

BDD

Zangla Jérémy

21 février 2020

1 TP1

1.1 Création des tables

Code fichier charger.ctl :

```
1 LOAD DATA INFILE *
2 INTO TABLE auteurs
3 fields terminated by ","
4 (num,nom,prenom nullif prenom="null",pays nullif pays="null",tel nullif tel="null")
5 BEGINDATA
6 1,Dupont,Jacques,FR,0473151585
7 2,Durand,Marie,GB,null
8 3,Dupont,Pierre,null,null
9 3,Dupont,null,null,null
```

Sur sqlplus :

```
1 CREATE TABLE AUTEURS (
2     NUM      NUMBER(6),
3     NOM      VARCHAR(50),
4     PRENOM   VARCHAR(50),
5     PA       VARCHAR(2),
6     TEL      VARCHAR(10)
7 );
8 CREATE TABLE OUVRAGE (
9     CODE     NUMBER(3),
10    TITRE    VARCHAR(50),
11    PRIX     NUMBER(6)
12 );
13 CREATE TABLE AUTEUR_OUVRAGE (
14     CODE_OUV NUMBER(6),
15     NUM_AUT  NUMBER(6)
16 );
17
18 INSERT INTO OUVRAGE VALUES (001, 'Intro aux BD', 260);
19 INSERT INTO OUVRAGE VALUES (002, 'Journal de Bolivie');
20 INSERT INTO OUVRAGE VALUES (003, 'L Homme aux sandales');
21
22 INSERT INTO AUTEUR_OUVRAGE VALUES (001, 1);
23 INSERT INTO AUTEUR_OUVRAGE VALUES (002, 2);
24 INSERT INTO AUTEUR_OUVRAGE VALUES (003, 2);
```

Résultat :

```
1 SQL> select * from auteurs;
2
3 NUM NOM                PRENOM                PA TEL
4 -----
5 1 DUPONT                Jacques                FR 0473151585
6 2 DURAND                Marie                GB
7 3 DUPONT                Pierre
8 3 DUPONT
9
10 SQL> select * from ouvrage;
11
12 CODE TITRE                PRIX
13 -----
14 1 Intro aux BD                260
15 2 Journal de Bolivie
16 3 L Homme aux sandales
17
18 SQL> select * from auteur_ouvrage;
19
20 CODE_OUV NUM_AUT
21 -----
22 1 1
23 2 2
24 3 2
```

1.2 Contraintes

```
1 alter table auteurs add constraint pk_auteurs primary key (num) exceptions into exceptions;
2 alter table ouvrage add constraint pk_ouvrage primary key (code) exceptions into exceptions;
3 alter table auteur_ouvrage add constraint pk_auteur_ouvrage primary key (code_ouv, num_aut)
  exceptions into exceptions;
4
5 alter table auteur_ouvrage add constraint fk_code foreign key code_ouv references ouvrage(code)
  exceptions into exceptions;
6 alter table auteur_ouvrage add constraint fk_aut foreign key num_aut references auteurs(num)
  exceptions into exceptions;
7
8 alter table auteurs add constraint maj_nom_auteurs check(nom = upper(nom)) exceptions into
  exceptions;
9
10 alter table auteurs drop constraint pk_auteurs;
```

2 TP2

2.1 Exercice 1

Code :

```
1 DECLARE
2     nom emp.ename%TYPE;
3     salaire emp.sal%TYPE;
4     commission emp.comm%TYPE;
5     departement dept.dname%TYPE;
6 BEGIN
7     SELECT ename,sal,comm,dname INTO nom,salaire,commission,departement FROM Emp NATURAL JOIN Dept
8     WHERE ename='MILLER';
9     DBMS_OUTPUT.PUT_LINE('Nom : ' || nom || ' Salaire : ' || salaire || ' Commission : ' ||
  commission || 'Departement : ' || departement);
10 END;
11 /
```

Résultat :

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

2.2 Exercice 2

Code :

```
1 DECLARE
2     num1 temp.num_col1%TYPE;
3     num2 temp.num_col2%TYPE;
4     char temp.char_col%TYPE;
5 BEGIN
6     FOR i IN 1..10 LOOP
7         IF MOD(i, 2) = 0 THEN
8             INSERT INTO temp VALUES (i, i * 100, CONCAT(TO_CHAR(i), ' est pair'));
9         ELSE
10            INSERT INTO temp VALUES (i, i * 100, CONCAT(TO_CHAR(i), ' est impair'));
11        END IF;
12    END LOOP;
13    COMMIT;
14 END;
15 /
```

Résultat :

```
1 SQL> select * from temp;
2
3  NUM_COL1  NUM_COL2 CHAR_COL
4  -----
5      5000 KING
6      1      100 1 est impair
7      2      200 2 est pair
8      3      300 3 est impair
```

```

9      4      400 4 est pair
10     5      500 5 est impair
11     6      600 6 est pair
12     7      700 7 est impair
13     8      800 8 est pair
14     9      900 9 est impair
15    10     1000 10 est pair

```

2.3 Exercice 3

Code :

```

1 DECLARE
2   Cursor c IS SELECT sal, empno, ename FROM emp ORDER BY sal DESC;
3   salaire emp.sal%TYPE;
4   numero emp.empno%TYPE;
5   nom emp.ename%TYPE;
6 BEGIN
7   OPEN c;
8   FOR i IN 1..5 LOOP
9     FETCH c INTO salaire, numero, nom;
10    INSERT INTO temp VALUES (salaire, numero, nom);
11  END LOOP;
12 END;
13 /

```

Résultat :

```

1 SQL> select * from temp;
2
3   NUM_COL1    NUM_COL2 CHAR_COL
4  -----
5         5000        7839 KING
6         3000        7902 FORD
7         3000        7788 SCOTT
8         2975        7566 JONES
9         2850        7698 BLAKE

```

2.4 Exercice 4

Code :

```

1 DECLARE
2   Cursor c IS SELECT UNIQUE sal, NVL(comm, 0), empno, ename FROM emp WHERE sal + NVL(comm, 0) >
3   2000;
4   salaire emp.sal%TYPE;
5   numero emp.empno%TYPE;
6   nom emp.ename%TYPE;
7   comm emp.comm%TYPE;
8 BEGIN
9   OPEN c;
10  LOOP
11    FETCH c INTO salaire, comm, numero, nom;
12    EXIT WHEN (c%notfound);
13    INSERT INTO temp VALUES (salaire + comm, numero, nom);
14  END LOOP;
15 END;
16 /

```

Résultat :

```

1 SQL> select * from temp;
2
3   NUM_COL1    NUM_COL2 CHAR_COL
4  -----
5         2975        7566 JONES
6         2650        7654 MARTIN
7         2850        7698 BLAKE
8         2450        7782 CLARK
9         3000        7788 SCOTT

```

```

10      5000      7839 KING
11      3000      7902 FORD
12      2200      7000 Zangla

```

2.5 Exercice 5

Code :

```

1 DECLARE
2   Cursor c IS SELECT sal, ename, empno, mgr FROM emp;
3   salaire emp.sal%TYPE;
4   nom emp.ename%TYPE;
5   empno emp.empno%TYPE;
6   mgrEmp emp.mgr%TYPE;
7   chaineMgr emp.mgr%TYPE;
8 BEGIN
9   OPEN c;
10  LOOP
11    FETCH c INTO salaire, nom, empno, mgrEmp;
12    EXIT WHEN (c%NOTFOUND);
13    IF salaire >= 4000 THEN
14      SELECT mgr INTO chaineMgr FROM emp WHERE empno=7902;
15      LOOP
16        EXIT WHEN (chaineMgr IS NULL OR chaineMgr=mgrEmp);
17        SELECT mgr INTO chaineMgr FROM emp where empno=chaineMgr;
18      END LOOP;
19      IF chaineMgr=mgrEmp OR mgrEmp IS NULL THEN
20        INSERT INTO temp VALUES (null, salaire, nom);
21      END IF;
22    END IF;
23  END LOOP;
24 END;
25 /

```

Résultat :

```

1 SQL> select * from temp;
2
3   NUM_COL1    NUM_COL2  CHAR_COL
4  -----
5      5000 KING

```

3 TP3

3.1 Exercice 1 (A)

Code :

```
1 CREATE OR REPLACE PROCEDURE createdept_zangla(num IN NUMBER, name IN VARCHAR2, loc IN VARCHAR2)
2 IS
3     d NUMBER;
4 BEGIN
5     SELECT deptno INTO d FROM dept WHERE deptno = num;
6     RAISE_APPLICATION_ERROR(-20001, 'Numero de departement deja existant');
7     EXCEPTION
8         WHEN NO_DATA_FOUND THEN
9             INSERT INTO dept VALUES(num, name, loc);
10 END;
11 /
```

Résultat :

```
1 ## Déjà existant ##
2 SQL> exec createdept_zangla(30, 'SALES', 'CHICAGO');
3 BEGIN createdept_zangla(30, 'SALES', 'CHICAGO'); END;
4
5 *
6 ERREUR a la ligne 1 :
7 ORA-20001: Numero de departement deja existant
8 ORA-06512: a "BD10.CREATEDEPT_ZANGLA", ligne 6
9 ORA-06512: a ligne 1
10
11 ## Ajout ##
12 SQL> exec createdept_zangla(16, 'TEST', 'AUBIERE');
13
14 Procedure PL/SQL terminee avec succes.
15
16 SQL> select * from dept;
17
18 DEPTNO      DNAME          LOC
19 -----
20      10      ACCOUNTING      NEW YORK
21      20      RESEARCH        DALLAS
22      30      SALES             CHICAGO
23      16      TEST              AUBIERE
```

3.2 Exercice 2 (A)

Code :

```
1 CREATE OR REPLACE FUNCTION salok_zangla(jobselect IN VARCHAR2, salaire IN NUMBER) RETURN NUMBER
2 IS
3     mi NUMBER;
4     ma NUMBER;
5 BEGIN
6     SELECT lsal, hsal INTO mi, ma FROM salintervalle_f2 WHERE job = jobselect;
7     IF salaire >= mi AND salaire <= ma THEN RETURN 1;
8     ELSE RETURN 0;
9     END IF;
10    EXCEPTION WHEN NO_DATA_FOUND THEN RETURN 0;
11 END;
12 /
```

Résultat :

```
1 SQL> select * from salintervalle_f2;
2
3 JOB      LSAL      HSAL
4 -----
5 ANALYST   2500      3000
6 CLERK     900        1300
7 MANAGER   2400      3000
8 PRESIDENT 4500      4900
```

```

9 SALESMAN 1200 1700
10
11 SQL> variable vrai number;
12 SQL> execute :vrai := salok_zangla('ANALYST', 2900);
13
14 Procedure PL/SQL terminee avec succes.
15
16 SQL> print vrai;
17
18      VRAI
19 -----
20      1
21 SQL> variable faux number;
22 SQL> execute :faux := salok_zangla('PRESIDENT', 4000);
23
24 Procedure PL/SQL terminee avec succes.
25
26 SQL> print faux;
27
28      FAUX
29 -----
30      0

```

3.3 Exercice 3 (A)

Code :

```

1 CREATE OR REPLACE PROCEDURE raisesalary_zangla(emp_id IN NUMBER, amount IN NUMBER)
2 IS
3     e NUMBER;
4     j VARCHAR(9);
5     s NUMBER;
6     possible NUMBER;
7 BEGIN
8     SELECT empno, job, sal INTO e, j, s FROM emp WHERE empno = emp_id;
9     possible := salok_zangla(j, s + amount);
10    IF possible = 1 THEN
11        UPDATE emp SET sal = s + amount WHERE empno = e;
12    ELSE
13        RAISE_APPLICATION_ERROR(-20002, 'Maximum deja atteint');
14    END IF;
15 END;
16 /

```

Résultat :

```

1 SQL> select * from emp where empno=7876;
2
3      EMPNO ENAME      JOB          MGR HIREDATE      SAL        COMM      DEPTNO
4 -----
5      7876 ADAMS      CLERK          7788 13/07/87      1100             20
6
7 SQL> execute raisesalary_zangla(7876, 100);
8
9 SQL> select * from emp where empno=7876;
10
11      EMPNO ENAME      JOB          MGR HIREDATE      SAL        COMM      DEPTNO
12 -----
13      7876 ADAMS      CLERK          7788 13/07/87      1200             20
14
15 SQL> execute raisesalary_zangla(7876, 500);
16 BEGIN raisesalary_zangla(7876, 500); END;
17
18 *
19 ERREUR a la ligne 1 :
20 ORA-20002: Maximum deja atteint
21 ORA-06512: a "BD10.RAISESALARY_ZANGLA", ligne 13
22 ORA-06512: a ligne 1

```

3.4 Exercice 4 (B)

Code :

```
1 DECLARE
2   Cursor c IS SELECT table_name FROM user_tables WHERE table_name NOT LIKE '%_OLD';
3   Cursor cold IS SELECT table_name FROM user_tables WHERE table_name LIKE '%_OLD';
4   t_name user_tables.table_name%TYPE;
5 BEGIN
6   OPEN cold;
7   LOOP
8     FETCH cold INTO t_name;
9     EXIT WHEN (cold%NOTFOUND);
10    EXECUTE IMMEDIATE 'DROP TABLE ' || t_name;
11  END LOOP;
12  OPEN c;
13  LOOP
14    FETCH c INTO t_name;
15    EXIT WHEN (c%NOTFOUND);
16    EXECUTE IMMEDIATE 'CREATE TABLE ' || t_name || '_old AS (SELECT * FROM ' || t_name || ')';
17    DBMS_OUTPUT.PUT_LINE(t_name);
18  END LOOP;
19 END;
20 /
```

Résultat (avec plusieurs exécutions pour vérifier) :

```
1 SQL> select table_name from user_tables;
2
3 TABLE_NAME
4 -----
5 AUTEURS
6 EXCEPTIONS
7 TEMP
8 SALINTERVALLE_F2
9 OUVRAGE
10 AUTEUR_OUVRAGE
11 DEPT
12 EMP
13
14 SQL> start b.sql
15
16 Procedure PL/SQL terminee avec succes.
17
18 SQL> select table_name from user_tables;
19
20 TABLE_NAME
21 -----
22 AUTEURS
23 EXCEPTIONS
24 TEMP
25 SALINTERVALLE_F2
26 OUVRAGE
27 AUTEUR_OUVRAGE
28 DEPT
29 EMP
30 AUTEURS_OLD
31 AUTEUR_OUVRAGE_OLD
32 DEPT_OLD
33 EMP_OLD
34 OUVRAGE_OLD
35 SALINTERVALLE_F2_OLD
36 TEMP_OLD
37 EXCEPTIONS_OLD
38
39 SQL> start b.sql
40
41 Procedure PL/SQL terminee avec succes.
42
43 SQL> select table_name from user_tables;
44
45 TABLE_NAME
```



```
46 -----
47 AUTEURS
48 EMP_OLD
49 EXCEPTIONS
50 AUTEUR_OUVRAGE_OLD
51 TEMP
52 SALINTERVALLE_F2
53 OUVRAGE
54 AUTEUR_OUVRAGE
55 DEPT
56 EMP
57 TEMP_OLD
58 DEPT_OLD
59 SALINTERVALLE_F2_OLD
60 AUTEURS_OLD
61 OUVRAGE_OLD
62 EXCEPTIONS_OLD
```

4 TP4

4.1 Exercice Package

Code :

```
1 CREATE OR REPLACE PACKAGE zangla AS
2   TYPE emp_cursor IS RECORD (emp_id NUMBER, nom VARCHAR2(10));
3   CURSOR emp_par_dep_zangla(dep IN NUMBER) RETURN emp_cursor;
4   PROCEDURE raise_salary_zangla(emp_id IN NUMBER, amount IN NUMBER);
5   PROCEDURE afficher_emp_zangla(deptno IN NUMBER);
6 END;
7 /
8
9 CREATE OR REPLACE PACKAGE BODY zangla AS
10  CURSOR emp_par_dep_zangla(dep IN NUMBER) RETURN emp_cursor IS
11    SELECT empno, ename FROM emp WHERE deptno = dep;
12  PROCEDURE raise_salary_zangla(emp_id IN NUMBER, amount IN NUMBER) IS
13    e NUMBER;
14    j VARCHAR(9);
15    s NUMBER;
16    possible NUMBER;
17  BEGIN
18    SELECT empno, job, sal INTO e, j, s FROM emp WHERE empno = emp_id;
19    possible := salok_zangla(j, s + amount);
20    IF possible = 1 THEN
21      UPDATE emp SET sal = s + amount WHERE empno = e;
22    ELSE
23      RAISE_APPLICATION_ERROR(-20002, 'Maximum deja atteint');
24    END IF;
25  END;
26  PROCEDURE afficher_emp_zangla(deptno IN NUMBER) IS
27    emp_id NUMBER;
28    nom VARCHAR2(9);
29  BEGIN
30    OPEN emp_par_dep_zangla(deptno);
31    LOOP
32      FETCH emp_par_dep_zangla INTO emp_id, nom;
33      EXIT WHEN (emp_par_dep_zangla%NOTFOUND);
34      DBMS_OUTPUT.PUT_LINE(emp_id || ' : ' || nom);
35    END LOOP;
36    CLOSE emp_par_dep_zangla;
37  END;
38 END;
39 /
```

Résultat procedure raise_salary :

```
1 Depuis la table emp:
2   EMPNO ENAME      JOB          MGR HIREDATE      SAL      COMM      DEPTNO
3   -----
4    7900 JAMES      CLERK       7698 03/12/81    1050
5
6 Depuis la table salintervalle_f2:
7   JOB   LSAL      HSAL
8   -----
9   CLERK   900      1300
10
11 SQL> exec zangla.raise_salary_zangla(7900, 50)
12
13 Procedure PL/SQL terminee avec succes.
14
15 Depuis la table emp:
16   EMPNO ENAME      JOB          MGR HIREDATE      SAL      COMM      DEPTNO
17   -----
18   7900 JAMES      CLERK       7698 03/12/81    1100
19
```

Résultat procedure afficher_emp :

```
1 SQL> select * from emp;
2
3      EMPNO  ENAME      JOB              MGR HIREDATE          SAL         COMM         DEPTNO
4 -----
5      7369 SMITH        CLERK             7902 17/12/80          800
6      7499 ALLEN         SALESMAN          7698 20/02/81         1600          300          30
7      7521 WARD           SALESMAN          7698 22/02/81         1250          500          30
8      7566 JONES          MANAGER           7839 02/04/81         2975
9      7654 MARTIN        SALESMAN          7698 28/09/81         1250          1400          30
10     7698 BLAKE         MANAGER           7839 01/05/81         2850
11     7782 CLARK         MANAGER           7839 09/06/81         2450          10
12     7788 SCOTT        ANALYST           7566 13/07/87         3000          20
13     7839 KING           PRESIDENT         17/11/81         5000          10
14     7844 TURNER        SALESMAN          7698 08/09/81         1500          0            30
15     7876 ADAMS         CLERK             7788 13/07/87         1200          20
16     7900 JAMES         CLERK             7698 03/12/81         1100          30
17     7902 FORD         ANALYST           7566 03/12/81         3000          20
18     7934 MILLER        CLERK             7782 23/01/82         1300          10
19     7000 Zangla        SALESMAN          7566 17/12/80         2200          20
20
21 SQL> exec zangla.afficher_emp_zangla(20)
22      7369 : SMITH
23      7566 : JONES
24      7788 : SCOTT
25      7876 : ADAMS
26      7902 : FORD
27      7000 : Zangla
28
29      Procedure PL/SQL terminee avec succes.
```

4.2 Exercice Trigger 1

Code :

```
1 CREATE OR REPLACE TRIGGER raise_zangla
2 BEFORE UPDATE ON emp
3 FOR EACH ROW
4 WHEN (new.sal < old.sal)
5 BEGIN
6     RAISE_APPLICATION_ERROR(-20003, 'Impossible de diminuer le salaire !');
7 END;
8 /
```

Résultat :

```
1 SQL> update emp set sal=1100 where empno=7934;
2 update emp set sal=1100 where empno=7934
3      *
4 ERREUR a la ligne 1 :
5 ORA-20003: Impossible de diminuer le salaire !
6 ORA-06512: a "BD10.RAISE_ZANGLA", ligne 2
7 ORA-04088: erreur lors d'execution du declencheur 'BD10.RAISE_ZANGLA'
```

4.3 Exercice Trigger 2

Code :

```
1 CREATE OR REPLACE TRIGGER numdept_zangla
2 BEFORE INSERT ON dept
3 FOR EACH ROW
4 WHEN (new.deptno < 61 OR new.deptno > 69)
5 BEGIN
6     RAISE_APPLICATION_ERROR(-20004, 'Numéro de departement doit etre entre 61 et 69');
7 END;
8 /
```

Résultat :

```

1 SQL> select * from dept;
2
3      DEPTNO DNAME      LOC
4 -----
5      10 ACCOUNTING    NEW YORK
6      20 RESEARCH      DALLAS
7      30 SALES         CHICAGO
8
9 3 lignes selectionnees.
10
11 SQL> insert into dept values (90, 'test', 'aubiere');
12 insert into dept values (90, 'test', 'aubiere')
13      *
14 ERREUR a la ligne 1 :
15 ORA-20004: Num??ro de departement doit etre entre 61 et 69
16 ORA-06512: a "BD10.NUMDEPT_ZANGLA", ligne 2
17 ORA-04088: erreur lors d execution du declencheur 'BD10.NUMDEPT_ZANGLA'
18
19 SQL> insert into dept values (65, 'test', 'aubiere');
20
21 1 ligne creee.
22
23 SQL> select * from dept;
24
25      DEPTNO DNAME      LOC
26 -----
27      10 ACCOUNTING    NEW YORK
28      20 RESEARCH      DALLAS
29      30 SALES         CHICAGO
30      65 test         aubiere
31
32 4 lignes selectionnees.

```

4.4 Exercice Trigger 3

Code :

```

1 CREATE OR REPLACE TRIGGER dept_zangla
2 BEFORE INSERT OR UPDATE ON emp
3 FOR EACH ROW
4 DECLARE
5     cpt NUMBER := 0;
6 BEGIN
7     SELECT COUNT(deptno) INTO cpt FROM dept WHERE :new.deptno = deptno;
8     IF cpt = 0 THEN
9         INSERT INTO dept VALUES(:new.deptno, 'A SAISIR', 'A SAISIR');
10    END IF;
11 END;
12 /

```

Résultat :

```

1 SQL> select * from dept;
2
3      DEPTNO DNAME      LOC
4 -----
5      10 ACCOUNTING    NEW YORK
6      20 RESEARCH      DALLAS
7      30 SALES         CHICAGO
8      65 test         aubiere
9
10 4 lignes selectionnees.
11
12 SQL> insert into emp (empno, ename, mgr, deptno) values (8000, 'jacques', 7839, 64);
13
14 1 ligne creee.
15
16 SQL> select * from dept;
17
18      DEPTNO DNAME      LOC
19 -----

```

```

20 10 ACCOUNTING NEW YORK
21 20 RESEARCH DALLAS
22 30 SALES CHICAGO
23 65 test aubiere
24 64 A SAISIR A SAISIR
25
26 5 lignes selectionnees.

```

4.5 Exercice Trigger 4

Code :

```

1 CREATE OR REPLACE TRIGGER nowweek_zangla
2 BEFORE INSERT OR UPDATE OR DELETE ON emp
3 FOR EACH ROW
4 BEGIN
5     IF to_char(SYSDATE, 'FMDAY', 'NLS_DATE_LANGUAGE=FRENCH') IN ('SAMEDI', 'DIMANCHE') THEN
6         RAISE_APPLICATION_ERROR(-20005, 'Pas de modifications le weekend');
7     END IF;
8 END;
9 /

```

Résultat :

```

1 SQL> insert into emp (empno, ename, mgr) values (8001, 'jean-louis', 7839);
2 insert into emp (empno, ename, mgr) values (8001, 'jean-louis', 7839)
3 *
4 ERREUR a la ligne 1 :
5 ORA-20005: Pas de modifications le weekend
6 ORA-06512: a "BD10.NOWEEK_ZANGLA", ligne 3
7 ORA-04088: erreur lors d'execution du declencheur 'BD10.NOWEEK_ZANGLA'

```

4.6 Exercice Trigger 5

Code :

```

1 SQL> ALTER TRIGGER nowweek_zangla DISABLE;
2
3 Declencheur modifie.

```

Résultat :

```

1 SQL> insert into emp (empno, ename, mgr, deptno) values (8001, 'jean-louis', 7839, 65);
2
3 1 ligne creee.
4
5 SQL> select * from emp;
6
7      EMPNO  ENAME      JOB              MGR HIREDATE          SAL         COMM         DEPTNO
8  -----
9      7369 SMITH      CLERK            7902 17/12/80           800             20
10     7499 ALLEN      SALESMAN         7698 20/02/81          1600           300         30
11     7521 WARD       SALESMAN         7698 22/02/81          1250           500         30
12     7566 JONES      MANAGER          7839 02/04/81          2975             20
13     7654 MARTIN    SALESMAN         7698 28/09/81          1250          1400         30
14     7698 BLAKE      MANAGER          7839 01/05/81          2850             30
15     7782 CLARK      MANAGER          7839 09/06/81          2450             10
16     7788 SCOTT     ANALYST          7566 13/07/87          3000             20
17     7839 KING       PRESIDENT        17/11/81          5000             10
18     7844 TURNER    SALESMAN         7698 08/09/81          1500            0           30
19     7876 ADAMS      CLERK            7788 13/07/87          1200             20
20     7900 JAMES      CLERK            7698 03/12/81          1100             30
21     7902 FORD       ANALYST          7566 03/12/81          3000             20
22     7934 MILLER    CLERK            7782 23/01/82          1300             10
23     7000 Zangla     SALESMAN         7566 17/12/80          2200             20
24     8000 jacques      7839             64
25     8001 jean-louis    7839             65

```

4.7 Exercice Trigger 6

Code :

```
1 SQL> ALTER TRIGGER nowweek_zangla ENABLE;
2
3 Declencheur modifie.
```

4.8 Exercice Trigger 7

4.8.1 Création de la table

Code :

```
1 CREATE TABLE STATS_zangla (
2     TypeMaj VARCHAR(6),
3     NbMaj NUMBER(3),
4     Date_derniere_Maj DATE
5 );
6
7 INSERT INTO STATS_zangla VALUES('UPDATE', 0, null);
8 INSERT INTO STATS_zangla VALUES('INSERT', 0, null);
9 INSERT INTO STATS_zangla VALUES('DELETE', 0, null);
```

4.8.2 A - trigger stats

Code :

```
1 CREATE OR REPLACE TRIGGER stat_zangla
2 BEFORE INSERT OR UPDATE OR DELETE ON emp
3 FOR EACH ROW
4 BEGIN
5     IF INSERTING THEN
6         UPDATE STATS_zangla SET NbMaj = NbMaj + 1 WHERE TypeMaj='INSERT';
7         UPDATE STATS_zangla SET Date_derniere_Maj = SYSDATE WHERE TypeMaj='INSERT';
8     END IF;
9     IF UPDATING THEN
10        UPDATE STATS_zangla SET NbMaj = NbMaj + 1 WHERE TypeMaj='UPDATE';
11        UPDATE STATS_zangla SET Date_derniere_Maj = SYSDATE WHERE TypeMaj='UPDATE';
12    END IF;
13    IF DELETING THEN
14        UPDATE STATS_zangla SET NbMaj = NbMaj + 1 WHERE TypeMaj='DELETE';
15        UPDATE STATS_zangla SET Date_derniere_Maj = SYSDATE WHERE TypeMaj='DELETE';
16    END IF;
17 END;
18 /
```

Résultat :

```
1 SQL> insert into emp (empno, ename, mgr, deptno) values (8002, 'jean', 7839, 65);
2
3 1 ligne creee.
4
5 SQL> insert into emp (empno, ename, mgr, deptno) values (8003, 'louis', 7839, 65);
6
7 1 ligne creee.
8
9 SQL> delete from emp where empno=8001;
10
11 1 ligne supprimee.
12
13 SQL> select * from STATS_zangla;
14
15 TYPEMA      NBMAJ DATE_DER
16 -----
17 UPDATE          0
18 INSERT         2 20/02/20
19 DELETE         1 20/02/20
```

4.8.3 B - for each row

Résultat :

```
1 SQL> UPDATE EMP SET SAL = SAL * 1.05;
2
3 17 lignes mises a jour.
4
5 SQL> select * from stats_zangla;
6
7 TYPEMA      NBMAJ DATE_DER
8 -----
9 UPDATE      17    20/02/20
10 INSERT      2    20/02/20
11 DELETE      1    20/02/20
```

4.9 Exercice Trigger 8

Code :

```
1 CREATE OR REPLACE TRIGGER checksal_zangla
2 BEFORE UPDATE OF job ON emp
3 FOR EACH ROW
4 DECLARE
5     borne_sup SalIntervalle_F2.hsal%type;
6     borne_inf SalIntervalle_F2.lsal%type;
7 BEGIN
8     IF :old.job != 'PRESIDENT' THEN
9         :new.sal := :old.sal + 100;
10    END IF;
11    SELECT lsal, hsal INTO borne_inf, borne_sup
12    FROM salintervalle_f2
13    WHERE job = :new.job;
14
15    IF :new.sal < borne_inf THEN
16        :new.sal := borne_inf;
17    ELSE
18        IF :new.sal > borne_sup THEN
19            :new.sal := borne_sup;
20        END IF;
21    END IF;
22 END;
23 /
```

Résultat :

```
1 SQL> select * from emp;
2
3 EMPNO ENAME      JOB          MGR HIREDATE      SAL        COMM        DEPTNO
4 -----
5     7369 SMITH      CLERK        7902 17/12/80      840
6     7499 ALLEN      SALESMAN     7698 20/02/81     1680         300         30
7     7521 WARD      SALESMAN     7698 22/02/81    1312,5         500         30
8     7566 JONES      MANAGER      7839 02/04/81    3123,75
9     7654 MARTIN     SALESMAN     7698 28/09/81    1312,5        1400         30
10    7698 BLAKE      MANAGER      7839 01/05/81    2992,5
11    7782 CLARK      MANAGER      7839 09/06/81    2572,5
12    7788 SCOTT      ANALYST      7566 13/07/87     3150
13    7839 KING      PRESIDENT    17/11/81     5250
14    7844 TURNER     SALESMAN     7698 08/09/81     1575         0           30
15    7876 ADAMS      CLERK        7788 13/07/87     1260
16    7900 JAMES      CLERK        7698 03/12/81     1155         30
17    7902 FORD      ANALYST      7566 03/12/81     3150         20
18    7934 MILLER     CLERK        7782 23/01/82     1365         10
19    7000 Zangla     SALESMAN     7566 17/12/80     2310         20
20    8000 jacques    7839
21
22 SQL> update emp set job='SALESMAN' where empno=7876;
23
24 1 ligne mise a jour.
25
```

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	840		20
7499	ALLEN	SALESMAN	7698	20/02/81	1680	300	30
7521	WARD	SALESMAN	7698	22/02/81	1312,5	500	30
7566	JONES	MANAGER	7839	02/04/81	3123,75		20
7654	MARTIN	SALESMAN	7698	28/09/81	1312,5	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2992,5		30
7782	CLARK	MANAGER	7839	09/06/81	2572,5		10
7788	SCOTT	ANALYST	7566	13/07/87	3150		20
7839	KING	PRESIDENT		17/11/81	5250		10
7844	TURNER	SALESMAN	7698	08/09/81	1575	0	30
7876	ADAMS	SALESMAN	7788	13/07/87	1360		20
7900	JAMES	CLERK	7698	03/12/81	1155		30
7902	FORD	ANALYST	7566	03/12/81	3150		20
7934	MILLER	CLERK	7782	23/01/82	1365		10
7000	Zangla	SALESMAN	7566	17/12/80	2310		20
8000	jacques		7839			64	

```
SQL> update emp set job='PRESIDENT' where empno=7876;
```

```
1 ligne mise a jour.
```

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	840		20
7499	ALLEN	SALESMAN	7698	20/02/81	1680	300	30
7521	WARD	SALESMAN	7698	22/02/81	1312,5	500	30
7566	JONES	MANAGER	7839	02/04/81	3123,75		20
7654	MARTIN	SALESMAN	7698	28/09/81	1312,5	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2992,5		30
7782	CLARK	MANAGER	7839	09/06/81	2572,5		10
7788	SCOTT	ANALYST	7566	13/07/87	3150		20
7839	KING	PRESIDENT		17/11/81	5250		10
7844	TURNER	SALESMAN	7698	08/09/81	1575	0	30
7876	ADAMS	PRESIDENT	7788	13/07/87	4500		20
7900	JAMES	CLERK	7698	03/12/81	1155		30
7902	FORD	ANALYST	7566	03/12/81	3150		20
7934	MILLER	CLERK	7782	23/01/82	1365		10
7000	Zangla	SALESMAN	7566	17/12/80	2310		20
8000	jacques		7839			64	