Creating Views With CF Query: DDL Follies

Posted At : November 14, 2007 7:35 PM | Posted By : Mark Kruger
Related Categories: MS SQL Server, Coldfusion & Databases

Most queries in your Coldfusion code do one of four things - Select, Insert, Update or Delete. Maybe you did *not* know that, given the proper permissions, you can do just about anything that can be done on the DB server from within a query. You can backup and restore, drop users, even execute shell commands. That's why you should never create a datasource using the SA user. Instead you should define what you want a datasource to do and create a user for that purpose. Still, sometimes it is useful to be able to do *other* things using Coldfusion and Cfquery.

For example, I have a generic table with rows that look like "col1, col2" that holds form data. In this particular application the customer creates custom forms to collect data from specific clients. All the forms look different. One might have fullname, address, city, Postal code, and the next one might see first name, last name zip. When the data is submitted it is put in col1, col2, col3. But he has a reporting tool that allows him to query tables from the database and run reports for his customer. What can we do to make it easier for him to report? Surely "select col1, col2" isn't going to do it. The answer is to use T-SQL Data Definition Language (DDL) to create a view for each customer.

In the tool that the customer uses to create the forms I figure out which form element is going to be stored in which column. I added a "colAlias" to the form configuration data that the user works with. That way, the customer can name his own alias for the column. When the form data is submitted or updated I run the following routine:

```
<cfquery name="createView" datasource="myDSN" result="t">
CREATE VIEW dbo.#getprofile.viewName#
AS

SELECT authData_id, authp_id, userID, <cfloop query="getCols">
        #dCol# AS #colAlias#,</cfloop>
        cono
FROM    FormDataTable
WHERE    clientID = '#getprofile.clientID#'

</cfquery>
```

The result was a view where the aliases are descriptive of the content the generic columns contain. It looks something like this.

```
<cfquery name="createView" datasource="myDSN" result="t">
CREATE VIEW dbo.vwUsers_123456
AS

SELECT authData_id, authp_id, userID,
        charColShort1 AS EmployeeName,
        charColShort2 AS SSN,
        charColShort3 AS address1,
        charColShort4 AS address2,
        charColShort5 AS city,
        charColShort6 AS state,
        charColShort7 AS zip,
        charColShort8 AS phone,
        charColShort9 AS emial,
        charColShort10 AS isManager,
        cono
FROM    FormDataTable
WHERE    ClientID = '123456'
</cfquery>
```

Now in his reporting tool he can choose the view "vwUsers_123456" to select from knowing that the columns will be meaningfully named and the data can only match the ClientID '123456'. Now that is pretty neat and a very useful way to make a system designed to handle large pools of disparate data dumped into a table in a way that makes it easier to figure it out. But there's a gotcha.

The Gotcha

The gotcha is that this create statement has the potential to ***fail silently***. That's right! Some DDL is like the proverbial tree falling in the woods with no one to hear it. If it *does* fail it's going to leave you scratching your head. No exception will be thrown and if you look in the debug the query will be there as if it was successful. Even the "results" attribute (a handy recent addition to CF) will not give you any clues. For example, as a test I ran this query:

```
<cfquery name="createView" datasource="myDSN" result="t">
CREATE VIEW dbo.test
AS

SELECT authData_id, authp_id, userID,
        charColShort1 AS EmployeeName,
        charColShort2 AS SSN,
        charColShort3 AS address1,
        charColShort4 AS address1,
        charColShort5 AS city,
        charColShort6 AS state,
        charColShort7 AS zip,
        charColShort8 AS phone,
        charColShort9 AS emial,
        charColShort10 AS isManager,
        cono
FROM    FormDataTable
WHERE    ClientID = '123456'
</cfquery>
```

Clearly this *should* fail because I am aliasing 2 different column names as "address1". When I run this query in query analyzer it does indeed fail as I expect. But CF treats it like Thanksgiving with your extended family - lots of nice chatter like everything is fine but under the surface things are broken. If you run into this you will find yourself looking in the view list on query analyzer and pondering how it is that the Cfquery went through but the view simply isn't showing up.

The Fix

Well there isn't one really. What I chose to do was to query the view immediately (select top 1 from #viewname#) using a try catch block and throwing a message back to the calling page to check for duplicate columns. The real lesson is to be extra careful running DDL in your Coldfusion queries - you may not be able to predict the result. As a rule of thumb I am going to set up a test to intentionally throw an error in any DDL type queries I intend to run.