Inserting Custom SQL into DataMgr

Posted At : January 9, 2008 9:00 AM | Posted By : Steve
Related Categories: DataMgr

We ran into a situation on a current project where we needed to return a column based on complicated logic. We needed to have a column that indicated if the value of an answer had changed since the user had originally answered the question. This logic was sufficiently complicated that even a custom relation field wouldn't do the job.

In situations where a layer of abstraction isn't able to handle the complexity required, it makes sense to break out of that layer of abstraction. That is why a ColdFusion programmer may sometimes (rarely) have to revert to Java. For DataMgr, this suggests writing SQL manually in situations like this.

In our case, however, we had already defined several relation fields that we needed to use and we didn't want to have to redefine them (or to ask DataMgr for the SQL for each relation field individually). We needed a way to add a field composed of complex SQL to our query without losing the benefit of our DataMgr relation fields.

Here is some SQL that represents what we were trying to accomplish:

```
CASE
WHEN EXISTS (
SELECT QuestionID
FROM answers
WHERE answers.QuestionID = questions.QuestionID
AND answers.PerformanceID = #Val(QuizID)#
--Logic here to indicate that chosen option value has changed
) THEN 1
ELSE 0
END isQuestionChanged
```

After a bit of thinking, I realized that I could use a feature that I added in 2.1 as an experimental feature (mentioning, but not covering, it in the documentation).

The getRecords() method has an optional argument called "advsql" that takes a structure with keys for the different sections of a SELECT query (SELECT,FROM,WHERE,ORDER BY). The first three keys add SQL to those sections, whereas the ORDER BY key replaces the SQL for that section.

So, to solve our problem, we used code like the following:

```
<cfset sAdvSQL = StructNew()>
<cfsavecontent variable="sAdvSQL.SELECT"><cfoutput>
CASE
WHEN EXISTS (
SELECT QuestionID
FROM answers
WHERE answers.QuestionID = questions.QuestionID
AND answers.PerformanceID = #Val(QuizID)#
--Logic here to indicate that chosen option value has changed
) THEN 1
ELSE 0
END isQuestionChanged
</cfoutput></cfsavecontent>

<cfset qQuestions = variables.DataMgr.getRecords(tablename="questions",advsql=sAdvSQL)>
```

Now the qQuestions query has all of the relation fields defined for the questions table as well as the isQuestionChanged field that we defined in the advsql code.

I also just got asked about how to filter by operators other than equality. The advsql argument could be used for this as well:

```
<cfset sAdvSQL = StructNew()>
<cfsavecontent variable="sAdvSQL.WHERE"><cfoutput>
lastname LIKE 'sm%'
</cfoutput></cfsavecontent>

<cfset qUsers = variables.DataMgr.getRecords(tablename="users",advsql=sAdvSQL)>
```

This could also allow for more complicated logic:

```
<cfset sAdvSQL = StructNew()>
<cfsavecontent variable="sAdvSQL.WHERE"><cfoutput>
(
isOver21 = 1
OR hasParentalApproval = 1
)
</cfoutput></cfsavecontent>

<cfset qUsers = variables.DataMgr.getRecords(tablename="users",advsql=sAdvSQL)>
```

As you can see the advsql argument opens up the power of SQL to be used in conjunction with the convenience of DataMgr.

DataMgr is open source and free for any use.