Article
Brendan Bannon · Sep 1, 2016 9m read

The Art of Mapping Globals to Classes (2 of 3)

The Art of Mapping Globals to Classes (2 of 3)

If you are looking to breathe new life into an old MUMPS application follow these steps to map your globals to classes and expose all that beautiful data to Objects and SQL.

This example is going to cram in 4 or 5 different things beyond what was covered in Part 1

All that is left after this is the Parent Child mapping example and then you will be on your way.

Same disclaimer: If you can ' t make heads or tails of your globals after reviewing these articles please contact the WRC and we will try to help you out: Support@InterSystems.com

Steps for Mapping a Global to a Class:

1. Identify a repeating pattern in the global data.
2. Identify what makes up a unique key.
3. Identify the properties and their types.
4. Define the properties in the class (don ' t forget the properties from the variable subscripts).
5. Defined the IdKey index.
6. Define the Storage Definition:
    a. Define the Subscripts up to and including the IdKey.
    b. Define the Data section.
    c. Ignore the Row ID section. 99% of the time the default is what you want so let the system fill that in.
7. Compile and test your class / table.

Say you have 2 globals that look something like this:

```
^mapping("Less Simple",1,1)="Bannon,Brendan^Father"
```

^mapping("Less Simple",1,1,"Activity")="Rock Climbing"

^mapping("Less Simple",1,2)="Bannon,Sharon^Mother"

^mapping("Less Simple",1,2,"Activity")="Yoga"

^mapping("Less Simple",1,3)="Bannon,Kaitlin^Daughter"

^mapping("Less Simple",1,3,"Activity")="Lighting Design"

^mapping("Less Simple",1,4)="Bannon,Melissa^Daughter"

^mapping("Less Simple",1,4,"Activity")="Marching Band"

^mapping("Less Simple",1,5)="Bannon,Robin^Daughter"

^mapping("Less Simple",1,5,"Activity")="reading"

^mapping("Less Simple",1,6)="Bannon,Kieran^Son"

^mapping("Less Simple",1,6,"Activity")="Marching Band"

^index("Less Simple","FName","BRENDAN",1,1)=""

^index("Less Simple","FName","KAITLIN",1,3)=""

^index("Less Simple","FName","KIERAN",1,6)=""

^index("Less Simple","FName","MELISSA",1,4)=""

^index("Less Simple","FName","ROBIN",1,5)=""

^index("Less Simple","FName","SHARON",1,2)=""

Yes it is a requirement that you learn about my family as you learn about mapping globals. I am not creative enough to invent data.

Step 1:

This time it looks like the repeating data is spread out over 2 global nodes instead of 1.

Step 2:

Hard to tell for sure with just this bit of data. Either the first or the first and second subscripts are constants. For this example we are going to assume that the second subscript is variable, FamilyId, and the third subscript is the PersonId.

Step 3:

So we have the two variable subscripts FamilyId and PersonId, and then we have the name, relation, and activity. Looking at the index global again for a hint I am going to map the name to two properties, FirstName and LastName. So that is a total of 6 properties we need to define.

Step 4:

A couple of things are new here. All the properties have an SQL Field Name and FirstName has a collation of UPPER. Defining SQL Field Names will make it clearer when we are talking about a Property and when we are talking about an SQL Field. The collation we will talk about when we look at mapping the index.

```
Property FamilyId As %Integer [ SqlFieldName = FamilyId ];

Property PersonId As %Integer[ SqlFieldName = PersonId ];

Property FirstName As %String(COLLATION = "UPPER")[ SqlFieldName = FirstName ];

Property LastName As %String[ SqlFieldName = LastName ];

Property Relation As %String[ SqlFieldName = Relation ];

Property Activity As %String[ SqlFieldName = Activity ];
```

Step 5:

There are 2 variable subscripts so that means the IdKey will be based on two properties:

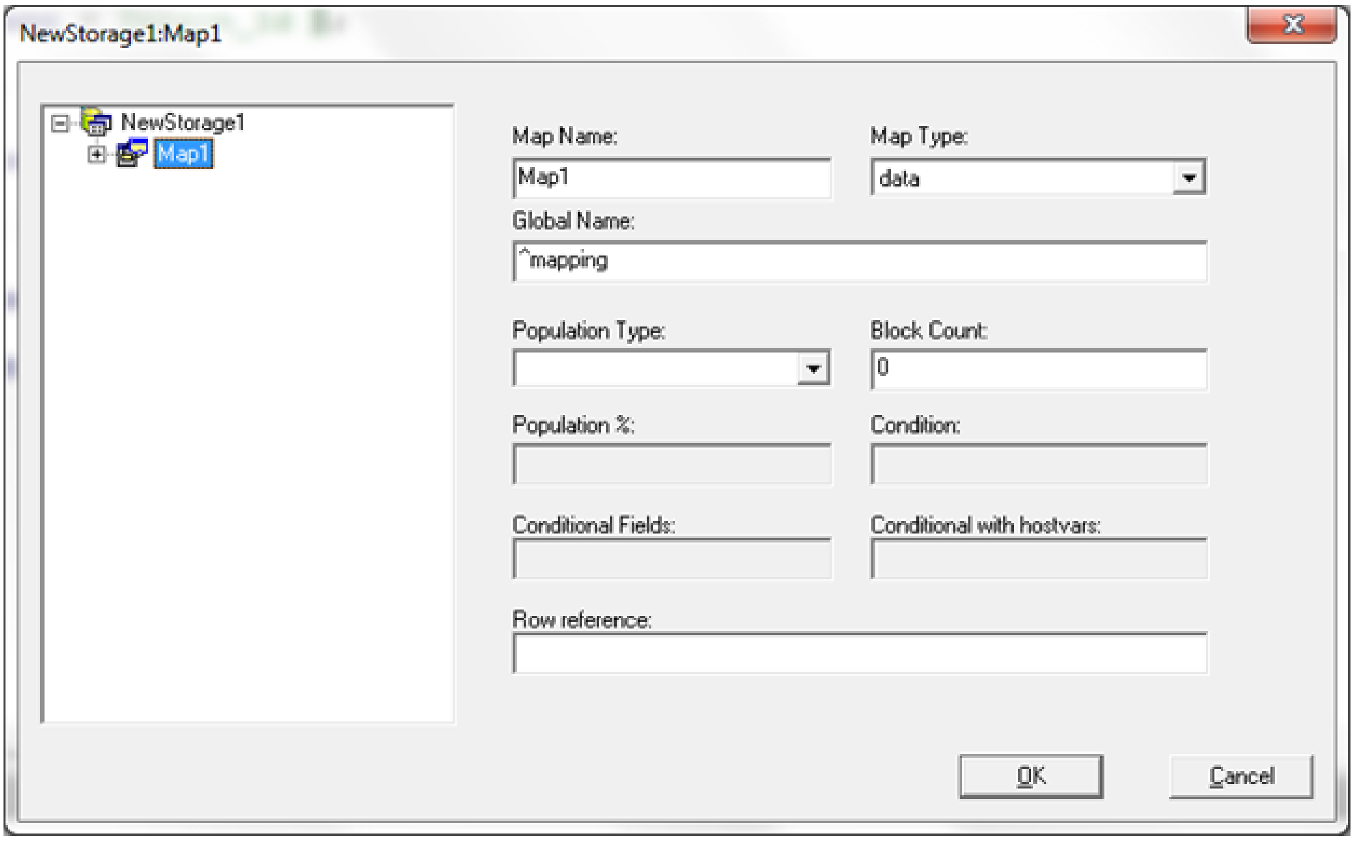
```
Index Master On (FamilyId, PersonId)[IdKey];
```

Again we have 1 index defined on 1 property, FirstName.

```
Index FNameIndex On FirstName;
```

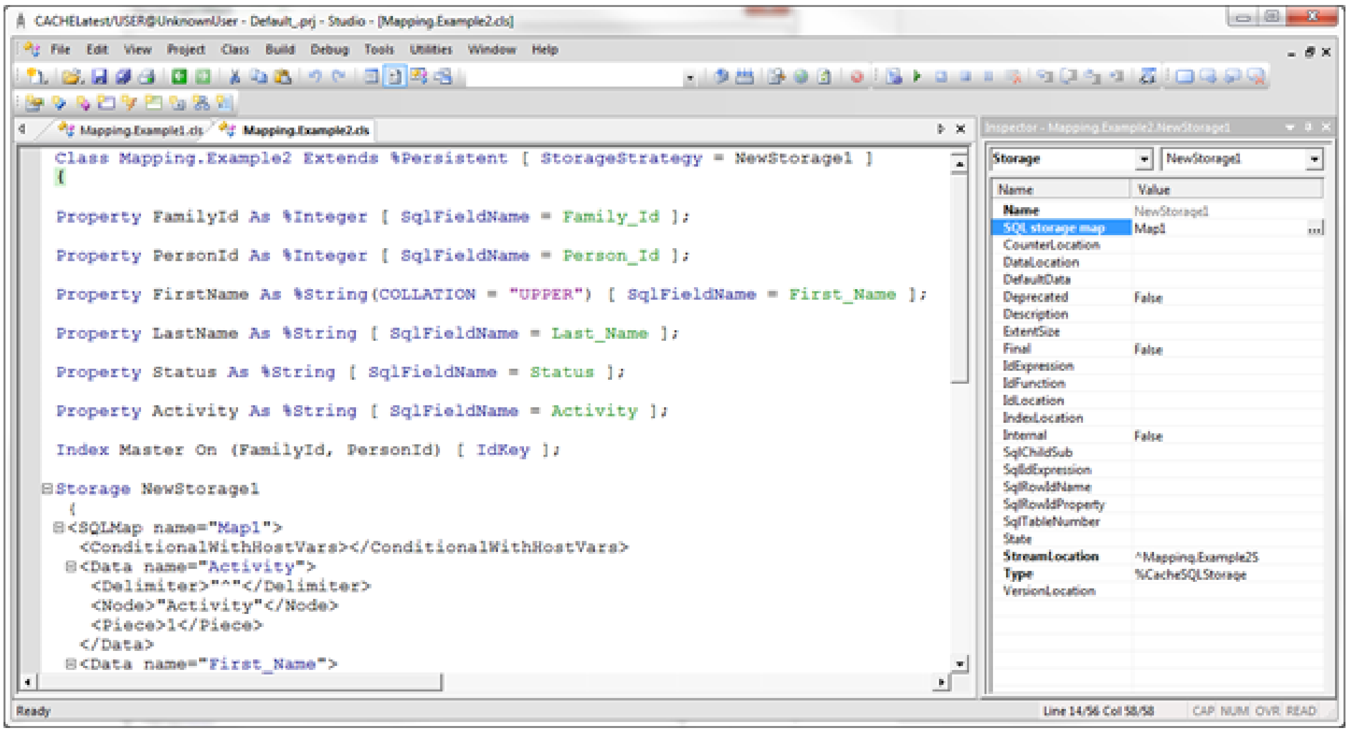
Step 6:

First thing to do is to create the storage definition. Click on the Storage icon or use the Inspector, Storage, right click, New Storage. This time I did leave the default value for the Map Name, just filling in the Global Name, ^mapping.



Map Name: Map1
Map Type: data
Global Name: ^mapping
Population Type:
Block Count: 0
Population %:
Condition:
Conditional Fields:
Conditional with hostvars:
Row reference:

One thing I did not show in the first example is how to get back to the wizard after you have created a storage definition. From the Inspector select Storage, then pick the Storage Name (in this case NewStorage1) click on SQL Storage Map, and then click on the box to the far right to open the wizard.


CACHELatest/USER@UnknownUser - Default_prj - Studio - [Mapping.Example2.cls]
File Edit View Project Class Build Debug Tools Utilities Window Help
Mapping.Example1.cls
Mapping.Example2.cls
Inspector - Mapping.Example2.NewStorage1
Ready
Line 14/56 Col 58/58
CAP NUM OVR READ

```
Class Mapping.Example2 Extends %Persistent [ StorageStrategy = NewStorage1 ]
{

Property FamilyId As %Integer [ SqlFieldName = Family_Id ];

Property PersonId As %Integer [ SqlFieldName = Person_Id ];

Property FirstName As %String(COLLATION = "UPPER") [ SqlFieldName = First_Name ];

Property LastName As %String [ SqlFieldName = Last_Name ];

Property Status As %String [ SqlFieldName = Status ];

Property Activity As %String [ SqlFieldName = Activity ];

Index Master On (FamilyId, PersonId) [ IdKey ];

Storage NewStorage1
{
<SQLMap name="Map1">
<ConditionalWithHostVars></ConditionalWithHostVars>
<Data name="Activity">
 <Delimiter>"^"</Delimiter>
 <Node>"Activity"</Node>
 <Piece>1</Piece>
</Data>
<Data name="First_Name">
```

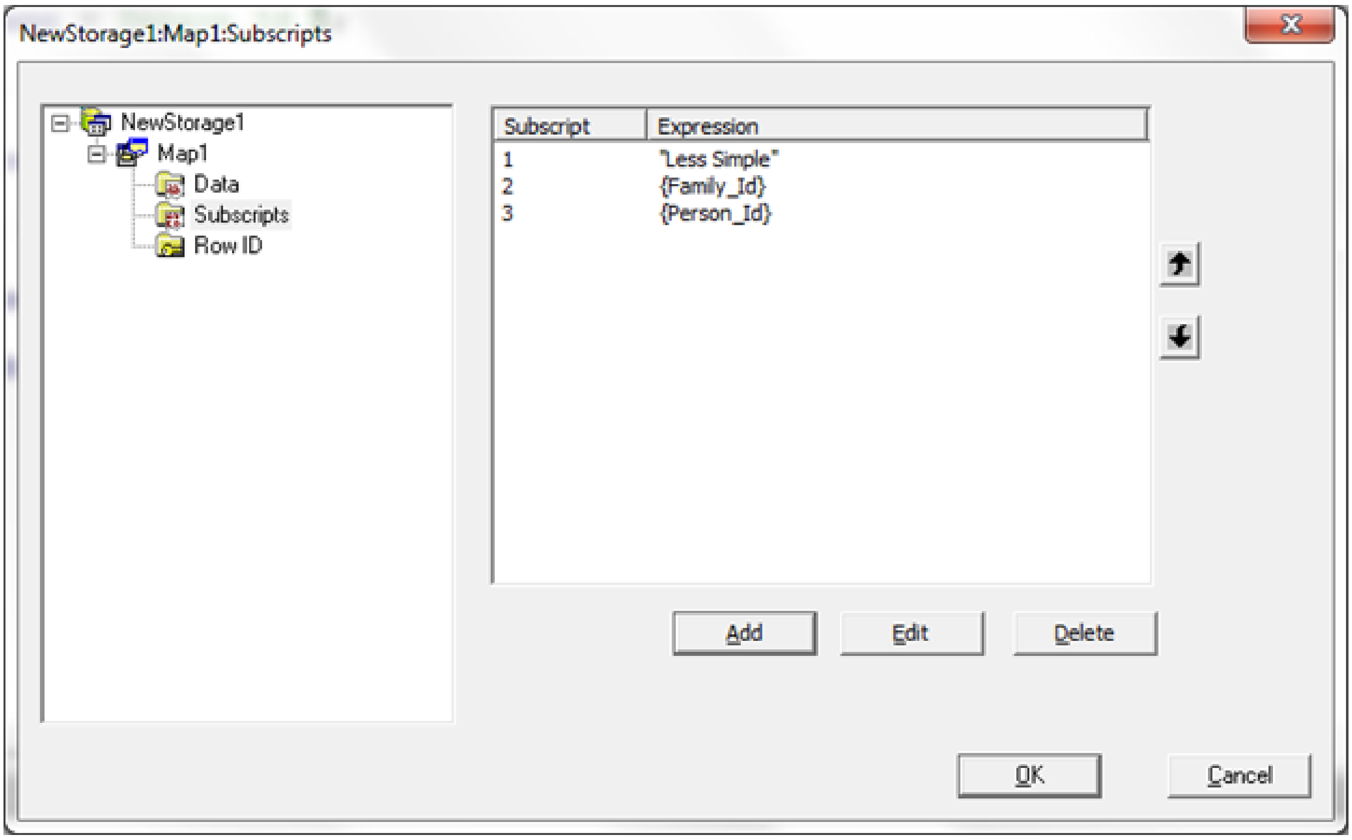
Storage | NewStorage1

Name	Value
Name	NewStorage1
SQL storage map	Map1
CounterLocation	
DataLocation	
DefaultData	
Deprecated	False
Description	
ExtentSize	
Final	False
IdExpression	
IdFunction	
IdLocation	
IndexLocation	
Internal	False
SqlChildSub	
SqlIdExpression	
SqlRowIdName	
SqlRowIdProperty	
SqlTableNumber	
State	
StreamLocation	^Mapping.Example2S
Type	%CacheSQLStorage
VersionLocation	

Step 6a:

Starting with the Subscripts section again. First we will need 3 subscript levels to define everything up to and including the IdKey: 1 constant and 2 fields. The fields that are listed in this window need to match the properties that are defined for the IdKey index.

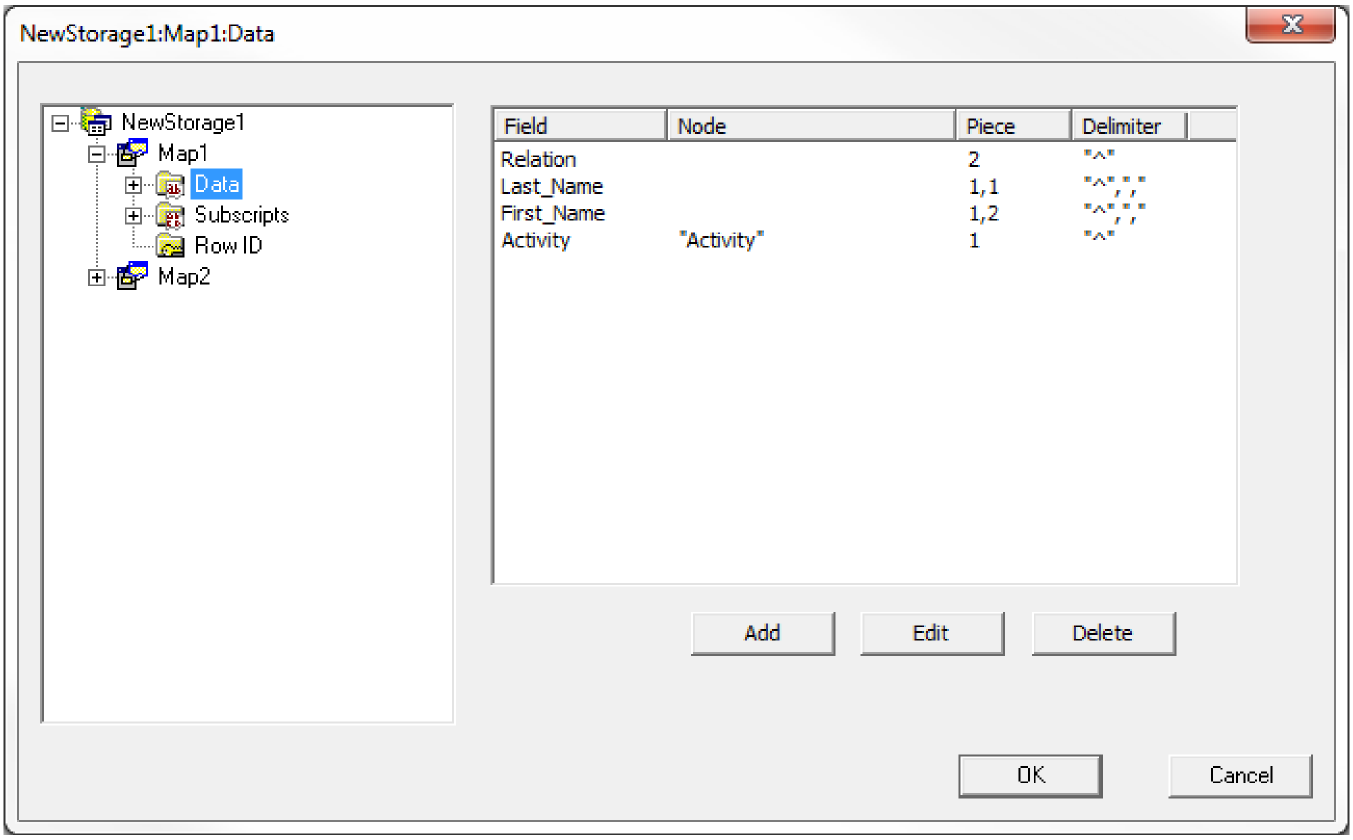
Note: In the storage definition we are always referencing the SQL field name, FamilyId, not the property name, FamilyId.



Subscript	Expression
1	"Less Simple"
2	{Family_Id}
3	{Person_Id}

Step 6b:

For the Data section we have 3 fields stored in 1 global node and then we have 1 field stored at a lower subscript level (“ Activity ”). To get at the first name and the last name as 2 different fields we will need to use multiple pieces and delimiters corresponding to a nested $PIECE(). In ObjectScript this would look like: set FirstName=$PIECE($PIECE(^mapping(“ Less Simple ” ,1,1), ” ^ ” ,1), ” , ” ,2). In the wizard we need to list all the Delimiters and Pieces starting on the innermost $PIECE() and working our way out. Activity is on a different global node so we add the constant to Node. Piece and Delimiter default to 1 and “ ^ ” so I just left them there.

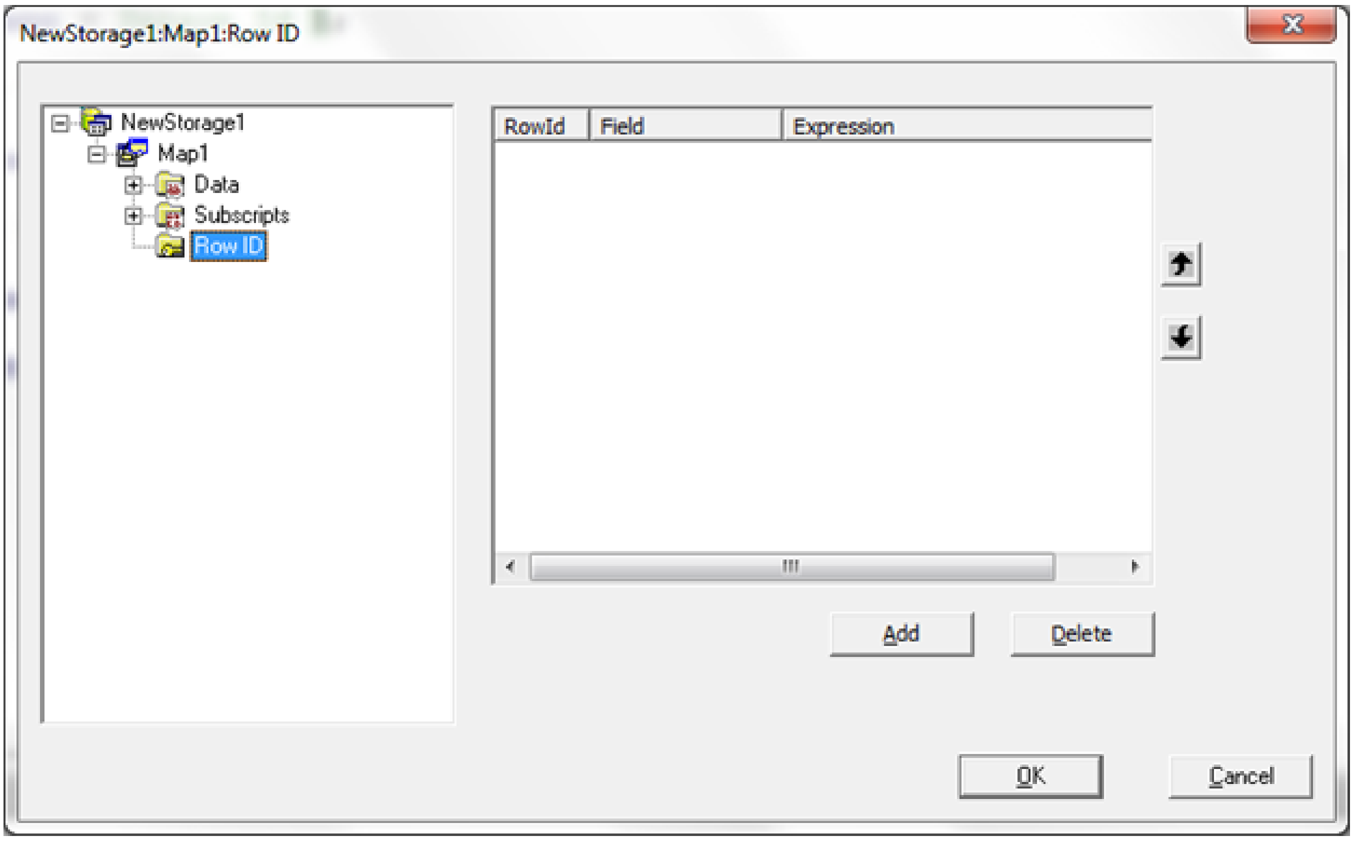


Field	Node	Piece	Delimiter
Relation		2	"^"
Last_Name		1,1	"^",","
First_Name		1,2	"^",","
Activity	"Activity"	1	"^"

Step 6c:

Still nothing to see here.

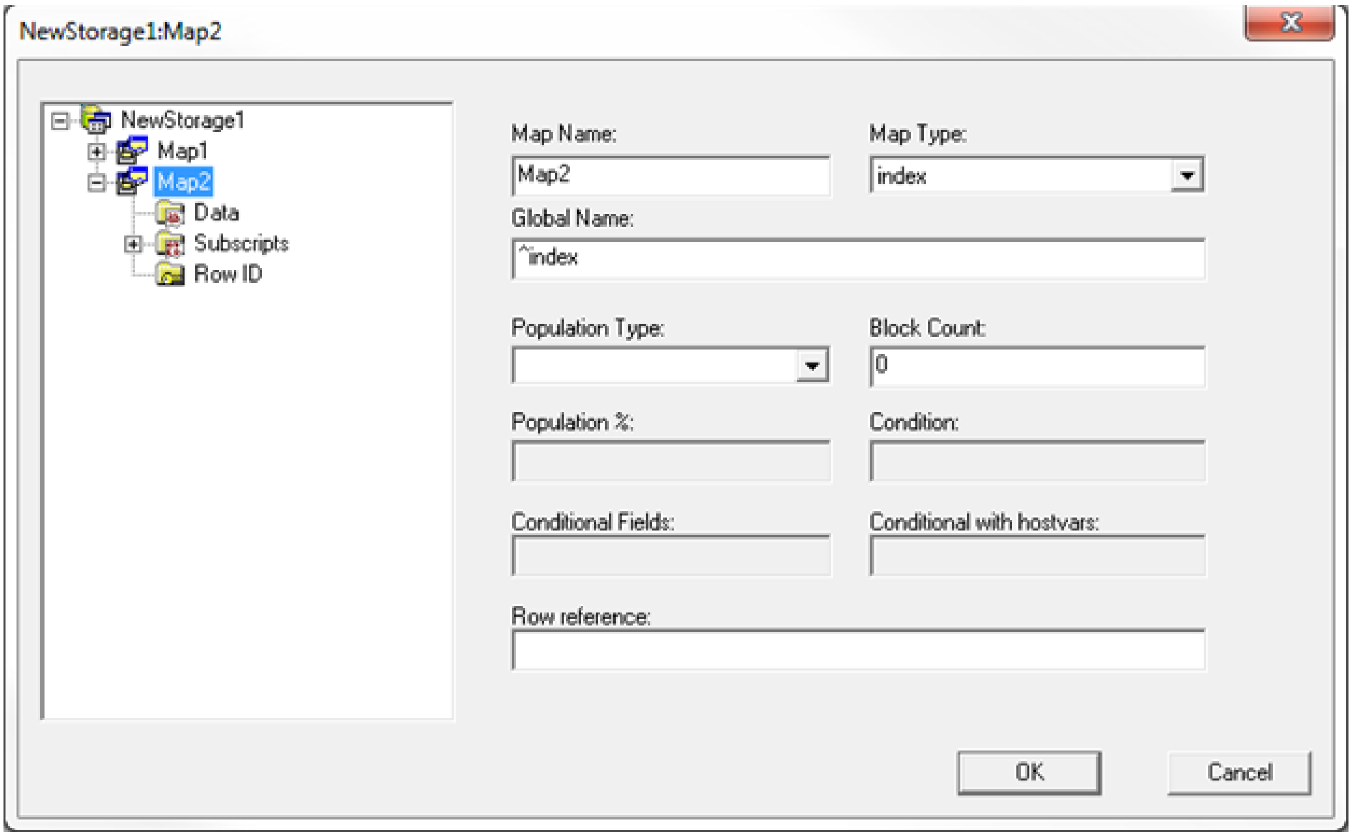
OK I know you are dying to see an example of when you need to define the Row ID section. When I post the zip file with all the examples there will be one called Mapping.RowIdSpec that will show a case where this is needed.



Now we can repeat Step 6 for the index map.

Step 6

Create Map2 and set the Global Name to ^index.



Map Name: Map2
Map Type: index
Global Name: ^index
Population Type:
Block Count: 0
Population %:
Condition:
Conditional Fields:
Conditional with hostvars:
Row reference:

Step 6a

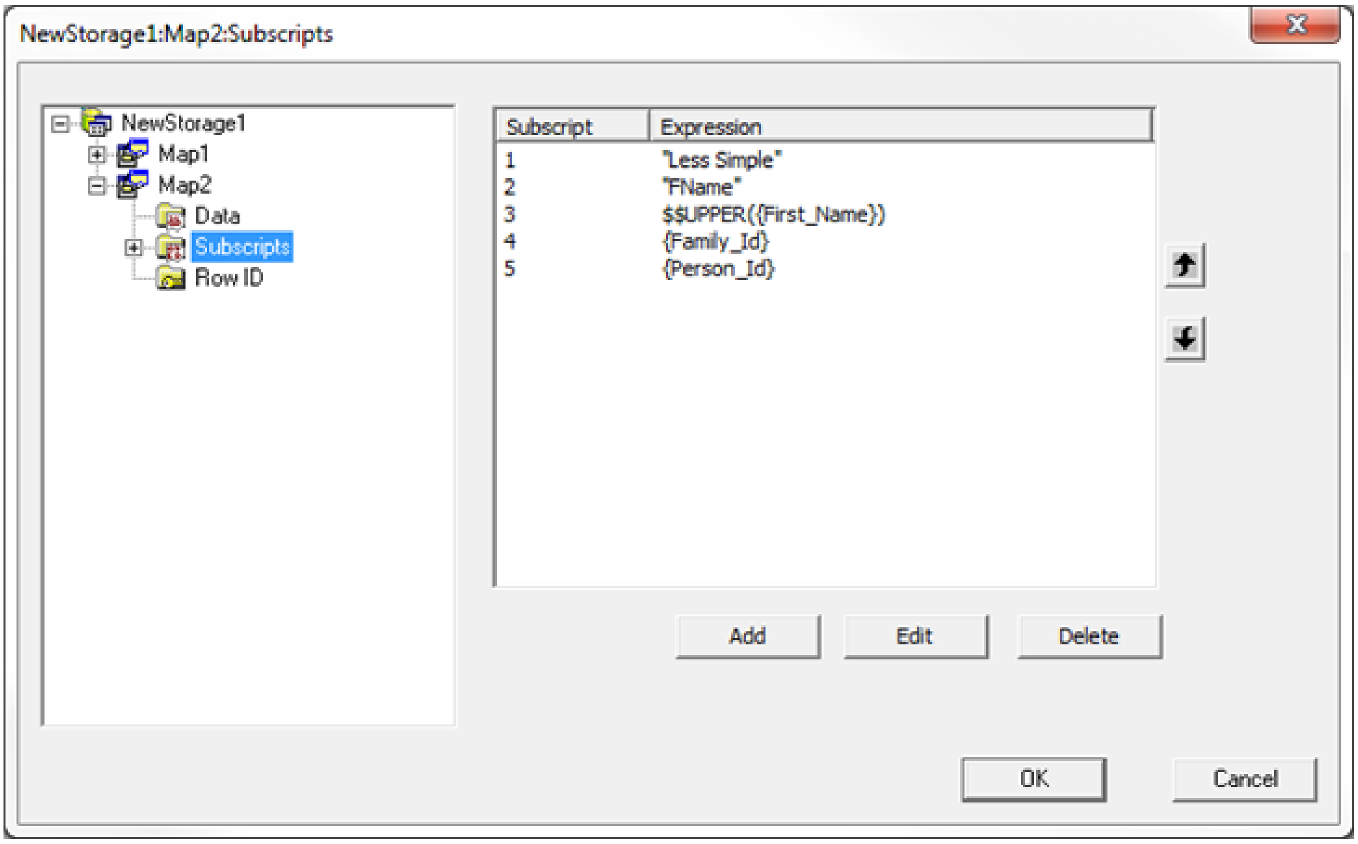
The index global has 5 subscripts: 2 constants, 2 IdKey fields, and the indexed field, FirstName. Looking at the ^mapping global and the ^index global the values for FirstName are different. In the ^mapping global the names are mixed case while in the ^index global the names are all UPPER case. Caché has several built in collation functions you can use to transform string data. Check out this link for info on the different functions you can use:

http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLbasics#GSQLbasicscollation

For Caché SQL Storage there is a good chance you might need to go to the legacy collation types:

http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLbasics#GSQLbasicscollationlegacy

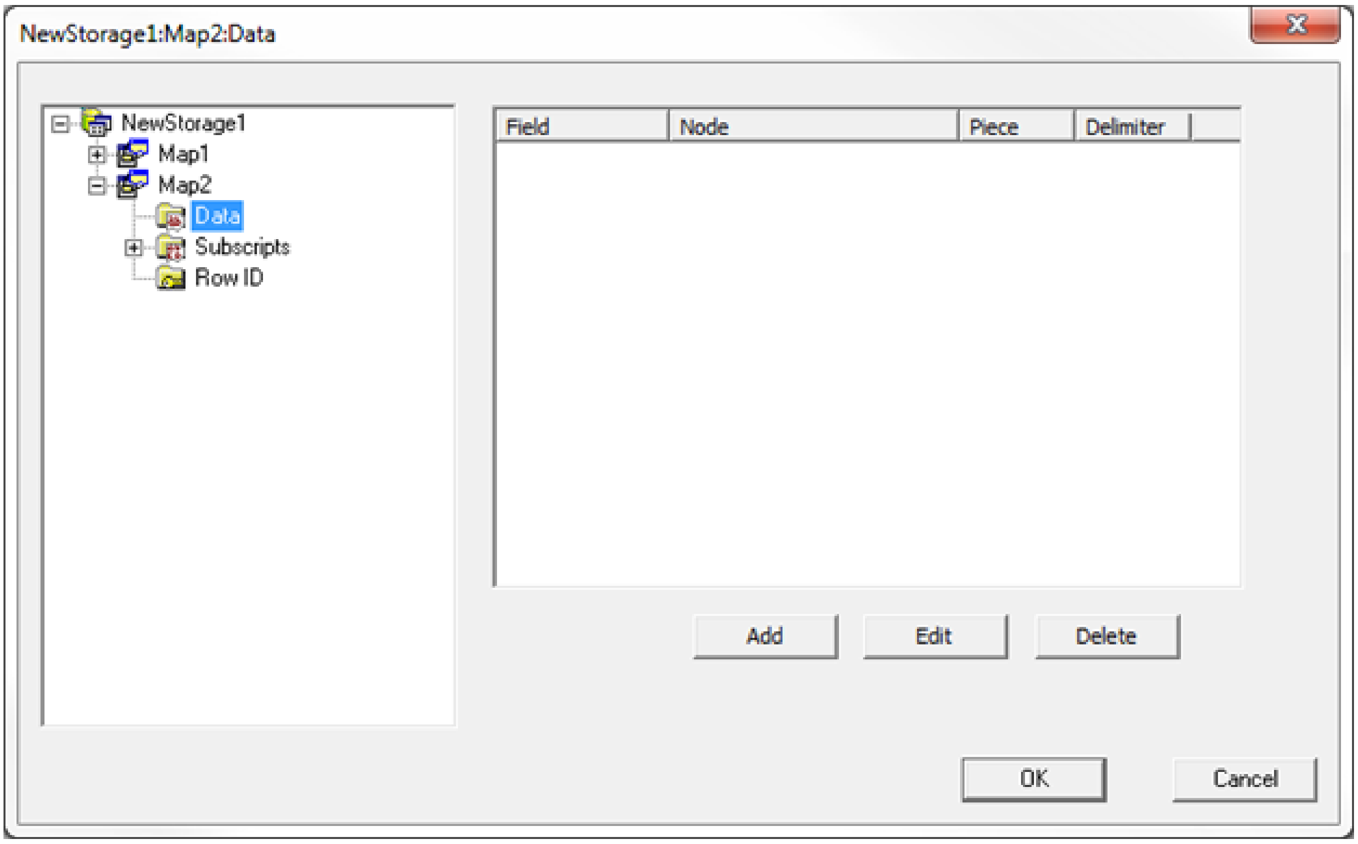
It is very important that the collation function used here matches the collation in the property definition. Having these not match is the most common reason an index map is not used by the query optimizer. Remember %String has a default collation of SQLUPPER while not providing a function in the mapping would be the same as EXACT collation.



Subscript	Expression
1	"Less Simple"
2	"FName"
3	$$UPPER({First_Name})
4	{Family_Id}
5	{Person_Id}

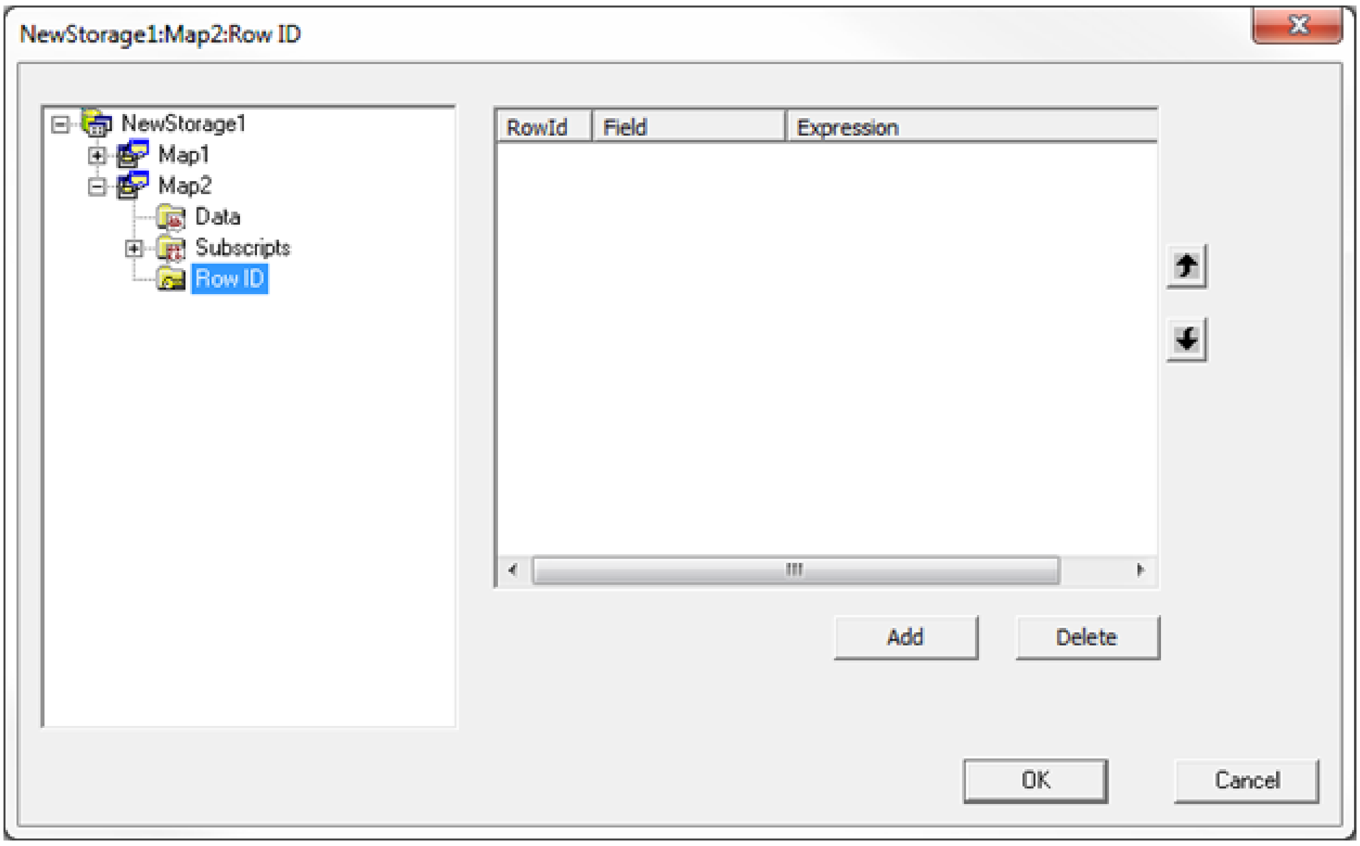
Step 6b

The ^index global has no data so nothing to define here.



Step 6c

By now you know, nothing here either.



Step 7:

```
Compilation started on 08/22/2016 15:42:16 with qualifiers 'fck /checkuptodate=expandedonly'
Compiling class Mapping.Example2
Compiling table Mapping.Example2
Compiling routine Mapping.Example2.1
Compilation finished successfully in 0.144s.
```

```
SELECT FamilyID, PersonID, FirstName, LastName, Relation, Activity

FROM Mapping.Example2
```

```
FamilyId        PersonId        FirstName      LastName       Relation           Activity

1          1          Brendan             Bannon Father   Rock Climbing

1          2          Sharon  Bannon Mother Yoga

1          3          Kaitlin    Bannon Daughter           Lighting Design

1          4          Melissa Bannon Daughter           Marching Band
```

1	5	Robin	Bannon	Daughter	Reading
1	6	Kieran	Bannon	Son	Marching Band

OK this time I am not going to bother with the storage definition. If you want to look at that you can load the xml and look at it there: mapping.example2.zip

#Globals #Mapping #SQL #Caché

Source URL:https://community.intersystems.com/post/art-mapping-globals-classes%C2%A0-2-3