

DB2 et le SQL récursif.

par Jean-Alain Baeyens ([autres articles](#))

Date de publication : 11/06/2007

Dernière mise à jour :

Cet article a pour vocation d'expliquer le fonctionnement du SQL récursif. Ce tutoriel est basé sur DB2 mais il est globalement applicable à d'autres gestionnaires de bases de données comme par exemple SQL Server. Cette technologie est à la fois simple et très puissante et permet d'éviter des procédures stockées complexes ou d'implémenter la récursivité au niveau du client qui entraîne un important trafic de données sur le réseau.

- I - Présentation de l'exemple
- II - La syntaxe
- III - Exemple: La liste des produits de base
- IV - La récursivité maîtrisée
- V - Conclusion

I - Présentation de l'exemple

Pour illustrer les possibilités du SQL récursif dans DB2, prenons l'exemple classique de la composition d'un produit fini. Le produit fini est composé de produits de base mais également de produits semi-finis. Un produit fini peut également être un produit semi-fini pour un autre produit.

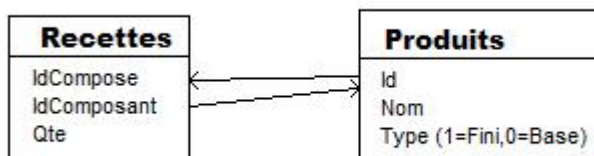


Schéma des données

L'approche la plus simple pour obtenir des informations comme "Quels produits de base dois-je prendre pour réaliser complètement mon produit ?" ou "Quelle est la composition d'un produit et de chaque produit qui le compose ?" est évidemment la récursivité. Vous pouvez bien sûr la développer dans le programme client. Ce qui entrainera un important trafic entre le client et le serveur. Une possibilité souvent méconnue ou oubliée est l'utilisation du SQL récursif.

II - La syntaxe

L'instruction permettant de réaliser un "Select" récursif est divisé en quatre parties:

- La première partie définit la table temporaire qui recevra le résultat tout d'abord du "Select" d'initialisation et ensuite des résultats successifs du "Select" de récursion.
- La seconde est le "Select" d'initialisation.
- La troisième est le "Select" qui sera exécuté de manière récursive.
- La quatrième partie est le "Select" final qui est exécuté sur la table temporaire

La syntaxe du SQL récursif

```
WITH  
nomDeLaTableTemporaire (champ1, champ2, champ3, ...) AS  
(  
    SELECT d'initialisation  
  
    UNION ALL  
  
    SELECT de récursion  
)  
SELECT de résultat
```

Concrètement, c'est comme si la table temporaire recevait le résultat du premier select. Ensuite, pour chaque ligne contenue dans cette table temporaire, le "Select" de récursion est exécuté. Le résultat est ajouté à la table temporaire et servira donc d'entrée pour autant de nouvelle exécution du "Select" que de lignes ajoutées et ainsi de suite. Au niveau moteur DB2, il est probable que cela ne se passe pas exactement de cette manière mais c'est l'idée générale.

III - Exemple: La liste des produits de base


Pour obtenir la liste des produits de base nécessaires à la réalisation du produit fini dont l'Id est 1, la commande SQL sera:


Commande SQL

```
WITH
  ProduitsDeBase (Id, Qte) AS
  (
    SELECT Racine.IdComposant, Racine.Qte
    FROM Recettes Racine
    WHERE Racine.IdCompose = 1

    UNION ALL

    SELECT Enfant.IdComposant, Enfant.Qte * Parent.Qte
    FROM ProduitsDeBase Parent, Recettes Enfant
    WHERE Parent.Id = Enfant.IdCompose
  )
  SELECT ProduitsDeBase.Id, Sum(Qte), Produits.Nom
  FROM ProduitsDeBase, Produits
  WHERE ProduitsDeBase.Id = Produits.ID AND Produits.Type = 0
  GROUP BY ProduitsDeBase.Id, Produits.Nom
  ORDER BY Produits.nom
```

 Dans le "Select" de récursion, une jointure est réalisée sur la table parent (table temporaire) de manière à obtenir tout les composants d'un élément. Le type de produit sera nécessaire pour la sélection finale.


 Notez que pour obtenir le nombre total d'élément, il est nécessaire de multiplier la quantité du composant par la quantité du composé.

Dans notre exemple, le produit fini 'A' concerné est composé de 2 produits 'B', 1 'C' et 1 'D'. Le produit 'C' est lui même composé de 2 'D' et 3 'E'. 'B' quant a lui est composé de 1 'E', 2 'F' et 2 'C'. Le résultat de la requête donne donc:

Résultat

ID	Qte	NOM
4	6	D
5	17	E
6	4	F

3 record(s) selected with 1 warning messages printed.

 Notez le message d'avertissement. Nous verrons dans le chapitre suivant pourquoi ce message est affiché et comment éviter ce problème.

IV - La récursivité maîtrisée

Imaginons que dans la recette, un produit fini contienne un autre produit fini qui lui même contient le premier. Soit A composé de B et B composé de A. Dans ce cas, la récursivité va entrer en boucle.

Pour éviter ce piège, vous devez limiter la profondeur de la récursivité en utilisant un compteur. Dans l'exemple, nous allons utiliser une colonne appelée "Niveau". Elle est initialisée à 0 et ensuite incrémentée de 1 à chaque appel récursif. Une condition sur cette colonne permet alors de bloquer la récursivité. A vous de choisir un nombre suffisamment grand pour que le traitement puisse être complet mais suffisamment petit pour éviter une boucle très pénalisante en temps en cas d'anomalie dans les données.

Commande SQL

```
WITH
  ProduitsDeBase (Niveau, Id, Qte) AS
  (
    SELECT 0, Racine.IdComposant, Racine.Qte
    FROM Recettes Racine
    WHERE Racine.IdCompose = 1

    UNION ALL

    SELECT Parent.Niveau + 1, Enfant.IdComposant, Enfant.Qte * Parent.Qte
    FROM ProduitsDeBase Parent, Recettes Enfant
    WHERE Parent.Id = Enfant.IdCompose AND Parent.Niveau < 10
  )
SELECT ProduitsDeBase.Id, Sum(Qte), Produits.Nom
FROM ProduitsDeBase, Produits
WHERE ProduitsDeBase.Id = Produits.ID AND Produits.Type = 0
GROUP BY ProduitsDeBase.Id, Produits.Nom
ORDER BY Produits.nom
```

Résultat

ID	2	NOM
4	6	D
5	17	E
6	4	F

3 record(s) selected.

La profondeur maximum a, dans ce cas, été limitée à dix.

V - Conclusion

Souvent méconnu, le SQL récursif est un outil puissant et pratique qui permet d'économiser un trafic important entre le serveur et le client qui autrement aurait dû assurer la récursivité par une succession de lecture. Toutefois, il est impératif de contrôler la récursion sous peine de voir le serveur entrer en boucle ou créer une table intermédiaire d'une taille astronomique.

