Rapport TP 4 – Base de données avancées

Nom : Lyes Ahfir
Date : 30 avril 2025

Exercice 1 :

1. Demander deux entiers et afficher leur somme

```
ACCEPT a NUMBER PROMPT 'Saisir le premier entier : ';
ACCEPT b NUMBER PROMPT 'Saisir le deuxieme entier : ';

DECLARE
    v_a NUMBER := &a;
    v_b NUMBER := &b;
BEGIN
    DBMS_OUTPUT.PUT_LINE('La somme est : ' || (v_a + v_b));
END;
```

2. Afficher la table de multiplication d'un entier

```
ACCEPT n NUMBER PROMPT 'Saisir un nombre : ';

DECLARE
    v_n NUMBER := &n;
BEGIN
    FOR i IN 1..10 LOOP
        DBMS_OUTPUT.PUT_LINE(v_n || ' x ' || i || ' = ' || (v_n * i));
    END LOOP;
END;
```

3. Fonction récursive pour calculer x^n

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

4. Calcul de la factorielle d'un entier et insertion dans une table

Création de la table :

```
CREATE TABLE resultatFactoriel (
    valeur NUMBER
);
```

Pour calculer la factorielle et remplir la table :

```
ACCEPT n NUMBER PROMPT 'Saisir un nombre strictement positif : ';

DECLARE
    v_n NUMBER := &n;
    v_fact NUMBER := 1;
BEGIN
    FOR i IN 1..v_n LOOP
        v_fact := v_fact * i;
    END LOOP;

    INSERT INTO resultatFactoriel VALUES (v_fact);
    DBMS_OUTPUT.PUT_LINE('Factorielle de ' || v_n || ' est : ' || v_fact);
END;
```

5. Calcul et stockage des 20 premières factorielles

Création de la table :

```
CREATE TABLE resultatsFactoriels (
    n NUMBER,
    factorielle NUMBER
);
```

Pour calculer les factorielles et remplir la table :

```
BEGIN
    FOR i IN 1..20 LOOP
        DECLARE
            v_fact NUMBER := 1;
        BEGIN
            FOR j IN 1..i LOOP
                v_fact := v_fact * j;
            END LOOP;
            INSERT INTO resultatsFactoriels VALUES (i, v_fact);
        END;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Les 20 premieres factorielles ont ete
       enregistrees.');
END;
```

Exercice 2 :

Soit le schéma de la base de données de gestion des employés, constituée d'une seule table emp.

Création de la table :

```
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. Insérer un employé dans la table

```
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 Republique';
  v_employe.dep := 92002;

  INSERT INTO emp VALUES v_employe;
END;
```

2. Supprimer les employés selon **dep** connu et afficher le nombre de suppressions

```
SET SERVEROUTPUT ON;

DECLARE
  v_nb_lignes NUMBER;
BEGIN
  DELETE FROM emp WHERE dep = 10;
  v_nb_lignes := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('v_nb_lignes : ' || v_nb_lignes);
END;
```

3. Afficher la somme des salaires avec curseur explicite et boucle `LOOP`

```
DECLARE
  v_salaire EMP.sal%TYPE;
  v_total   EMP.sal%TYPE := 0;
  CURSOR c_salaires IS
    SELECT sal FROM emp;
BEGIN
  OPEN c_salaires;

  LOOP
    FETCH c_salaires INTO v_salaire;
    EXIT WHEN c_salaires%NOTFOUND;

    IF v_salaire IS NOT NULL THEN
      v_total := v_total + v_salaire;
    END IF;
  END LOOP;

  CLOSE c_salaires;
  DBMS_OUTPUT.PUT_LINE('Total : ' || v_total);
END;
```

4. Calcul du salaire moyen (AVG)

```
DECLARE
  v_salaire EMP.sal%TYPE;
  v_total   EMP.sal%TYPE := 0;
  v_count   INTEGER := 0;
  v_moyenne EMP.sal%TYPE;

  CURSOR c_salaires IS
    SELECT sal FROM emp;
BEGIN
  OPEN c_salaires;

  LOOP
    FETCH c_salaires INTO v_salaire;
    EXIT WHEN c_salaires%NOTFOUND;

    IF v_salaire IS NOT NULL THEN
      v_total := v_total + v_salaire;
      v_count := v_count + 1;
    END IF;
  END LOOP;

  CLOSE c_salaires;

  IF v_count > 0 THEN
    v_moyenne := v_total / v_count;
    DBMS_OUTPUT.PUT_LINE('Moyenne des salaires : ' || v_moyenne);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Aucun salaire trouve.');
  END IF;
END;
```

5. Refaire les deux procédures précédentes avec la boucle **FOR IN**

```
DECLARE
  v_total   EMP.sal%TYPE := 0;
  v_count   INTEGER := 0;
  v_moyenne EMP.sal%TYPE;

BEGIN
  FOR emp_rec IN (SELECT sal FROM emp) LOOP
    IF emp_rec.sal IS NOT NULL THEN
      v_total := v_total + emp_rec.sal;
      v_count := v_count + 1;
    END IF;
  END LOOP;

  IF v_count > 0 THEN
    v_moyenne := v_total / v_count;
    DBMS_OUTPUT.PUT_LINE('Somme des salaires : ' || v_total);
    DBMS_OUTPUT.PUT_LINE('Moyenne des salaires : ' || v_moyenne);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Aucun salaire trouve.');
  END IF;
END;
```

6. Afficher les noms d'employés d'un département via un curseur paramétré

```
DECLARE
  CURSOR c(p_dep EMP.dep%TYPE) IS
    SELECT dep, nom FROM emp WHERE dep = p_dep;
BEGIN
  FOR v_employe IN c(92000) LOOP
    DBMS_OUTPUT.PUT_LINE('Dep 1 : ' || v_employe.nom);
  END LOOP;

  FOR v_employe IN c(75000) LOOP
    DBMS_OUTPUT.PUT_LINE('Dep 2 : ' || v_employe.nom);
  END LOOP;
END;
```

Exercice 3 :

Spécification du package **packageClient**

```
CREATE OR REPLACE PACKAGE packageClient IS

  - Procedure avec age
  PROCEDURE addClient(
    p_idClient          client.idClient%TYPE,
    p_nomClient         client.nomClient%TYPE,
    p_prenomClient      client.prenomClient%TYPE,
    p_codePostalClient  client.codePostalClient%TYPE,
    p_ageClient         client.ageClient%TYPE
  );

  - Procedure sans age
  PROCEDURE addClient(
    p_idClient          client.idClient%TYPE,
    p_nomClient         client.nomClient%TYPE,
    p_prenomClient      client.prenomClient%TYPE,
    p_codePostalClient  client.codePostalClient%TYPE
  );

  - Fonction moyenne age
  FUNCTION getAgeAvg RETURN NUMBER;

END packageClient;
```

Implémentation du package

```
CREATE OR REPLACE PACKAGE BODY packageClient IS

  PROCEDURE addClient(
    p_idClient          client.idClient%TYPE,
    p_nomClient         client.nomClient%TYPE,
    p_prenomClient      client.prenomClient%TYPE,
    p_codePostalClient  client.codePostalClient%TYPE,
    p_ageClient         client.ageClient%TYPE
  ) IS
  BEGIN
    INSERT INTO client VALUES (
      p_idClient, p_nomClient, p_prenomClient, p_codePostalClient,
          p_ageClient
    );
  EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      DBMS_OUTPUT.PUT_LINE('Erreur : client deja existant.');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Erreur inconnue : ' || SQLERRM);
  END;
```

```
  PROCEDURE addClient(
    p_idClient          client.idClient%TYPE,
    p_nomClient         client.nomClient%TYPE,
    p_prenomClient      client.prenomClient%TYPE,
    p_codePostalClient  client.codePostalClient%TYPE
  ) IS
  BEGIN
    INSERT INTO client (
      idClient, nomClient, prenomClient, codePostalClient
    ) VALUES (
      p_idClient, p_nomClient, p_prenomClient, p_codePostalClient
    );
  EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      DBMS_OUTPUT.PUT_LINE('Erreur : client deja existant.');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Erreur inconnue : ' || SQLERRM);
  END;



  FUNCTION getAgeAvg RETURN NUMBER IS
    v_moyenne NUMBER;
  BEGIN
    SELECT AVG(ageClient) INTO v_moyenne FROM client;
    RETURN v_moyenne;
  EXCEPTION
    WHEN OTHERS THEN
      RETURN NULL;
  END;

END packageClient;
```