TD 5 : Initiation au langage PL/pgSQL

Objectifs : Créer des fonctions en pl/pgSQL en :
• utilisant les structures de contrôle de PL/pgSQL ;
• manipulant les fonctions prédéfinies dans postgreSQL

Sommaire

Exercice 1 :

Écrire la fonction calculer_longueur_max() qui calcule la longueur de deux chaînes de caractères fournies en argument et qui retourne la longueur la plus longue.
• Arguments en entrée : 2 chaînes de caractères.
• Argument en sortie : longueur maximale des deux chaînes de caractères.

```
CREATE OR REPLACE FUNCTION calculer_longueur_max(chaine1 VARCHAR, chaine2 VARCHAR)
  RETURNS TEXT
  LANGUAGE plpgsql
  AS $plpgsql$
  DECLARE a1 INT; a2 INT; x TEXT;
  BEGIN
   a1=CHAR_LENGTH(chaine1);
   a2=CHAR_LENGTH(chaine2);
  IF a1>a2 THEN
    x = (chaine1,'est plus long');
  ELSIF a1<a2 THEN
    x = (chaine2,'est plus long');
  ELSE
     x = 'Ils sont de la même longueur';
  END IF;
  RETURN x;
  END;
$plpgsql$;

SELECT calculer_longueur_max('salut', 'test');
```

calculer_longueur_max
text
(salut,"est plus long")

Exercice 2 :

Écrire la fonction nb_occurrences() qui compte et retourne le nombre d'occurrences d'un caractère dans un intervalle d'une chaîne de caractères. L'intervalle est indiqué par une position de départ et de fin dans la chaîne.

- Arguments en entrée :
  - paramètre 1 : un caractère ;
  - paramètre 2 : chaîne de caractères ;
  - paramètre 3 : indice de début de l'intervalle ;
  - paramètre 4 : indice de la fin de l'intervalle.
- Argument en sortie : nombre d'occurrences du caractère dans l'intervalle.

```
CREATE OR REPLACE FUNCTION nb_occurence(carac TEXT, chaine TEXT, i_debut INT, i_fin INT)
  RETURNS INT
  LANGUAGE plpgsql
  AS $plpgsql$
  DECLARE a1 text;
          nb0 INT;
          chaine1 text;
  BEGIN
   chaine1 = chaine;
  nb0 = 0;
  FOR i IN i_debut..i_fin LOOP
    a1=SUBSTR(chaine1, i, i_fin);
    IF a1=carac THEN
      nb0 = nb0+1;
      END IF;
      END LOOP;
    RETURN nb0;
    END;
$plpgsql$;

SELECT nb_occurences ('a', 'allo', '1', '4');
```

nb_occurences
varchar
1

Exercice 3 :

Écrire la fonction getNbJoursParMois() qui calcule le nombre de jours dans un mois pour une date fournie en paramètre.
• Argument en entrée : une date (sous format SQL).
• Argument en sortie : nombre de jours du mois de la date

```
CREATE OR REPLACE FUNCTION getNbJourParMois(myDate DATE) RETURNS DATE AS
  $$
    SELECT (date_trunc('MONTH', $1) + INTERVAL '1 MONTH - 1 day')::DATE;

 $$
LANGUAGE 'sql';

SELECT DATE_PART('day', getNbJourParMois('2020-11-05'));
```

date_part double...
30

Exercice 4 :

Écrire la fonction dateSqlToDatefr() qui vous permet de convertir la date fournie en paramètre au format SQL en une date au format JJ/MM/AA.
• Argument en entrée : une date au format AAAA-MM-JJ
• Argument en sortie : une date au format JJ/MM/AA.

```sql
CREATE OR REPLACE FUNCTION dateSqlToDatefr(myDate DATE)
 RETURNS DATE
  LANGUAGE plpgsql
   AS $plpgsql$
  BEGIN
    RETURN TO_CHAR(myDate, 'DD/MM/YYYY');
    END;
$plpgsql$;

SELECT dateSqlToDatefr('2020-10-12');
```

datesqltodatefr date
12/10/2020

Exercice 5 :

Écrire la fonction getNomJour() qui vous permet de retourner le nom du jour de la semaine correspondant à la date fournie en paramètre.
• Argument en entrée : une date au format SQL
• Argument en sortie : le nom du jour de la date.
Contrainte : utiliser la fonction extract().

```
CREATE OR REPLACE FUNCTION getNomJour(myDate DATE)
 RETURNS VARCHAR
  LANGUAGE plpgsql
   AS $plpgsql$
  DECLARE
    b INT;
    c TEXT[];

  BEGIN
    b=EXTRACT(ISODOW FROM myDate);
    c='{"Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"}';
    RETURN c[b];
    END;
$plpgsql$;

SELECT getNomJour('2020-10-12');
```

getnomjour
varchar
Lundi

Exercice 6 :

Contexte : HHB-Direct (Ressource : e-sio/slam3/hhb_direct_v1.sql)
Écrire une fonction qui retourne le nombre de clients débiteurs.
Contrainte : utiliser la fonction date_part() de PostgreSQL.
Écrire une fonction qui retourne le nombre de clients habitant dans une ville. Le nom de la ville est un paramètre de la fonction.

```
CREATE OR REPLACE FUNCTION countClient(count VARCHAR)
  RETURNS INT
  LANGUAGE plpgsql
  AS $plpgsql$

  BEGIN
    RETURN DISTINCT COUNT(DISTINCT client.num_client)
    FROM client
      WHERE client.adresse_client LIKE('%'||count||'%');
  END;

$plpgsql$;

SELECT countClient('LANNIO');
```

countclient int
2