will return the IDENTITY value inserted in T1. This was the last insert that occurred in the same scope. The SCOPE\_IDENTITY() function will return the null value if the function is invoked before any INSERT statements into an identity column occur in the scope.

Failed statements and transactions can change the current identity for a table and create gaps in the identity column values. The identity value is never rolled back even though the transaction that tried to insert the value into the table is not committed. For example, if an INSERT statement fails because of an IGNORE\_DUP\_KEY violation, the current identity value for the table is still incremented.

http://i.msdn.microsoft.com/Global/Images/clear.gif Examples

**A. Using @@IDENTITY and SCOPE\_IDENTITY with triggers**

The following example creates two tables, TZ and TY, and an INSERT trigger on TZ. When a row is inserted to table TZ, the trigger (Ztrig) fires and inserts a row in TY.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_ctl00_ctl00_code');" \o "Copy Code)

USE tempdb

GO

CREATE TABLE TZ (

Z\_id int IDENTITY(1,1)PRIMARY KEY,

Z\_name varchar(20) NOT NULL)

INSERT TZ

VALUES ('Lisa')

INSERT TZ

VALUES ('Mike')

INSERT TZ

VALUES ('Carla')

SELECT \* FROM TZ

--Result set: This is how table TZ looks.

Z\_id Z\_name

-------------

1 Lisa

2 Mike

3 Carla

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_ctl00_ctl01_code');" \o "Copy Code)

CREATE TABLE TY (

Y\_id int IDENTITY(100,5)PRIMARY KEY,

Y\_name varchar(20) NULL)

INSERT TY (Y\_name)

VALUES ('boathouse')

INSERT TY (Y\_name)

VALUES ('rocks')

INSERT TY (Y\_name)

VALUES ('elevator')

SELECT \* FROM TY

--Result set: This is how TY looks:

Y\_id Y\_name

---------------

100 boathouse

105 rocks

110 elevator

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_ctl00_ctl02_code');" \o "Copy Code)

/\*Create the trigger that inserts a row in table TY

when a row is inserted in table TZ.\*/

CREATE TRIGGER Ztrig

ON TZ

FOR INSERT AS

BEGIN

INSERT TY VALUES ('')

END

/\*FIRE the trigger and determine what identity values you obtain

with the @@IDENTITY and SCOPE\_IDENTITY functions.\*/

INSERT TZ VALUES ('Rosalie')

SELECT SCOPE\_IDENTITY() AS [SCOPE\_IDENTITY]

GO

SELECT @@IDENTITY AS [@@IDENTITY]

GO

Here is the result set.

SCOPE\_IDENTITY

4

/\*SCOPE\_IDENTITY returned the last identity value in the same scope. This was the insert on table TZ.\*/

@@IDENTITY

115

/\*@@IDENTITY returned the last identity value inserted to TY by the trigger. This fired because of an earlier insert on TZ.\*/

**B. Using @@IDENTITY and SCOPE\_IDENTITY() with replication**

The following examples show how to use @@IDENTITY and SCOPE\_IDENTITY() for inserts in a database that is published for merge replication. Both tables in the examples are in the AdventureWorks2008R2 sample database: Person.ContactType is not published, and Sales.Customer is published. Merge replication adds triggers to tables that are published. Therefore, @@IDENTITY can return the value from the insert into a replication system table instead of the insert into a user table.

The Person.ContactType table has a maximum identity value of 20. If you insert a row into the table, @@IDENTITY and SCOPE\_IDENTITY() return the same value.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_ctl00_ctl03_code');" \o "Copy Code)

USE AdventureWorks2008R2;

GO

INSERT INTO Person.ContactType ([Name]) VALUES ('Assistant to the Manager');

GO

SELECT SCOPE\_IDENTITY() AS [SCOPE\_IDENTITY];

GO

SELECT @@IDENTITY AS [@@IDENTITY];

GO

Here is the result set.

SCOPE\_IDENTITY

21

@@IDENTITY

21

The Sales.Customer table has a maximum identity value of 29483. If you insert a row into the table, @@IDENTITY and SCOPE\_IDENTITY() return different values. SCOPE\_IDENTITY() returns the value from the insert into the user table, whereas @@IDENTITY returns the value from the insert into the replication system table. Use SCOPE\_IDENTITY() for applications that require access to the inserted identity value.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_ctl00_ctl04_code');" \o "Copy Code)

INSERT INTO Sales.Customer ([TerritoryID],[PersonID]) VALUES (8,NULL);

GO

SELECT SCOPE\_IDENTITY() AS [SCOPE\_IDENTITY];

GO

SELECT @@IDENTITY AS [@@IDENTITY];

GO

Here is the result set.

SCOPE\_IDENTITY

29484

@@IDENTITY

89

|  |  |  |
| --- | --- | --- |
| **A couple of items worth noting regarding SCOPE\_IDENTITY() behavior** |  |  |

Please Wait  Please Wait

Batch cannot be considered a valid scope for this function. The GO statement, which signifies the end of a batch, does not impact the value of SCOPE\_IDENTITY() as can be clearly seen in the already provided examples as well at the example included below.  
  
Failure of a statement to insert also does not impact the value. In the exaple below with the IGNORE\_DUP\_KEY set on the unique index on name, the attempt to insert 'Lisa' again still returns the value 2 which was the value of the previous insert for 'Mike'. Even if IGNORE\_DUP\_KEY is removed and the statement results in an actual error, SCOPE\_IDENTITY() still maintains the value of 2 rather than what might be expected to be NULL   
  
CREATE TABLE ScopeTest (  
[ID] int NOT NULL IDENTITY(1,1) PRIMARY KEY,  
[Name] varchar(20) NOT NULL)  
GO  
  
CREATE UNIQUE NONCLUSTERED INDEX [ScopeTest\_name] ON [ScopeTest] ([Name]) WITH (IGNORE\_DUP\_KEY = ON)   
GO  
  
INSERT ScopeTest  
VALUES ('Lisa')  
select SCOPE\_IDENTITY(), @@ROWCOUNT, IDENT\_CURRENT('ScopeTest'), @@IDENTITY  
GO  
  
INSERT ScopeTest  
VALUES ('Mike')  
select SCOPE\_IDENTITY(), @@ROWCOUNT, IDENT\_CURRENT('ScopeTest'), @@IDENTITY  
GO  
  
INSERT ScopeTest  
VALUES ('Lisa')  
select SCOPE\_IDENTITY(), @@ROWCOUNT, IDENT\_CURRENT('ScopeTest'), @@IDENTITY  
GO  
  
DROP TABLE ScopeTest  
GO