Article
Guillaume Rongier · Mars 29, 2022 9m de lecture

Les globales sont des épées magiques pour la gestion des données (partie 1)

Les globales, ces épées magiques destinées à stocker des données, existent depuis un certain temps, mais peu de gens savent les utiliser efficacement ou connaissent cette super-arme.

Si vous utilisez les globales pour des tâches où ils sont vraiment utiles, les résultats peuvent être étonnants, que ce soit en termes d'amélioration des performances ou de simplification spectaculaire de la solution globale (1, 2).

Les globales offrent une façon particulière de stocker et de traiter les données, qui est complètement différente des tableaux SQL. Ils ont été introduits en 1966 dans le langage de programmation M(UMPS), qui était initialement utilisé dans les bases de données médicales. Il est toujours utilisé de la même manière, mais a également été adopté par d'autres secteurs où la fiabilité et les hautes performances sont des priorités absolues : finance, commerce, etc.

Plus tard, M(UMPS) a évolué vers Caché ObjectScript (COS). COS a été développé par InterSystems comme un superset de M. Le langage original est toujours accepté par la communauté des développeurs et existe dans quelques implémentations. Il y a plusieurs signes d'activité sur le web : MUMPS Google group, Mumps User's group), effective ISO Standard, etc.

Les DBMS (systèmes de base de données) globales modernes prennent en charge les transactions, la journalisation, la réplication et le partitionnement. Cela signifie qu'ils peuvent être utilisés pour construire des systèmes distribués modernes, fiables et rapides.

Les globales ne vous limitent pas aux frontières du modèle relationnel. Ils vous donnent la liberté de créer des structures de données optimisées pour des tâches particulières. Pour de nombreuses applications, l'utilisation raisonnable des globales peut être une véritable solution miracle offrant des vitesses dont les développeurs d'applications relationnelles classiques ne peuvent que rêver.

Les globales, en tant que méthode de stockage des données, peuvent être utilisés dans de nombreux langages de programmation modernes, tant de haut niveau que de bas niveau. Par conséquent, cet article se concentrera spécifiquement sur les globales et non sur le langage dont ils sont issus.

2. Comment fonctionnent les globales

Commençons par comprendre comment fonctionnent les globales et quels sont leurs avantages. Les globales peuvent être vus sous différents angles. Dans cette partie de l'article, nous les verrons comme des arbres ou des stockages de données hiérarchiques.

En termes simples, une globale est une liste de données persistantes. Une liste qui est automatiquement sauvegardé sur le disque.

Il est difficile d'imaginer quelque chose de plus simple pour stocker des données. Dans le code du programme (écrit en langage COS/M), la seule différence avec un tableau associatif ordinaire est le symbole **^** qui précède leur nom.

Il n'est pas nécessaire de connaître SQL pour enregistrer des données dans une globale, car toutes les commandes nécessaires sont très simples et peuvent être apprises en une heure.

Commençons par l'exemple le plus simple, un arborescence mono niveau avec deux branches. Les exemples sont écrits en COS.

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

Lorsque des données sont insérées dans une globale (la commande Set), 3 choses se produisent automatiquement:

1. **Sauvegarde** des données sur le disque.
2. **Indexation**. Ce qui est entre les parenthèses est un indice, ce qui est à droite du signe égal est la valeur du nœud.
3. **Tri**. Les données sont triées par une clé. La prochaine traversée mettra "Sergey Smith" en première position, suivi de "John Sidorov". Lorsque l'on obtient une liste d'utilisateurs à partir d'une globale, la base de données ne passe pas de temps à trier. Vous pouvez en fait demander une liste triée à partir de n'importe quelle clé, même une clé inexistante (la sortie commencera à partir de la première clé réelle suivant celle-ci).

Toutes ces opérations sont effectuées à une vitesse incroyable. Sur mon système personnel (i5-3340, 16GB, HDD WD 1TB Blue), j'ai réussi à atteindre 1 050 000 insertions/sec en un seul processus. Sur les systèmes multi-cœurs, les vitesses peuvent atteindre des dizaines de millions of insertions/sec.

Bien sûr, la vitesse de saisie des données en elle-même ne dit pas grand-chose. Nous pouvons, par exemple, saisir des données dans des fichiers texte - c'est ainsi que le traitement fonctionne chez Visa, selon les rumeurs. Cependant, avec les globales, nous obtenons un stockage structuré et indexé avec lequel vous pouvez travailler en profitant de sa grande vitesse et de sa facilité d'utilisation.

- **La plus grande force des globales est la vitesse d'insertion de nouveaux nœuds dans ceux-ci.**
- **Les données sont toujours indexées dans une globale. Les traversées en profondeur et les traversées arborescentes mono-niveau sont toujours très rapides.**

Ajoutons quelques branches de deuxième et troisième niveau de la globale.

```
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
```

Apparemment, vous pouvez construire des arbres à plusieurs niveaux en utilisant des globales. L'accès à n'importe quel nœud est presque instantané grâce à l'indexation automatique après chaque insertion. Les branches de l'arbre à n'importe quel niveau sont triées par une clé.

Comme vous pouvez le constater, les données peuvent être stockées à la fois sous forme de clés et de valeurs. La longueur combinée d'une clé (la somme des longueurs de tous les index) peut atteindre 511 octets et les valeurs peuvent atteindre une taille de 3,6 MB dans Caché. Le nombre de niveaux dans un arbre (nombre de dimensions) est plafonné à 31.

Une autre chose intéressante : vous pouvez construire un arbre sans définir les valeurs des nœuds de niveau supérieur.

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

Les cercles vides sont des nœuds sans valeur.

Pour mieux comprendre les globales, comparons-les à d'autres arbres : les arbres de jardin et les arbres de noms de systèmes de fichiers.

Comparons les globales aux structures hiérarchiques les plus familières : les arbres réguliers qui poussent dans les jardins et les champs, ainsi que les systèmes de fichiers.

Comme nous pouvons le constater, les feuilles et les fruits ne poussent qu'à l'extrémité des branches des arbres ordinaires.
Systèmes de fichiers - les informations sont également stockées à l'extrémité des branches, également connues comme des noms de fichiers complets.

Et voici la structure de données d'une globale.

Differences:
1. **Nœuds internes:** les informations d'une globale peuvent être stockées dans tous les nœuds et pas seulement aux extrémités des branches.
2. **Nœuds externes:** les globales doivent avoir des extrémités de branche définies (extrémités avec des valeurs), ce qui n'est pas obligatoire pour les systèmes de fichiers et les arbres de jardin.

En ce qui concerne les noeuds internes, nous pouvons traiter la structure de globale comme un sur-ensemble des

arbres de noms des systèmes de fichiers et des arbres de jardins. La structure de globale est donc plus flexible.

En général, une globale est un **arbre structuré qui prend en charge la sauvegarde des données dans chaque nœud.**

Afin de mieux comprendre le fonctionnement des globales, imaginons ce qui se passerait si les créateurs d'un système de fichiers utilisaient une approche identique à celle des globales pour stocker les informations ?

1. Si le dernier fichier d'un dossier était supprimé, le dossier lui-même serait également supprimé, ainsi que tous les dossiers de niveau supérieur qui ne contenaient que ce dossier supprimé.
2. Il n'y aurait pas besoin de dossiers du tout. Il y aurait des fichiers avec des sous-fichiers et des fichiers sans sous-fichiers. Si vous le comparez à un arbre normal, chaque branche deviendrait un fruit.

3. Des choses comme README.txt ne seraient probablement plus nécessaires. Tout ce que vous avez besoin de dire sur le contenu d'un dossier pourrait être écrit dans le fichier du dossier lui-même. En général, les noms de fichiers ne peuvent pas être distingués des noms de dossiers (par exemple, /etc/readme peut être soit un dossier, soit un fichier), ce qui signifie que nous pourrions nous contenter d'exploiter des fichiers.
4. Les dossiers comportant des sous-dossiers et des fichiers pourraient être supprimés beaucoup plus rapidement. Il existe des articles sur le net qui racontent à quel point il est long et difficile de supprimer des millions de petits fichiers (1, 2, 3). En revanche, si vous créez un pseudo-système de fichiers basé sur une globale, cela prendra quelques secondes, voire des fractions de seconde. Lorsque j'ai testé la suppression de sous-arbres sur mon ordinateur personnel, j'ai réussi à supprimer 96 à 341 millions de nœuds d'un arbre à deux niveaux sur un disque dur (pas un disque dur externe). Et il convient de mentionner que nous parlons de la suppression d'une partie d'un arbre global, et non de la suppression d'un fichier entier contenant une globale.

La suppression des sous-arbres est encore un autre avantage des globaux : vous n'avez pas besoin de récursion pour cela.
C'est incroyablement rapide.
Dans notre arbre, cela pourrait être fait avec une commande **Kill**.

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

Vous trouverez ci-dessous un petit tableau qui vous permettra de mieux comprendre les actions que vous pouvez effectuer sur une globale.

Commandes et fonctions clés liées aux globales dans COS	
Set	Paramétrage (initialisation) des branches jusqu'à un noeud (si non défini) et valeur du noeud
Merge	Copie d'un sous-arbre
Kill	Suppression d'un sous-arbre
ZKill	Suppression de la valeur d'un nœud particulier. Le sous-arbre issu de ce noeud n'est pas affecté
$Query	Traversée complète et approfondie de l'arbre
$Order	Renvoie l'indice suivant au même niveau
$Data	Vérifier si un nœud est défini
$Increment	Incrémentation atomique de la valeur d'un nœud afin d'éviter la lecture et l'écriture pour ACID. La dernière recommandation est d'utiliser $Sequence à la place.

Merci de votre attention, je serai heureux de répondre à vos questions.

Démenti: Cet article reflète l'opinion privée de l'auteur et n'a aucun rapport avec la position officielle d'InterSystems.

#Débutant #Globals #Node.js #Performances #Tables relationnelles #Caché #InterSystems IRIS

URL de la source:https://fr.community.intersystems.com/post/les-globales-sont-des-%C3%A9p%C3%A9es-magiques-pour-la-gestion-des-donn%C3%A9es-partie-1