文章
Qiao Peng · 一月 14, 2021 阅读大约需 8 分钟

将 global 映射到类的技巧（第 4/3 部分）

三部曲中的第四部，有人是《银河系漫游指南》的粉丝吗？

如果你希望为旧的 MUMPS 应用程序注入新的生命，请按照以下步骤将 global 映射到类，并将所有这些漂亮的数据公开给 Objects 和 SQL。

如果上述内容听起来很陌生，请阅读以下几篇文章从头开始：[将 global 映射到类的技巧（第 1/3 部分）](https://community.intersystems.com/post/art-mapping-globals-classes-1-3) [将 global 映射到类的技巧（第 2/3 部分）](https://community.intersystems.com/post/art-mapping-globals-classes%C2%A...) [将 global 映射到类的技巧（第 3/3 部分）](https://community.intersystems.com/post/art-mapping-globals-classes-3-3)

这篇文章是献给你的，Joel！ 我们将在上一个示例中定义的父子关系的基础上，创建一个孙子类来处理添加到 ^ParentChild Global 中的新季节信息。

同样的免责声明：如果你在阅读完这些文章后仍然对 global 没有头绪，请联系 WRC，我们将尽力为你提供帮助：Support@InterSystems.com。

将 Global 映射到类的步骤：

1. 1. 确定 global 数据中的重复模式。
2. 2. 确定唯一键的组成。
3. 3. 确定属性及其类型。
4. 4. 在类中定义属性（不要忘记变量下标中的属性）。
5. 5. 定义 IdKey 索引。
6. 6. 定义存储定义：
    a. 定义直到 IdKey（包括 IdKey）的下标。
    b. 定义 “ Data(数据) ” 部分。
    c. 忽略 “ Row ID(行 ID) ” 部分。 99% 的时候，默认值就是你需要的，所以让系统填写。
7. 7. 编译并测试你的类/表。

```
^ParentChild(1)="Brendan^45956"

^ParentChild(1,"Hobbies",1)="Pit Crew"

^ParentChild(1,"Hobbies",1,"Seasons")="Fall*Winter"

^ParentChild(1,"Hobbies",2)="Kayaking"

^ParentChild(1,"Hobbies",2,"Seasons")="Spring\*Summer\*Fall"

^ParentChild(1,"Hobbies",3)="Skiing"

^ParentChild(1,"Hobbies",3,"Seasons")="Summer*Winter"

^ParentChild(2)="Sharon^46647"

^ParentChild(2,"Hobbies",1)="Yoga"
```

^ParentChild(2,"Hobbies",1,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(2,"Hobbies",2)="Scrap booking"

^ParentChild(2,"Hobbies",2,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(3)="Kaitlin^56009"

^ParentChild(3,"Hobbies",1)="Lighting Design"

^ParentChild(3,"Hobbies",1,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(3,"Hobbies",2)="pets"

^ParentChild(3,"Hobbies",2,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(4)="Melissa^56894"

^ParentChild(4,"Hobbies",1)="Marching Band"

^ParentChild(4,"Hobbies",1,"Seasons")="Fall"

^ParentChild(4,"Hobbies",2)="Pep Band"

^ParentChild(4,"Hobbies",2,"Seasons")="Winter"

^ParentChild(4,"Hobbies",3)="Concert Band"

^ParentChild(4,"Hobbies",3,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(5)="Robin^57079"

^ParentChild(5,"Hobbies",1)="Baking"

^ParentChild(5,"Hobbies",1,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(5,"Hobbies",2)="Reading"

^ParentChild(5,"Hobbies",2,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(6)="Kieran^58210"

^ParentChild(6,"Hobbies",1)="SUBA"

^ParentChild(6,"Hobbies",1,"Seasons")="Summer"

^ParentChild(6,"Hobbies",2)="Marching Band"

^ParentChild(6,"Hobbies",2,"Seasons")="Fall"

^ParentChild(6,"Hobbies",3)="Rock Climbing"

^ParentChild(6,"Hobbies",3,"Seasons")="Spring*Summer*Fall"

^ParentChild(6,"Hobbies",4)="Ice Climbing"

^ParentChild(6,"Hobbies",4,"Seasons")="Winter"

步骤 1：

为我们的新类找到重复数据并不是很难，就是 Seasons 子节点。
棘手的部分在于，我不希望 “ Spring*Summer*Fall ” 都在一行，我想要的是三行：“ Spring ”、“ Summer ”、“ Fall ”。

^ParentChild(1)="Brendan^45956"

^ParentChild(1,"Hobbies",1)="Pit Crew"

^ParentChild(1,"Hobbies",1,"Seasons")="Fall*Winter"

^ParentChild(1,"Hobbies",2)="Kayaking"

^ParentChild(1,"Hobbies",2,"Seasons")="Spring*Summer*Fall"

^ParentChild(1,"Hobbies",3)="Skiing"

^ParentChild(1,"Hobbies",3,"Seasons")="Summer*Winter"

每个爱好可以有 1 到 4 个季节。 我想我们可以在 Example3Child 类中再创建 4
个属性，但是如果有人发明了一个新季节，我们该怎么办？
更灵活的解决方案是创建一个孙子表，这样季节的数量可以保持动态。

步骤 2：

我们都知道要查看下标来获取 IdKey 的各个部分，因此我们知道下标 1 和下标 3 将是 IdKey
的一部分，我们还需要一个值来唯一标识不同的季节，但是没有下标了！

我们放在映射中的信息用于生成查询代码。 也许如果我们考虑需要什么 COS
命令来获取此信息，这将帮助我们定义映射。 我们需要做的 3 大件事是：

SET sub1=$ORDER(^Parentchild(sub1))

SET sub2=$ORDER(^Parentchild(sub1, ” Hobbies ” ,sub2))

SET season=$PIECE(^ParentChild(sub1, ” Hobbies ” ,sub2 ” , ” Seasons ”), ” * ” ,PC)

我们可以在映射的 “ 下标 ” 部分中执行相同的操作。 Caché SQL 存储支持 4 种不同类型的下标：Piece、Global、Sub
和 Other。 默认是 Sub，也就是到目前为止我们一直在使用的，它让我们获得了所需的 $ORDER() 循环。

在此示例中，我们将介绍 Piece 选项。 在此级别使用的属性将用作 Piece 计数器（上面示例中的 PC）。
它的默认行为是递增 1，直到到达字符串末尾。

步骤 3：

3 个属性：数据很简单：Season，然后是关系属性：HobbyRef，最后还需要一个子下标：PieceCount

步骤 4：

Property Season As %String; Property PieceCounter As %Integer; Relationship HobbyRef
As Mapping.Example3Child [Cardinality = parent, Inverse = Seasons];

步骤 5：

查看下标映射时，可以看到 3 个变量级别，但是对于 IdKey 索引，我们只引用 2 个属性：HobbyRef 和
PieceCounter

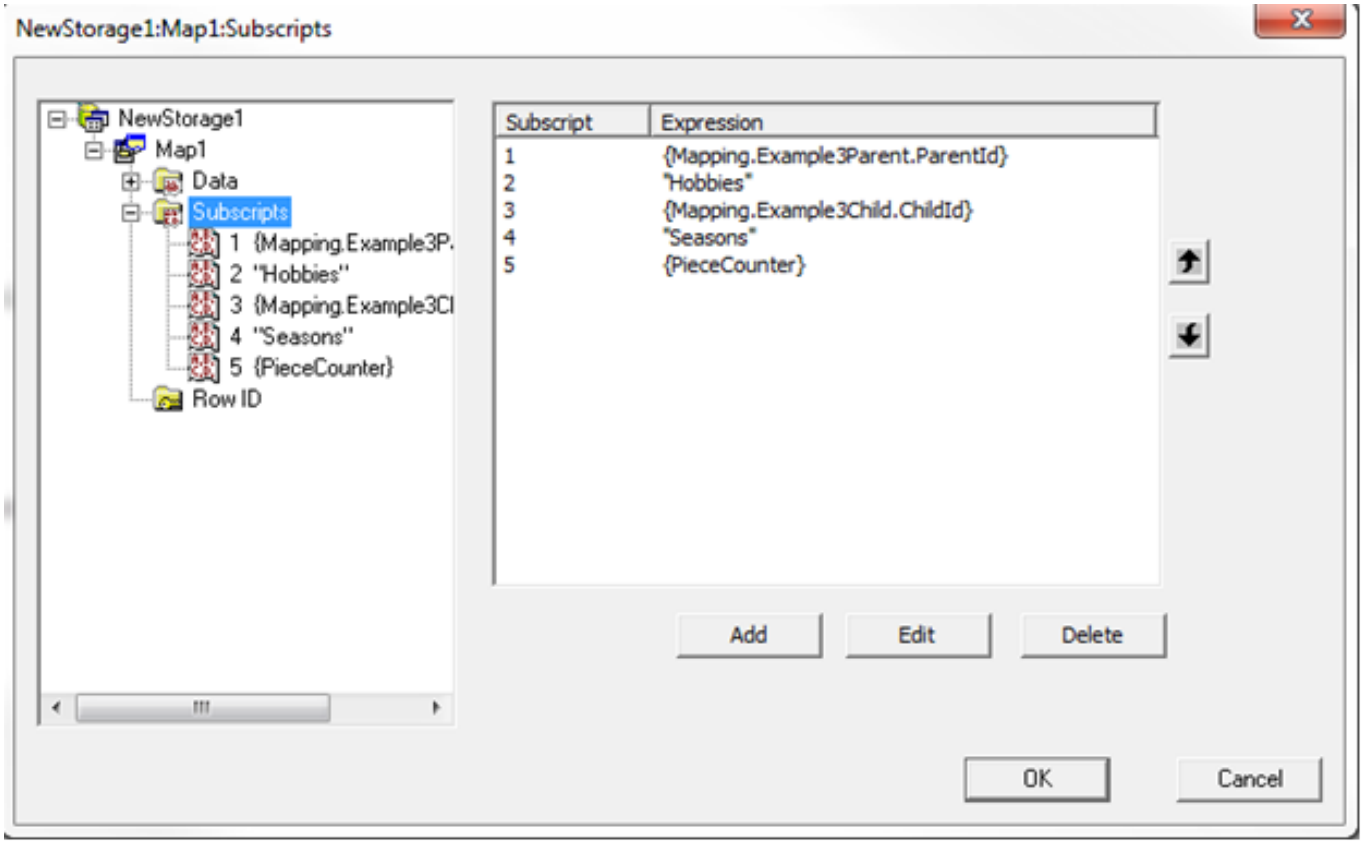
Index Master On (HobbyRef, PieceCounter) [IdKey];

步骤 6：

还是我们一直在讨论的三个部分。 我们仍然忽略行 ID。 这次，我们需要更详细地说明下标，并定义访问类型。
此类将使用 “ Sub ” 和 “ Piece ” 。 如果你想看 “ Global ” 和 “ Other ” 的示例，你需要获得我的示例压缩文件。

步骤 6a：

主下标页看起来与往常一样，只添加了两个级别。
记住，对于父引用的两个部分，我们需要重新引用我们引用的类：{Mapping.Example3Parent} 和 {Mapping.Example3Child}。 当点击窗口左侧的一个下标级别时，就会显示出差异。



在下图中，你可以看到在每个下标级别可以执行的所有不同操作。 对于访问类型 “ Sub ” ，假设你要从 “ ” 开始，然后 $ORDER()，直到到达 “ ” 。 如果不是这种情况，你可以提供一个 “ 开始值 ” 或一个 “ 停止值 ” 。

“ Data Access(数据访问) ” 允许你更改从上一个级别看到的内容，例如，你可以更改要循环的 global。

“ Next Code(后续代码) ” 和 “ Invalid Conditions(无效条件) ” 结合在一起，是该窗口中最常使用的内容。 如果简单的 $ORDER() 不能让你从一个有效值转到下一个有效值，则可以编写你自己的代码让我们使用。

“ Next Code(后续代码) ” 将用于从一个有效值转到下一个有效值，就像 $ORDER() 一样。

“ Invalid Conditions(无效条件) ” 用于测试给定值。 如果你为 “ subscriptX ” 提供了一个值，不需要调用 Next 来找到它，你已经提供了它。 需要的是一些代码，用于确定该值是否有效。

我的长期承诺的示例压缩文件中有许多使用 “ Next Code(后续代码) ” 和 “ Invalid Conditions(无效条件) ” 的类。

“ Access Variables(访问变量) ” 是该页的最后一项内容，很少使用。
基本上，你可以在这里设置变量，在一个下标级别为其赋一个值，然后在更高的下标级别中使用它。
生成的表代码将为你处理范围。

对于下标级别 5，“ Access Type(访问类型) ”为“ Piece ”，“ Delimiter(分隔符) ”为“ * ”。 生成的代码将从 Piece 1 开始，然后递增 1，直到用完 $PIECE 值为止。 重申一次，我们可以提供“ Start Values(开始值) ”或“ Stop Values(停止值) ”来对此进行控制。

步骤 6b：

数据部分只是一个属性；对于“ Piece ”或“ Delimiter(分隔符) ”来说不需要。
如果此表有更多字段，我们很可能需要提供“ Piece ”和“ Delimiter(分隔符) ”，那样也没问题。

步骤 6c： 仍然将此处留空。

步骤 7：

所有代码都编译得干净漂亮。

```
Compilation started on 11/30/2016 08:17:42 with qualifiers 'uk/importselectivity=1 /checkuptodate=expandedonly'
Compiling 2 classes, using 2 worker jobs
Compiling class Mapping.Example3Child
Compiling class Mapping.Example3GrandChild
Compiling table Mapping.Example3GrandChild
Compiling table Mapping.Example3Child
Compiling routine Mapping.Example3Child.1
Compiling routine Mapping.Example3GrandChild.1
Compilation finished successfully in 1.021s.
```

三个表的连接：

```
SELECT P.ID, P.Name, P.DateOfBirth,

C.ID, C.Hobby, G.ID, G.Season

FROM Mapping.Example3Parent P

JOIN Mapping.Example3Child C ON P.ID = C.ParentRef

JOIN Mapping.Example3Grandchild G ON C.ID = G.HobbyRef

WHERE P.Name = 'Kieran'
```

得出：

ID	姓名	出生日期	ID	爱好	ID	季节
6	Kieran	05/16/2000	6||1	SUBA	6||1||1	Summer
6	Kieran	05/16/2000	6||2	Marching Band	6||2||1	Fall
6	Kieran	05/16/2000	6||3	Rock Climbing	6||3||1	Spring

6	Kieran	05/16/2000	6||3	Rock Climbing	6||3||2	Summer
6	Kieran	05/16/2000	6||3	Rock Climbing	6||3||3	Fall
6	Kieran	05/16/2000	6||4	Ice Climbing	6||4||1	Winter

记住，我说过子表 IdKey 总是由两部分组成：父引用和子下标。

在第一行中，Example3Child ID 为 6||1：父引用 = 6，子下标 = 1。

对于 Example3GrandChild，IdKey 由三部分组成：6||1||1，但仍然是父引用和子下标。
父引用只是更复杂一些：父引用 = 6||1，子下标 = 1。

在 Example3Child 中，下标中的属性数与 IdKey 中的属性数相匹配。 在父子结构中嵌套更深入时，IdKey 会变成复合形式，并且下标的数量将增加。

这是此示例中使用的 3 个类的导出：MappingExample4.zip。

#映射 #对象数据模型 #SQL #Globals #Caché

源 URL: https://cn.community.intersystems.com/post/%E5%B0%86-global-%E6%98%A0%E5%B0%84%E5%88%B0%E7%B1%BB%E7%9A%84%E6%8A%80%E5%B7%A7%EF%BC%88%E7%AC%AC-43-%E9%83%A8%E5%88%86%EF%BC%89