Article
Guillaume Rongier · Avr 4, 2022 13m de lecture

Les globales sont des épées magiques pour stocker des données. Arbres (partie 2)

3. Variantes des structures lors de l'utilisation de globales

Une structure, telle qu'un arbre ordonné, présente plusieurs cas particuliers. Examinons ceux qui ont une valeur pratique pour le travail avec les globales.

3.1 Cas particulier 1. Un nœud sans branches

Les globales peuvent être utilisées non seulement comme une liste de données, mais aussi comme des variables ordinaires. Par exemple, pour créer un compteur :

```
Set ^counter = 0  ; setting counter
Set id=$Increment(^counter) ;  atomic incrementation
```

En même temps, une globale peut avoir des branches outre sa valeur. L'un n'exclut pas l'autre.

3.2 Cas particulier 2. Un nœud et plusieurs branches

En fait, il s'agit d'une base classique clé-valeur. Et si nous enregistrons des tuples de valeurs au lieu de valeurs, nous obtiendrons une table ordinaire avec une clé primaire.

Afin d'implémenter une table basé sur des globales, nous devrons former des chaînes de caractères à partir des valeurs des colonnes, puis les enregistrer dans une globale par la clé primaire. Afin de pouvoir diviser la chaîne en colonnes lors de la lecture, nous pouvons utiliser ce qui suit :

1. Caractère de délimitation.

```
Set ^t(id1) = "col11/col21/col31"
Set ^t(id2) = "col12/col22/col32"
```

2. Un schéma fixe, dans lequel chaque champ occupe un nombre particulier d'octets. C'est ainsi qu'on procède généralement dans les bases de données relationnelles.

3. Une fonction spéciale $LB (introduite dans Caché) qui compose une chaîne de caractères à partir de valeurs.

```
Set ^t(id1) = $LB("col11", "col21", "col31")
Set ^t(id2) = $LB("col12", "col22", "col32")
```

Ce qui est intéressant, c'est qu'il n'est pas difficile de faire quelque chose de similaire aux clés étrangères dans les bases de données relationnelles en utilisant des globales. Appelons ces structures des index globaux. Un index global est un arbre supplémentaire permettant d'effectuer des recherches rapides sur des champs qui ne font pas partie intégrante de la clé primaire de la globale principale. Vous devez écrire un code supplémentaire pour le remplir et l'utiliser.

Nous créons un index global basé sur la première colonne.

```
Set ^i("col11", id1) = 1
Set ^i("col12", id2) = 1
```

Pour effectuer une recherche rapide par la première colonne, vous devrez regarder dans la ^i globale et trouver les clés primaires (id) correspondant à la valeur nécessaire dans la première colonne.

Lors de l'insertion d'une valeur, nous pouvons créer à la fois des valeurs et des index globaux pour les champs nécessaires. Pour plus de fiabilité, nous allons l'intégrer dans une transaction.

```
TSTART
Set ^t(id1) = $LB("col11", "col21", "col31")
Set ^i("col11", id1) = 1
TCOMMIT
```

Plus d'informations sont disponibles ici making tables in M using globals and emulation of secondary keys.

Ces tables fonctionneront aussi rapidement que dans les bases de données traditionnelles (ou même plus rapidement) si les fonctions d'insertion/mise à jour/suppression sont écrites en COS/M et compilées.
J'ai vérifié cette affirmation en appliquant un grand nombre d'opérations INSERT et SELECT à une seule table à deux colonnes, en utilisant également les commandes TSTART et TCOMMIT (transactions).
Je n'ai pas testé de scénarios plus complexes avec des accès concurrents et des transactions parallèles.

Sans utiliser de transactions, la vitesse d'insertion pour un million de valeurs était de 778 361 insertions/seconde.

Pour 300 millions de valeurs, la vitesse était de 422 141 insertions/seconde.

Lorsque des transactions ont été utilisées, la vitesse a atteint 572 082 insertions/seconde pour 50 millions de

valeurs. Toutes les opérations ont été exécutées à partir du code M compilé. J'ai utilisé des disques durs ordinaires, pas des SSD. RAID5 avec Write-back. Le tout fonctionnant sur un processeur Phenom II 1100T.

Pour effectuer le même test pour une base de données SQL, il faudrait écrire une procédure stockée qui effectuerait les insertions en boucle. En testant MySQL 5.5 (stockage InnoDB) avec la même méthode, je n'ai jamais obtenu plus de 11K insertions par seconde.

En effet, l'implémentation de tables avec des globales est plus complexe que de faire la même chose dans des bases de données relationnelles. C'est pourquoi les bases de données industrielles basées sur les globales ont un accès SQL pour simplifier le travail avec les données tabulaires.

En général, si le schéma de données ne change pas souvent, que la vitesse d'insertion n'est pas critique et que l'ensemble de la base de données peut être facilement représenté par des tables normalisées, il est plus facile de travailler avec SQL, car il offre un niveau d'abstraction plus élevé.

Dans ce cas, je voulais montrer que les globales peuvent être utilisées comme un constructeur pour créer d'autres bases de données. Comme le langage assembleur qui peut être utilisé pour créer d'autres langages. Et voici quelques exemples d'utilisation des globales pour créer des contreparties de key-values, lists, sets, tabular, document-oriented DB's.

Si vous devez créer une base de données non standard avec un minimum d'efforts, vous devriez envisager d'utiliser les globales.

3.3 Cas particulier 3. Un arbre à deux niveaux dont chaque nœud de deuxième niveau a un nombre fixe de branches

Vous l'avez probablement deviné : il s'agit d'une implémentation alternative des tables utilisant des globales. Comparons-la avec la précédente.

Tables dans un arborescence deux niveaux vs. Tables dans un arborescence mono niveau.	
Cons	Pros
Insertions plus lentes, car le nombre de nœuds doit être égal au nombre de colonnes. Une plus grande consommation d'espace sur le disque dur, car les index globaux (comme les index de table) avec les noms de colonne occupent de l'espace sur le disque dur et sont dupliqués pour chaque ligne.	Un accès plus rapide aux valeurs de certaines colonnes, puisque vous n'avez pas besoin d'analyser la chaîne de caractères. D'après mes tests, c'est 11,5 % plus rapide pour 2 colonnes et encore plus rapide pour plus de colonnes. Il est plus facile de modifier le schéma de données et de lire le code.

Conclusion: Rien d'extraordinaire. Les performances étant l'un des principaux avantages des globales, il n'y a pratiquement aucun intérêt à utiliser cette approche, car il est peu probable qu'elle soit plus rapide que les tables ordinaires des bases de données relationnelles.

3.4 Cas général. Arbres et clés ordonnées

Toute structure de données qui peut être représentée comme un arbre s'adapte parfaitement aux globales.

3.4.1 Objets avec des sous-objets

C'est dans ce domaine que les globales sont traditionnellement utilisées. Il existe de nombreuses maladies, médicaments, symptômes et méthodes de traitement dans le domaine médical. Il est irrationnel de créer une table avec un million de champs pour chaque patient, d'autant plus que 99% d'entre eux seront vides.

Imaginez une base de données SQL composée des tables suivants : " Patient " ~100 000 champs, " Médicament " 100 000 champs, " Thérapie " 100 000 champs, " Complications " 100 000 champs et ainsi de suite. Comme alternative, vous pouvez créer une BD avec des milliers de tableaux, chacun pour un type de patient particulier (et

ils peuvent aussi se superposer !), un traitement, un médicament, ainsi que des milliers de tables pour les relations entre ces tables.

Les globales s'adaptent parfaitement aux soins de santé, puisqu'elles permettent à chaque patient de disposer d'un dossier complet, de la liste des thérapies, des médicaments administrés et de leurs effets, le tout sous la forme d'un arbre, sans gaspiller trop d'espace disque en colonnes vides, comme ce serait le cas avec les bases de données relationnelles.

Les globales fonctionnent bien pour les bases de données contenant des données personnelles, lorsque la tâche consiste à accumuler et à systématiser le maximum de données personnelles diverses sur un client. C'est particulièrement important dans les domaines de la santé, de la banque, du marketing, de l'archivage et autres.

Il est évident que SQL permet également d'émuler un arbre en utilisant seulement quelques tables (EAV, 1,2,3,4,5,6, 7,8), mais c'est beaucoup plus complexe et plus lent. En fait, nous devrions écrire une globale basé sur des tables et cacher toutes les routines liées aux tables sous une couche d'abstraction. Il n'est pas correct d'émuler une technologie de niveau inférieur (les globales) à l'aide d'une technologie de niveau supérieur (SQL). C'est tout simplement injustifié.

Ce n'est pas un secret que la modification d'un schéma de données dans des tableaux gigantesques (ALTER TABLE) peut prendre un temps considérable. MySQL, par exemple, effectue l'opération ALTER TABLE ADD¦DROP COLUMN en copiant toutes les données de l'ancienne tableau vers la nouvelle (je l'ai testé sur MyISAM et InnoDB). Cela peut bloquer une base de données de production contenant des milliards d'enregistrements pendant des jours, voire des semaines.

Si nous utilisons des globales, la modification de la structure des données ne nous coûte rien Nous pouvons ajouter de nouvelles propriétés à n'importe quel objet, à n'importe quel niveau de la hiérarchie et à n'importe quel moment. Les changements qui nécessitent de renommer les branches peuvent être appliqués en arrière-plan avec la base de données en fonctionnement.

Par conséquent, lorsqu'il s'agit de stocker des objets comportant un grand nombre de propriétés facultatives, les globales fonctionnent parfaitement.

Je vous rappelle que l'accès à l'une des propriétés est instantané, puisque dans une globale, tous les chemins sont un B-Arbre.

Dans le cas général, les bases de données basées sur des globales sont un type de bases de données orientées documents qui supportent le stockage d'informations hiérarchiques. Par conséquent, les bases de données orientées documents peuvent concurrencer efficacement les globales dans le domaine du stockage des cartes médicales.

Mais ce n'est pas encore le cas.
Prenons MongoDB, par exemple. **Dans ce champ**, il perd face aux globales pour les raisons suivantes :

1. **Taille du document.** L'unité de stockage est un texte au format JSON (BSON, pour être exact) dont la taille maximale est d'environ 16 Mo. Cette limitation a été introduite dans le but de s'assurer que la base de données JSON ne devienne pas trop lente lors de l'analyse syntaxique, lorsqu'un énorme document JSON y est enregistré et que des valeurs de champ particulières sont traitées. Ce document est censé contenir des informations complètes sur un patient. Nous savons tous à quel point les cartes de patient peuvent être épaisses. Si la taille maximale de la carte est plafonnée à 16 Mo, cela permet de filtrer immédiatement les patients dont les cartes contiennent des IRM, des radiographies et d'autres documents. Une seule branche d'une entreprise mondiale peut contenir des gigaoctets, des pétaoctets ou des téraoctets de données. Tout est dit, mais laissez-moi vous en dire plus.
2. **Le temps nécessaire à la création/modification/suppression de nouvelles propriétés de la carte du patient.** Une telle base de données devrait copier la carte entière dans la mémoire (beaucoup de données !), analyser les données BSON, ajouter/modifier/supprimer le nouveau nœud, mettre à jour les index, remballer le tout en BSON et sauvegarder sur le disque. Une globale n'aurait besoin que d'adresser la propriété nécessaire et d'effectuer l'opération nécessaire.
3. **La vitesse d'accès à des propriétés particulières.** Si le document possède de nombreuses propriétés et

une structure à plusieurs niveaux, l'accès à des propriétés particulières sera plus rapide car chaque chemin dans la globale est le B-Arbre. En BSON, vous devrez analyser linéairement le document pour trouver la propriété nécessaire.

3.3.2 Tables associatives

Les tables associatives (même avec les tables imbriquées) fonctionnent parfaitement avec les globales. Par exemple, cette table PHP ressemblera à la première illustration en 3.3.1.

```
$a = array(
  "name" => "Vince Medvedev",
  "city" => "Moscow",
  "threatments" => array(
    "surgeries" => array("apedicectomy", "biopsy"),
    "radiation" => array("gamma", "x-rays"),
    "physiotherapy" => array("knee", "shoulder")
  )
);
```

3.3.3 Documents hiérarchiques : XML, JSON

Ils peuvent également être facilement stockés dans des globales et décomposés de manières différentes.

XML

La méthode la plus simple pour décomposer le XML en globales consiste à stocker les attributs des balises dans les nœuds. Et si vous avez besoin d'un accès rapide aux attributs des attributs, nous pouvons les placer dans des branches séparées.

```
<note id=5>
<to>Alex</to>
<from>Sveta</from>
<heading>Reminder</heading>
<body>Call me tomorrow!</body>
</note>
```

Dans COS, le code ressemblera à ceci :

```
Set ^xml("note")="id=5"
Set ^xml("note","to")="Alex"
Set ^xml("note","from")="Sveta"
Set ^xml("note","heading")="Reminder"
Set ^xml("note","body")="Call me tomorrow!"
```

Note: Pour XML, JSON et les tables associatives, vous pouvez imaginer un certain nombre de méthodes pour les afficher dans les globales. Dans ce cas particulier, nous n'avons pas reflété l'ordre des balises imbriquées dans la balise "note". Dans la globale **^xml**, les balises imbriquées seront affichés dans l'ordre alphabétique. Pour un affichage précis de l'ordre, vous pouvez utiliser le modèle suivant, par exemple :

JSON.

Le contenu de ce document JSON est présenté dans la première illustration de la section 3.3.1 :

```
var document = {
  "name": "Vince Medvedev",
  "city": "Moscow",
```

```
  "threatments": {
    "surgeries": ["apedicectomy", "biopsy"],
    "radiation": ["gamma", "x-rays"],
    "physiotherapy": ["knee", "shoulder"]
  },
};
```

3.3.4 Structures identiques liées par des relations hiérarchiques

Exemples : structure des bureaux de vente, positions des personnes dans une structure MLM, base des débuts aux échecs.

Base de données des débuts. Vous pouvez utiliser une évaluation de la force du mouvement comme valeur de l'indice de nœud d'une globale. Dans ce cas, vous devrez sélectionner une branche ayant le poids le plus élevé pour déterminer le meilleur déplacement. Dans la globale, toutes les branches de chaque niveau seront triées en fonction de la force du mouvement.

La structure des bureaux de vente, des personnes dans une société MLM. Les noeuds peuvent stocker certaines valeurs de cache reflétant les caractéristiques de la sous-arborescence entière. Par exemple, les ventes de cette sous-arborescence particulière. Nous pouvons obtenir des informations exactes sur les réalisations de n'importe quelle branche à tout moment.

4. Situations où l'utilisation des globales est avantageuse

La première colonne contient une liste de cas où l'utilisation des globales vous donnera un avantage considérable en termes de performance, et la seconde - une liste de situations où elles simplifieront le développement ou le modèle de données.

Vitesse	Commodité du traitement/de la présentation des données
1. Insertion [avec tri automatique à chaque niveau], [indexation par la clé primaire]	1. Objets/instances avec un grand nombre de propriétés/instances non requises [et/ou imbriquées]
2. Suppression de sous-arbres	2. Données sans schéma - de nouvelles propriétés peuvent souvent être ajoutées et d'anciennes supprimées
3. Objets comportant de nombreuses propriétés imbriquées auxquelles vous devez accéder individuellement	3. Vous devez créer une BD non standard.Bases de données de chemins et arbres de solutions
4. Structure hiérarchique avec possibilité de parcourir les branches enfant à partir de n'importe quelle branche, même inexistante	4. Lorsque les chemins peuvent être représentés de manière pratique sous forme d'arbre
5. Parcours en profondeur de l'arbre	5. On doit supprimer les structures hiérarchiques sans utiliser la récursion

Clause de non-responsabilité: *cet article et les commentaires le concernant reflètent uniquement mon opinion et n'ont rien à voir avec la position officielle de la société InterSystems.*

#Débutant #Globals #Modèle de données #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-stocker-des-donn%C3%A9es-arbres-partie-2