

Solutions proposées TP

BDA : TP4

Enseigné par :

Samir YUCEF

Réalisé par l'étudiant :

- Lucas ZHENG

lucas.zheng@edu.univ-paris13.fr

Table des questions

Exercice 1.....	2
Exercice 2.....	10
Exercice 3.....	10

Contexte

Le travail réalisé dans le cadre de ce TP a été effectué avec PostgreSQL.
Il existe certaines différences de syntaxe entre PostgreSQL et Oracle.

En particulier, PostgreSQL ne permet pas la saisie interactive de données via des instructions comme ACCEPT, contrairement à Oracle.

Par conséquent, les valeurs utilisées dans les scripts ont été insérées manuellement dans le code afin de pouvoir exécuter et tester correctement les programmes.

Exercice 1

1. Écrire une procédure anonyme PL/SQL qui permet de demander à un utilisateur de saisir deux entiers et d'afficher leur somme.

```
DO $$  
DECLARE  
    v_1 INTEGER := 42;  
    v_2 INTEGER := 16;  
BEGIN  
    RAISE NOTICE 'La somme des deux entiers fournis est : %', v_1 + v_2;  
END;  
$$;
```



The screenshot shows a PostgreSQL IDE window titled 'TP5/postgres@PostgreSQL 17*'. The interface includes a toolbar with icons for file operations, query execution, and settings. The 'Query' tab is active, displaying a PL/SQL block with the following code:

```
1 DO $$
2 DECLARE
3     v_1 INTEGER := 42;
4     v_2 INTEGER := 16;
5 BEGIN
6     RAISE NOTICE 'La somme des deux entiers fournis est : %', v_1 + v_2;
7 END;
8 $$;
```

The 'Messages' tab is also active, showing the output of the query:

```
NOTICE: La somme des deux entiers fournis est : 58
DO

Query returned successfully in 28 msec.
```

2. Écrire une procédure anonyme PL/SQL qui permet de demander à un utilisateur de saisir un nombre et d'afficher sa table de multiplication.

```
DO $$
DECLARE
    v_n INTEGER := 9;
BEGIN
    FOR i IN 1..10 LOOP
        RAISE NOTICE '% x % = %', v_n, i, v_n * i;
    END LOOP;
END;
$$;
```

Welcome TP5/postgres@PostgreSQL 17* x

TP5/postgres@PostgreSQL 17

Query Query History

```

1 DO $$
2 DECLARE
3     v_n INTEGER := 9;
4 BEGIN
5     FOR i IN 1..10 LOOP
6         RAISE NOTICE '% x % = %', v_n, i, v_n * i;
7     END LOOP;
8 END;
9 $$;
10

```

Data Output Messages Notifications

```

NOTICE:  9 x 1 = 9
NOTICE:  9 x 2 = 18
NOTICE:  9 x 3 = 27
NOTICE:  9 x 4 = 36
NOTICE:  9 x 5 = 45
NOTICE:  9 x 6 = 54
NOTICE:  9 x 7 = 63
NOTICE:  9 x 8 = 72
NOTICE:  9 x 9 = 81
NOTICE:  9 x 10 = 90
DO

```

Query returned successfully in 651 msec.

3. Écrire une fonction récursive qui permet de retourner x^n , x et n sont deux entiers positifs.

```

CREATE OR REPLACE FUNCTION puissance(x INT, n INT) RETURNS INT AS $$
BEGIN
    IF n = 0 THEN
        RETURN 1;
    ELSE
        RETURN x * puissance(x, n - 1);
    END IF;
END;
$$ LANGUAGE plpgsql;
SELECT puissance(4, 2);

```

Welcome TP5/postgres@PostgreSQL 17* x

TP5/postgres@PostgreSQL 17

Query Query History

```

1 CREATE OR REPLACE FUNCTION puissance(x INT, n INT) RETURNS INT AS $$
2 BEGIN
3     IF n = 0 THEN
4         RETURN 1;
5     ELSE
6         RETURN x * puissance(x, n - 1);
7     END IF;
8 END;
9 $$ LANGUAGE plpgsql;
10 SELECT puissance(4, 2);

```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	puissance integer
1	16

4. Écrire une procédure anonyme PL/SQL qui calcule la factorielle d'un nombre strictement positif saisi par l'utilisateur. Le résultat sera stocké dans une table resultatFactoriel.

```

CREATE TABLE IF NOT EXISTS resultatFactoriel
(
    valeur INT,
    factorielle INT
);

DO $$
DECLARE
    v_n INT := 5;
    res INT := 1;
BEGIN
    FOR i IN 1..v_n LOOP

```

```

        res := res * i;
    END LOOP;

    INSERT INTO resultatFactoriel VALUES (v_n, res);
    RAISE NOTICE 'Factorielle de % = %', v_n, res;
END;
$$;

```

Welcome TP5/postgres@PostgreSQL 17* x

TP5/postgres@PostgreSQL 17

Query Query History

```

1 CREATE TABLE IF NOT EXISTS resultatFactoriel
2 (
3     valeur INT,
4     factorielle INT
5 );
6
7 DO $$
8 DECLARE
9     v_n INT := 5;
10    res INT := 1;
11 BEGIN
12     FOR i IN 1..v_n LOOP
13         res := res * i;
14     END LOOP;
15
16     INSERT INTO resultatFactoriel VALUES (v_n, res);
17     RAISE NOTICE 'Factorielle de % = %', v_n, res;
18 END;
19 $$;
20 select * from resultatFactoriel;

```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	valeur integer	factorielle integer
1	5	120

5. Modifier le programme précédent pour qu'il calcule et stocke dans une table `resultatsFactoriels` les factorielles des 20 premiers nombres entiers.

```
CREATE TABLE IF NOT EXISTS resultatsFactoriels
(
    valeur INT,
    factorielle BIGINT
);
DO $$

DECLARE
    n INT;
    res BIGINT;
BEGIN
    FOR n IN 1..20 LOOP
        res := 1;
        FOR i IN 1..n LOOP
            res := res * i;
        END LOOP;
        INSERT INTO resultatsFactoriels VALUES (n, res);
    END LOOP;
    RAISE NOTICE 'Factorielles insérées.';
END;
$$;
```


Welcome TP5/postgres@PostgreSQL 17* x

TP5/postgres@PostgreSQL 17

Query Query History

```

1 CREATE TABLE IF NOT EXISTS resultatsFactoriels
2 (
3     valeur INT,
4     factorielle BIGINT
5 );
6 DO $$
7
8 DECLARE
9     n INT;
10    res BIGINT;
11 BEGIN
12     FOR n IN 1..20 LOOP
13         res := 1;
14         FOR i IN 1..n LOOP
15             res := res * i;
16         END LOOP;
17         INSERT INTO resultatsFactoriels VALUES (n, res);
18     END LOOP;
19     RAISE NOTICE 'Factorielles insérées.';
20 END;
21 $$;
22 select * from resultatsFactoriels;

```

Data Output Messages Notifications

	valeur integer	factorielle bigint
1	1	1
2	2	2
3	3	6
4	4	24
5	5	120
6	6	720
7	7	5040
8	8	40320
9	9	362880
10	10	3628800

Exercice 2

- Créer la table et y insérer quelques données. Tester vos méthodes au fur et à mesure.

```
CREATE TABLE IF NOT EXISTS emp (  
    matr    INT          PRIMARY KEY,  
    nom VARCHAR(50) NOT NULL,  
    sal    NUMERIC(7,2),  
    adresse VARCHAR(96),  
    dep    INT          NOT NULL  
);
```

- Ecrire un bloc anonyme qui permet d'insérer un nouveau employé, dont les valeurs des attributs matr, nom, sal, adresse et dep sont respectivement 4, Youcef, 2500, avenue de la République, 92002.

DO \$\$

DECLARE

 v_emp emp%ROWTYPE;

BEGIN

 v_emp.matr := 4;

 v_emp.nom := 'Youcef';

 v_emp.sal := 2500;

 v_emp.adresse := 'avenue de la République';

 v_emp.dep := 92002;

 INSERT INTO emp VALUES (v_emp.*);

 RAISE NOTICE 'Employé inséré : % - %', v_emp.matr, v_emp.nom;

END;

\$\$;

- Ecrire un bloc anonyme qui permet de supprimer tous les clients dont le dep est connu, et utiliser la fonction ROWCOUNT pour afficher le nombre de n-uplets supprimés.

DO \$\$

DECLARE

 v_nb_lignes INT;

BEGIN

 DELETE FROM emp

 WHERE dep IS NOT NULL;

 GET DIAGNOSTICS v_nb_lignes = ROW_COUNT;

```

    RAISE NOTICE 'Nombre de lignes supprimées : %', v_nb_lignes;
END;
$$;

```

- Ecrire un bloc anonyme qui permet d'afficher la somme des salaires de tous les employés, en utilisant un curseur explicite et la boucle LOOP AND LOOP. Cette procédure doit vous afficher le même résultat que la requête suivante :

```

DO $$
DECLARE
    v_salaire NUMERIC(7,2);
    v_total   NUMERIC(14,2) := 0;
    cur_sal   CURSOR FOR SELECT sal FROM emp;
BEGIN
    OPEN cur_sal;
    LOOP
        FETCH cur_sal INTO v_salaire;
        EXIT WHEN NOT FOUND;
        IF v_salaire IS NOT NULL THEN
            v_total := v_total + v_salaire;
        END IF;
    END LOOP;
    CLOSE cur_sal;

    RAISE NOTICE 'total = %', v_total;
END;
$$;

```

- Modifier la procédure précédente pour calculer le salaire moyen. Cette procédure doit vous afficher le même résultat que la requête suivante :

```

DO $$
DECLARE
    v_salaire NUMERIC(7,2);
    v_total   NUMERIC(14,2) := 0;
    v_nombre  INT           := 0;
    cur_sal   CURSOR FOR SELECT sal FROM emp;
BEGIN
    OPEN cur_sal;
    LOOP
        FETCH cur_sal INTO v_salaire;
        EXIT WHEN NOT FOUND;
        IF v_salaire IS NOT NULL THEN

```

```

        v_total := v_total + v_salaire;
        v_nombre := v_nombre + 1;
    END IF;
END LOOP;
CLOSE cur_sal;

IF v_count > 0 THEN
    RAISE NOTICE 'moyenne = %', v_total / v_count;
END IF;
END;
$$;

```

- Modifier les deux procédures précédentes, en utilisant la boucle FOR IN.
- Somme bis

```

DO $$
DECLARE
    v_total NUMERIC := 0;
    salaire NUMERIC;
BEGIN
    FOR salaire IN SELECT sal FROM emp
    LOOP
        IF salaire IS NOT NULL THEN
            v_total := v_total + salaire;
        END IF;
    END LOOP;

    RAISE NOTICE 'total bis = %', v_total;
END;
$$;

```

- Moyenne bis

```

DO $$
DECLARE
    v_total NUMERIC := 0;
    salaire NUMERIC;
    nombre_sal INT := 0;
BEGIN
    FOR salaire IN SELECT sal FROM emp
    LOOP
        IF salaire IS NOT NULL THEN
            v_total := v_total + salaire;
            nombre_sal = nombre_sal + 1;
        END IF;
    END LOOP;
END;

```

```

        END LOOP;

        RAISE NOTICE 'moyenne bis = %', v_total/nombre_sal;
END;
$$;

```

- Ecrire une procédure anonyme qui permet d'afficher les noms des employés du département 92000 et 75000, en utilisant un curseur paramètre.

- Dans mes données, j'ai Alice dans le département 10 et Youcef dans le département 92002

```

DO $$
DECLARE
    cur_emp CURSOR(p_dep INT) FOR
        SELECT nom FROM emp WHERE dep = p_dep;
    v_emp_rec RECORD;
BEGIN
    -- Département 92002
    OPEN cur_emp(92002);
    LOOP
        FETCH cur_emp INTO v_emp_rec;
        EXIT WHEN NOT FOUND;
        RAISE NOTICE 'Dep 92002 : %', v_emp_rec.nom;
    END LOOP;
    CLOSE cur_emp;

    -- Département 10
    OPEN cur_emp(10);
    LOOP
        FETCH cur_emp INTO v_emp_rec;
        EXIT WHEN NOT FOUND;
        RAISE NOTICE 'Dep 10 : %', v_emp_rec.nom;
    END LOOP;
    CLOSE cur_emp;
END;
$$;

```


TP5/postgres@PostgreSQL 17

Query Query History

```

1 DO $$
2 DECLARE
3     v_emp emp%ROWTYPE;
4 BEGIN
5     v_emp.matr      := 4;
6     v_emp.nom       := 'Youcef';
7     v_emp.sal       := 2500;
8     v_emp.adresse   := 'avenue de la République';
9     v_emp.dep       := 92002;
10
11     INSERT INTO emp VALUES (v_emp.*);
12     RAISE NOTICE 'Employé inséré : % - %', v_emp.matr, v_emp.nom;
13 END;
14 $$;
15 SELECT * FROM emp;

```

Data Output Messages Notifications

Showing rows: 1 to 2 Page No: 1 of 1

	matr [PK] integer	nom character varying (50)	sal numeric (7,2)	adresse character varying (96)	dep integer
1	1	Alice	2500.00	rue de la base de données	10
2	4	Youcef	2500.00	avenue de la République	92002

TP5/postgres@PostgreSQL 17

Query Query History

```

1  DO $$
2  DECLARE
3      v_nb_lignes INT;
4  BEGIN
5      DELETE FROM emp
6      WHERE dep IS NOT NULL;
7      GET DIAGNOSTICS v_nb_lignes = ROW_COUNT;
8      RAISE NOTICE 'Nombre de lignes supprimées : %', v_nb_lignes;
9  END;
10 $$;

```

Data Output Messages Notifications

NOTICE: Nombre de lignes supprimées : 2
DO

Query returned successfully in 49 msec.

On remet les deux lignes supprimées qui ont chacun 2500 sur la colonne "Salaire"

Welcome TP5/postgres@PostgreSQL 17* x

TP5/postgres@PostgreSQL 17

Query Query History

```
1 DO $$
2 DECLARE
3     v_salaire NUMERIC(7,2);
4     v_total   NUMERIC(14,2) := 0;
5     cur_sal   CURSOR FOR SELECT sal FROM emp;
6 BEGIN
7     OPEN cur_sal;
8     LOOP
9         FETCH cur_sal INTO v_salaire;
10        EXIT WHEN NOT FOUND;
11        IF v_salaire IS NOT NULL THEN
12            v_total := v_total + v_salaire;
13        END IF;
14    END LOOP;
15    CLOSE cur_sal;
16
17    RAISE NOTICE 'total = %', v_total;
18 END;
19 $$;
```

Data Output Messages Notifications

NOTICE: total = 5000.00

DO

Query returned successfully in 27 msec.

TP5/postgres@PostgreSQL 17

No limit

E

Query

Query History

1 DO \$\$
2 DECLARE
3 v_total NUMERIC := 0;
4 salaire NUMERIC;
5 BEGIN
6 FOR salaire IN SELECT sal FROM emp
7 LOOP
8 IF salaire IS NOT NULL THEN
9 v_total := v_total + salaire;
10 END IF;
11 END LOOP;
12
13 RAISE NOTICE 'total bis = %', v_total;
14 END;
15 \$\$;

Data Output Messages Notifications

[Query](#) [Query History](#)

```

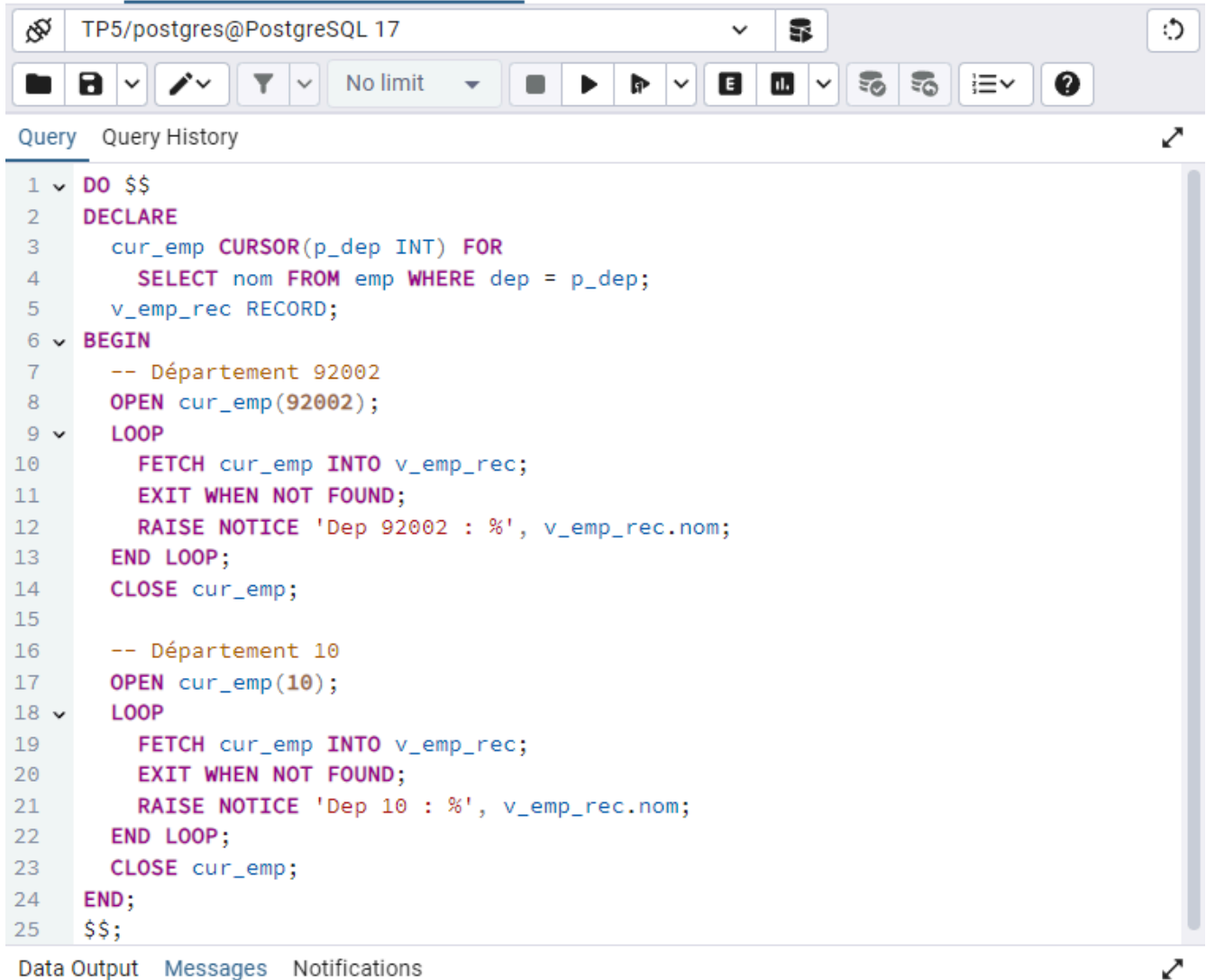
1  DO $$
2  DECLARE
3      v_total NUMERIC := 0;
4      salaire  NUMERIC;
5      nombre_sal INT := 0;
6  BEGIN
7      FOR salaire IN SELECT sal FROM emp
8      LOOP
9          IF salaire IS NOT NULL THEN
10             v_total := v_total + salaire;
11             nombre_sal = nombre_sal + 1;
12          END IF;
13      END LOOP;
14
15      RAISE NOTICE 'moyenne bis = %', v_total/nombre_sal;
16  END;
17  $$;

```

Data Output Messages Notifications

```
NOTICE:  moyenne bis = 2500.000000000000000000
DO
```

Query returned successfully in 27 msec.



TP5/postgres@PostgreSQL 17

Query Query History

```
1 DO $$
2 DECLARE
3     cur_emp CURSOR(p_dep INT) FOR
4         SELECT nom FROM emp WHERE dep = p_dep;
5     v_emp_rec RECORD;
6 BEGIN
7     -- Département 92002
8     OPEN cur_emp(92002);
9     LOOP
10         FETCH cur_emp INTO v_emp_rec;
11         EXIT WHEN NOT FOUND;
12         RAISE NOTICE 'Dep 92002 : %', v_emp_rec.nom;
13     END LOOP;
14     CLOSE cur_emp;
15
16     -- Département 10
17     OPEN cur_emp(10);
18     LOOP
19         FETCH cur_emp INTO v_emp_rec;
20         EXIT WHEN NOT FOUND;
21         RAISE NOTICE 'Dep 10 : %', v_emp_rec.nom;
22     END LOOP;
23     CLOSE cur_emp;
24 END;
25 $$;
```

Data Output Messages Notifications

NOTICE: Dep 92002 : Youcef
NOTICE: Dep 10 : Alice
DO

Query returned successfully in 51 msec.

Exercice 3