MAMAN Alexandre
BTS SIO2

Compte rendu TP :
Langage PL/pgSQL

Sommaire :

1) Contexte

Ce TP consiste à continuer à nous initier au langage PL/PGSQL.

2) Exercices

Exercice 1 :

Exécuter les fonctions suivantes et indiquer le résultat obtenu.

```
CREATE OR REPLACE FUNCTION getOneTuple() RETURNS record
LANGUAGE plpgsql
AS
$function$
declare
res record ;
begin
select into res * from client ;
return res ;
end;
$function$ ;



CREATE OR REPLACE FUNCTION getOneTupleTable()
RETURNS SETOF client
LANGUAGE plpgsql
AS $function$
begin
return query select * from client ;
end;
$function$ ;



CREATE OR REPLACE FUNCTION lesClients() RETURNS SETOF client
LANGUAGE plpgsql
AS $function$
declare
res client%ROWTYPE ;
begin
for res in select * from client
loop
return next res ;
end loop;
return;
end ;
$function$ ;


Select getOneTuple();
Select getOneTupleTable();
Select lesClients();
```

On reprend les fonctions données en énoncé puis on Select getOneTuple(), Select getOneTupleTable(), Select lesClients().

```
(1,DUPONT,Pierre,"5 Rue du Port, 22300 LANNION",Pdupont,Pdupont)
```

En exécutant la fonction, cela nous renvoie la première ligne de la table client.

Exercice 2 :

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 Nbclientsville(ville VARCHAR)
RETURNS INT
LANGUAGE plpgsql
AS $plpgsql$
  BEGIN
  RETURN DISTINCT COUNT (DISTINCT client.num_client)
  FROM client
  WHERE client.adresse_client LIKE ('%'||ville||'%');
  end;
$plpgsql$;

SELECT Nbclientsville('LANNION');
```

On commence par écrire la requête qui permet de retourner le nombre de clients habitant dans une ville fourni en paramètre «SELECT * FROM client WHERE adresse_client=$1 ; ». Ensuite on entre l'adresse voulu.

nbclientsville
int
2

En exécutant la fonction, cela nous renvoie qu'à 'Lannion', il y a 2 client qui habitent ici.

Exercice 3 :

Écrire une fonction qui retourne le nombre de clients débiteurs.

```
CREATE OR REPLACE FUNCTION NbClients()
RETURNS SETOF client
LANGUAGE plpgsql
AS $function$
begin
return query SELECT COUNT(num_client) FROM client ;
end;
$function$ ;


Select NbClients();
```

On écrit la requête qui permet de retourner le nombre de clients débiteurs « SELECT COUNT(num_client) FROM client ; »

Exercice 4 :

Écrire une fonction qui permet d'insérer un tuple dans la table client. Les valeurs des colonnes seront fournies en argument à la fonction sauf pour le numéro du client. Le numéro du client devra être calculé dans le fonction. La fonction récupère le dernier numéro et l'incrémente pour créer le nouveau numéro du client à enregistrer. Une information devra indiquer si le tuple a été bien enregistré ou pas

```
CREATE OR REPLACE FUNCTION Insertion(num,nom,prenom,adresse,identifiant,mdp)
RETURNS SETOF client
LANGUAGE plpgsql
AS $function$
begin
return query INSERT INTO client
  (num_client,nom_client,prenom_client,adresse_client,identifiant_internet,mdp_internet)
  VALUES ($1,$2,$3,$4,$5,$6);
end;
$function$ ;


SELECT Insertion('550','Maman','Alexandre','54Ad Georges Pompidou','MAlexandre','azerty12');
```

On commence par écrire la requête qui permet d'insérer un tuple dans la table client avec les valeurs fournies en paramètre « INSERT INTO client (num_client,nom_client,prenom_client,adresse_client,identifiant_internet,mdp_internet) VALUES ($1,$2,$3,$4,$5,$6); ». Ensuite on entre les différentes valeurs dans les différentes colonne.

Exercice 5 :

Écrire une fonction nommée nb_operation_compte_mois() qui permet de calculer le nombre d'opérations pour un mois passé en argument pour un compte bien précis dont le numéro est passé en argument.

```
CREATE OR REPLACE FUNCTION  nb_operation_compte_mois(mois INT, year INT, numcompte int)
RETURNS INT
LANGUAGE plpgsql
AS $plpgsql$
BEGIN
  RETURN COUNT(DISTINCT operation.id_operation)
  FROM operation
  WHERE numcompte=operation.num_compte
  AND date_part('month',operation.date)=mois
  AND date_part('year',operation.date)=year;
END;
$plpgsql$;


SELECT nb_operation_compte_mois(10, 2016, 5) ;
```

On commence par écrire la requête qui permet de calculer le nombre d'opérations pour un mois pour un compte donnée en paramètre : « SELECT COUNT(id_operation) FROM compte,operation WHERE compte.id_type=operation.id_type AND num_compte=$1 AND date=$2; ». Ensuite on entre le numéro de compte en paramètre.

nb_operation_compte_mois int
0

En exécutant la fonction, cela nous renvoie qu'il n'y a eu aucune opération à la date choisie.

Exercice 6 :

Écrire une fonction nommée creer_id_internet() qui crée l'identifiant et le mot de passe internet du client passé en argument.

```
CREATE OR REPLACE FUNCTION create_id_internet(numcliente INT, nom VARCHAR, prenom VARCHAR) RETURNS VARCHAR
LANGUAGE plpgsql
AS $plpgsql$
 DECLARE
  var1 TEXT;
  var2 TEXT;
  var3 TEXT;
BEGIN
  var1 = UPPER(SUBSTR(prenom, 0,2)) ;
  var2 = LOWER(nom);
  var3 = var1 || var2;
  UPDATE client SET identifiant_internet = var3, mdp_internet=var3 WHERE client.num_client=numcliente;
  RETURN var3;
END;
$plpgsql$;


SELECT create_id_internet(5,'Maman', 'Alexandre') ;
```

On commence par écrire la requête qui permet de créer l'identifiant et le mot de passe internet du client passé en paramère : « INSERT INTO client (identifiant_internet,mdp_internet) VALUES ($2,$3) WHERE num_client=$1; ». Ensuite on entre l'identifiant et le mot de passe à créer.

create_id_internet varchar
Amaman

En exécutant la fonction, cela nous crée l'identifiant internet « Amaman »

Exercice 7 :

Écrire une fonction nommée creer_date() qui insère dans la table date toutes les dates pour le mois et l'année passées en argument

```sql
CREATE OR REPLACE FUNCTION  getNbJoursParMois(datee date) RETURNS DATE AS
$$
  SELECT (date_trunc('MONTH', $1) + INTERVAL '1 MONTH - 1 day')::DATE;

$$
LANGUAGE 'sql';



CREATE OR REPLACE FUNCTION creer_date(mois varchar, annee varchar) RETURNS VARCHAR
LANGUAGE plpgsql
AS $plpgsql$
DECLARE
    nbjour INT;
    i VARCHAR;
    datee DATE;
BEGIN

 /*   i = annee ||'-' || mois || '-'|| '01';*/
    datee = TO_DATE(annee || mois || '01','YYYYMMDD');
    nbjour = date_part('day',getNbJoursParMois(datee)) ;


    FOR i IN 1..nbJour LOOP
       INSERT INTO DATE(date) VALUES ( TO_DATE(annee || mois || i,'YYYYMMDD'));

END LOOP;

    RETURN TO_DATE(annee || mois || i,'YYYYMMDD');

END;

$plpgsql$;

SELECT creer_date('11','2001') ;
```

On rentre en paramètre un mois et une année. Tous les jours du mois sont ajoutés dans la base de données.

3) Conclusion

Ce TP ne m'a posé aucune difficultés hormis quelques requêtes qui ne fonctionnait pas.