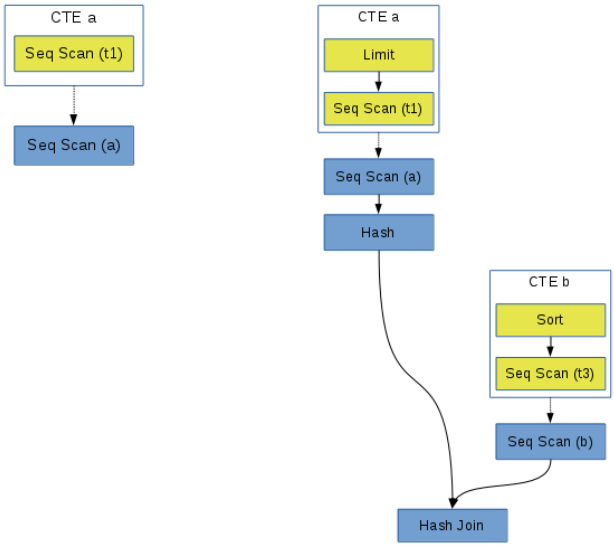
PostgreSQL
Architecture et notions avancées

3e édition
Version imprimée 3.0

par Guillaume Lelarge et Julien Rouhaud

ERRATUM

Page 278 du livre imprimé, une figure a été omise, illustrant l'utilisation de deux CTE dans une requête appelante



Ci-après, successivement, la page erronée et la page corrigée.

L'exécution se passe ainsi. Tout d'abord, la table `t1` est parcourue de façon séquentielle. Le résultat est utilisé par le parcours de CTE, pendant lequel un filtre est exécuté pour ne récupérer que les lignes dont la colonne `id` a une valeur inférieure à 10.

Avec deux CTE, cela peut donner ceci :

```
Hash Join  ...
  Hash Cond: (b.c1 = a.id)
  CTE a
    ->  Limit  ...
          ->  Seq Scan on t1  ...
  CTE b
    ->  Sort  ...
          ->  Seq Scan on t3  ...
  ->  CTE Scan on b  ...
  ->  Hash  ...
        ->  CTE Scan on a  ...
              Filter: (id < 10)
```

Enfin, il existe aussi le *Subquery Scan* pour les parcours de sous-requêtes.

Jointures

Une jointure prend deux ensembles de données en entrée et les combine pour produire en sortie un seul ensemble de données.

PostgreSQL implémente trois algorithmes de jointures : `Nested Loop`, `Merge Join` et `Hash Join`. Ces algorithmes disposent de deux variantes supplémentaires (`Semi Join` et `Anti Join`) lors de l'utilisation de jointures externes ou de clauses `EXISTS` et `IN`, ou `Left Join` et `Right Join` pour les jointures de même type.

PostgreSQL gère la parallélisation sur les différents types de jointure (`Nested Loop` et `Merge Join` en version 9.6, et `Hash Join` en version 11).

Nested Loop

Sa traduction littérale serait "boucle imbriquée", et c'est en effet cela. L'exécuteur va lire toutes les lignes du premier ensemble. Pour chaque ligne, il va lire le deuxième ensemble en filtrant par rapport à la condition de jointure. Les lignes résultantes composent le résultat. De ceci, il découle que la durée d'exécution de cette jointure est proportionnelle à la taille des deux ensembles. En fait, elle dépend surtout de la recherche dans le deuxième ensemble. Cet algorithme est efficace quand le deuxième ensemble est très petit ou lorsqu'un index permet de récupérer rapidement les lignes satisfaisant la condition de jointure. Dans le second cas, on parle de *Parameterized Nested Loop*.

L'exécution se passe ainsi. Tout d'abord, la table `t1` est parcourue de façon séquentielle. Le résultat est utilisé par le parcours de CTE, pendant lequel un filtre est exécuté pour ne récupérer que les lignes dont la colonne `id` a une valeur inférieure à 10.

Avec deux CTE, cela peut donner ceci :

```
Hash Join  ...
  Hash Cond: (b.c1 = a.id)
  CTE a
    ->  Limit  ...
          ->  Seq Scan on t1  ...
  CTE b
    ->  Sort  ...
          ->  Seq Scan on t3  ...
  ->  CTE Scan on b  ...
  ->  Hash  ...
        ->  CTE Scan on a  ...
              Filter: (id < 10)
```

Les schémas de la Figure 10.1 tentent de représenter graphiquement cela.

Figure 10.1 : *Un CTE unique (droite) et deux CTE (gauche) dans une requête appelante*