

TD 6 : PostgreSQL

Sommaire :

Exercice 1-----*Page 2.*

Exercice 2-----*Page 3.*

Exercice 3, 4, 5-----*Page 4.*

Exercice 6-----*Page 5.*

Exercice 7-----*Page 6,7.*

Exercice 8, Conclusion-----*Page 8.*

Exercice 1 : Exécuter les fonctions suivantes et indiquer le résultat obtenu

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

```
SELECT getOneTupleTable();
```

getonetupletable
public.client

1,DUPONT,Pierre,"5 Rue du Port, 22300 LANNION",Pdupont,Pdupont)
2,DUPONT,Annie,"5 Rue du Port, 22300 LANNION",Adupont,Adupont)

```
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 lesClients();
```

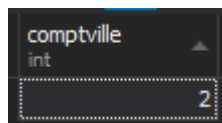
lesclients
public.client

1,DUPONT,Pierre,"5 Rue du Port, 22300 LANNION",Pdupont,Pdupont)
2,DUPONT,Annie,"5 Rue du Port, 22300 LANNION",Adupont,Adupont)
3,DELAVAL,Jean,"12 Bd de l'Orne, 14234 OUISTREHAM",Jdelaval,Jdelaval)
4,HANOT,Eric,"13 Avenue de Neuilly, 94230 NOGENT SUR MARNE",Ehanot,Ehanot)
5,LEVY,Sarah,"1 Rue Neuve, 14110 CONDE SUR NOIREAU",Slevy,Slevy)
6,Test,test,test,test)

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

SELECT comptville('LANNION');
```



comptville
int
2

On met dans le SELECT tout les habitants de la ville donnée, ici LANNION contient 2 habitants.

Exercice 3 : Écrire une fonction qui retourne le nombre de clients débiteurs.

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.

Ces exercices ont été réalisées dans le TD 5 de SLAM 3

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, 2012, 5);
```

nb_operation_compte_mois	
	int
▶	2

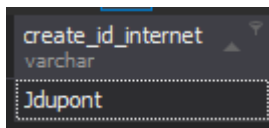
On obtient 2 car on a mis en paramètre de selection les valeurs qui nous intéressait (10, 2012, 5) .

On utilise date_part('month' , unedate) qui permet de récupérer le mois de la date rentrée en paramètre.

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 creer_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 creer_id_internet(42, 'Dupont', 'Jacques');
```



creer_id_internet
varchar
Jdupont

On rentre en paramètre un numéro ainsi qu'un nom et un prénom afin de générer un identifiant en fonction du nom et du prénom.

Ici il s'agit de la première lettre du prénom suivi du nom de famille.

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.

```
CREATE OR REPLACE FUNCTION getNbJoursParMois3(datee date) RETURNS DATE AS
$$
    SELECT (date_trunc('MONTH', $1) + INTERVAL '1 MONTH - 1 day')::DATE;
$$
LANGUAGE 'sql';

CREATE OR REPLACE FUNCTION creer_date2(mois varchar, annee varchar) RETURNS VARCHAR
LANGUAGE plpgsql
AS $plpgsql$
DECLARE
    nbjour INT;
    i VARCHAR;
    datee DATE;
BEGIN
    datee = TO_DATE(annee || mois || '01', 'YYYYMMDD');
    nbjour = date_part('day', getNbJoursParMois3(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_date2('08', '2001') ;
```

01/08/2001
02/08/2001
03/08/2001
04/08/2001
05/08/2001
06/08/2001
07/08/2001
08/08/2001
09/08/2001
10/08/2001
11/08/2001
12/08/2001
13/08/2001
14/08/2001
15/08/2001
16/08/2001
17/08/2001
18/08/2001
19/08/2001
20/08/2001
21/08/2001
22/08/2001
23/08/2001
24/08/2001
25/08/2001
26/08/2001
27/08/2001
28/08/2001
29/08/2001
30/08/2001
31/08/2001

On vérifie que les données ont bien été insérées au mois et à l'année inséré dans la fonction `creer_date2`.

Ici j'ai mis le huitième mois de l'année 2001 ce qui a inséré tout les jours.

Exercice 8 : Écrire une fonction nommée `creer_user_client()` qui crée un utilisateur postgresSQL pour chaque client de la banque.

```
CREATE OR REPLACE FUNCTION creer_user_client() RETURNS VARCHAR
LANGUAGE plpgsql
AS $plpgsql$
DECLARE
    nbcli INT;
    nbcli2 INT;
BEGIN
    nbcli = COUNT(identifiant_internet) FROM client;
    nbcli2 = 0;

    FOR i IN 1..nbcli LOOP
        IF NOT EXISTS (SELECT * FROM pg_user WHERE id_internet = 'identifiant_internet') THEN
            CREATE ROLE identifiant_internet with PASSWORD 'mdp_internet';
            nbcli2 = nbcli2+1;
        END IF;
    END LOOP;

    RETURN nbcli2;
END;
$plpgsql$;

SELECT creer_user_client();
```

Cela crée un utilisateur pour chaque client de la banque présent dans la base de données.

Conclusion : Certains exercices sont compliqués car il faut chercher les bonnes méthodes pour les réaliser.