Les CTE ou la récursivité avec SQL Server 2005

Comment exécuter des requêtes récursives en SQL

Comment effectuer des requêtes récursives en SQL 2005. Ceci est une problématique classique en développement qui est souvent déportée sur le langage externe, ce qui provoque une multiplication du nombre des requêtes envoyées au serveur.
Ce problème est ajouté dans la norme SQL 3, et donc a été intégré dans SQL Server 2005.

Introduction

Les Common Table Expressions (CTE) ou les Expressions de Table Communes en français sont comparables à des tables temporaires qui peuvent être appelées en cours de requête mais dont l'existence n'est pas réelle.

Il permet aussi d'éviter la création d'une VUE pour le cas d'un besoin très ponctuel.

On peut représenter un CTE comme une table virtuelle qui permet par exemple de faire une boucle récursive dont les résultats seront stockés dans ce CTE.

Nous allons dans cet article expliquer comment utiliser un CTE, puis montrer un cas de calcul mathématique et enfin un exemple avec un arbre.

Exemple de base d'utilisation d'un CTE

Un CTE est toujours défini par le mot WITH associé avec le nom de celui-ci. On transmet ensuite les colonnes retournées par celui-ci.

Nous ne spécifions pas de type de donnée dans ce cadre, cette déclaration se fait automatiquement lors de l'exécution.

Ensuite, on définit la sélection que retourne ce CTE, ce qui dans notre cas est la valeur 1234567.

```
------------------------------------------------------------------
-- Test de base
------------------------------------------------------------------
WITH MonCTEbasique (MonExemple)
AS
(
   SELECT 1234567
)
SELECT MonExemple FROM MonCTEbasique
------------------------------------------------------------------
```

On n'a alors plus qu'à faire la requête d'appel de ce CTE.

Comme vous pouvez le voir, la conception d'un CTE simple est similaire à une vue. On peut donc faire des agrégations de valeurs directement utilisables dans la suite d'un script SQL avec ce procédé.

Voyons le cas d'une requête récursive mathématique.

Exemple avancé de calcul mathématique

Nous allons dans cet exemple voir comment faire la somme des différentes valeurs entre celui transmis (5 dans notre exemple) et 0.

Ainsi, ce calcul est le suivant :

- 5 + 4 + 3 + 2 + 1 + 0 = 15

Donc le principe de l'usage du CTE dans ce cadre est de boucler sur la requête de sommation jusqu'à atteindre 0. On transmet donc à chaque étape la somme temporaire et la valeur à ajouter.

Pour comprendre la problématique de la récursivité, je vous invite à lire l'article suivant :

- **Cours sur la récursivité**

Le principe de ce code est donc de définir les variables qui vont être utilisée au cours du calcul :

- @Calcul : qui stockera la somme au cours des passages de la boucle
- @Valeur : qui définit la valeur de départ (5 dans notre exemple)

On déclare alors la fonction de calcul (le CTE) avec le travail à faire lors de chaque passage :

- Somme des valeurs dans la variable locale à la boucle (Total)
- Diminution d'un cran de la variable locale (MaValeur)

On spécifie le point d'arrêt de la boucle :

- Lorsque MaValeur est égale à 0

On n'a alors plus qu'à spécifier le SELECT sur ce CTE.

```
DECLARE @Calcul AS int;
DECLARE @Valeur AS int;

SET @Valeur = 5;

-----------------------------------------------------------------
WITH MaSommeChiffreCTE(Total, MaValeur)
AS
(
   SELECT 0, @Valeur
   UNION ALL
   SELECT Total+MaValeur, MaValeur-1
   FROM MaSommeChiffreCTE
   WHERE MaValeur > 0
)
-----------------------------------------------------------------

SELECT @Calcul = Total
FROM MaSommeChiffreCTE

OPTION (MAXRECURSION 100)

PRINT @Calcul
```

Attention :

Pour des raisons évidentes de sécurité et donc pour éviter une boucle infinie, on spécifie une limite du nombre de boucle à 100, avec l'option MAXRECURSION.

Maintenant que nous avons vu un exemple de récursivité mathématique, voyons un cas pratique pour la navigation dans les arbres de décision.

Cas des arborescences

Le cas classique des arborescences est un "casse-tête" en développement que l'on retrouve dans les organigrammes, dans les menus en cascades ou encore dans les treeviews. Ainsi, on est obligé de passer par un développement spécifique dans l'application que fera des appels successifs à la base de données.

La question est donc simple, pourquoi ne pas passer par le CTE, étant donné que ceux-ci permettent d'intégrer la récursivité.

Vous pourrez donc trouver un exemple de solution sur le site de la documentation MSDN de SQL Server :

- **Requêtes récursives utilisant des expressions de table communes**

```
USE AdventureWorks;
GO
WITH DirectReports (ManagerID, EmployeeID, Title, DeptID, Level)
AS
(
-- Anchor member definition
    SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID,
        0 AS Level
    FROM HumanResources.Employee AS e
    INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh
        ON e.EmployeeID = edh.EmployeeID AND edh.EndDate IS NULL
    WHERE ManagerID IS NULL
    UNION ALL
-- Recursive member definition
    SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID,
        Level + 1
    FROM HumanResources.Employee AS e
    INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh
        ON e.EmployeeID = edh.EmployeeID AND edh.EndDate IS NULL
    INNER JOIN DirectReports AS d
        ON e.ManagerID = d.EmployeeID
)
-- Statement that executes the CTE
SELECT ManagerID, EmployeeID, Title, Level
FROM DirectReports
INNER JOIN HumanResources.Department AS dp
    ON DirectReports.DeptID = dp.DepartmentID
WHERE dp.GroupName = N'Research and Development' OR Level = 0;
GO
```

Je vous invite à lire la documentation d'explication qui est très précise et détaillée sur le sujet.

Conclusion

La récursivité est un besoin courant dans les développements d'application. On peut retrouver cela dans les TreeView, les menus en cascades, les organigrammes, ...

Cet article nous a permis de voir une des fonctionnalités majeures ajoutées au moteur SQL Server 2005. Celle-ci est accompagnée de nombreuses autres que nous verrons dans de prochains articles.

Voici quelques liens utiles si cet article vous a intéressé :

- **Requêtes récursives utilisant des expressions de table communes**
- **Utilisation d'expressions de table communes**
- **SQL Server 2005 Recursive Functions**
- **Arborescence SQL par intervalles**
- **Gestion d'arbres par représentation intervallaire**
- **L'avenir de SQL : SQL 3, XML, XQL**
- **Cours sur la récursivité**

En vous souhaitant de bons projets de développement.

Romelard Fabrice (alias F___)

Consultant Technique **ilem SA**