Développement des Bases de Données

Arquillière Mathieu

7 février 2020

1 TP1

2 TP2

2.1 Exercice 1

Code :

```
DECLARE
    nom emp.ename%TYPE;
    salaire emp.sal%TYPE;
    commission emp.comm%TYPE;
    departement dept.dname%TYPE;
BEGIN
    SELECT ename,sal,comm,dname INTO nom,salaire,commission,departement FROM Emp NATURAL JOIN Dept
    WHERE ename='MILLER';
    DBMS_OUTPUT.PUT_LINE('Nom : ' || nom || ' Salaire : ' || salaire || ' Commission : ' ||
    commission || 'Departement : ' || departement);
END;
/
```

Résultat :

```
Nom : MILLER Salaire : 1300 Commission : Departement : ACCOUNTING
```

2.2 Exercice 2

Code :

```
DECLARE
    num1 temp.num_col1%TYPE;
    num2 temp.num_col2%TYPE;
    char temp.char_col%TYPE;
BEGIN
  FOR i IN 1..10 LOOP
    IF MOD(i, 2) = 0 THEN
      INSERT INTO temp VALUES (i, i * 100, CONCAT(TO_CHAR(i), ' est pair'));
    ELSE
      INSERT INTO temp VALUES (i, i * 100, CONCAT(TO_CHAR(i), ' est impair'));
    END IF;
  END LOOP;
  COMMIT;
END;
/
```

Résultat :

```
SQL> select * from temp;

  NUM_COL1   NUM_COL2 CHAR_COL
---------- ---------- -------------------------------------------------------
      5000 KING
   1     100 1 est impair
   2     200 2 est pair
   3     300 3 est impair
   4     400 4 est pair
   5     500 5 est impair
   6     600 6 est pair
   7     700 7 est impair
   8     800 8 est pair
   9     900 9 est impair
  10    1000 10 est pair
```

2.3 Exercice 3

Code :

```
DECLARE
  Cursor c IS SELECT sal, empno, ename FROM emp ORDER BY sal DESC;
  salaire emp.sal%TYPE;
  numero emp.empno%TYPE;
  nom emp.ename%TYPE;
BEGIN
  OPEN c;
  FOR i IN 1..5 LOOP
    FETCH c INTO salaire, numero, nom;
    INSERT INTO temp VALUES (salaire, numero, nom);
  END LOOP;
END;
/
```

Résultat :

```
SQL> select * from temp;

  NUM_COL1   NUM_COL2 CHAR_COL
---------- ---------- -------------------------------------------------------
      5000   7839 KING
      3000   7902 FORD
      3000   7788 SCOTT
      2975   7566 JONES
      2850   7698 BLAKE
```

2.4 Exercice 4

Code :

```
DECLARE
    Cursor c IS SELECT UNIQUE sal, NVL(comm, 0), empno, ename FROM emp WHERE sal + NVL(comm, 0) >
    2000;
    salaire emp.sal%TYPE;
    numero emp.empno%TYPE;
    nom emp.ename%TYPE;
    comm emp.comm%TYPE;
BEGIN
  OPEN c;
  LOOP
    FETCH c INTO salaire, comm, numero, nom;
    EXIT WHEN (c%notfound);
    INSERT INTO temp VALUES (salaire + comm, numero, nom);
  END LOOP;
END;
/
```

Résultat :

```
SQL> select * from temp;

  NUM_COL1   NUM_COL2 CHAR_COL
---------- ---------- -------------------------------------------------------
      2975   7566 JONES
      2650   7654 MARTIN
      2850   7698 BLAKE
      2450   7782 CLARK
      3000   7788 SCOTT
      5000   7839 KING
      3000   7902 FORD
      2200   7000 Zangla
```

2.5 Exercice 5

Code :

```
DECLARE
  Cursor c IS SELECT sal, ename, empno, mgr FROM emp;
  salaire emp.sal%TYPE;
  nom emp.ename%TYPE;
  empno emp.empno%TYPE;
  mgrEmp emp.mgr%TYPE;
  chaineMgr emp.mgr%TYPE;
BEGIN
  OPEN c;
  LOOP
    FETCH c INTO salaire, nom, empno, mgrEmp;
    EXIT WHEN(c%NOTFOUND);
    IF salaire >= 4000 THEN
      SELECT mgr INTO chaineMgr FROM emp WHERE empno=7902;
      LOOP
        EXIT WHEN(chaineMgr IS NULL OR chaineMgr=mgrEmp);
        SELECT mgr INTO chaineMgr FROM emp where empno=chaineMgr;
      END LOOP;
      IF chaineMgr=mgrEmp OR mgrEmp IS NULL THEN
        INSERT INTO temp VALUES (null, salaire, nom);
      END IF;
    END IF;
  END LOOP;
END;
/
```

Résultat :

```
SQL> select * from temp;

  NUM_COL1   NUM_COL2 CHAR_COL
---------- ---------- -------------------------------------------------------
     5000 KING
```

3 TP3

3.1 Exercice 1 (A)

Code :

```
CREATE OR REPLACE PROCEDURE createdept_zangla(num IN NUMBER, name IN VARCHAR2, loc IN VARCHAR2)
IS
    d NUMBER;
BEGIN
    SELECT deptno INTO d FROM dept WHERE deptno = num;
    RAISE_APPLICATION_ERROR(-20001, 'Numero de departement deja existant');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            INSERT INTO dept VALUES(num, name, loc);
END;
/
```

Résultat :

```
## Déjà existant ##
SQL> exec createdept_zangla(30, 'SALES', 'CHICAGO');
BEGIN createdept_zangla(30, 'SALES', 'CHICAGO'); END;

*
ERREUR a la ligne 1 :
ORA-20001: Numero de departement deja existant
ORA-06512: a "BD10.CREATEDEPT_ZANGLA", ligne 6
ORA-06512: a ligne 1


## Ajout ##
SQL> exec createdept_zangla(16, 'VACHE', 'GUERET');

Procedure PL/SQL terminee avec succes.

SQL> select * from dept;

DEPTNO      DNAME        LOC
---------- -------------- -------------
    10     ACCOUNTING    NEW YORK
    20     RESEARCH    DALLAS
    30     SALES       CHICAGO
    16     VACHE       GUERET
```

3.2 Exercice 2 (A)

Code :

```
CREATE OR REPLACE FUNCTION salok_zangla(jobselect in VARCHAR2, salaire in NUMBER) RETURN NUMBER
IS
    mi NUMBER;
    ma NUMBER;
BEGIN
    SELECT lsal, hsal INTO mi, ma FROM salintervalle_f2 WHERE job = jobselect;
    IF salaire >= mi AND salaire <= ma THEN RETURN 1;
    ELSE RETURN 0;
    END IF;
    EXCEPTION WHEN NO_DATA_FOUND THEN RETURN 0;
END;
/
```

Résultat :

```
SQL> select * from salintervalle_f2;

JOB    LSAL      HSAL
--------- ---------- ----------
ANALYST   2500      3000
CLERK     900      1300
MANAGER   2400      3000
```

```
PRESIDENT 4500      4900
SALESMAN  1200      1700

SQL> variable vrai number;
SQL> execute :vrai := salok_zangla('ANALYST', 2900);

Procedure PL/SQL terminee avec succes.

SQL> print vrai;

      VRAI
----------
   1
SQL> variable faux number;
SQL> execute :faux := salok_zangla('PRESIDENT', 4000);

Procedure PL/SQL terminee avec succes.

SQL> print faux;

      FAUX
----------
   0
```

3.3 Exercice 3 (A)

Code :

```
CREATE OR REPLACE PROCEDURE raisesalary_zangla(emp_id IN NUMBER, amount IN NUMBER)
IS
    e NUMBER;
    j VARCHAR(9);
    s NUMBER;
    possible NUMBER;
BEGIN
    SELECT empno, job, sal INTO e, j, s FROM emp WHERE empno = emp_id;
    possible := salok_zangla(j, s + amount);
    IF possible = 1 THEN
        UPDATE emp SET sal = s + amount WHERE empno = e;
    ELSE
        RAISE_APPLICATION_ERROR(-20002, 'Maximum deja atteint');
    END IF;
END;
/
```

Résultat :

```
SQL> select * from emp where empno=7876;

    EMPNO ENAME      JOB          MGR HIREDATE    SAL        COMM      DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
     7876 ADAMS      CLERK        7788 13/07/87   1100                  20

SQL> execute raisesalary_zangla(7876, 100);

SQL> select * from emp where empno=7876;

    EMPNO ENAME      JOB          MGR HIREDATE    SAL        COMM      DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
     7876 ADAMS      CLERK        7788 13/07/87   1200                  20

SQL> execute raisesalary_zangla(7876, 500);
BEGIN raisesalary_zangla(7876, 500); END;

*
ERREUR a la ligne 1 :
ORA-20002: Maximum deja atteint
ORA-06512: a "BD10.RAISESALARY_ZANGLA", ligne 13
ORA-06512: a ligne 1
```

3.4 Exercice 4 (B)

Code :

```
DECLARE
  Cursor c IS SELECT table_name FROM user_tables WHERE table_name NOT LIKE '%_OLD';
  Cursor cold IS SELECT table_name FROM user_tables WHERE table_name LIKE '%_OLD';
  t_name user_tables.table_name%TYPE;
BEGIN
  OPEN cold;
  LOOP
    FETCH cold INTO t_name;
    EXIT WHEN (cold%NOTFOUND);
    EXECUTE IMMEDIATE 'DROP TABLE ' || t_name;
  END LOOP;
  OPEN c;
  LOOP
    FETCH c INTO t_name;
    EXIT WHEN(c%NOTFOUND);
    EXECUTE IMMEDIATE 'CREATE TABLE ' || t_name || '_old AS (SELECT * FROM ' || t_name || ')';
    DBMS_OUTPUT.PUT_LINE(t_name);
  END LOOP;
END;
/
```

Résultat (avec plusieurs exécutions pour vérifier) :

```
SQL> select table_name from user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
AUTEURS
EXCEPTIONS
TEMP
SALINTERVALLE_F2
OUVRAGE
AUTEUR_OUVRAGE
DEPT
EMP

SQL> start b.sql

Procedure PL/SQL terminee avec succes.

SQL> select table_name from user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
AUTEURS
EXCEPTIONS
TEMP
SALINTERVALLE_F2
OUVRAGE
AUTEUR_OUVRAGE
DEPT
EMP
AUTEURS_OLD
AUTEUR_OUVRAGE_OLD
DEPT_OLD
EMP_OLD
OUVRAGE_OLD
SALINTERVALLE_F2_OLD
TEMP_OLD
EXCEPTIONS_OLD

SQL> start b.sql

Procedure PL/SQL terminee avec succes.

SQL> select table_name from user_tables;

TABLE_NAME
```

```
--------------------------------------------------------------------------------
AUTEURS
EMP_OLD
EXCEPTIONS
AUTEUR_OUVRAGE_OLD
TEMP
SALINTERVALLE_F2
OUVRAGE
AUTEUR_OUVRAGE
DEPT
EMP
TEMP_OLD
DEPT_OLD
SALINTERVALLE_F2_OLD
AUTEURS_OLD
OUVRAGE_OLD
EXCEPTIONS_OLD
```

4 TP4

4.1 Exercice Package

Code :

```
CREATE OR REPLACE PACKAGE zangla AS
  TYPE emp_cursor IS RECORD (emp_id NUMBER, nom VARCHAR2(10));
  CURSOR emp_par_dep_zangla(dep IN NUMBER) RETURN emp_cursor;
  PROCEDURE raise_salary_zangla(emp_id IN NUMBER, amount IN NUMBER);
  PROCEDURE afficher_emp_zangla(deptno IN NUMBER);
END;
/

CREATE OR REPLACE PACKAGE BODY zangla AS
  CURSOR emp_par_dep_zangla(dep IN NUMBER) RETURN emp_cursor IS
    SELECT empno, ename FROM emp WHERE deptno = dep;
  PROCEDURE raise_salary_zangla(emp_id IN NUMBER, amount IN NUMBER) IS
    e NUMBER;
    j VARCHAR(9);
    s NUMBER;
    possible NUMBER;
  BEGIN
    SELECT empno, job, sal INTO e, j, s FROM emp WHERE empno = emp_id;
    possible := salok_zangla(j, s + amount);
    IF possible = 1 THEN
      UPDATE emp SET sal = s + amount WHERE empno = e;
    ELSE
      RAISE_APPLICATION_ERROR(-20002, 'Maximum deja atteint');
    END IF;
  END;
  PROCEDURE afficher_emp_zangla(deptno IN NUMBER) IS
    emp_id NUMBER;
    nom VARCHAR2(9);
  BEGIN
    OPEN emp_par_dep_zangla(deptno);
    LOOP
      FETCH emp_par_dep_zangla INTO emp_id, nom;
      EXIT WHEN(emp_par_dep_zangla%NOTFOUND);
      DBMS_OUTPUT.PUT_LINE(emp_id || ' : ' || nom);
    END LOOP;
    CLOSE emp_par_dep_zangla;
  END;
END;
/
```

Résultat procedure raise_salary :

```
Depuis la table emp:
    EMPNO ENAME      JOB         MGR HIREDATE    SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
    7900 JAMES      CLERK       7698 03/12/81   1050                  30

Depuis la table salintervalle_f2:
JOB   LSAL     HSAL
--------- ---------- ----------
CLERK    900     1300

SQL> exec zangla.raise_salary_zangla(7900, 50)

Procedure PL/SQL terminee avec succes.

Depuis la table emp:
EMPNO ENAME      JOB         MGR HIREDATE    SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
7900 JAMES      CLERK       7698 03/12/81   1100                  30
```

Résultat procedure afficher_emp :

```
SQL> select * from emp;

    EMPNO ENAME      JOB         MGR HIREDATE    SAL       COMM      DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
     7369 SMITH      CLERK        7902 17/12/80    800                  20
     7499 ALLEN      SALESMAN   7698 20/02/81   1600        300         30
     7521 WARD       SALESMAN   7698 22/02/81   1250        500         30
     7566 JONES      MANAGER    7839 02/04/81   2975                  20
     7654 MARTIN     SALESMAN   7698 28/09/81   1250       1400         30
     7698 BLAKE      MANAGER    7839 01/05/81   2850                  30
     7782 CLARK      MANAGER    7839 09/06/81   2450                  10
     7788 SCOTT      ANALYST    7566 13/07/87   3000                  20
     7839 KING       PRESIDENT         17/11/81   5000                  10
     7844 TURNER     SALESMAN   7698 08/09/81   1500        0           30
     7876 ADAMS      CLERK        7788 13/07/87   1200                  20
     7900 JAMES      CLERK        7698 03/12/81   1100                  30
     7902 FORD       ANALYST    7566 03/12/81   3000                  20
     7934 MILLER     CLERK        7782 23/01/82   1300                  10
     7000 Zangla     SALESMAN   7566 17/12/80   2200                  20

SQL> exec zangla.afficher_emp_zangla(20)
    7369 : SMITH
    7566 : JONES
    7788 : SCOTT
    7876 : ADAMS
    7902 : FORD
    7000 : Zangla

    Procedure PL/SQL terminee avec succes.
```

4.2 Exercice Trigger 1

Code :

```
CREATE OR REPLACE TRIGGER raise_zangla
  BEFORE UPDATE ON emp
  FOR EACH ROW
  WHEN (new.sal < old.sal)
  BEGIN
    RAISE_APPLICATION_ERROR(-20003, 'Impossible de diminuer le salaire !');
END;
/
```

Résultat :

```
SQL> update emp set sal=1100 where empno=7934;
update emp set sal=1100 where empno=7934
        *
ERREUR a la ligne 1 :
ORA-20003: Impossible de diminuer le salaire !
ORA-06512: a "BD10.RAISE_ZANGLA", ligne 2
ORA-04088: erreur lors d'execution du declencheur 'BD10.RAISE_ZANGLA'
```