記事
Toshihiko Minamoto · 2021年2月18日 9m read

グローバルをクラスにマッピングする技術 (2/3)

古くなった MUMPS アプリケーションの新たな生命を吹き込みたいとお考えでしたら、以下にご紹介するステップを実行すれば、グローバルをクラスにマッピングし、美しいデータを Object や SQL に公開できます。

今回ご紹介する例には、パート 1 ではカバーしなかった内容を 4 つないし 5 つ程度盛り込んでいます。

その後は親子マッピングの例を紹介して完結となります。それを修得したらマッピングはもう完璧でしょう。

前回と同じ免責事項: これらの記事を読んでもグローバルがよく理解できないという方は、WRC (Support@InterSystems.com) までメールでお問い合わせください。喜んでサポートさせていただきます。

グローバルをクラスにマッピングするステップ。

1. グローバルデータが繰り返し使用されるパターンを特定する。
2. 固有キーの構成を特定する。
3. プロパティとそれぞれの型を特定する。
4. クラス内のプロパティを定義する (変数の添え字をお忘れなく)。
5. IdKey のインデックスを定義する。
6. Storage Definition を以下の手順で定義する。

   a. 添え字を IdKey まで (IdKey を含む) 定義する。
   b. Data セクションを定義する。
   c. Row ID セクションには触れない。 デフォルトが 99% の割合で適切なので、これはシステムに任せます。

7. クラス / テーブルをコンパイルし、テストします。

以下のようなグローバルが 2 種類 (mapping と index) あるとします。

^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)=""

そうです、グローバルマッピングについて学ぶと同時に、私の家族についても知っていただく必要があります。
創造力が足らなくて、これ以外のデータは思いつきませんでした。

ステップ 1:

今回は、繰り返しデータが、1 つではなく、2 つのグローバルノードに広がっています。

ステップ 2:

データがこれしかないので、確信は持てませんが、 1 つ目の添え字、もしくは 1 つ目と 2
つ目の添え字の両方が定数だと思います。 今回の例では、2 つ目の添え字が変数 FamilyId、3 つ目が PersonId
であると想定します。

ステップ 3:

変数の添え字は FamilyId と PersonId の 2 つ、さらに Name、Relation、Activity があります。
ヒントを求めてインデックスのグローバルをもう一度確認した結果、Name を 2 つのプロパティ FirstName と
LastName にマッピングすることにしました。 これで、定義する必要のあるプロパティが全部で 6
個になりました。

ステップ 4:

ここで新しいものが 2 つ登場します。 すべてのプロパティに SQL Field Name があり、FirstName の照合が
UPPER になっています。 SQL Field Names
を定義しておけば、プロパティの話なのか、SQLフィールドの話なのかがはっきりします。
照合については、後ほどインデックスのマッピングと一緒に説明します。

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];

ステップ 5:

変数の添え字が 2 つあるため、IdKey は 2 つのプロパティに基づくことを意味します。

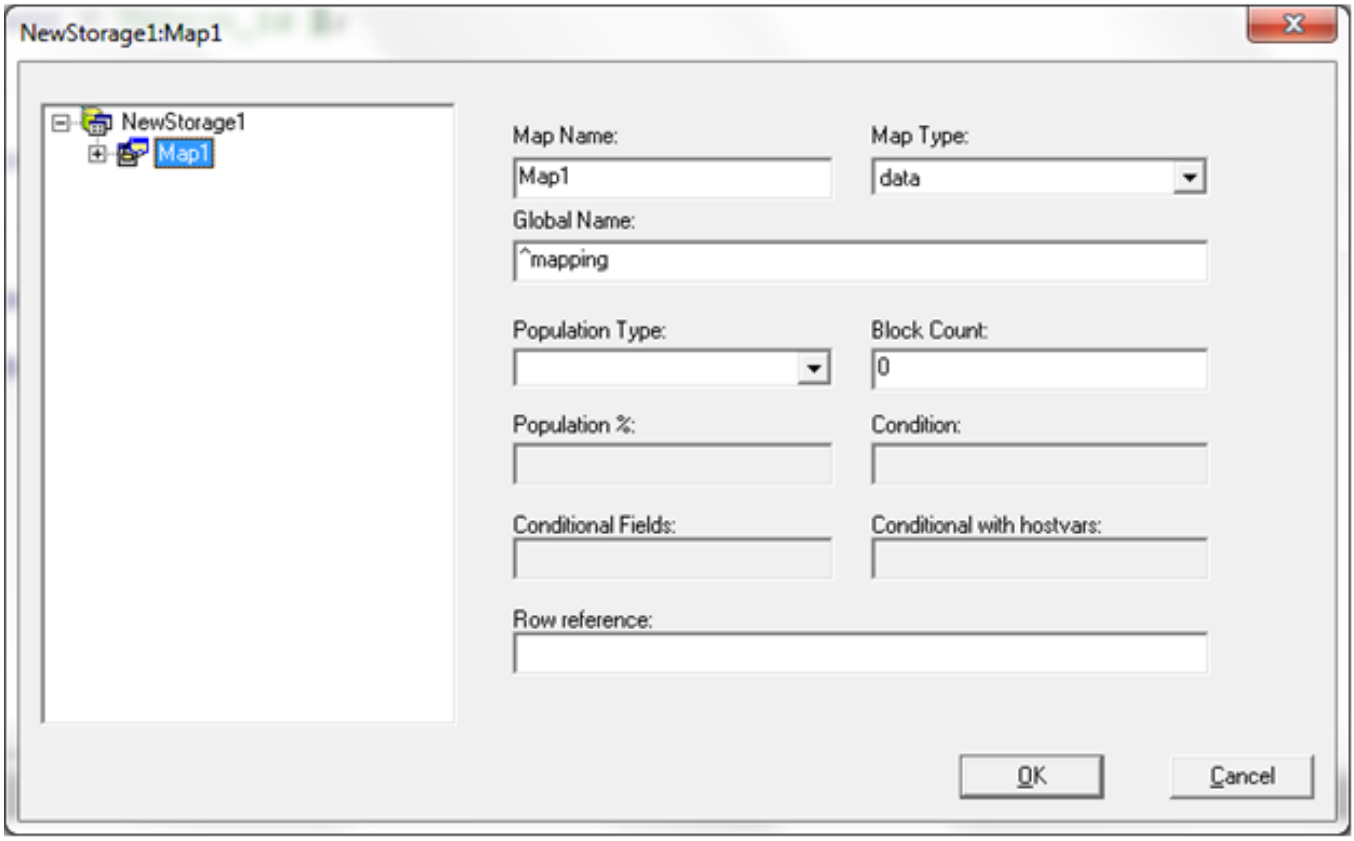
Index Master On (FamilyId, PersonId)[IdKey];

ここでも、1 つのインデックスが 1 つのプロパティ FirstName に対して定義されています。

Index FNameIndex On FirstName;

ステップ 6:

まず最初に Storage Definition を作成します。 Storage アイコンをクリックするか、Inspector を使って、Storage を選択し、New Storage を右クリックします。 今回は、Map Name をデフォルト値のままにして、Global Name と ^mapping だけを入力しました。



最初の例では、Storage Definition
を作成してからウィザードに戻る方法を説明していなかったので、ここでご紹介いたします。
ウィザードに戻るには、Inspector から Storage を選択した後に、Storage Name (今回は NewStorage1)
を選択し、SQL Storage Map と右側の端の方にあるボックスを順にクリックします。

ステップ 6a:

もう一度 Subscripts セクションから始めます。 まず最初に、IdKey
以下のすべてを定義するために添え字のレベルが 3 つ (定数が 1 つ、フィールドが 2 つ) 必要になります。
このウィンドウに表示されるフィールドは、IdKey
のインデックスに定義されているプロパティと一致している必要があります。

注意: Storage Definition では、常に、プロパティ名 FamilyId ではなく、SQL のフィールド名 FamilyId
を参照しています。

ステップ 6b:

Data セクションでは、1 つのグローバルノードにフィールドが 3 つ、そして添え字の下位レベル (“ Activity ”) にはフィールドが 1 つ格納されています。 ファーストネームとラストネームを 2 つの個別のフィールドとして取得するには、入れ子になった $PIECE() に対応する Piece と Delimiter を複数個使う必要があります。 これは、ObjectScript では「set FirstName=$PIECE($PIECE(^mapping(“ Less Simple ” ,1,1), ” ^ ” ,1), ” , ” ,2)」のように記述されます。 ウィザードでは、最も内側にある $PIECE() からすべての Delimiter と Piece を順にリストアップする必要があります。 Activity は別のグローバルノードにあるため、Node には定数を追加します。 Piece と Delimiter の既定値は、それぞれ 1 と “ ^ ” なので、そのままにしています。

ステップ 6c:

ここはまだお見せできるものがありません。

Row ID
セクションを定義する必要があるケースの例を見たくてしょうがない、という皆さまのお気持はよく分かります。後ほどすべての例を取り込んだ zip ファイルをお見せするときに、Mapping.RowIdSpec という名前の例を入れておきますので、それを見てご確認ください。

それでは、インデックスマップに対してステップ 6 を実行します。

ステップ 6

Map2 を作成し、Global Name を「^index」に設定します。

ステップ 6a

インデックスグローバルには、定数が 2 個、IdKey フィールドが 2 個、インデックス付きフィールド FirstName が 1 つ、と添え字は全部で 5 つあります。 ^mapping グローバルと ^index グローバルを見ると、FirstName の値が違うことが分かります。 ^mapping グローバルでは、名前に大文字と小文字が混ざっている一方で、^index グローバルの名前は大文字だけが使われています。 Caché には、文字列データの変換に使えるコレーション機能がいくつか備え付けられています。
使用できる各機能については、こちらのリンクをご覧ください。

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

Caché SQL Storage では、コレーションのレガシータイプが必要になる可能性が強いです。

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

ここで使用する照合機能がプロパティ定義の照合と一致するものであることは極めて重要です。 インデックスマップがクエリオプティマイザーに使用されない最も一般的な理由は、これらが一致しないということがあるためです。 %String のデフォルトの照合は SQLUPPER となっていますが、マッピングで照合機能を提供しないことは、EXACT 照合を使うのと同じであることを覚えておきましょう。

ステップ 6b

^index グローバルにはデータがないため、ここで定義するものはありません。

ステップ 6c

もうお分かりでしょう、ここもなしです。

ステップ 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

今回、Storage Definition はスキップしておきます。 確認したい方は、mapping.example2.zip
をダウンロードして、XML ファイルを読み込んでください。

#マッピング #SQL #グローバル #Caché

ソースURL:
https://jp.community.intersystems.com/post/%E3%82%B0%E3%83%AD%E3%83%BC%E3%83%90%E3%83%AB%E3%82%92%E3%82%AF%E3%83%A9%E3%82%B9%E3%81%AB%E3%83%9E%E3%83%83%E3%83%94%E3%83%B3%E3%82%B0%E3%81%99%E3%82%8B%E6%8A%80%E8%A1%93%C2%A023