Getting the ID field after an Insert

Posted At : August 8, 2005 11:19 AM | Posted By : Mark Kruger
Related Categories: SQL tips, MS SQL Server, Coldfusion Tips and Techniques

About every 2 months or so someone asks about reliably returning the primary key record from an insert query. The problem they are trying to solve usually has to do with additional inserts into related tables. For example, if you are adding a new user and you want to set group permissions as well - but group permissions require inserting into another table. One way to do it is to do an insert, then do a second query that pulls back the "max(id)" and uses another qualifier - like an email address. This requires 2 connections to the database, but it is a very common method. If you are using SQL server and your primary key is an "identity" type field then you have another option. You can insert and get back the identity value in the same query. This is preferred because of SQL treats a single query statement as an implicit transaction - meaning you are assured of data integrity, and that you will return the right value. Here's the way it works.

```
<cfquery name="putUser" datasource="#dsn#">
      SET NOCOUNT ON
         INSERT INTO USERS (username, email)
         VALUES
            (<cfqueryparam cfsqltype="CF_SQL_CHAR" value="#usersname#">,
             <cfqueryparam cfsqltype="CF_SQL_CHAR" value="#email#"> )
      SELECT @@Identity AS newId
      SET NOCOUNT OFF
   </cfquery>
```

If you output the value of "#putUser.newId#" you will see it's value is the new value of the "identity" field for the record you just created. Notice also the "NOCOUNT" attribute that I toggle on and off. This is to prevent some "extra" data from being sent back to the driver during the operation. If you've ever run multiple statements in query analyzer you will have seen this extra data in the message pane. It's the data that says "1 row affected" or "3 rows affected". SQL keeps track of the number of rows and outputs data to the driver telling it what is happening. This used to cause exceptions to be thrown on ODBC drivers when trying to access server level vars like @@Identity. Setting NOCOUNT ON and back OFF again alleviates this issue. I have to say candidly that I have no idea if this is an issue on a CFMX server having never tested it. Either way, setting nocount on and off makes a multi-statement query faster and more efficient.

You might be wondering about the variable @@identity. The variable @@identity contains the last identity value generated *in the current session*(the current session meaning "this" connection). That means if you do 2 inserts to 2 different tables and then select @@identity (which you must alias by the way), you will get the identity value of the last statement. Just be careful about how you arrange your statements and be cautious about updating more than 1 table in single statement. If you need to manipulate multiple table identity fields in a single statement the SCOPE_IDENTITY() is a better choice.

One more word of caution - watch out for triggers! A common use of triggers is to insert into table B when an insert is done on table A. For example, suppose you had a log table that contained the create date for inserting a new user and you put an "insert" trigger on the users table that inserted the userid and getDate(). If that second "log" table has an Identity field, @@identity will return the ID of the log table - since it was the "last identity generated in the session." In that case your only option is to use the SCOPE_IDENTITY() function as in: *SELECT SCOPE_IDENTITY() AS newID*.