Article
Minoru Horita · Jun 3, 2020 10m read

グローバルはデータを保存するための魔法の剣です パート3 - 疎な配列



前のパート（1、2）では、ツリーとしてのグローバルを話題に取り上げました。
この記事では、それらを疎な配列と見なします。

疎な配列は、ほとんどの値が同一であると想定される配列の種類です。

疎な配列は実際には非常に大きいため、同一の要素でメモリを占有することには意味がありません。 したがって、疎な配列を整理し、重複した値の格納にメモリが浪費されないようにすることには意味があります。

疎な配列は、J、MATLABなど一部のプログラミング言語では言語の一部になっています。
他の言語では、疎な配列を使用できるようにする特別なライブラリが存在します。
C++の場合は、Eigenなどがあります。

次の理由により、グローバルは疎な配列を実装するのに適した候補であると言えます。

1. 特定のノード値のみを保存し、未定義のノード値を保存しないこと。

2. ノード値のアクセスインターフェースが、多くのプログラミング言語が多次元配列の要素にアクセスするために提供しているものとよく似ていること。

```
Set ^a(1, 2, 3)=5
Write ^a(1, 2, 3)
```

3. グローバルはデータを格納するためにかなり低レベルの構造を採用しているため、優れたパフォーマンス

[1]をご覧ください）。

グローバルは永続的な構造であるため、グローバル用に十分なメモリを確保できることが事前に分かっている場合のみ、グローバルに基づいて疎な配列を作成する意味があること。

疎な配列の実装には、未定義の要素を処理する場合にデフォルトで特定の値を返すようにするという意味合いもあること。

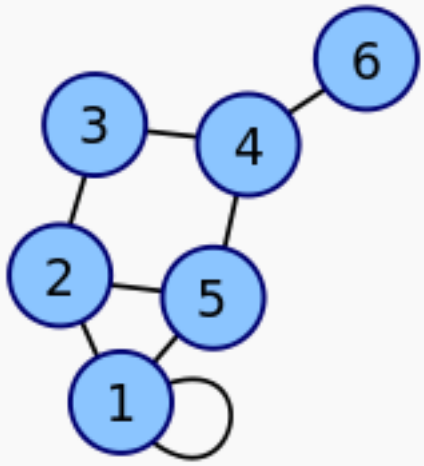
これは、COSの$GET関数を使用して実装できます。 この例では、次のような3次元配列を見てみましょう。

```
SET a = $GET(^a(x,y,z), defValue)
```

では、どのような種類のタスクで疎な配列が必要になり、どのようにグローバルは役立つのでしょうか？

隣接行列

このような行列はグラフを表すために使用されています。

Graph	Adjacency matrix
	$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

グラフが大きいほど、行列に含まれるゼロが多くなることは明らかです。 例えば社会的ネットワークのグラフをこの種の行列で表すと、ほとんどがゼロで構成されることになります。つまり、疎な配列になります。

```
Set ^m(id1, id2) = 1
Set ^m(id1, id3) = 1
Set ^m(id1, id4) = 1
Set ^m(id1) = 3
Set ^m(id2, id4) = 1
Set ^m(id2, id5) = 1
Set ^m(id2) = 2
....
```

この例では隣接行列と各ノードのエッジ数（誰とつながっているか、およびつながりの数）を ^m
グローバルに保存します。

グラフの要素数が2,900万を超えない場合（この数は、8
* 最大文字列長
で計算されます）、このような行列を格納するには、ビット文字列を使うのがより経済的です。ビット文字列は大きなギャップを特別な方法で最適化するからです。

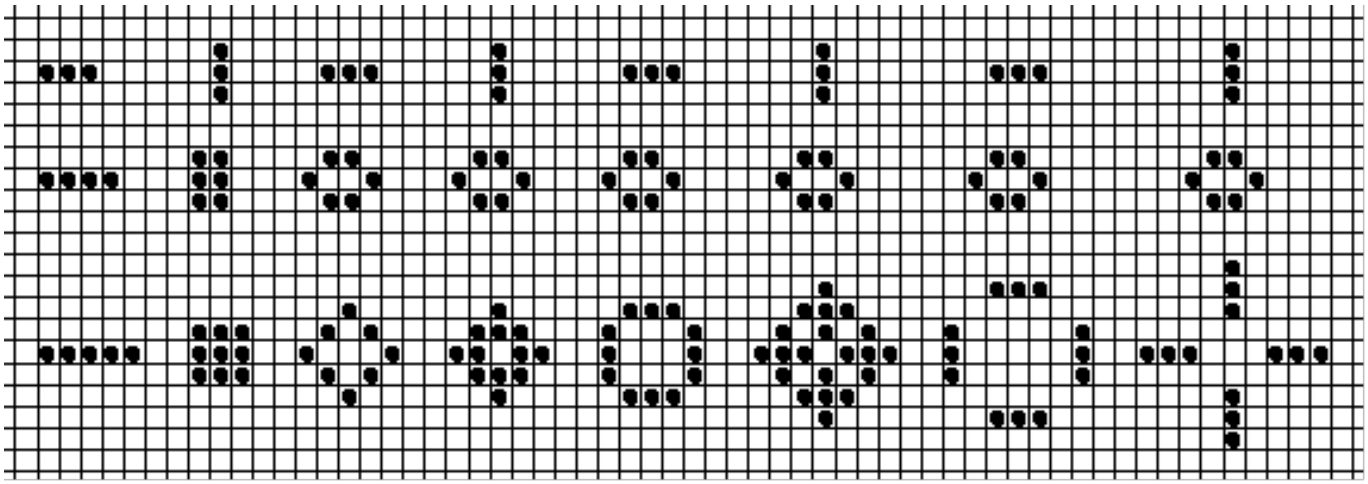
ビット文字列の操作は、$BIT 関数を使用して実行されます。

```
; ??????
SET $BIT(rowID, positionID) = 1
; ??????
Write $BIT(rowID, positionID)
```

FSMスイッチのテーブル

FSMスイッチのグラフは通常のグラフであるため、FSMスイッチのテーブルは基本的に上述したのと同じ隣接行列です。

セル・オートマトン



最も有名なセル・オートマトンである「ライフ」ゲーム
では、そのルール（セルに多数の隣接セルがある場合、セルが死ぬ）によって本質的に疎な配列が形成されます。

スティーブン・ウルフラム氏は、セル・オートマトンを新しい科学分野であると考えています。 スティーブン氏は2002年に「新しい種類の科学」と呼ばれる1,280ページの本を出版しました。同著には、セル・オートマトンの分野での成果は分離されていないものの、非常に安定しており、すべての科学分野にとって重要であることが記されています。

コンピューターで処理できるアルゴリズムがセル・オートマトンを使用して実装できることも証明されています。セル・オートマトンは動的な環境やシステムのシミュレーション、アルゴリズムの問題の解決、その他の目的に使用されています。

巨大なフィールドがあり、セル・オートマトンのすべての中間状態を登録する必要がある場合、グローバルの使用は合理的であると言えます。

地図の作成

疎な配列の使用に関して最初に思い浮かぶのは、地図の作成です。

一般的に、地図には何も無い空間がたくさんあります。 世界地図が大きなピクセルで構成されていると想定した場合、地球上の全ピクセルの71%が海を表す疎な配列で占められていることになります。
また、地図に人工的な構造物を追加するだけの場合、95%超は何も無い空間になります。

もちろん、地図をビットマップ配列として保存する人などいません。誰もが代わりにベクトル表現を使用しています。[文字列の折り返しの区切り]しかし、ベクトル地図とは何でしょうか？ これは、ポリラインとポリゴンを備えたある種の構造物です。[文字列の折り返しの区切り]本質的には、ポイントとこれらの関係を記録したデータベー

スです。

地図作成で最も困難な作業の1つには、ガイア宇宙望遠鏡が実行している銀河地図の作成が挙げられます。
例えて言えば、銀河は1つの巨大な疎な配列になっています。 その99,999999.....%は完全に空の空間です。
銀河地図の保存には、グローバルに基づくデータベースであるCacheが選ばれました。

このプロジェクトにおけるグローバルの正確な構造は分かりませんが、おそらく次のようなものであると想定できます。

```
Set ^galaxy(b, l, d) = 1; ??????????????
Set ^galaxy(b, l, d, "name") = "??"
Set ^galaxy(b, l, d, "type") = "??" ; ????????????????????????????????????
Set ^galaxy(b, l, d, "weight") = 14E50
Set ^galaxy(b, l, d, "planetes") = 7
Set ^galaxy(b, l, d, "planetes", 1) = "??"
Set ^galaxy(b, l, d, "planetes", 1, weight) = 1E20
```

...

b、l、dは銀河座標であり、それぞれ緯度、経度、太陽からの距離を表しています。

グローバルベースのデータベースはスキーマレスであるため、グローバルの柔軟な構造を活かして星と惑星の特性を保存することができます。

Cacheは柔軟性に優れているというだけでなく、一連のデータを素早く保存しながら同時にインデックスグローバルを作成できるため、高速な検索を実行できるということが理由で選ばれました。

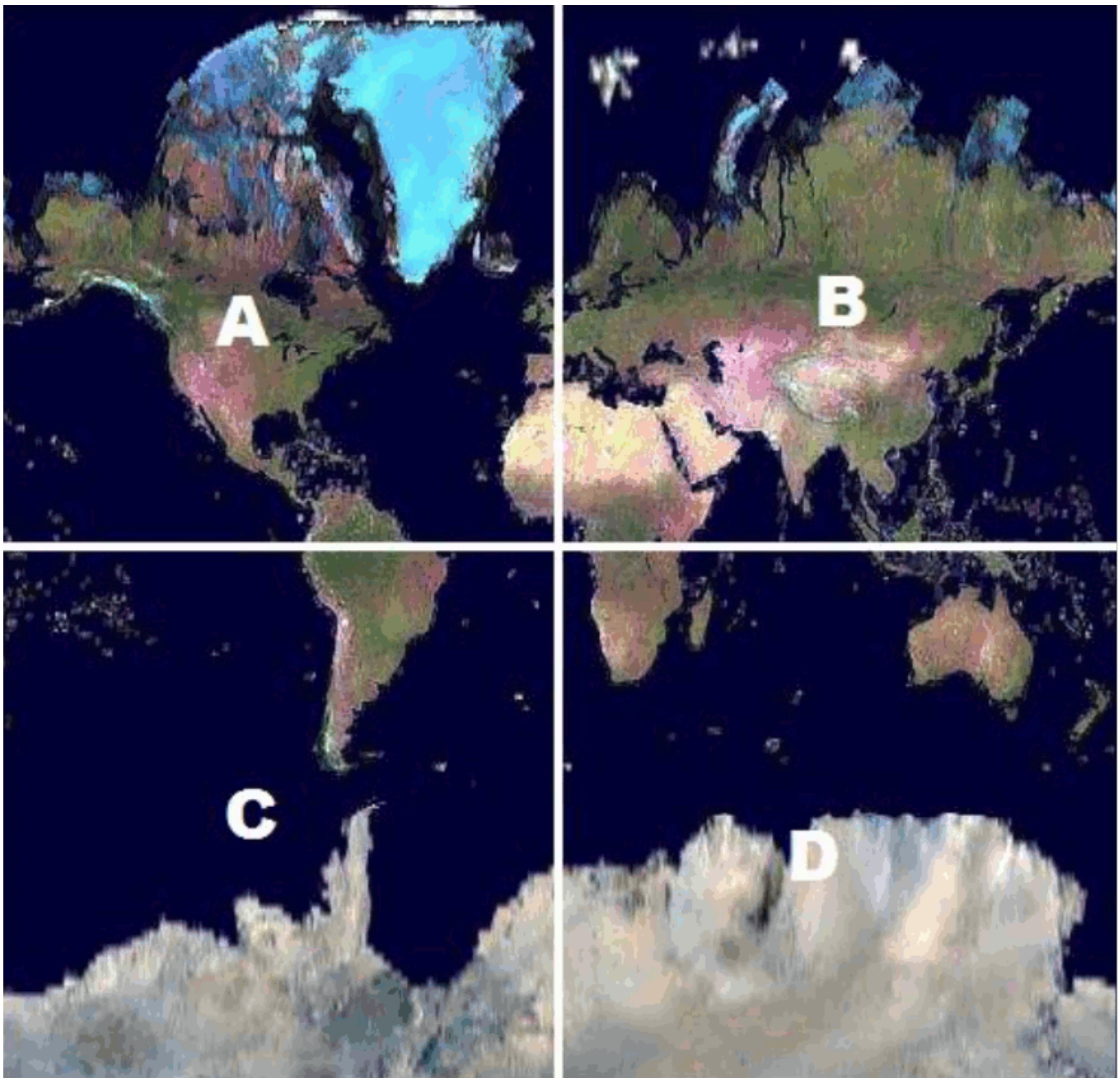
地球に話を戻すと、グローバルはこのような地図に特化したプロジェクトであるOpenStreetMap XAPI
やOpenStreetMapのフォークであるFOSMで使用されていました。

つい最近のCachéハッカソンでは
、ある開発者のグループがこの技術を使用して地理空間インデックスを実装していました。
詳細については、こちらの記事をご覧ください。

OpenStreetMap XAPIでグローバルを使用した地理空間インデックスの実装

イラストはこちらのプレゼン資料から引用しています。

地球全体を正方形に分割し、それをさらに小さな正方形に再帰的に分割していきます。
最終的には、グローバルを作成するための階層構造を取得します。



これで、いつでも素早く任意の正方形を要求したり、空にしたりできます。その場合はその正方形の子孫となるすべての正方形も返されるか、空になります。

グローバルに基づく詳細なスキームは、以下のようにいくつかの方法で実装できます。

方法1:

```
Set ^m(a, b, a, c, d, a, b,c, d, a, b, a, c, d, a, b,c, d, a, 1) = idPointOne
Set ^m(a, b, a, c, d, a, b,c, d, a, b, a, c, d, a, b,c, d, a, 2) = idPointTwo
...
```

方法2:

```
Set ^m('abacdabcdabacdabcda', 1) = idPointOne
```

```
Set ^m('abacdabcdabacdabcda', 2) = idPointTwo
...
```

どちらの場合も、COS/Mで任意のレベルの正方形にあるポイントを要求するのはそれほど面倒ではありません。最初の方法では、任意のレベルで正方形の空間断片を消去するのが多少簡単になりますが、この処理が必要になることはほとんどありません。

下位レベルの正方形の例:



そして、こちらはXAPIプロジェクトで使用されているグローバルの例です。グローバルに基づいてインデックスが表現されています。



^way
グローバルは、ポリライン
（道路、小川など）やポリゴン（建物や森林などの閉じられた空間）の頂点を格納するために使用されています。

グローバルでの疎な配列の使用方法の大まかな分類

1. 一部のオブジェクトの座標とその状態（地図作成、セル・オートマトン）を格納します。
2. 疎行列を格納します。

方法2）では特定の座標が要求され、要素に値が割り当てられていない場合、疎な配列の要素のデフォルト値を取得する必要があります。

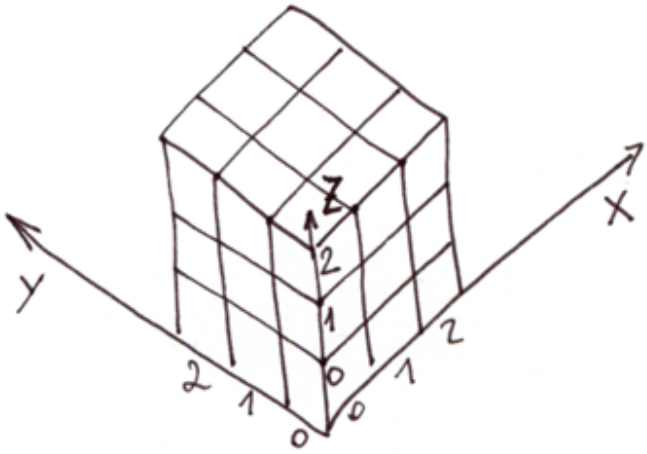
グローバルに多次元行列を格納するメリット

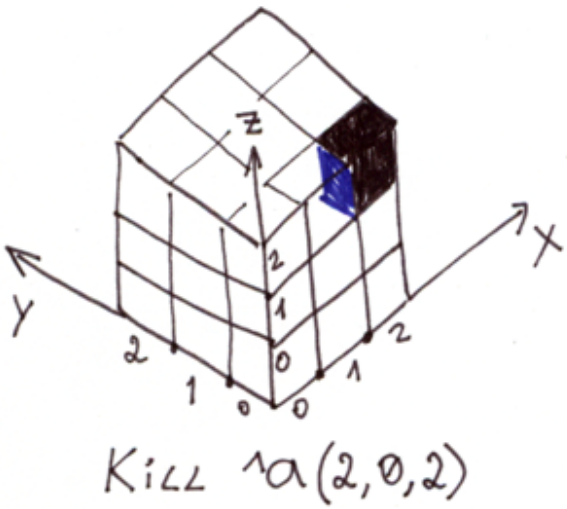
文字列、面、立方体などの複数の空間断片を素早く削除または選択できます。整数インデックスの場合、文字列、

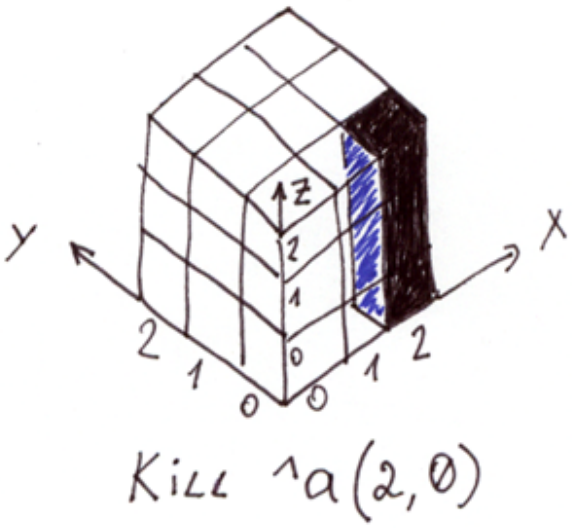
面、立方体などの複数の空間断片を素早く削除または選択できると便利です。

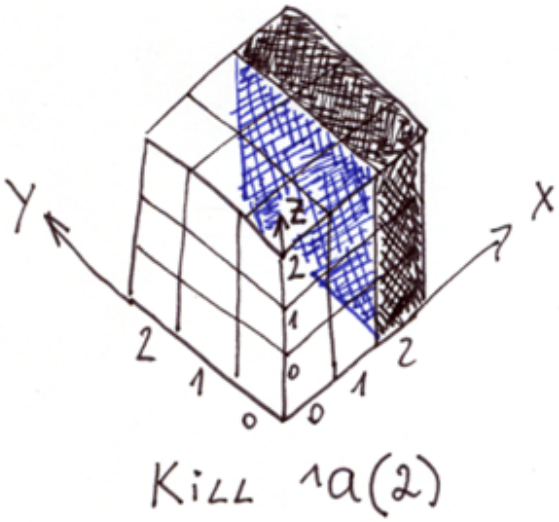
Killコマンドは、単独の要素、文字列、さらには面全体を削除できます。
グローバルにはプロパティがあるため、要素ごとに削除するよりも1000倍高速に削除できます。

この図は、グローバル ^a の3次元配列とさまざまな削除処理を表現しています。









既知のインデックスで空間断片を選択するには、Merge コマンドを使用できます。

行列の列をColumn変数に抜き出します。

```
; 3x3x3???3??????????
Set ^a(0,0,0)=1,^a(2,2,0)=1,^a(2,0,1)=1,^a(0,2,1)=1,^a(2,2,2)=1,^a(2,1,2)=1
Merge Column = ^a(2,2)
; Column????????
Zwrite Column
```

出力:

```
Column(0)=1

Column(2)=1
```

興味深いことに、$GET
経由でアドレスを指定できるColumn変数に疎な配列が含まれています。これは、デフォルト値が格納されていないためです。

$Order 関数を使用する小さなプログラムを使用して空間断片を選択することもできます。
これは、量子化されていないインデックスがある空間で特に便利です（地図の作成）。

まとめ

今日の現実は、新しい課題を提起しています。
グラフは数十
億の頂点で構成でき、地図
は数十億のポイントで構成できます。セル・オートマ
トン（1、2）に基づいて独自の世界を作りたいと考えている人もいるかもしれません。

疎な配列のデータ量はRAMに収まる大きさには圧縮できませんが、それでも疎な配列を処理する必要がある場合は、グローバルとCOSを使用してそのようなプロジェクトを実装することを検討する必要があります。

最後までお読みいただき、ありがとうございました！ コメント欄で質問やリクエストをお待ちしています。

> 免責事項：この記事と記事に対する筆者(英語原文はSergey Kamenev氏によるものです)のコメントは単なる筆者の私見であり、InterSystemsの公式見解とは関係ありません。

また、前のパート「グローバルはデータを保存するための魔法の剣です パート2 - ツリー」も確認してください。

#Key Value #Indexing #Globals #Data Model #Performance #Relational Tables #Beginner #Caché #InterSystems IRIS

Source URL:https://community.intersystems.com/node/477586