Université Sorbonne Paris Nord
École d'ingénieur Sup Galilée

TP n° 4 : Introduction au Procedural Language (PL/SQL) d'Oracle

Rapport de TP

Réalisé par :

Célian WIRTZ (celian.wirtz@edu.univ-paris13.fr)

12100311

Date • 28/04/25

Année universitaire : 2024/2025

Responsable

M. Youcef

Exercice n° 1

Question 1 :

```
SET SERVEROUTPUT ON;
DECLARE
    a NUMBER;
    b NUMBER;
BEGIN
    DBMS_OUTPUT.PUT('Entrez le premier entier : ');
    a := &a;

    DBMS_OUTPUT.PUT('Entrez le deuxième entier : ');
    b := &b;

    DBMS_OUTPUT.PUT_LINE('La somme est : ' || (a + b));
END;
/
```

Question 2 :

```
DECLARE
    i NUMBER := 0;
    nombre NUMBER;
BEGIN
    DBMS_OUTPUT.PUT('Entrez un nombre : ');
    nombre := &nombre;

    WHILE i <= 10 LOOP
        DBMS_OUTPUT.PUT_LINE(nombre || ' x ' || i || ' = ' || (nombre * i));
        i := i + 1;
    END LOOP;
END;
/
```

Question 3 :

```
CREATE OR REPLACE FUNCTION puissance(x IN NUMBER, n IN NUMBER) RETURN NUMBER IS
BEGIN
    IF n = 0 THEN
        RETURN 1;
    ELSE
        RETURN puissance(x, n-1) * x;
    END IF;
END;
/
```

Question 4 :

```
CREATE TABLE IF NOT EXISTS resultatFactoriel(res IN NUMBER PRIMARY KEY, factorielle
NUMBER);

DECLARE
    a NUMBER := 1;
    i NUMBER := 1;
    n NUMBER;
BEGIN
    DBMS_OUTPUT.PUT('Entrez un nombre > 0 : ');
    n := &n;

    WHILE i <= n LOOP
        a := a * i;
        i := i + 1;
    END LOOP;
    INSERT INTO resultatFactoriel(res, factorielle) VALUES (n, a);
    DBMS_OUTPUT.PUT_LINE(n || '! = ' || a);
END;
/
```

Question 5 :

```
CREATE TABLE IF NOT EXISTS resultatFactoriel(res IN NUMBER PRIMARY KEY, factorielle
NUMBER);

DECLARE
    a NUMBER := 1;
    n NUMBER := 1;
BEGIN
    WHILE n <= 20 LOOP
        a := a * n;
        INSERT INTO resultatFactoriel(res, factorielle) VALUES (n, a);
        DBMS_OUTPUT.PUT_LINE(n || '! = ' || a);
        n := n + 1;
    END LOOP;
END;
/
```

Exercice n° 2

```
CREATE TABLE emp(
    matr NUMBER(10) NOT NULL,
    nom VARCHAR2(50) NOT NULL,
    sal number(7, 2),
    adresse VARCHAR2(96),
    dep NUMBER(10) NOT NULL,
    CONSTRAINT emp_pk PRIMARY KEY(matr)
);
```

-- 1

```
SET SERVEROUTPUT ON;
DECLARE
    v_employe emp%ROWTYPE;
BEGIN
    v_employe.matr := 4;
    v_employe.nom := 'Youcef ';
    v_employe.sal := 2500;
    v_employe.adresse := 'avenue de la République ';
    v_employe.dep := 92002;
    INSERT into emp VALUES v_employe;
    COMMIT;
END;
```

On insère un employé, dont les valeurs des attributs matr, nom, sal, adresse et dep sont respectivement 4, Youcef, 2500, avenue de la République et 92002.

-- 2

```
DECLARE
    nb_lignes NUMBER;
BEGIN
    DELETE FROM emp WHERE dep IS NOT NULL;
    nb_lignes := SQL%ROWCOUNT;
    DBMS_OUTPUT.PUT_LINE('Nombre de n-uplets supprimés : ' || nb_lignes);
    COMMIT;
END;
```

-- 3

```
DECLARE
    v_sal EMP.sal%TYPE;
    v_tot NUMBER := 0;
    CURSOR c_sal IS
        SELECT sal FROM emp;
BEGIN
    OPEN c_sal;
    LOOP
        FETCH c_sal INTO v_sal;
        EXIT WHEN c_sal%NOTFOUND;
        v_tot := v_tot + v_sal;
    END LOOP;
    CLOSE c_sal;
    DBMS_OUTPUT.PUT_LINE('Total : ' || v_tot);
END;
```

-- 4

```
DECLARE
    v_sal EMP.sal%TYPE;
    v_tot NUMBER := 0;
    v_nb NUMBER := 0;
CURSOR c_sal IS
        SELECT sal FROM emp;
BEGIN
    OPEN c_sal;
    LOOP
        FETCH c_sal INTO v_sal;
        EXIT WHEN c_sal%NOTFOUND;
        v_tot := v_tot + v_sal;
        v_nb := v_nb + 1;
    END LOOP;
    CLOSE c_sal;
    DBMS_OUTPUT.PUT_LINE('Total : ' || v_tot);
    DBMS_OUTPUT.PUT_LINE('Moyenne : ' | | v_tot / v_nb);
END;
```

Pour avoir la moyenne, il suffit de diviser par le nombre de valeur.

-- 5

```
DECLARE
    v_tot NUMBER := 0;
    v_nb NUMBER := 0;
BEGIN
    FOR i IN (SELECT sal FROM emp) LOOP
        v_tot := v_tot + i.sal;
        v_nb := v_nb + 1;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Total : ' || v_tot);
    DBMS_OUTPUT.PUT_LINE('Moyenne : ' | | v_tot / v_nb);
END;
```

-- 6

```
DECLARE
    CURSOR c(p_dep EMP.dep%TYPE) IS
        SELECT nom FROM emp WHERE dep = p_dep;
BEGIN
    OPEN c_emp(92000);
    LOOP
        FETCH c_emp INTO v_nom;
        EXIT WHEN c_emp % NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Dep 1 : ' || v_emp.nom);
    END LOOP;
    CLOSE c_emp;

    OPEN c_emp(75000);
    LOOP
        FETCH c_emp INTO v_nom;
        EXIT WHEN c_emp % NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Dep 2 : ' || v_emp.nom);
    END LOOP;
    CLOSE c_emp;
END;
```

Exercice n° 3

Si il manque la table on commence par la créer.

```
CREATE TABLE IF NOT EXISTS CLIENT(
    ID NUMBER PRIMARY KEY,
    NOM VARCHAR2(50),
    ADRESSE VARCHAR2(100),
    EMAIL VARCHAR2(100)
);
```

Ensuite on fait deux procédures qui sont donc surchargés et ont le même nom :

```
CREATE OR REPLACE PACKAGE pkg_c IS
    PROCEDURE add_c(p_id NUMBER, p_nom VARCHAR2);
    PROCEDURE add_c(p_id NUMBER, p_nom VARCHAR2, p_adresse VARCHAR2, p_email VAR-
CHAR2);
END plg_c;
```

On fait le body en gérant les exceptions (ex : on ne peut pas ajouter un client avec un id déjà utilisé par un autre client) :

```
CREATE OR REPLACE PACKAGE BODY pkg_c IS
    PROCEDURE add_c(p_id NUMBER, p_nom VARCHAR2) IS
    BEGIN
        INSERT INTO client (ID, NOM)
        VALUES (p_id, p_nom);
    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Attention un client a déjà cet id !');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('ERROOOOOOORRR : ' || SQLERRM);
    END;

    PROCEDURE add_c(p_id NUMBER, p_nom VARCHAR2, p_adresse VARCHAR2, p_email VAR-
CHAR2) IS
    BEGIN
        INSERT INTO client (ID, NOM, ADRESSE, EMAIL)
        VALUES (p_id, p_nom, p_adresse, p_email);
    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Attention un client a déjà cet id !');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('ERROOOOOOORRR : ' || SQLERRM);
    END;
END pkg_c;
```