Using a DSN connection for Connectionless Access

Posted At : June 3, 2005 10:02 AM | Posted By : Mark Kruger
Related Categories: Coldfusion MX 7, Coldfusion & Databases, Coldfusion Tips and Techniques

Macromedia added "connectionless" DSN's in CF 5, then took them away again in CFMX. This much lamented feature was useful in certain instances. One of my favorites was for data export. For one of our e-commerce sites suppliers wanted a daily Access file for orders. We created an export process that copied an Access template to a new location (an Access db with the tables needed). Then we created an ODBC connection to it using the "connectionstring" attribute of CFQUERY and ran an insert routine to copy in the orders. The file was then zipped and automatically emailed to the supplier for drop shipment. In the words of Jimmy Neutron's Dad, "Now you gotta admit that's pretty neat!".

In CFMX however the connectionstring attribute is gone. This is because the system no longer interacts directly through ODBC. Instead it uses JDBC as the data access layer. That's a very good thing. We have had great results with speed and reliability through JDBC. It does put a crimp in our data export plan however. Fortunately there's a work-around. It's not perfect, but it works pretty well.

The way it works is by using an existing DSN connection as a proxy for a DSN connection. Think back to CF 4.5 when you were still using Access in production (ouch!) - remember that time you said to yourself "I wish I could join 2 tables from separate datasources." Then you stumbled upon a solution. Inside your CFQUERY you used the "IN '<%path%>\filename.MDB'" syntax? Why not use the same approach. All you need is to expand on it a little.

The following steps and test code were sent to me by Brian at StudioCart.com and they follow and expand on this MM Technote. First, you have to set up an Access DB Connection in the Coldfusion Administrator. It can be to a blank DB or you can even use any existing Access DSN. It's really just a placeholder for the socket. Then it's just a matter of using the "IN" clause to specify the filename in your queries. Here's a great sample.

```
<!--- Dynamic DSN TEST: Datasource = "pass_through_DSN" DSN points to "blank.mdb" Select and
Insert to "Orders.mdb" --->
<!--- Select test --->
<cfquery name="select" datasource="pass_through_DSN">
Select * from ordertbl
IN 'C:\ATLANTA_DSN\Orders.mdb'
</cfquery>
<cfoutput query="select">#idorderno#</cfoutput>
<!--- End Select test --->

<!--- Insert Test --->
<cfquery name="test" datasource="pass_through_DSN" >
Insert into stafftbl (idunique,firstname,surname,employeetype)
IN 'C:\ATLANTA_DSN\Orders.mdb'
Values ('22','bobyb','smith','10')
</cfquery>
<!--- End Insert Test --->

<!--- Verify Insert --->
<cfquery name="verify" datasource="pass_through_DSN">
```

```
Select firstname from stafftbl
IN 'C:\ATLANTA_DSN\Orders.mdb'
Where idunique = 22
</cfquery>

<cfoutput query="verify">#firstname#</cfoutput>
```

Remember you are specifying the file name as part of the *"what to fetch"* section of the query - not as part of the Where clause or as some additional attribute to CFQUERY. That means the "IN" clause goes immediately after the columns and tablename or names. This works:

```
<cfquery name="verify" datasource="pass_through_DSN">
Select firstname from stafftbl
IN 'C:\ATLANTA_DSN\Orders.mdb'
Where idunique = 22
</cfquery>
```

But this does *not* work.

```
<cfquery name="verify" datasource="pass_through_DSN">
Select firstname from stafftbl
Where idunique = 22
IN 'C:\ATLANTA_DSN\Orders.mdb'
</cfquery>
```

The same rule applies to inserts - you have to place the "IN" in the right section of the query. This works:

```
<cfquery name="test" datasource="pass_through_DSN" >
Insert into stafftbl (idunique,firstname,surname,employeetype)
IN 'C:\ATLANTA_DSN\Orders.mdb'
Values ('22','bobyb','smith','10')
</cfquery>
```

But this does *not* work:

```
<cfquery name="test" datasource="pass_through_DSN" >
Insert into stafftbl (idunique,firstname,surname,employeetype)
Values ('22','bobyb','smith','10')
IN 'C:\ATLANTA_DSN\Orders.mdb'
</cfquery>
```

Two important things to note. While this is a good work around it *does* require that a DSN be set up as a proxy or placeholder. So in reality it isn't really "dsnless". Secondly, this technique demonstrates the absolute inherent weakness of Access as a production DB. It is simply not a secure medium. When you use access you are storing your data in a FILE - not in a "system" like an RDBMS with lots of security features. Using it for exporting data is one thing (and quite useful), but using as your production db is a liability.