Article
Guillaume Rongier · Juin 10, 2022 8m de lecture

Stockage des données - Informations à connaître pour prendre de bonnes décisions lors du développement

Cette publication est le résultat direct d'une collaboration avec un client d'InterSystems qui est venu me consulter pour le problème suivant :

```
SELECT COUNT(*) FROM MyCustomTable
```

Cela prend 0,005 secondes, pour un total de 2300 lignes. Cependant :

```
SELECT * FROM MyCustomTable
```

Prenait des minutes. La raison en est subtile et suffisamment intéressante pour que j'écrive un article à ce sujet. Cet article est long, mais si vous faites défiler la page jusqu'en bas, je vous donnerai un résumé rapide. Si vous êtes arrivé jusqu'ici et que vous pensez en avoir lu assez, faites défiler la page jusqu'à la fin pour connaître l'essentiel. Vérifiez la phrase en **gras**.

Lors de la création de vos classes, il faut tenir compte de la question du stockage. Comme beaucoup d'entre vous le savent, toutes les données dans Caché sont stockées dans des Globales.

Si vous ne le savez pas, je pense que cet article sera un peu trop long. Je vous recommande de consulter un excellent tutoriel dans notre documentation :

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

Si vous n'avez jamais utilisé Caché/Ensemble/HealthShare, le tutoriel ci-dessus est très utile, et même si vous l'avez fait, il vaut la peine de le consulter !

Maintenant, comme toutes les données sont stockées dans des globales, il est important de comprendre comment les définitions de vos classes correspondent aux globales. Construisons une application ensemble ! Nous allons examiner certains pièges courants et discuter de la façon dont le développement de vos classes affecte vos stratégies de stockage, avec un regard particulier sur les performances SQL.

Imaginons que nous soyons le Bureau du recensement des États-Unis et que nous voulions disposer d'une base de données pour stocker les informations concernant tous les habitants des États-Unis. Nous construisons donc une classe de la manière suivante :

```
Class USA.Person extends %Persistent
{
 Property Name as %String;
 Property SSN as %String;
```

```
 Property Address as %String;
 Property DateOfBirth as %Date;
}
```

SSN est l'abréviation de "Social Security Number" (numéro de sécurité sociale) qui, bien qu'il n'ait pas été conçu à l'origine pour être un numéro d'identification des personnes, est leur numéro d'identification de facto. Cependant, comme nous sommes traditionalistes, nous ne l'utiliserons pas pour l'identification. Cela dit, nous tenons à ce que cet élément soit indexé, car c'est un excellent moyen de rechercher une personne. Nous savons que nous devrons parfois rechercher des personnes par le nom, c'est pourquoi nous voulons également un index des noms. Et parce que notre patron aime ses rapports basés sur des tranches d'âge, nous pensons qu'un index des dates de naissance pourrait également être utile. Ajoutons-les donc à notre classe

```
Class USA.Person extends %Persistent
{
 Property Name as %String;
 Property SSN as %String;
 Property Address as %String;
 Property DateOfBirth as %Date;

 Index NameIDX On Name;
 Index SSNIDX On SSN [Unique];
 Index DOBIDX on DateOfBirth;

}
```

Très bien. Alors ajoutons une ligne et voyons à quoi ressemblent nos globales. Notre instruction INSERT est la suivante :

```
INSERT INTO USA.Person (Name,SSN,Address,DateOfBirth) VALUES
   ('Baxter, Kyle','111-11-1111','1 Memorial Drive, Cambridge, MA 02142','1985-07-20'
)
```

Et la globale:

```
USER>zw ^USA.PersonD
^USA.PersonD=1
^USA.PersonD(1)=$lb("","Baxter, Kyle","111-11-1111","1 Memorial Drive, Cambridge, MA
02142",52796)
```

Le stockage par défaut d'une classe stocke vos données dans ^Package.ClassD. Si le nom de la classe est trop long, il peut être haché, et vous pouvez le trouver dans la définition de stockage au bas de votre définition de classe. Les index, à quoi ressemblent-ils ?

```
USER>zw ^USA.PersonI
^USA.PersonI("DOBIDX",52796,1)=""
^USA.PersonI("NameIDX"," BAXTER, KYLE",1)=""
^USA.PersonI("SSNIDX"," 111-11-1111",1)=""
```

Excellent, notre stockage est plutôt bon pour l'instant. Donc on ajoute nos 320 millions de personnes et on peut trouver des gens assez rapidement. Mais maintenant nous avons un problème, car nous voulons traiter le

président et tous les ex-présidents avec une considération spéciale. Nous ajoutons donc une classe spéciale pour le président :

```
Class USA.President extends USA.Person
{
Property PresNumber as %Integer;

Index PresNumberIDX on PresNumber;
}
```

Bien. En raison de l'héritage, nous récupérons toutes les propriétés de USA.Person, et nous en ajoutons une pour nous permettre de savoir quel numéro de président il était. Puisque je veux faire un peu de politique, je vais INSÉRER notre PROCHAIN président. Voici l'instruction :

```
INSERT INTO USA.President (Name,SSN,DateOfBirth,Address,PresNumber) VALUES ('McDonald
,Ronald','221-18-7518','01-01-1963','1600 Pennsylvania Ave NW, Washington, DC 20006',
45)
```

Note : Son numéro de sécurité sociale s'écrit 'Burger'. Désolé si c'est le vôtre.

Alors c'est génial ! Regardons votre Globale du Président :

```
USER>zw ^USA.PresidentD


```

Pas de données ! Et c'est là que nous arrivons à l'essentiel de cet article. Parce que nous avons décidé d'hériter de USA.Person FIRST, nous avons hérité non seulement de ses propriétés et index, mais aussi de son stockage ! Donc pour localiser le président McDonald, nous devons regarder dans ^USA.PersonD. Et nous pouvons voir ce qui suit :

```
^USA.PersonD(2)=$lb("~USA.President~","McDonald,Ronald","221-18-7518","1600 Pennsylva
nia Ave NW, Washington, DC 20006",44560)
^USA.PersonD(2,"President")=$lb(45&)
```

Deux choses à noter ici. La première est que nous pouvons voir que le nœud (2) possède toutes les informations déjà stockées dans USA.Person. Alors que le noeud (2, "President") ne contient que les informations spécifiques à la classe USA.President.

Qu'est-ce que cela signifie en pratique ? Eh bien, si nous voulons faire une opération de type : SELECT * FROM USA.President, nous aurons BESOIN de parcourir l'ensemble du tableau des personnes. Si nous pensons que le tableau des personnes contient 320 000 000 lignes et que le tableau des présidents en contient 45, alors nous devons faire plus de 320 000 045 références globales pour extraire 45 lignes ! En effet, si l'on regarde le plan de requête :

- `Lire la carte maîtresse USA.President.IDKEY, en bouclant sur ID.`
- `Pour chaque ligne:`
- `Résultat de la ligne.`

Nous observons ce que nous attendons. Cependant, nous avons déjà vu que cela signifie qu'il faut nécessairement regarder dans la globale ^USA.PersonD. Donc, cela va être une référence globale de 320 000 000+ car nous devons tester CHAQUE ^USA.PersonD pour vérifier s'il y a des données dans ^USA.PersonD(i, "Président")

puisque nous ne savons pas quelles personnes seront présidents. Eh bien, c'est mauvais ! Ce n'est pas du tout ce que nous voulions ! Que pouvons-nous faire ? Eh bien, nous avons deux options :

Option 1

Ajouter un index d'éxtent. Si nous faisons cela, nous obtenons une liste d'identifiants qui nous permet de savoir quelles personnes sont des présidents et nous pouvons utiliser cette information pour lire des nœuds spécifiques de la globale ^USA.Person. Comme je dispose d'un stockage par défaut, je peux utiliser un index bitmap, ce qui rendra l'opération encore plus rapide. Nous ajoutons l'index comme suit :

```
Index Extent [Type=Bitmap, Extent];
```

Et quand nous regardons notre plan de requête pour SELECT * FROM USA.President nous pouvons voir :

- Lecture de l'extent du bitmap USA.President.Extent, en bouclant sur l'ID.
- Pour chaque ligne :
- Lecture de la carte maîtresse USA.President.IDKEY, en utilisant la valeur idkey donnée.
  Résultat de la ligne.

Ah, maintenant ça va être sympa et rapide. Une référence globale pour lire l'Extent et ensuite 45 autres pour les présidents. C'est plutôt efficace.

Les inconvénients ? La connexion à ce tableau devient un peu plus compliquée et peut impliquer un plus grand nombre de tableaux temporaires que vous ne le souhaiteriez.

Option 2

Changement de la définition de la classe en ::

```
Class USA.President extends (%Persistent, USA.Person)
```

En faisant de %Persistent la première classe étendue, USA.President aura sa propre définition de stockage. Ainsi, les présidents seront stockés de la manière suivante :

```
USER>zw ^USA.PresidentD
^USA.PresidentD=1
^USA.PresidentD(1)=$lb("","McDonald,Ronald","221-18-7518","1600 Pennsylvania Ave NW,
Washington, DC 20006",44560,45)
```

C'est donc une bonne chose, car choisir USA.President signifie simplement lire les 45 membres de cette globale. C'est facile et agréable, et le design est clair.

Les inconvénients ? Eh bien maintenant, les présidents ne sont PAS dans le tableau des personnes Person. Donc si vous voulez des informations sur les présidents ET les non-présidents, vous devez faire *SELECT ... FROM USA.Person UNION ALL SELECT ... FROM USA.President*

Si vous avez arrêté de lire au début, recommencez ici !

Lors de la création d'un héritage, nous avons deux options

Option 1: L'héritage de la superclasse est le premier. Cela permet de stocker les données dans la même globale que la superclasse. Utile si vous voulez avoir toutes les informations ensemble, et vous pouvez atténuer les problèmes de performance dans la sous-classe en ayant un index extent.

Option 2: Héritage de %Persistent first. Cela permet de stocker les données dans une nouvelle globale. C'est utile si vous interrogez beaucoup la sous-classe, mais si vous voulez voir les données de la super-classe et de la sous-classe, vous devez utiliser une requête UNION.

Laquelle de ces solutions est la meilleure ? Cela dépend de la façon dont vous allez utiliser votre application. Si vous souhaitez effectuer un grand nombre de requêtes sur l'ensemble des données, vous opterez probablement pour la première approche. En revanche, si vous ne pensez pas interroger les données dans leur ensemble, vous opterez probablement pour la seconde approche. Les deux approches sont tout à fait acceptables, à condition de ne pas oublier l'index extent de l'option 1.

Des questions ? Des commentaires ? De longues pensées contradictoires ? Laissez-les ci-dessous !

#Conseils et astuces #Globals #Modèle de données #Object Data Model #ObjectScript #SQL #Tutoriel