420-C42

Langages d'exploitation des bases de données

Partie 13

DQL V

Expression de table commune et requêtes récursives

Expression de table commune WITH

- Les ETC (**E**xpressions de **T**able **C**ommune ou, en anglais, *CTE* pour ***C**ommon **T**able **E**xpressions*) permettent de définir des sous-requêtes temporaires utilisées dans une requête principale.
- Elles améliorent la lisibilité et la réutilisation du code SQL, notamment pour les requêtes complexes.
- Elles offrent ces caractéristiques :
  - Définissent une table temporaire utilisable uniquement dans la requête principale avec les optimisations appropriées.
  - Peuvent être utilisées plusieurs fois dans la requête principale.
  - Améliorent la modularité des requêtes SQL.
  - Peuvent être récursives (présentée plus loin).
- Elles font partie de la norme SQL (depuis SQL3:1999) mais ne sont pas toujours implémentées dans les différents SGBD (le sont dans PostgreSQL).

Expression de table commune WITH

- La clause WITH permet la définition des ETC. Voici le synopsis de base:

```
WITH cte_name_1 AS (
    requête_sql
) [,… cte_name_2 …]
SELECT * FROM cte_name_1;
```

- La clause WITH crée et exécute les requêtes internes cte_name_1, cte_name_2, cte_name_n avant la requête principale identifiée par le dernier SELECT.
- Toutes les requêtes peuvent utiliser les résultats obtenus par les requêtes antérieures.
- Au final, les résultats intermédiaires générés par WITH sont supprimés.

Expression de table commune WITH

- Exemple 1 : Sélection des employés en informatique. Requête triviale où le WITH n’est pas nécessairement pertinent.

```
WITH filtered_employees AS (
    SELECT id,
                name,
                department
    FROM employees
    WHERE department = 'Informatique'
)
SELECT * FROM filtered_employees;
```

Expression de table commune WITH

- Exemple 2 : Identifie les départements où le salaire moyen est supérieur à 50 000, en calculant préalablement ce salaire moyen pour chaque département.

```
WITH ventes_par_categorie AS (
    SELECT categorie,
                SUM(montant) AS total_ventes
    FROM ventes
   GROUP BY categorie
)
SELECT categorie, total_ventes
  FROM  ventes_par_categorie
WHERE total_ventes > 50000;
```

Expression de table commune WITH

- Exemple 3 : Cet exemple extrait les employés ayant un haut salaire, puis filtre ceux qui ont été embauchés avant une certaine année.

```
WITH high_salaries AS (
    SELECT id, name, salary, hire_date
    FROM employees
    WHERE salary > 70000
), old_rich_employees AS (
    SELECT id, name, salary, hire_date
    FROM high_salaries
    WHERE EXTRACT(YEAR FROM hire_date) < 2015
)
SELECT id, name, salary
FROM old_rich_employees;
```

Expression de table commune WITH

- Exemple 4 : Cet exemple sélectionne les employés ayant un salaire supérieur à 70 000 et travaillant dans des départements comptant plus de 10 employés.

```
WITH high_salaries AS (
    SELECT id, name, salary, department_id
    FROM employees
    WHERE salary > 70000
), large_departments (department_id, employee_count)  AS (    -- avec noms de col
    SELECT department_id, COUNT(*)                            -- ou AS employee_count
    FROM employees
    GROUP BY department_id
    HAVING COUNT(*) > 10
)
SELECT hs.id, hs.name, hs.salary, ld.employee_count
FROM high_salaries hs
  JOIN large_departments AS ld
    ON hs.department_id = ld.department_id;
```

ETC récursives WITH RECURSIVE

- Les ETC permettent des requêtes récursives qui permettent des résultats impossibles autrement.
- Elles sont particulièrement utiles pour traiter des structures hiérarchiques telles que les arbres et les graphes. Par exemple:
  - organigrammes
  - relations parent-enfant
  - chemins dans un graphe
  - sous-composants dans un composant
- On suppose la notion de récursivité déjà acquise.
- Néanmoins, on rappelle qu'elle implique deux notions fondamentales :
  - logique récurrente avec réduction de la taille du problème
  - condition d'arrêt

CTE récursives WITH RECURSIVE

- Le synopsis général des requêtes récursives est :

```
WITH RECURSIVE cte AS (                          -- <  nom de la cte
    -- Partie d'ancrage
    initial_query                                -- <  conditions de départ

    UNION [ALL | DISTINCT]                       -- <  fusion d'information

    -- Partie récursive
    recursive_query (référence à cte)            -- <  appel récursif en faisant référence à cte*
    -- SELECT ...                                -- <  même schéma que initial_query
    -- FROM cte                                  -- <  *même nom que la cte
    -- WHERE x < y;                              -- <  critère d'arrêt
) -- Requête principale
SELECT * FROM cte;
```

Requête récursive WITH RECURSIVE

- Exemple 1 : Requête récursive très simple qui génère une série de nombres. Produit les nombres : 1, 2, 3, 4, 5

```
WITH RECURSIVE compteur(n) AS (
    VALUES (1)
    UNION ALL
    SELECT n + 1 FROM compteur WHERE n < 5
)
SELECT n FROM compteur;
```

Requête récursive WITH RECURSIVE

- Exemple 2 : Calcule 3^5.

```
WITH RECURSIVE Exponentielle(base, exposant, resultat) AS (
    -- Cas de base: Initialiser les valeurs
    VALUES (3, 5, 1)  -- base=3, exposant=4, résultat initial=1

   UNION ALL

    -- Cas récursif: Multiplier le résultat par la base et décrémenter l'exposant
    SELECT base, exposant - 1, resultat * base
    FROM Exponentielle
    WHERE exposant > 0 -- Condition d'arrêt : exposant atteint 1
)
-- Sélectionner le résultat final     -- > si c'était
SELECT resultat                       -- > SELECT *
FROM Exponentielle                    -- > FROM Exponentielle;
WHERE exposant = 0;
```

	base integer	exposant integer	resultat integer
1	3	5	1
2	3	4	3
3	3	3	9
4	3	2	27
5	3	1	81
6	3	0	243

Requête récursive WITH RECURSIVE

- Exemple 3 : Requête récursive déterminant la hiérarchie d'employés liée au superviseur.

```
WITH RECURSIVE hierarchie(nas, nom, boss, niveau) AS (
    -- Partie d'ancrage : Sélectionner les employés sans superviseur (niveau 1)
    SELECT nas, nom, '-'::VARCHAR(32) AS boss, 1 AS niveau
    FROM employe
    WHERE superviseur IS NULL

    UNION ALL

    -- Partie récursive : Trouver les employés supervisés et incrémenter le niveau
    SELECT e.nas, e.nom, h.nom, h.niveau + 1
    FROM employe e
    JOIN hierarchie h ON e.superviseur = h.nas
)
-- Requête principale : Sélectionner tous les employés
SELECT nas, nom, boss, niveau
FROM hiérarchie
ORDER BY niveau, nom;
```

	nas integer	nom character varying (32)	boss character varying (32)	niveau integer
1	111	Dupuis	-	1
2	222	Bordeleau	Dupuis	2
3	444	Lebel	Dupuis	2
4	123	Sasseur	Dupuis	2
5	555	Tangay	Dupuis	2
6	777	Brochant	Bordeleau	3
7	666	Brochant	Bordeleau	3
8	333	Fontaine	Bordeleau	3
9	999	Leblanc	Tangay	3
10	888	Pignon	Tangay	3