Article
Robert Cemper · Apr 27, 2020 4m read

Multidimensional Property Persistence - Part 1 (Classic)

As you know in Caché / IRIS you have the possibility to define a property as Multidimensional as documented here:
https://docs.intersystems.com/iris20201/csp/docbook/DocBook.UI.Page.cls?KEY=GOBJ_proplit#GOBJ_proplit_multidim
and the explanation of how to use it
https://docs.intersystems.com/iris20201/csp/docbook/Doc.View.cls?KEY=GOBJ_proplit#GOBJ_proplit_multidim_values

Though the access is quite comfortable (in traditional COS sense) there are 2 main restrictions that hurt:

#1) It is not saved to disk unless your application includes code to save it specifically.
#2) It cannot be stored in or exposed through SQL tables
  there are some more
I'll show how to overcome these limits

#1) Let's take this simple class as example:

```
Class DC.Multi Extends (%Persistent, %Populate) [ Final ]
{
    Property Name As %String;
    Property DOB As %Date;
    Property Multi As %String [ MultiDimensional ] ;
```

The storage map already shows issue #1 no place for "Multi"

```
Storage Default
{
  <Data name="MultiDefaultData">
    <Value name="1">
      <Value>Name</Value>
    </Value>
    <Value name="2">
      <Value>DOB</Value>
    </Value>
  </Data>
  <DataLocation>^DC.MultiD</DataLocation>
  <DefaultData>MultiDefaultData</DefaultData>
  <IdLocation>^DC.MultiD</IdLocation>
  <IndexLocation>^DC.MultiI</IndexLocation>
  <StreamLocation>^DC.MultiS</StreamLocation>
  <Type>%Storage.Persistent</Type>
}
```

so we add 2 Methods:

```
/// save Multidimensional property
Method %OnAfterSave(insert As %Boolean) As %Status [
Private, ServerOnly = 1 ]
{  Merge ^(..%Id(),"Multi")=i%Multi quit $$$OK   }

/// load Multidimensional property
Method %OnOpen() As %Status [ Private, ServerOnly = 1 ]
{  Merge i%Multi=^(..%Id(),"Multi") quit $$$OK  }
```

we just attach the orphaned structure to our actual object.

To be honest:
This is not my invention, but the (simplified) approach that was used
in class %CSP.Session when it was written around the start of the millennium.

With the next simple add-on your multidimensional structure becomes persistent.

The object in memory looks like this:

```
CACHE>zw o2
o2=3@DC.Multi  ; <OREF>
+----------------- general information ---------------
¦     oref value: 3
¦     class name: DC.Multi
¦         %%OID: $lb("2","DC.Multi")
¦ reference count: 2
+----------------- attribute values ------------------
¦      %Concurrency = 1  <Set>
¦               DOB = 62459
¦          Multi("a") = 1
¦    Multi("rob",1) = "rcc"
¦    Multi("rob",2) = 2222
¦              Name = "Klingman,Uma C."
```

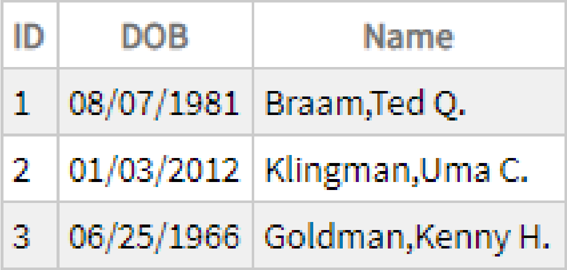
```
+---------------------------------------------------
```

and that's the related storage, a nice multidimensional global:

```
CACHE>zw ^DC.MultiD(2)
^DC.MultiD(2)=$lb("Klingman,Uma C.",62459)
^DC.MultiD(2,"Multi","a")=1
^DC.MultiD(2,"Multi","rob",1)="rcc"
^DC.MultiD(2,"Multi","rob",2)=2222
```

OK so far:

#2) **SELECT * from DC.Multi**
has no idea what a column "Multi" might be.

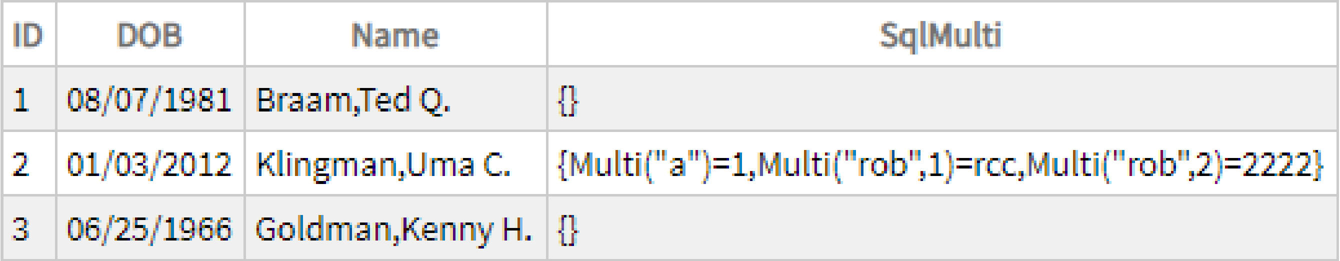


ID	DOB	Name
1	08/07/1981	Braam,Ted Q.
2	01/03/2012	Klingman,Uma C.
3	06/25/1966	Goldman,Kenny H.

so we add an SQLfriedly calculated property and some appropriate styling.

```
Property SqlMulti As %String(MAXLEN = "")
[ Calculated, SqlComputed,
SqlComputeCode = { set {*}= ##class(DC.Multi).ShowMulti({ID}) } ];
ClassMethod ShowMulti(id) As %String(MAXLEN="")
{  set res="{"
      ,obj=..%OpenId(id)
   if $isobject(obj) {
      set qry=$query(obj.Multi(""),1,value)
      while qry'="" {
         set res=res_$piece(qry,".",2,99)_"="value_  ","
            ,qry=$query(@qry,1,value)
         }
     set $extract(res,*)=""
   }
   quit res_"}"
}
```

And it looks like this



ID	DOB	Name	SqlMulti
1	08/07/1981	Braam,Ted Q.	{}
2	01/03/2012	Klingman,Uma C.	{Multi("a")=1,Multi("rob",1)=rcc,Multi("rob",2)=2222}
3	06/25/1966	Goldman,Kenny H.	{}

No need to say that the design is totally in your hands.

As there is evident progress from the past the next article will show you a solution more appropriate to the new century.

#Other

Source URL:https://community.intersystems.com/post/multidimensional-property-persistence-part-1-classic