TP n° 4 : Introduction à PL/SQL

Exercice n° 1

1. Ecrire une proc´edure anonyme PL/SQL qui permet de demander `a un utilisateur ´ de saisir deux entiers et d'afficher leur somme

```
SQL> SET SERVEROUTPUT ON;
SQL> ACCEPT num1 NUMBER PROMPT 'Entrez le 1er entier  : ';
Entrez le 1er entier  : 1
SQL> ACCEPT num2 NUMBER PROMPT 'Entrez le 2ème entier : ';
Entrez le 2ème entier : 7
SQL> DECLARE
  2    v_num1 NUMBER := &num1;
  3    v_num2 NUMBER := &num2;
  4  BEGIN
  5    DBMS_OUTPUT.PUT_LINE('Somme : ' || (v_num1 + v_num2));
  6  END;
  7  /
ancien   2 :   v_num1 NUMBER := &num1;
nouveau   2 :   v_num1 NUMBER :=          1;
ancien   3 :   v_num2 NUMBER := &num2;
nouveau   3 :   v_num2 NUMBER :=          7;
Somme : 8

ProcÚdure PL/SQL terminÚe avec succÞs.
```

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.

```
SQL> ACCEPT num PROMPT 'Veuillez entrer un nombre : '
Veuillez entrer un nombre : 45
SQL> DECLARE
  2      n NUMBER := &num;
  3      i NUMBER := 1;
  4  BEGIN
  5      WHILE i <= 10 LOOP
  6          DBMS_OUTPUT.PUT_LINE(n || ' x ' || i || ' = ' || (n * i));
  7          i := i + 1;
  8      END LOOP;
  9  END;
 10  /
ancien   2 :      n NUMBER := &num;
nouveau   2 :      n NUMBER := 45;
45 x 1 = 45
45 x 2 = 90
45 x 3 = 135
45 x 4 = 180
45 x 5 = 225
45 x 6 = 270
45 x 7 = 315
45 x 8 = 360
45 x 9 = 405
45 x 10 = 450
```


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

```
SQL> CREATE OR REPLACE FUNCTION puissance(x IN NUMBER, n IN NUMBER) RETURN NUMBER IS
  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  /
```

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*.

```
SQL> CREATE TABLE resultatFactoriel (
  2      nombre NUMBER PRIMARY KEY,
  3      factorielle NUMBER
  4  );
```

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

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> DECLARE
  2      fact NUMBER := 1;
  3  BEGIN
  4      FOR n IN 1..20 LOOP
  5          IF n = 1 THEN
  6              fact := 1;
  7          ELSE
  8              fact := fact * n;
  9          END IF;
 10
 11          DBMS_OUTPUT.PUT_LINE('Factorielle de ' || n || ' = ' || fact);
 12
 13          INSERT INTO resultatsFactoriels (nombre, factorielle)
 14          VALUES (n, fact);
 15      END LOOP;
 16
 17      COMMIT;
 18  END;
```

Exercice n° 2

Soit le schéma de la base de données de gestion des employés, constituée d'une seule table *employe*. Créer la table et y insérer quelques données. Tester vos méthodes au fur et à mesure.

```
SQL> CREATE TABLE emp (
  2       matr NUMBER(10) NOT NULL,
  3       nom VARCHAR2(50) NOT NULL,
  4       sal NUMBER(7, 2),
  5       adresse VARCHAR2(96),
  6       dep NUMBER(10) NOT NULL,
  7       CONSTRAINT emp_pk PRIMARY KEY (matr)
  8   );

Table crÚÚe.
```

1. Écrire 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.

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2      v_employe emp%ROWTYPE; -- Déclaration d'une variable record basée sur la table emp
  3  BEGIN
  4      -- Attribution des valeurs avec la syntaxe correcte
  5      v_employe.matr := 4;
  6      v_employe.nom := 'Youcef';
  7      v_employe.sal := 2500;
  8      v_employe.adresse := 'avenue de la République';
  9      v_employe.dep := 92002;
 10
 11      -- Insertion du tuple dans la table emp
 12      INSERT INTO emp VALUES v_employe;
 13      COMMIT; -- Validation de la transaction
 14      DBMS_OUTPUT.PUT_LINE('Employé inséré avec succès.');
 15  END;
 16  /
Employé inséré avec succès.
```

Exercice n° 3

Nous avons défini, en cours, la spécification du package de gestion des clients : deux procédures qui permettent d'ajouter un client. Ces deux procédure portent le même nom, car la surchage en PL/SQL est possible. Implémenter cette spécification et lever toutes les exceptions.

```
CREATE OR REPLACE PACKAGE gestion_clients AS
   -- Première procédure : ajouter un client avec seulement le nom
   PROCEDURE ajouter_client(p_nom IN VARCHAR2);

   -- Deuxième procédure surchargée : ajouter un client avec nom et prénom
   PROCEDURE ajouter_client(p_nom IN VARCHAR2, p_prenom IN VARCHAR2);
END gestion_clients;
/
```

```
CREATE OR REPLACE PACKAGE BODY gestion_clients AS

   PROCEDURE ajouter_client(p_nom IN VARCHAR2) IS
   BEGIN
      -- Ici on simule l'ajout du client avec seulement le nom
      INSERT INTO clients (id_client, nom)
      VALUES (clients_seq.NEXTVAL, p_nom);

   EXCEPTION
      WHEN OTHERS THEN
         RAISE_APPLICATION_ERROR(-20001, 'Erreur lors de l''ajout du client avec
```

```
nom.');
    END ajouter_client;

    PROCEDURE ajouter_client(p_nom IN VARCHAR2, p_prenom IN VARCHAR2) IS
    BEGIN
        -- Ici on simule l'ajout du client avec nom et prénom
        INSERT INTO clients (id_client, nom, prenom)
        VALUES (clients_seq.NEXTVAL, p_nom, p_prenom);

    EXCEPTION
        WHEN OTHERS THEN
            RAISE_APPLICATION_ERROR(-20002, 'Erreur lors de l''ajout du client avec nom
et prénom.');
    END ajouter_client;

END gestion_clients;
/
```