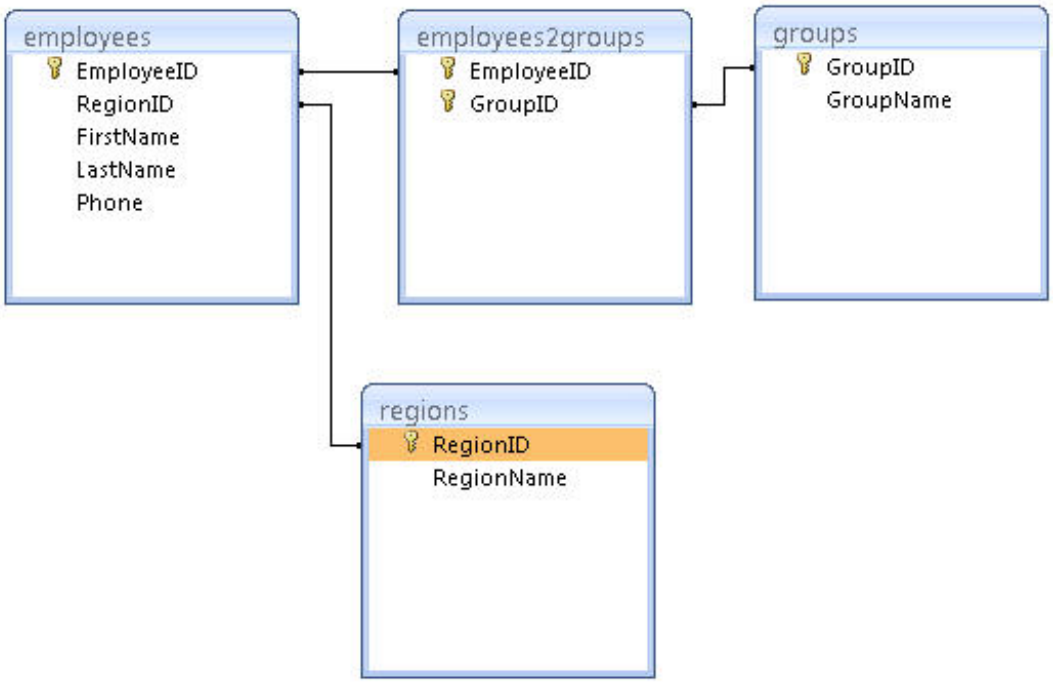
DataMgr List Relations

Posted At : November 26, 2008 7:30 AM | Posted By : Steve
Related Categories: DataMgr

DataMgr has had "list" relations for some time now, but I don't think I have yet done a good job of explaining them. Hopefully I can correct that today.

(**What is DataMgr?**)

So, let's start with this example:



employees

EmployeeID	RegionID	FirstName	LastName
1	1	Scott	Summers
2	2	Jean	Grey
3	3	Ororo	Munroe
4	4	Peter	Nicholas
5	1	Robert	Drake
6	2	Henry	McCoy
7	3	Kurt	Wagner
8	4	Charles	Xavier

groups

GroupID	GroupName
1	Blue
2	Gold

3	Red

regions

RegionID	RegionName
1	North
2	East
3	West
4	South

This presents use with several options for using a "list" relation.

One to Many

For example, I might want a list of all of the EmployeeIDs associated with a given region. For that, I could use the following XML (**see LoadXml**):

```
<table name="regions">
   <field ColumnName="RegionID" CF_Datatype="CF_SQL_INTEGER" PrimaryKey="true" Identity="true" />
   <field ColumnName="RegionName" CF_Datatype="CF_SQL_VARCHAR" Length="50" />
   <field ColumnName="employees">
      <relation
         type="list"
         table="employees"
         field="EmployeeID"
         join-field-local="RegionID"
         join-field-remote="RegionID"
      />
   </field>
</table>
```

Alternately:

```
<cfset sRelation = {
   type="list",
   table="employees",
   field="EmployeeID",
   join-field-local="RegionID",
   join-field-remote="RegionID"
}>
<cfset DataMgr.setColumn(tablename="regions",columnname="employees",Relation=sRelation)>
```

This simply references the "EmployeeID" field from the "employees" table where the value of the field indicated in "join-field-local" in the "regions" table ("RegionID") matches the value of the fields indicated in the "join-field-remote" ("RegionID") in the "employees" table.

Now, because both the "join-field-local" and "join-field-remote" values matched, I could have just used on "join-field" attribute as a shortcut, but I will stick with the more verbose syntax for clarity.

If I call DataMgr.getRecords("regions"), I will get back a query with an "employees" column that will have a comma delimited list with the values of every "EmployeeID" from a row where the "RegionID" of the "employees" table matches the "RegionID" for that record in the "region" table.

```
<cfset qEmployees = DataMgr.getRecords(tablename="regions")>
```

RegionID	RegionName	employees
1	North	1,5
2	East	2,6
3	West	3,7
4	South	4,8

The table in the database wouldn't have a "groups" field, but it would still be part of the query returned from that table by DataMgr.

Many to Many

I also might want a list of all of the GroupIDs associated with a given employee. For that, I could use the following XML:

```
<table name="employees">
   <field ColumnName="groups">
      <relation
         type="list"
         table="groups"
         join-table="employees2groups"
         field="GroupID"
         local-table-join-field="EmployeeID"
         join-table-field-local="EmployeeID"
         join-table-field-remote="GroupID"
         remote-table-join-field="GroupID"
      />
   </field>
</table>
```

Or the Following setColumn() call:

```
<cfset sRelation = {
   type="list",
   table="groups",
   join-table="employees2groups",
   field="GroupID",
   local-table-join-field="EmployeeID",
   join-table-field-local="EmployeeID",
   join-table-field-remote="GroupID",
   remote-table-join-field="GroupID"
}>
<cfset DataMgr.setColumn(tablename="employees",columnname="groups",Relation=sRelation)>
```

Then a getRecords() call on the employees table can include the "groups" column.

```
<cfset qEmployees = DataMgr.getRecords(tablename="employees",fieldlist="EmployeeID,FirstName,LastName,groups")>
```

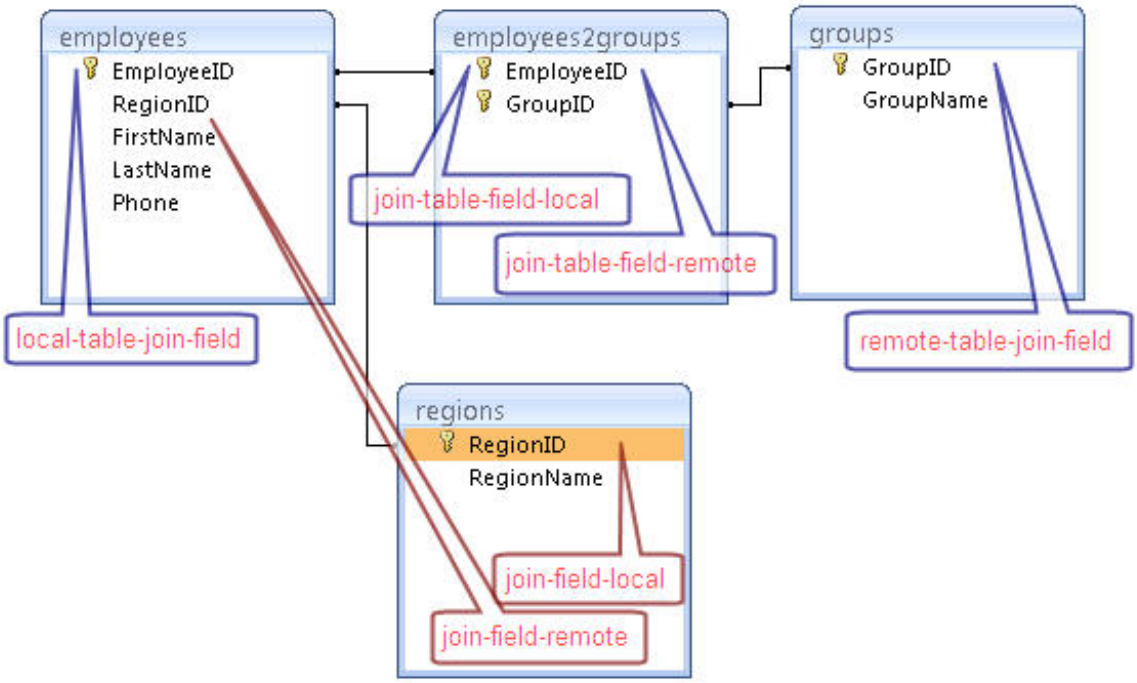
EmployeeID	FirstName	LastName	groups
1	Scott	Summers	1,3
2	Jean	Grey	1,2

3	Ororo	Munroe	2
4	Peter	Nicholas	2
5	Robert	Drake	
6	Henry	McCoy	1,3
7	Kurt	Wagner	
8	Charles	Xavier	1,2,3

Here are what each attribute indicates:

- type="list": indicates I want a list
- table="groups": Indicates that I want my column to contain a value from a field in the "groups" table
- join-table="employees2groups": Indicates that this is a many-to-many relationship stored in the "employees2groups" table
- field="GroupID": Indicates that I want to get a value from the "GroupID" field of the "groups" table
- local-table-join-field="EmployeeID": means that the "EmployeeID" field of the "employees" table should match a field in the "employees2groups" table
- join-table-field-local="EmployeeID": means the at the "EmployeeID" field of "employees2groups" should match a field in the "employees" table
- join-table-field-remote="GroupID": means that the "GroupID" field of "employees2groups" should match a field in the "groups" table
- remote-table-join-field="GroupID": means that the "GroupID" field of "groups" should match a field in the "employees2groups" table

The fields marked with red balloons indicate the One-to-Many example, while the fields with the blue balloons indicate the Many-to-Many example.



Since this list relation uses a join-table, the value of the field can be set when saving a record in the "employees" table.

```
<cfset sData = {EmployeeID=3,groups="1,2}>
<cfset Datamgr.saveRecord("employees",sData)>
```

This will add or remove records to the "employees2groups" table so that the value of the groups field is "1,2".

If I wanted to get a list of the "GroupName" fields from the "groups" table for groups associated with a given user, I could do that as well.

```
<table name="employees">
   <field ColumnName="groupnames">
      <relation
         type="list"
         table="groups"
         join-table="employees2groups"
         field="GroupName"
         local-table-join-field="EmployeeID"
         join-table-field-local="EmployeeID"
         join-table-field-remote="GroupID"
         remote-table-join-field="GroupID"
      />
   </field>
</table>
```

```
<cfset qEmployees = DataMgr.getRecords(tablename="employees",fieldlist="EmployeeID,FirstName,LastName,groupnames")>
```

EmployeeID	FirstName	LastName	groupnames
1	Scott	Summers	Blue,Red
2	Jean	Grey	Blue,Gold
3	Ororo	Munroe	Gold
4	Peter	Nicholas	Gold
5	Robert	Drake	
6	Henry	McCoy	Blue,Red
7	Kurt	Wagner	
8	Charles	Xavier	Blue,Gold,Red

Just like when saving the "groups" field, a value can be passed in to the "groupnames" field.

```
<cfset sData = {EmployeeID=3,GroupNames="Gold,Blue"}>
<cfset Datamgr.saveRecord("employees",sData)>
```

Note that this will only work if a join-table is included in the list relation.

Hopefully this provides a good explanation of list relations in DataMgr. If you have any questions, let me know.

DataMgr is open source and free for any use. Download 2.2 Beta from the **DataMgr page** on my site.