

SGBD : EXPLOITATION D'UNE BASE DE DONNÉES [R206]

TD(TP) N°1 – 2 - PROGRAMMATION AVANCÉE EN SQL

OBJECTIFS

- Programmation avancée en SQL
- Fonctions Stockées en SQL

ENONCÉS

Exercice I : `case, array`

Dans un langage procédural, une permutation de deux données `a` et `b` peut être exprimée par le code suivant :

```
1  temp=a;
2  a=b;
3  b=temp;
```

Question 1.1. *Ecrire une fonction `SQL` `permute` qui permet de permuter les valeurs de deux données `a` et `b` transmises en paramètres, comme dans l'exemple :*

```
(abir) [abir] => select a,b from permute(1,2);
a | b
---+---
2 | 1
(1 row)
```

On s'intéresse maintenant à la permutation de deux valeurs consécutives de rang `j-1` et `j` d'un tableau `t` ayant un nombre effectif de `n` éléments, comme décrit par le code procédural suivant :

```
1  if (j>=2 && j<=n) -- limites
2  {
3      if (t[j-1] > t[j]) -- ordre décroissant
4      {
5          temp=t[j];    -- permuter
6          t[j]=t[j-1];
7          t[j-1]=temp;
```

Date: February 1, 2023.

Hocine ABIR - IUT Villetaneuse .

```

8     }
9 }

```

Question 1.2. *Ecrire une fonction SQL `arp` qui prend en entrée deux paramètres :*

- (1) *un tableau `t` d'entiers*
- (2) *un entier `j` (indice dans `t`)*

et qui retourne le tableau `t` après avoir permuter les entrées `j-1` et `j` de `t` comme indiqué par le code ci-dessus.

Exercice II : `cte`

On considère la cte `pyramide` suivante :

```

1  /*
2   Pyramide
3  */
4  WITH RECURSIVE pyramide (h) AS
5  (
6      SELECT 1
7      UNION
8      SELECT h + 1
9          FROM pyramide
10         WHERE h < 4
11 )
12 SELECT repeat(' ', 4-h) || -- Blancs
13        repeat('*', 2*h-1) as "Pyramide" -- Etoiles
14 FROM pyramide;

```

Question 2.1. *Reécrire cette requête sans utiliser le deuxième appel*

`repeat('*', 2*h-1) as "Pyramide" -- Etoiles`
de la fonction `repeat`.

Question 2.2. *Reécrire la requête obtenue à la Question 2.1 sans utiliser le premier appel*

`repeat(' ', 4-h) | - Blancs/`
de la fonction `repeat`.

Question 2.3. *Modifier Votre requête pour afficher la pyramide inversée comme dans l'exemple ci-dessous :*

```

Pyramide
-----
*****
*****
***

```

```

*
(4 rows)

```

Question 2.4. *A partir de la requête obtenue à la question 2.2, décrire une fonction SQL qui permet d'obtenir une pyramide de hauteur arbitraire, comme dans l'exemple suivant :*

```

(abir) [abir] => select pyramid(3);
pyramid
-----

```

```

*
***
*****
(3 rows)

```

```

(abir) [abir] => select pyramid(5);
pyramid
-----

```

```

*
***
*****
*****
*****
(5 rows)

```

Exercice III : tri à bulle

Dans cet exercice, nous allons nous intéresser au tri à bulle dont le principe est résumé dans le code suivant pour un tableau `t` ayant `n` éléments :

```

1  for (int i=n; i<=2;i--)
2  {
3      for (int j=2;j<=i;j++)
4      {
5          if (t[j-1] > t[j]) -- ordre decroissant
6          {
7              temp=t[j];    -- permuter
8              t[j]=t[j-1];
9              t[j-1]=temp;
10         }
11     }
12 }

```

On considère la fonction SQL `nestedloop` suivante :

```

1 CREATE FUNCTION nestedloop(int [],
2                               out i int,out j int)
3 RETURNS SETOF record AS
4 $$
5 WITH RECURSIVE inl(k,f) AS
6 (
7     SELECT array_upper($1,1),2
8     UNION
9     SELECT k ,f+1
10    FROM inl
11    WHERE f<k
12 ),
13 outl (i,j) AS
14 (
15     SELECT k,f FROM inl
16     union
17     SELECT i-1 , j
18    FROM outl
19    WHERE i>j
20 )
21 SELECT * FROM outl;
22 $$ language SQL;

```

Question 3.1.

3.1.1. Créer la fonction `nestedloop` en exécutant la commande ci-dessus.

3.1.2. Exécuter la requête suivante :

```
select * from nestedloop(array[5,2,3,6,7]);
```

et étudier les résultats obtenus.

Question 3.2. Décrire une fonction `tribulle` qui permet de trier un tableau d'entier en utilisant les boucles imbriquées (voir la question 3.1) comme dans l'exemple suivant :

```
(abir) [abir] => select tribulle(array[7,5,2,4,3,1,6]);
tribulle
-----
{1,2,3,4,5,6,7}
(1 row)
```

Suggestion : décrire d'abord une fonction `bulle` qui sera appelée par `tribulle`. Pensez à utiliser la fonction `arp` de l'exercice précédent.