Bases de Données
-PL/PgSQL -
(Exemples)

Myriam Mokhtari-Brun

Exemple 1 :

```
DROP FUNCTION f1(real);

CREATE FUNCTION f1(real) RETURNS real AS $$
    DECLARE
        total ALIAS FOR $1;
    BEGIN
        RETURN total * 0.196;
    END;
$$ LANGUAGE plpgsql;
```

Exemple d'utilisation :

```
SELECT f1(100);

SELECT medic, prix, f1(prix) AS "Taxes sur prix"
FROM medicament ;
```


Exemple 2 :

```
DROP FUNCTION f2(client.typec%type);
CREATE OR REPLACE FUNCTION f2(client.typec%type) RETURNS
varchar AS $$
DECLARE
    v_typec ALIAS FOR $1;
    v_typeclient varchar;
BEGIN                          --existe aussi avec CASE WHEN
    IF   v_typec='1' THEN v_typeclient:='Particulier';
        ELSIF v_typec='2' THEN v_typeclient:='Entreprise Privée';
        ELSIF v_typec='3' THEN v_typeclient:='Etablissement public';
    END IF;
    RETURN v_typeclient;
END;
$$ LANGUAGE plpgsql;
```

Exemple d'utilisation :

```
SELECT f2('1');
SELECT numcl, nomcl, f2(typec) from client;
```


Exemple 3 :

```
DROP FUNCTION f3();
CREATE OR replace FUNCTION f3() RETURNS void
AS $$
BEGIN
    FOR i IN 1..3                          --idem avec while et LOOP exit
    LOOP
        EXECUTE 'DROP table t'  ||   i   ||   ' cascade';
        RAISE NOTICE 'Table  T% supprimée', i ;
    END LOOP;

END;
$$ LANGUAGE plpgsql;
```

Exemple d'utilisation :

```
SELECT f3();
→ NOTICE: Table T1 supprimée
    NOTICE: Table T2 supprimée
    NOTICE: Table T3 supprimée
```

Exemple 3bis : (avec while et LOOP exit)

```
CREATE OR replace FUNCTION f3bis() RETURNS void
As $$
    DECLARE
        i integer:=1 ;
    BEGIN
        WHILE i <=3
        LOOP
            EXECUTE 'DROP table t' || i || ' cascade';
            RAISE NOTICE 'Table  T% supprimée', i ;
            i:=i+1 :
        END LOOP;
    END ;$$ LANGUAGE plpgsql;

– avec LOOP … EXIT
        LOOP
            IF i>3 THEN EXIT ; END IF ;
            EXECUTE 'DROP table t' || i || ' cascade';
            i:=i+1 :
        END LOOP;
```


Exemple 4 : requête retournant une ligne

```
DROP FUNCTION f4(type_animal.typea%type);

CREATE OR replace FUNCTION  f4(animal.cotypa%type) RETURNS
real AS $$
    DECLARE
        v_poidsmoyen real;
        v_cotypa ALIAS FOR $1;

    BEGIN

        select avg(poids) INTO v_poidsmoyen
        from animal
        where cotypa=v_cotypa;

        RETURN TO_CHAR(v_poidsmoyen,'000.00');
    END;$$ LANGUAGE plpgsql;
```

Exemple d'utilisation :

```
select noman, poids, f4(cotypa) as "poids moyen"
from animal;
```


Exemple 5 : requête retournant une ligne avec test d'existence

```
CREATE OR replace FUNCTION  f5(animal.cotypa%type) RETURNS
real AS $$
    DECLARE
        v_poidsmoyen real;
        v_cotypa ALIAS FOR $1;

    BEGIN
        select avg(poids) INTO v_poidsmoyen from animal
        where cotypa=v_cotypa;
        IF FOUND
            THEN RETURN TO_CHAR(v_poidsmoyen,'00.00');
            ELSE RAISE EXCEPTION 'Type animal inconnu';
        END IF;
END ;$$ LANGUAGE plpgsql;
```

Exemple d'utilisation :

```
SELECT f5('99')
→ Erreur:  Type animal inconnu
```


Exemple 6 : requête retournant une ligne

```
/*Nombre d'animaux ayant subi le traitement de libellé x donné en
paramètre. Ne pas compter 2 fois le même animal.*/

CREATE OR REPLACE FUNCTION nb_traites(traitement.trait%type)
RETURNS INTEGER AS $$
    DECLARE
        nb INTEGER;
        x ALIAS FOR $1;

    BEGIN
        SELECT COUNT(DISTINCT idani) INTO nb
        FROM visite
        WHERE numvi IN (SELECT DISTINCT numvi
                        FROM detail_visite AS d, traitement AS t
                        WHERE d.cotra= t.cotra AND t.trait= x
                    );
        RETURN nb;
    END; $$ LANGUAGE plpgsql;
```

Exemple 6 suite :

Exemple d'utilisation :

```
SELECT trait, nb_traites(trait) AS "Nb traites" from traitement;

SELECT nb_traites('Brossage');
```

```
/* exemple d'utilisation à travers une autre fonction */
CREATE OR REPLACE FUNCTION appel_nb_traites() RETURNS
void AS $$
    DECLARE
        nb INTEGER;

    BEGIN
        nb:=nb_traites('Brossage');
        RAISE NOTICE 'nb d''animaux ayant subi un brossage %', nb;
    END;
$$ LANGUAGE plpgsql;
```


Exemple 7 : Appel d'une fonction retournant void (=procédure) :

```
CREATE FUNCTION del_client(client.numcl%type) RETURNS void
AS $$
    DECLARE
        v_numcl ALIAS FOR $1;
    BEGIN
        DELETE FROM client where numcl=v_numcl;
    END; $$ LANGUAGE plpgsql;
```

Exemple d'utilisation :

```
/* l'appel peut se faire avec PERFORM */

CREATE FUNCTION appel_del_client() RETURNS int AS $$
    BEGIN
        PERFORM del_client('AC001');
        ...
    END ;$$ LANGUAGE plpgsql;
```


Exemple 8 : requête retournant plus d'une ligne

```
CREATE OR replace FUNCTION f8() RETURNS void
AS $$
    DECLARE
        i RECORD;
    BEGIN
        FOR i IN select tablename from pg_tables
                    where schemaname='mbrun'
        LOOP
            EXECUTE 'drop table ' || i.tablename || ' cascade';
        END LOOP;

END; $$ LANGUAGE plpgsql;
```

Exemple d'utilisation :

```
SELECT f8();
```


Exemple 9 : requête retournant plus d'une ligne

Afficher les n°d'animaux ayant subit le traitement de libellé x donné en paramètre. Ne pas compter 2 fois le même animal.

```
CREATE OR REPLACE FUNCTION animaux_traites(traitement.trait
%type) RETURNS void AS $$
    DECLARE
        i RECORD;
        x ALIAS FOR $1;
        BEGIN
            RAISE NOTICE 'Animaux ayant subit le traitement %', x;
            FOR i IN select  distinct v.idani
                        from visite AS v, detail_visite AS d, traitement AS t
                        where v.numvi= d.numvi and d.cotra=t.cotra
                            and t.trait = x
            LOOP
                RAISE NOTICE i.idani;
            END LOOP;
END; $$ LANGUAGE plpgsql;
SELECT animaux_traites('Brossage');
```

Exemple 10 : fonction retournant une ligne d'une table

```
/*en plpgsql*/
CREATE OR REPLACE FUNCTION f10(char) RETURNS departement
AS $$
    DECLARE
        res departement;
    BEGIN
        SELECT * INTO res from departement where codep=$1;
        return res;
    END ; $$ LANGUAGE plpgsql ;
```

Exemple d'utilisation :

```
SELECT f10 ('30');   //retourne un élément structuré (champs entre ()) :
→  f10
    ----------------
    (30,Gard,0.005)

Select * from f10('30');  //retourne une ligne d'une table :
→ codep | depar | tauta
     -------+ ------- +-------
      30    | Gard  | 0.005
```


Exemple 10bis : fonction retournant une ligne d'une table

```
/*en sql*/
CREATE OR REPLACE FUNCTION f10bis(char)
RETURNS departement AS $$
        SELECT *  from departement where codep=$1;
$$ LANGUAGE sql ;
```

Exemple d'utilisation :

Idem fonction f10.

Exemple 11 : fonction retournant un RECORD

```
CREATE OR REPLACE FUNCTION f11(char) RETURNS RECORD AS
$$
    DECLARE
        res  RECORD;
        BEGIN
            SELECT codep, depar INTO res  from departement
            where codep=$1;
            return res;
        END ; $$ LANGUAGE plpgsql ;
```

Exemple d'utilisation :

```
SELECT f11 ('30');   //retourne un élément structuré (champs entre ()) :
→  f11
    ----------------
    (30,Gard)

Select * from f11('30') as (codep char(2), depar varchar); ;
→ codep |   depar              //retourne une ligne d'une table
    --------+------------
     30     | Gard
```


Exemple 12 : fonction retournant plusieurs lignes d'une table ou de type RECORD

```
CREATE OR REPLACE FUNCTION f12() RETURNS SETOF
departement AS $$
    DECLARE
        res departement%ROWTYPE ;
        BEGIN
            FOR res IN SELECT * from departement
            LOOP
                RETURN NEXT res;
            END LOOP;
            RETURN;
        END ;
$$
```

Exemple 12 suite : fonction retournant plusieurs lignes d'une table ou de type RECORD

Exemple d'utilisation :

SELECT f12 (); //retourne des éléments structurés (champs entre parenthèses) :

→ f13

(12,Aveyron,0.000)
(30, Gard,0.005)
...

Select * from f12(); //retourne les lignes d'une table :

→ codep | depar | tauta
-------+ ------- +-------
12 | Aveyron | 0.000
...