文章
Weiwei Gu · 六月 27, 2022 阅读大约需 6 分钟

Globals 是管理数据的魔剑 ： 第一部分

Globals，这些存储数据的魔剑，已经存在了一段时间，但是没有多少人能够有效地使用它们，也没有多少人知道这个超级武器。

如果你把Globals的东西用在它们真正能发挥作用的地方，其结果可能是惊人的，要么是性能的提高，要么是整体解决方案的大幅简化 (1, 2).

Globals提供了一种特殊的存储和处理数据的方式，它与SQL表完全不同。它们在1966年首次出现在 M(UMPS) 编程语言中， 该语言最初用于医学数据库。现在它仍然以同样的方式被使用
，但也被其他一些以可靠性和高性能为首要任务的行业所采用：金融、交易等。

后来M(UMPS)演变为 Caché ObjectScript (COS). COS是由InterSystems公司开发的，作为M的一个超集.
其原始语言仍然被开发者社区所接受，并在一些实现中保持活力。在网络上有几个活跃的网址，比如：MUMPS Google group, Mumps User's group), effective ISO Standard等等

现代基于Globals的数据库支持交易、日志、复制、分区等。这意味着它们可以被用来构建现代的、可靠的、快速的分布式系统。

Gloabls并不将你限制于关系模型的范围内。它们让你可以自由地创建为特定任务优化的数据结构。对于许多应用来说，合理地使用好的Globals就如一颗真正的银子弹头，它所提供的速度是传统关系型应用的开发者所梦寐以求的。

作为一种存储数据的方法，globals可以在许多现代编程语言中使用，包括高级和低级语言。因此，本文将特别关注Globals本身，而不是它们曾经来自的语言。

Globals 是如何工作的

让我们先了解一下globals是如何工作的，它们有哪些优点。

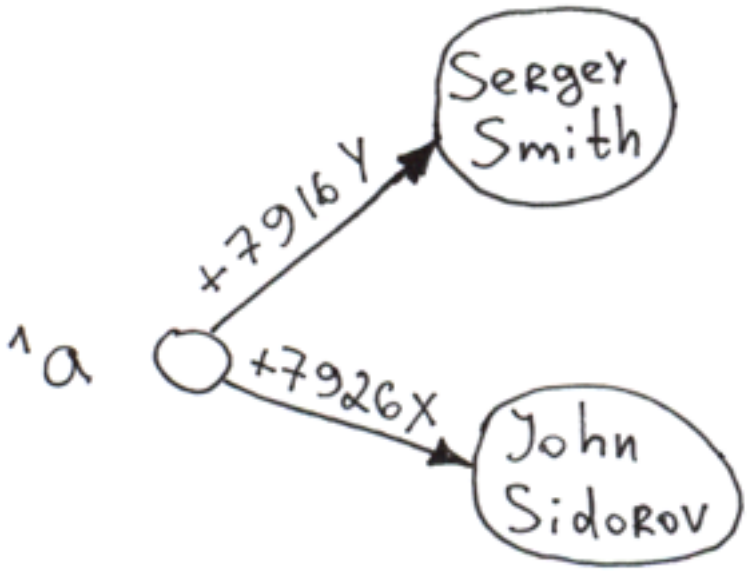
我们可以从不同的角度来看待globals。在文章的这一部分，我们可将把它们看成是树形状结构或分层的数据存储空间。

简单地说，Globals是一个持久化的数组。一个自动保存在磁盘上的数组。

很难想象有什么比这更简单的方法来存储数据。在程序代码中（用COS/M语言编写），与普通关联数组的唯一区别是站在它们名字前面的^符号。

若将数据保存为Globals, 你可以不需要知道SQL，因为所有必要的命令都非常简单，在一个小时内就可以学会。

让我们从最简单的例子开始，一个有两个分支的单层的树形结构。例子是用COS(Caché ObjectScript) 写的。




```
Set ^a("+7926X") = "John Sidorov"
Set ^a("+7916Y") = "Sergey Smith"
```

当数据被插入一个Global（Set命令）时，有3件事情会自动发生:

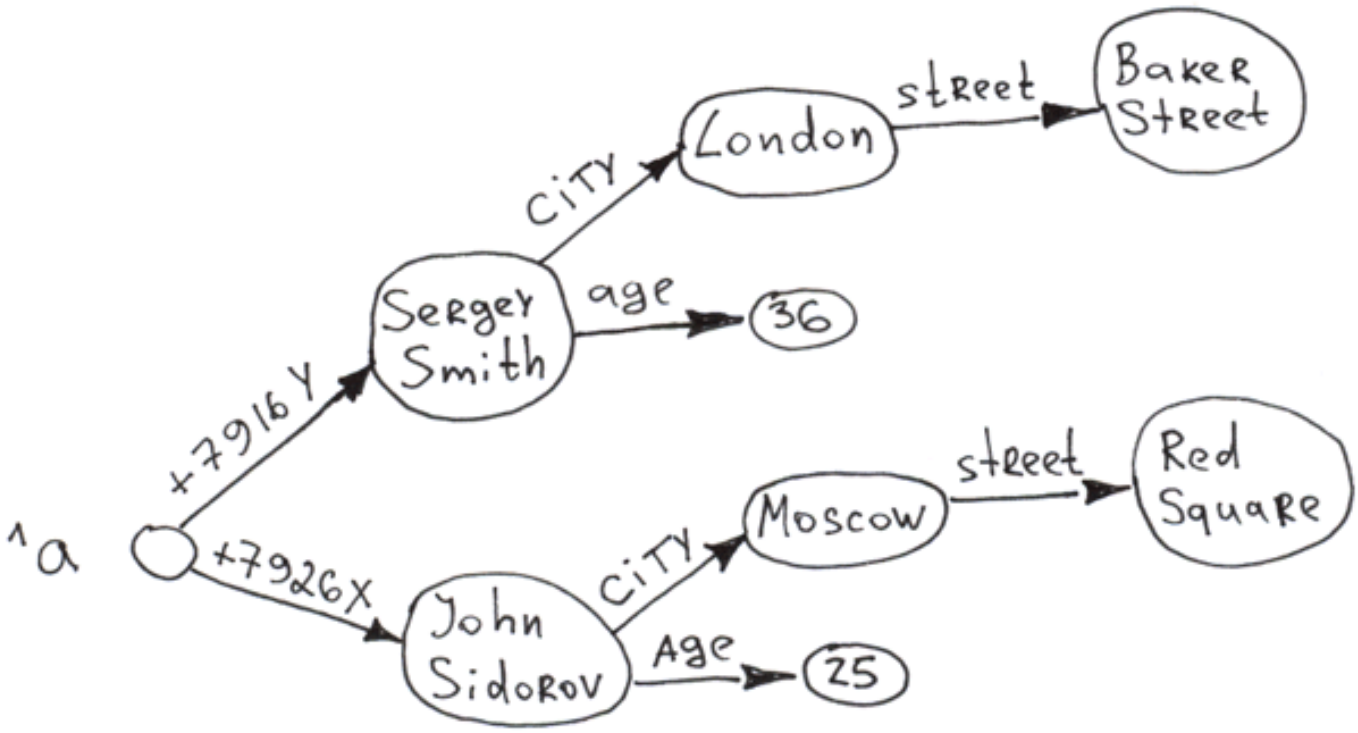
1.将数据保存到磁盘。
2.编制索引。
括号里的是下标，等号右边的是a节点值。
3. 排序。
数据是按一个键来排序的。下一次的遍历会把 "Sergey Smith "放到第一个位置，然后是 "John Sidorov"。当从global获得一个用户列表时，数据库不会在排序上花费时间。实际上你可以请求一个从任何键开始的排序列表，甚至是一个不存在的键（输出将从这个键之后的第一个真正的键开始）。
所有这些操作都以惊人的速度进行。在我的个人系统（i5-3340，16GB，HDD WD 1TB Blue）上，我设法在一个进程中达到105万次插入/秒。在多核系统上，速度可以达到几千万次/秒 的插入。

当然，记录插入速度本身并不能说明什么。例如，我们可以将数据写入文本文件--根据传言，这就是Visa的处理方式。然而，通过globals，我们得到了一个结构化和索引化的存储，你可以在工作中享受其高速和易用性。

- **globals最大的优势是在其中插入新节点的速度。**
- **数据在全局中总是有索引的。单层和深入的树形遍历总是非常快的。**

让我们在global中添加一些二级和三级的分支看看：

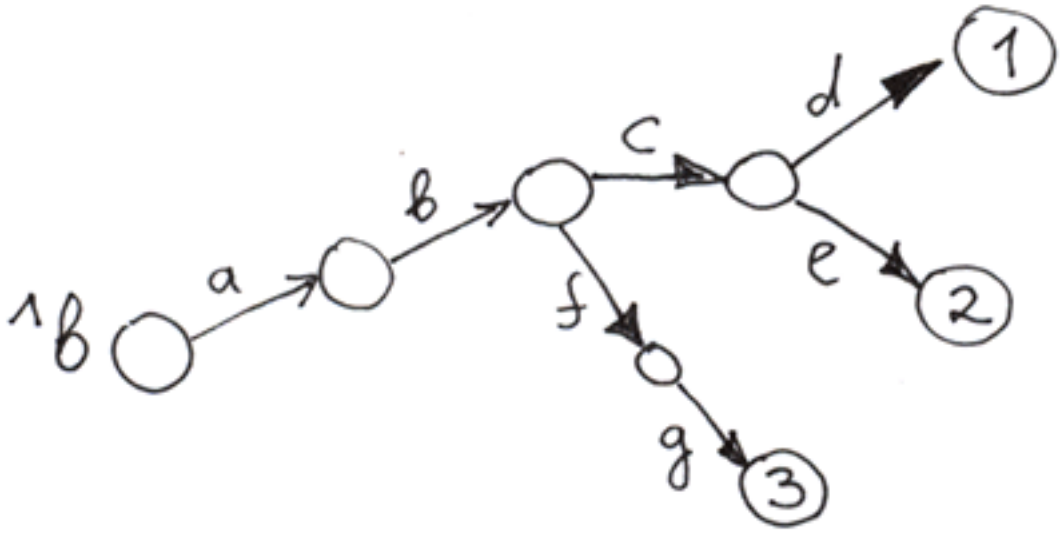
```
Set ^a("+7926X", "city") = "Moscow"
Set ^a("+7926X", "city", "street") = "Req Square"
Set ^a("+7926X", "age") = 25
Set ^a("+7916Y", "city") = "London"
Set ^a("+7916Y", "city", "street") = "Baker Street"
Set ^a("+7916Y", "age") = 36
```



显然，你可以使用globals建立多层级的 “ 树 ” 。由于每次插入后的自动索引，因而其对任何节点的访问几乎都是即时的。任何一级的树枝都可以按一个键进行排序。

正如你所看到的，数据可以被存储在键和值中。一个键的综合长度（所有索引的长度之和）可以达到511字节，而Caché中的值可以达到3.6MB的大小。树中的层数（维数）上限为31。

还有一件很酷的事情：你可以在不定义顶级节点的值的情况下建立一棵 “ 树 ” 。

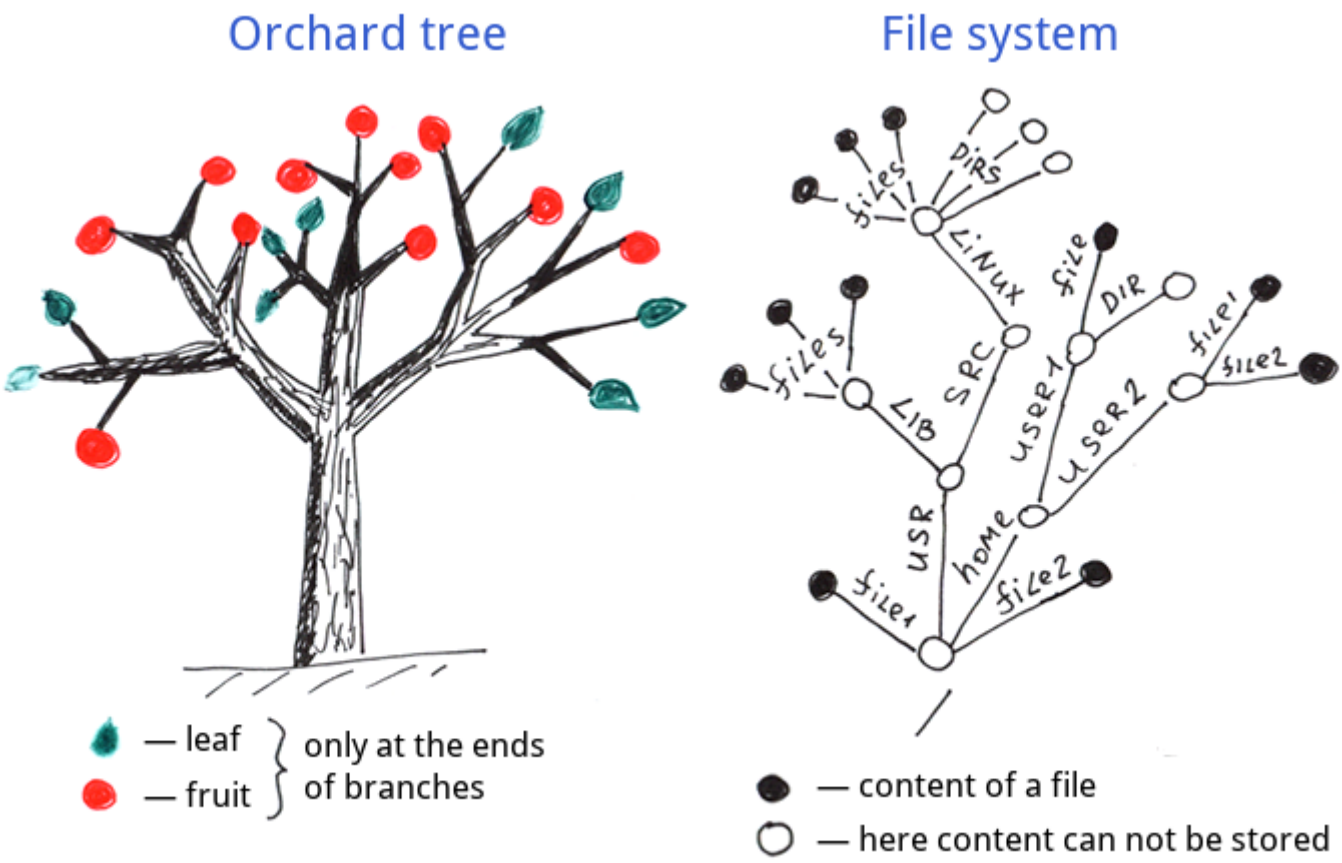



```
Set ^b("a", "b", "c", "d") = 1
Set ^b("a", "b", "c", "e") = 2
Set ^b("a", "b", "f", "g") = 3
```

空的圆圈是没有值的节点。

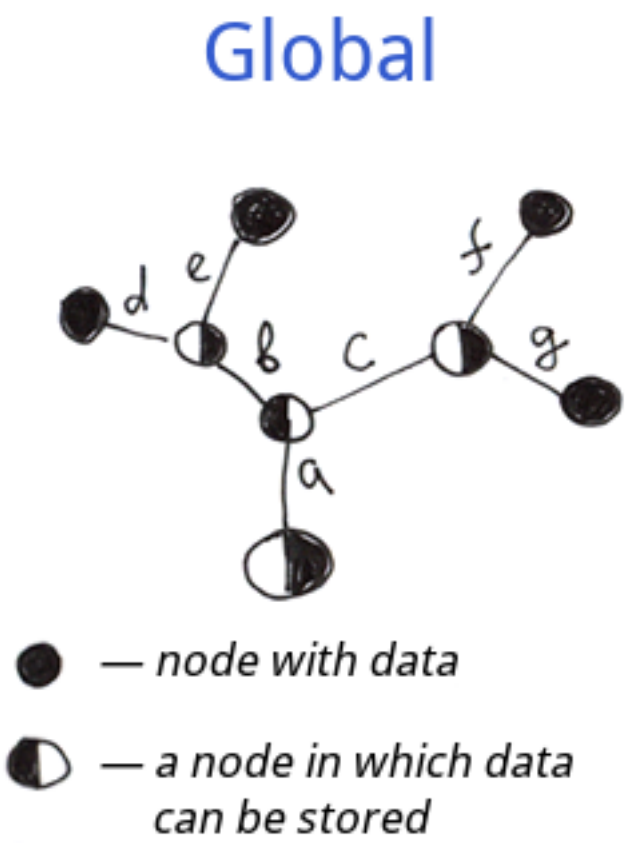
为了更好地理解globals，让我们把它们与其他树进行比较：所谓“ 花园树 ”和“ 文件系统名称树 ”。

让我们把globals与最熟悉的层次结构进行比较：如下的Orchard tree-“ 生长在花园和田野中的普通树 ”，以及File system-文件系统。



我们可以看到，叶子和果实只生长在普通树木的枝干末端。
文件系统--信息也被存储在树枝的末端，也被称为全文件名。

而下面这里是一个Global的数据结构：



不同：

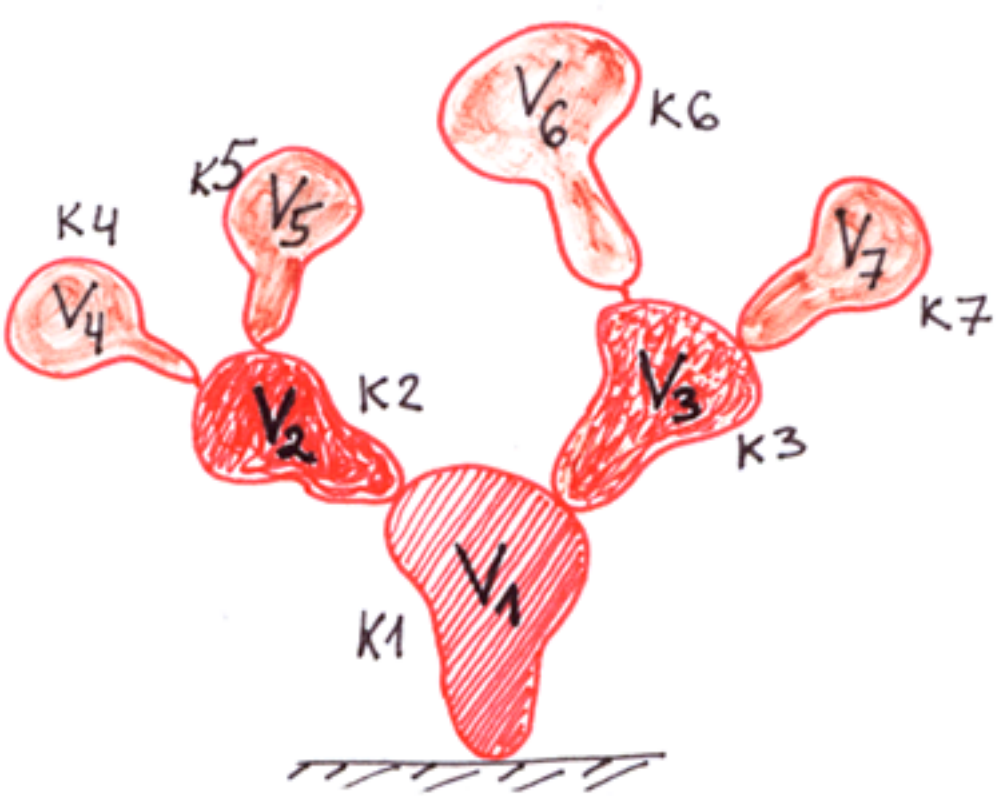
1.内部节点：Global中的信息可以存储在每个节点中，而不是只存储在分支末端。
2.外部节点：globals必须有定义的分支末端（有值的末端），这对文件系统和"花园树"来说不是强制性的。

关于内部节点，我们可以把global的结构看作是文件系统的名字树和花园树结构的超集。所以global的结构是一个更灵活的结构。

一般来说，global是一个结构化的树，支持在每个节点中保存数据。

为了更好地理解globals是如何工作的，让我们想象一下，如果文件系统的创建者使用与globals相同的方法来存储信息，会发生什么？

1. 如果一个文件夹中的最后一个文件被删除了，那么这个文件夹本身以及所有只包含这个被删除的文件夹的高层文件夹也会被删除。
2. 这样就根本不需要文件夹了。会有带子文件的文件和不带子文件的文件。如果你把它与普通的树作比较，每个分支都会变成一个果实。



If the orchard tree was a global ...

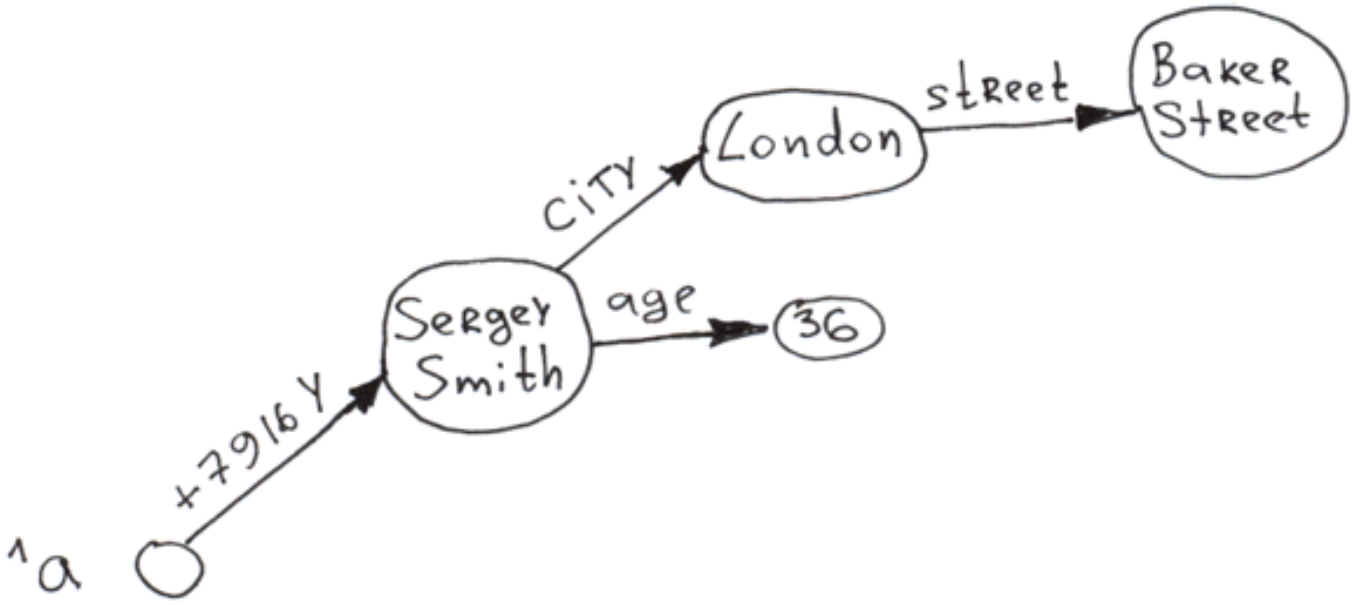
3. 像README.txt这样的东西可能就不再需要了。所有你需要说的关于文件夹的内容都可以写在文件夹文件本身。一般来说，文件名和文件夹名是没有区别的（例如，/etc/readme可以是文件夹，也可以是文件），这意味着我们只需要操作文件就可以了。

4. 带有子文件夹和文件的文件夹可以更快地被删除。网络上有一些文章讲述了删除数百万个小文件是多么的耗时和困难(1, 2, 3
). 然而，如果你创建一个基于Global的假的文件系统，它将只需要几秒钟甚至几分之一秒。当我在家里的电脑上测试删除子树时，我成功地从HDD（不是SDD）上的两级树上删除了96-341万个节点。值得一提的是，我们讨论的是删除Global树的一部分，而不是删除包含Global的整个文件。

子树的移除是globals的另一个优势：你不需要递归来做这个。它的速度快得令人难以置信。

在我们的树中，这可以通过一个Kill的命令来完成。

```
Kill ^a("+7926X")
```



下面是一个小表格，它可以让你更好地了解你可以在Global上执行的操作

Cache object script中与Globals有关的关键命令和功能	
Set设置	设置（初始化）分支到一个节点（如果未定义）和节点值
Merge合并	复制一棵子树
Kill	删除一棵字树
ZKill	删除一个特定节点的值。源自该节点的子树不受影响。
$Query	对树进行全面深入的遍历
$Order	返回同一级别的下一个下标
$Data	检查一个节点是否被定义
$Increment	节点值的原子递增，以避免ACID的读和写。最新的建议是使用 $Sequence 来代替

感谢你的关注，我很乐意回答你的任何问题。

免责声明：本文反映了作者的个人观点，与InterSystems的官方立场无关。

让我们期待下一篇继续 "Globals 是存储数据的魔剑-树 ：第二部分 *（待翻译）*

你将了解到哪些类型的数据可以显示在globals中，以及它们在哪些地方效果最好。

#Node.js #关系表 #性能 #新手 #Caché #Global Masters #InterSystems IRIS

源URL: https://cn.community.intersystems.com/post/globals-%E6%98%AF%E7%AE%A1%E7%90%86%E6%95%B0%E6%8D%AE%E7%9A%84%E9%AD%94%E5%89%91-%EF%BC%9A-%E7%AC%AC%E4%B8%80%E9%83%A8%E5%88%86