Article

[Lorenzo Scalese](#) · Mars 30, 2022 3m de lecture

Persistence des propriétés multidimensionnelles - Partie 1 (classique)

Comme vous le savez, dans Caché / IRIS, vous avez la possibilité de définir une propriété comme Multidimensionnelle, comme documenté [ici](#) et l'explication de la façon de l'utiliser est [ici](#).

Bien que l'accès soit assez confortable (au sens traditionnel du COS), il y a 2 restrictions principales qui font mal :

1. Il n'est pas sauvegardé sur le disque, sauf si votre application inclut du code pour le sauvegarder spécifiquement.
2. Il ne peut pas être stocké dans des tableaux SQL ou exposé à travers ceux-ci.

il y en a d'autres

Je vais vous montrer comment surmonter ces limites.

1. Prenons cette classe simple comme exemple :

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

La carte de stockage montre déjà le problème n°1 : pas de place pour "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>
}
```

donc nous ajoutons 2 méthodes :

```
/// 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 }
```

nous attachons juste la structure orpheline à notre objet réel.
Pour être franc, ce n'est pas mon invention, mais l'approche (simplifiée) qui était utilisée dans la classe %CSP.Session lorsqu'elle a été écrite vers le début du millénaire.
Avec le simple ajout suivant, votre structure multidimensionnelle devient persistante.

L'objet en mémoire ressemble à ceci :

```
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."
+-----
```

et c'est le stockage associé, une jolie globale multidimensionnelle :

```
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
```

Jusqu'à présent, tout va bien.

2. SELECT * from DC.Multi
n'a aucune idée de ce que peut être une colonne "Multi".

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

donc nous ajoutons une propriété calculée SQL et un style approprié.

```
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_"})"  
}
```

Et cela ressemble à ceci

Inutile de dire que la conception est totalement entre vos mains.

Comme il y a un progrès évident par rapport au passé, le prochain article vous montrera une solution plus appropriée au nouveau siècle.

[#Globals](#) [#JSON](#) [#ObjectScript](#) [#SQL](#) [#Valeur clé](#) [#Caché](#) [#Ensemble](#) [#InterSystems IRIS](#)

URL de la
source: <https://fr.community.intersystems.com/post/persistence-des-propri%C3%A9t%C3%A9s-multidimensionnelles-partie-1-classique>