

## Re: @@Identity

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.adonet/2004-10/0970.html>

---

**From:** Gabe Garza (*spam\_at\_nospam.com*)

**Date:** 10/14/04

Date: Thu, 14 Oct 2004 20:17:48 GMT

Michael,

This is from the MSDN Library – April 2004

---

-----  
How Does SQL Server Manage Identity Values?

When you add a row to a table with an Identity column you don't include a value for the Identity column because the DBMS server automatically adds an increment (usually 1) to the highest Identity value in the table and uses this value for the new row's Identity value. If you're using SQL Server, the value is also saved in a connection-global variable: @@IDENTITY. (GUID identifiers are supported in SQL Server and other full-featured DBMS systems, but not Access/JET.) That's fine, but what happens when a row is deleted? Is that row's Identity value forever orphaned? Yep, unless you reseed the Identity (DBCC CHECKIDENT), deleted Identity values are lost. Identity values are also orphaned when a transaction is rolled back. This means that when you use Identity columns, you'll need to be prepared for gaps in the series. It also means that eventually, the Integer you're using will overflow, so it's important that you use an Integer large enough for your needs—now, and well into the future. An "integer" in SQL Server can identify about 2 billion rows while a "bigint" can identify 9,223,372,036,854,775,807 rows (that's a lot of rows). However, a "smallint" can identify only about 32 thousand rows. I actually had someone complain that he ran out of Identity values—they had used a "tinyint" that ran out after 255 rows. Sigh. I'm not going to delve into techniques to recover orphaned identity values—it's tough to do and over the years I've found it's not worth the trouble. Just make sure to define an integer wide enough to get you through the next century or corporate take-over when they re-write everything anyway.

---

-----  
SCOPE\_IDENTITY

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. Thus, two statements are in the same scope if they are in the same

stored procedure, function, or batch.

---

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.

---

You have 3 ways of getting identities, @@IDENTITY, SCOPE\_IDENTITY and IDENT\_CURRENT.

Notice that the @@IDENTITY is a connection-global variable. So if 2 or more users connect to the database they'd have there own @@IDENTITY global variable. So you can do an INSERT statement then next a SELECT @@IDENTITY to get your own @@IDENTITY, not someone else's @@IDENTITY.

The problem you're describing is if you use IDENT\_CURRENT. That is for the current table. So if 2 or more people insert records into a database and used IDENT\_CURRENT to retrieve an identity value, then the last person who inserted, that's the value that you'd be getting which isn't going to be 100% your identity.

A good example that they give in the MSDN for SCOPE\_IDENTITY is this:  
TABLE1 – has a identity  
TABLE2 – has a identity and a trigger that'll insert a record into TABLE1.

If I insert into TABLE2, you'd use SCOPE\_IDENTITY to get your identity from TABLE2 and @@IDENTITY to get your identity from TABLE1.

Hope this clears up how identities work.

"Michael L John" <mjohn@emscharts.com> wrote in message news:evRL\$0MoEHA.3684@TK2MSFTNGP10.phx.gbl...  
> *Scott, you obviously are an idiot.*  
> *But, I will attempt to answer your LAST post because it is the only one that makes sense.*  
>  
> *There are now 4 highly skilled and very experienced people that did not understand anything you asked in your OP, so I guess all of them (us) are idiots.*  
>  
> *A SQL server database table can have one integer field designated as an identity field. This value will auto increment when a new row is inserted into a table that has an identity field. The starting values, as well as the number that is incremented, can be defined by the user.*

>  
> *If you are inserting a row with your ADO.Net code that will increment the  
> identity value, and need to reliably retrieve this value, you can,  
> immediately after the INSERT statement, run "SELECT @@Identity". This  
will  
> return an integer that is the LAST IDENTITY VALUE inserted into a table.*  
>  
> *As Bill pointed out, there are issues when using the GLOBAL SQL VARIABLE  
> @@identity.*  
>  
> *Assume you have 2 users that are both inserting a record in the same  
table.  
> User A inserts record 10, user B inserts record 11. Unless your code has  
> perfect timing, "SELECT @@identity" will return 11.  
> In a much worse case, assume there are 10 users inserting records into 10  
> tables with identity columns defined. In that case, who knows what value  
> you may get back from "SELECT @@identity"*  
>  
> *You will be better served by using "SELECT SCOPE\_IDENTITY()" in your code.  
> The value returned by this FUNCTION is the identity specific to the  
CURRENT  
> SCOPE.*  
>  
> *As Mary also said, READ BOOKS ONLINE. This will make things much clearer.  
> Unless you do not really know how to read.*  
>  
> *Good Luck  
> Michael L John*  
>  
>  
> *"Scott M." <NoSpam@NoSpam.com> wrote in message  
> news:ujJJ43FoEHA.3684@TK2MSFTNGP10.phx.gbl...  
>> Roy, I asked about @@Identity because I didn't know about it. If I  
>> misused the term "parameter", it was because I lacked the information I  
>> was looking for in the first place.  
>>  
>> The OP was not, as you say, "non-sensical". The other replies seemed to  
>> have gotten what I was asking. Might I suggest that you not reply at  
all  
>> if you don't understand the post?  
>>  
>>  
>> "Roy Fine" <rlfine@tw.t.obfuscate.net> wrote in message  
>> news:%23xhgyBEoEHA.3224@tk2msftngp13.phx.gbl...  
>>> Scott,  
>>>  
>>> read your original post – I said there is no such thing as an @@Identity  
>>> parameter – that was what you were asking how to set. Your original  
post  
>>> was nonsensical! your your own sake (and professional reputation) –  
one*

> >> *would have to strongly advise you against asking people to go back and*  
> >> *read*  
> >> *it.*  
> >> *roy*  
> >>  
> >>  
> >> *"Roy Fine" <rlfine@twf.obfuscate.net> wrote in message*  
> >> *news:%237Y%23LMDoEHA.592@TK2MSFTNGP11.phx.gbl...*  
> >>>  
> >>> *"Scott M." <NoSpam@NoSpam.com> wrote in message*  
> >>> *news:Oi\$nLKpnEHA.2028@TK2MSFTNGP10.phx.gbl...*  
> >>> > *Wow Mary, thanks for the attitude!*  
> >>> >  
> >>> > *I don't know why you keep telling me that @@Identity it not*  
> >>> > *technically*  
> >> *a*  
> >>> > *field. I know that and it is beside the point of my question. It*  
> >>> > *does*  
> >>> > *return a field and it doesn't take a rocket scientist to figure that*  
> >>> > *out*  
> >>> > *what I meant in my OP.*  
> >>> >  
> >>>  
> >>> >> *How do I set the @@Identity parameter for data that*  
> >>> >> *has been read in from a*  
> >>> >> *database? I have read in data via a command*  
> >>> >> *(SELECT statement) and am*  
> >>> >> *trying to build a good UPDATE statement and*  
> >>> >> *want to use the @@Identity field.*  
> >>>  
> >>> *Scott – I have read the above statement from your OP – and I have read*  
> >>> *it*  
> >>> *several times sincethis thread has started, and I must say "your OP*  
> >>> *makes*  
> >> *no*  
> >>> *sense". There is no such thing as an @@Identity parameter – so I*  
**would**  
> >> *walk*  
> >>> *away from the OP and try again.*  
> >>>  
> >>> *roy fine*  
> >>>  
> >>>  
> >>>  
> >>  
> >>  
> >  
> >  
>  
>  
>