
Développement Base De Donnée

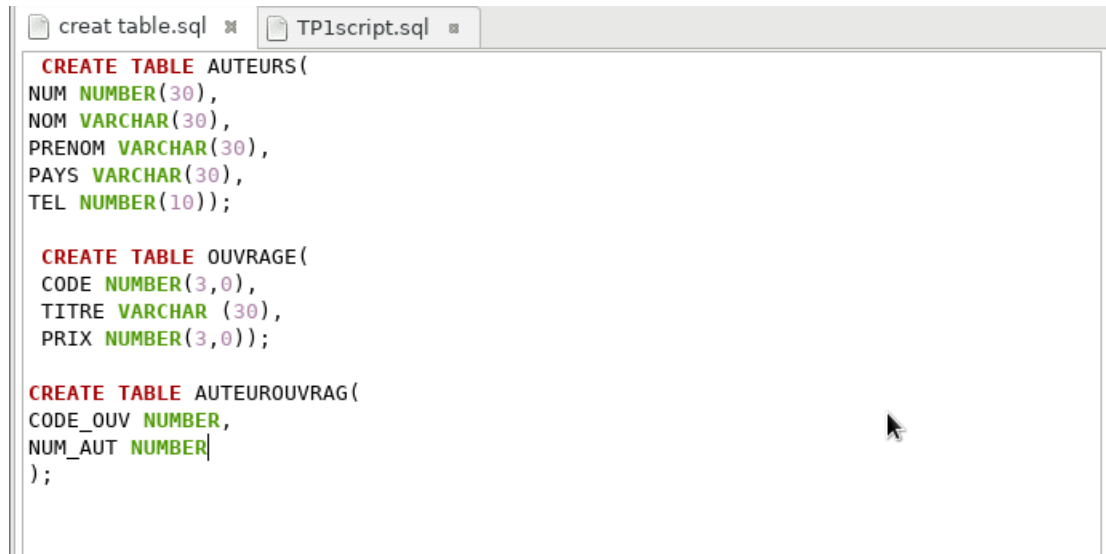
TP CR

Auteur : BIAN Xinran

TP 1 : PL/SQL

1. La création des tables

D'abord, créer ces tables sans contraintes et les remplir (remplir la table Auteurs avec SQL*Loader, la table Ouvrage avec la commande insert into)



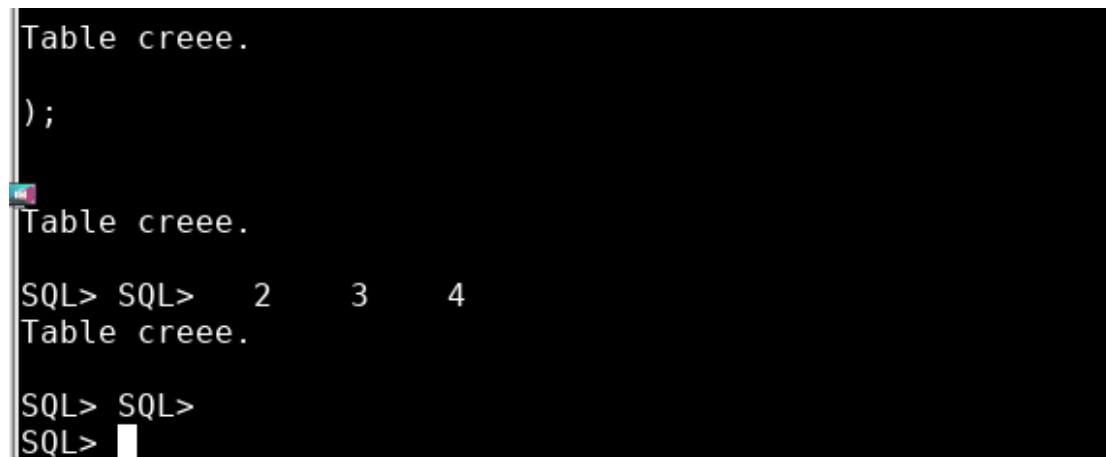
```
creat table.sql  TP1script.sql

CREATE TABLE AUTEURS(
NUM NUMBER(30),
NOM VARCHAR(30),
PRENOM VARCHAR(30),
PAYS VARCHAR(30),
TEL NUMBER(10));

CREATE TABLE OUVRAGE(
CODE NUMBER(3,0),
TITRE VARCHAR (30),
PRIX NUMBER(3,0));

CREATE TABLE AUTEUROUVRAG(
CODE_OUV NUMBER,
NUM_AUT NUMBER
);
```

Figure 1 – Création des tables



```
Table creee.

);

Table creee.

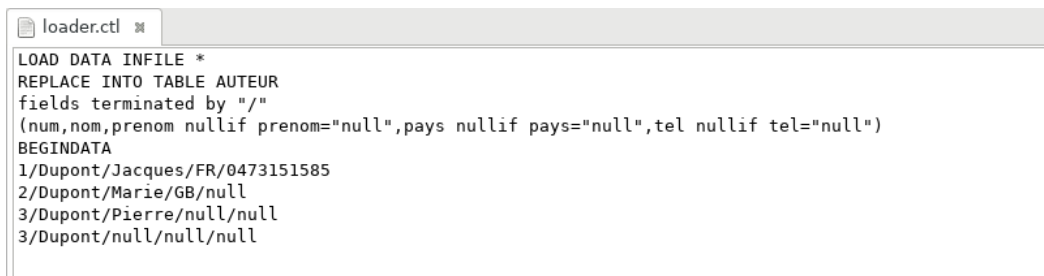
SQL> SQL> 2 3 4
Table creee.

SQL> SQL>
SQL> 
```

Figure 2 – Création des tables – Résultat

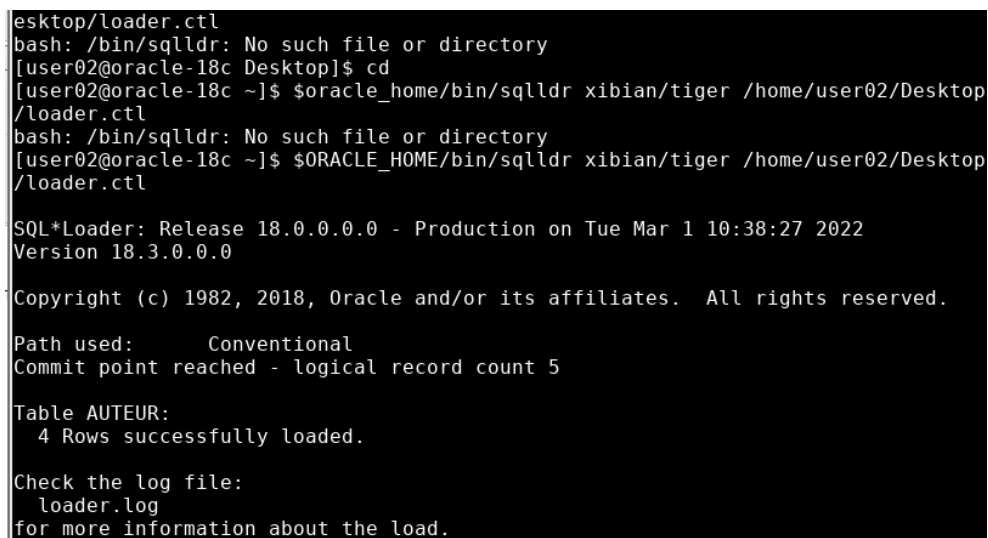
2. Insertion des données

● 2.1 Avec SQL Loader



```
loader.ctl
LOAD DATA INFILE *
REPLACE INTO TABLE AUTEUR
fields terminated by "/"
(num,nom,prenom nullif prenom="null",pays nullif pays="null",tel nullif tel="null")
BEGINDATA
1/Dupont/Jacques/FR/0473151585
2/Dupont/Marie/GB/null
3/Dupont/Pierre/null/null
3/Dupont/null/null/null
```

Figure 3 – fichier contrôle



```
esktop/loader.ctl
bash: /bin/sqlldr: No such file or directory
[user02@oracle-18c Desktop]$ cd
[user02@oracle-18c ~]$ $oracle_home/bin/sqlldr xibian/tiger /home/user02/Desktop
/loader.ctl
bash: /bin/sqlldr: No such file or directory
[user02@oracle-18c ~]$ $ORACLE_HOME/bin/sqlldr xibian/tiger /home/user02/Desktop
/loader.ctl

SQL*Loader: Release 18.0.0.0.0 - Production on Tue Mar 1 10:38:27 2022
Version 18.3.0.0.0

Copyright (c) 1982, 2018, Oracle and/or its affiliates. All rights reserved.

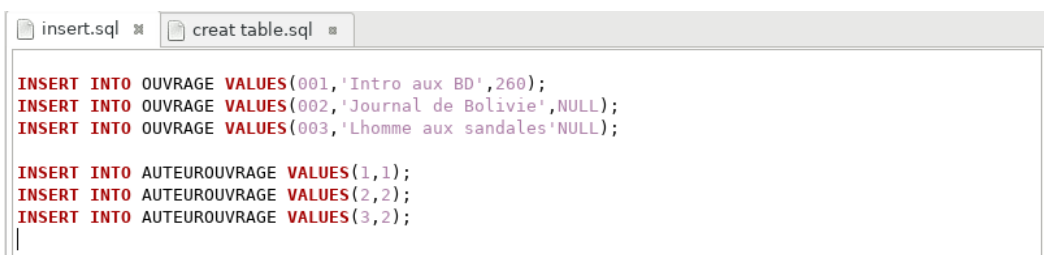
Path used:          Conventional
Commit point reached - logical record count 5

Table AUTEUR:
  4 Rows successfully loaded.

Check the log file:
  loader.log
for more information about the load.
```

Figure 4 – Execution de la commande sqlldr

● 2.2 Avec la requête insert



```
insert.sql  creat table.sql
INSERT INTO OUVRAGE VALUES(001,'Intro aux BD',260);
INSERT INTO OUVRAGE VALUES(002,'Journal de Bolivie',NULL);
INSERT INTO OUVRAGE VALUES(003,'L'homme aux sandales',NULL);

INSERT INTO AUTEUROUVRAGE VALUES(1,1);
INSERT INTO AUTEUROUVRAGE VALUES(2,2);
INSERT INTO AUTEUROUVRAGE VALUES(3,2);
```

Figure 5 – Insertion des données avec la commande insert

3. Ajout des contraintes

● 3.1 Les clés primaires

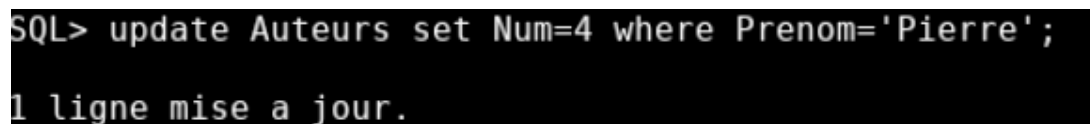
Ajouter la contrainte « clé primaire » sur la relation Auteurs en récupérant les lignes qui posent problèmes dans une table nommée aut_violation.



```
insert.sql creat table.sql cle.sql
CREATE TABLE PK_VIOLATION(
  row_id rowid,
  owner varchar2(30),
  table_name varchar2(30),
  constraint varchar2(120)
);

alter table auteurs add constraint PK_AUTEUR primary key(Num) exceptions into pk_violation;
alter table ouvrage add constraint PK_ouvrage primary key(Code) exceptions into pk_violation;
alter table AUTEUROUVRAGE add constraint PK_AUTEUROUVRAGE primary key(Code_ouv, Num_aut)
exceptions into pk_violation;
```

Figure 6 – L’ajout des contraintes

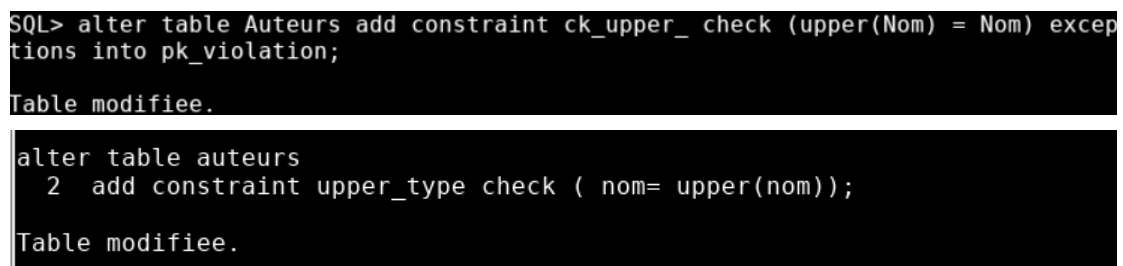


```
SQL> update Auteurs set Num=4 where Prenom='Pierre';
1 ligne mise a jour.
```

Figure 7 – Update de la table

● 3.2 le nom de auteur

Ajouter la contrainte sur la table ouvrage qui assure que les noms des auteurs sont toujours en majuscules en récupérant les lignes qui posent problèmes dans la table aut_violation.



```
SQL> alter table Auteurs add constraint ck_upper_ check (upper(Nom) = Nom) excep
tions into pk_violation;
Table modifiee.

alter table auteurs
  2 add constraint upper_type check ( nom= upper(nom));
Table modifiee.
```

Figure 8 – La contrainte Majuscule

4. Supprimer la contrainte clé primaire

Supprimer la contrainte clé primaire de la table Auteurs.

```
delete from auteurs  
  2 where num=3 and prenom is null;  
  
0 lignes supprimees.
```

Figure 9 – Supprimer la contrainte de la clé primaire

TP 2 : PL/SQL

1. Afficher à l'écran le nom , salaire, la commission de l'employé
'MILLER' ainsi que le nom du département dans lequel il travail.

```
DECLARE
nom varchar2(10);
salaire number(7,2);
commi number(7,2);
depmt varchar2(14);
BEGIN
    select ename,sal,comm,D.dname INTO nom,salaire,commi,depmt
    from Dept D, Emp E
    where E.ename = 'MILLER' and D.deptno = E.deptno;

    dbms_output.put_line('nom'||nom || ' ' || 'salaire' || salaire || ' ' || 'depmt' || depmt);
END;
/
```

Figure 10 – Affichage des informations sur un employé

```
SQL> set serveroutput on
SQL> start q1.sql
nomMILLER salaire1300 depmtACCOUNTING

Procedure PL/SQL terminee avec succes.
```

```
SQL> select ename,sal,deptno from emp
2  where ename='MILLER';
```

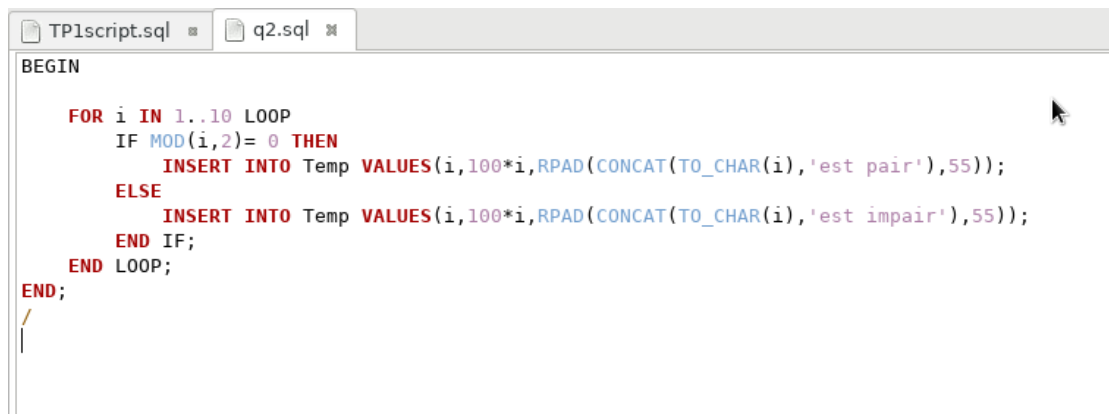
ENAME	SAL	DEPTNO
MILLER	1300	10

Figure 11 – resultat des informations sur un employé

2. Ecrire un programme PL/SQL permettant d'insérer dans la table Temp les 100 lignes suivantes :

Temp

Num COL1	Num COL2	Char COL
1	100	1 est impair
2	200	2 est paire
3	300	3 est impair
...
...
100	10000	100 est paire



```

BEGIN
  FOR i IN 1..100 LOOP
    IF MOD(i,2)= 0 THEN
      INSERT INTO Temp VALUES(i,100*i,RPAD(CONCAT(TO_CHAR(i),'est pair'),55));
    ELSE
      INSERT INTO Temp VALUES(i,100*i,RPAD(CONCAT(TO_CHAR(i),'est impair'),55));
    END IF;
  END LOOP;
END;

```

Figure 12 – Insertion des tuples

NUM_COL1	NUM_COL2	CHAR_COL
89	8900	89est impair
90	9000	90est pair
91	9100	91est impair
92	9200	92est pair
93	9300	93est impair
94	9400	94est pair
95	9500	95est impair
96	9600	96est pair
97	9700	97est impair
98	9800	98est pair
99	9900	99est impair
NUM_COL1	NUM_COL2	CHAR_COL
100	10000	100est pair

100 lignes selectionnees.

Figure 13– Resultat des tuples

3. Insérer dans la table Temp (sal, empno, ename) les 5 employés les mieux payés

```
TP1script.sql  q2.sql  q3.sql
DECLARE
CURSOR cur IS SELECT sal,empno,ename
               FROM Emp
               ORDER BY sal DESC;
sala emp.sal%type;
num  emp.empno%type;
name emp.ename%type;
BEGIN
  OPEN cur;
  FOR i IN 1..5 LOOP
    FETCH cur INTO sala,num,name;
    EXIT WHEN cur%NOTFOUND;
    INSERT INTO Temp VALUES(sala,num,name);
  END LOOP;
  CLOSE cur;
END;
```

Figure 14 – Insertion des tuples avec les curseurs

NUM_COL1	NUM_COL2	CHAR_COL
81	8100	81est impair
82	8200	82est pair
83	8300	83est impair
84	8400	84est pair
85	8500	85est impair
86	8600	86est pair

105 lignes selectionnees.

Figure 15 – Résultat d'insertion avec les curseurs

4. Récupérer dans une table Temp tous les employés dont les revenus mensuels (salaire + commission) sont supérieurs à \$2000.

```
DECLARE
CURSOR cur IS SELECT sal,comm,ename,empno
               FROM Emp;

sala Emp.sal%type;
num  Emp.empno%type;
name Emp.ename%type;
com  Emp.comm%type;
salaire number(8,0);
BEGIN
    OPEN cur;
    LOOP
        FETCH cur INTO sala,com,name,num;
        EXIT WHEN cur%NOTFOUND;
        salaire := sala;
        IF com IS NOT NULL THEN
            salaire := salaire + com;
        END IF;
        IF salaire > 2000 THEN
            INSERT INTO Temp VALUES(salaire,num,name);
        END IF;
    END LOOP;
    CLOSE cur;
END;
```

Figure 16 – Insertion des tuples avec les curseurs

```
SQL> select * from temp ;
```

NUM_COL1	NUM_COL2	CHAR_COL
5000	7839	KING
2975	7566	JONES
2850	7698	BLAKE
2450	7782	CLARK
3000	7902	FORD
3000	7788	SCOTT
3500	7000	BIAN

7 lignes selectionnees.

Figure 17 – Résultat d'insertion avec les curseurs

5. Insérer dans une table Temp(sal, ename) le premier employé qui a un salaire supérieur à \$4000 et qui est plus haut dans la chaîne de la commande que l'employé 7902.

```
q5.sql x
DECLARE
CURSOR cur IS SELECT sal,empno,ename
                FROM Emp WHERE sal > 4000 and empno NOT IN
                (SELECT empno
                 FROM Emp
                 CONNECT BY PRIOR empno = mgr
                 START WITH empno = 7902);

sala emp.sal%type;
num  emp.empno%type;
name emp.ename%type;
BEGIN
    OPEN cur;
    FETCH cur INTO sala,num,name;
    INSERT INTO Temp VALUES(sala,num,name);
    CLOSE cur;
END;
/
```

Figure 18 – Insertion des tuples avec les curseurs

```
SQL> select * from temp;

  NUM_COL1  NUM_COL2  CHAR_COL
-----
      5000      7839  KING
      5000      7839  KING
      2975      7566  JONES
      2850      7698  BLAKE
      2450      7782  CLARK
      3000      7902  FORD
      3000      7788  SCOTT
      3500      7000  BIAN

8 lignes selectionnees.
```

Figure 19 – Résultat d'insertion avec les curseurs

TP 3 : Programmation en PL/SQL

1. Procédures et fonctions stockés

- **1.1 Procédure createdept_votrenom (numéro_dept, dept_name, localisation) :** permettant de créer de nouveaux départements dans la table Dept. Vérifier si le numéro_dept existe déjà dans la table.

```
create or replace procedure createdept_bian (numero_dept IN NUMBER, dept_name IN VARCHAR2, localisation IN VARCHAR2) IS
d NUMBER;
DUPLICATE EXCEPTION; CURSOR c is select deptno from dept;
flag integer :=0;
v_deptno dept.deptno%type;
Begin
  OPEN c;
  LOOP
    FETCH c INTO v_deptno;
    EXIT WHEN (c%notfound);
    if v_deptno = numero_dept then
      flag:=1;
    end if;
  END LOOP;
  if flag = 1 then
    raise DUPLICATE;
  end if;
  DBMS_OUTPUT.PUT(' bien ajouter ');
  Insert Into Dept Values(numero_dept,dept_name,localisation);
exception
  when DUPLICATE THEN
    DBMS_OUTPUT.PUT(' ce numero existe deja ');
END ;
/
```

Figure 20– Procédure

Cette procédure nous permet d'avoir des enregistrements qui ont un deptno different.

```
SQL> start q1.sql

Procedure creee.

SQL> exec createdept_bian( 1,'wuhan','china');

Procedure PL/SQL terminee avec succes.
```

Figure 21 – Résultat de procedure

- **1.2 Fonction salok_votrenom (job, salaire) Return Number** : fonction qui vérifie que le salaire d'un job est dans un intervalle donné (nécessité d'une table SalIntervalle (job, lsal, hsal)). Retourner 1 si c'est la cas, sinon 0.

```
create table SalIntervalle(job varchar2(9), lsal number(7,2), hsal number(7,2));
insert into SalIntervalle values('ANALYST',2500,3000);
insert into SalIntervalle values('CLERK',900,1300);
insert into SalIntervalle values('MANAGER',2400,3000);
insert into SalIntervalle values('PRESIDENT',4500,4900);
insert into SalIntervalle values('SALESMAN',1200,1700);

create or replace function sal_ok_bian
(job varchar2, salaire number)
return number
IS
cpt number(5);
BEGIN
select COUNT(*) INTO cpt FROM SalIntervalle S WHERE S.job = job
AND salaire > lsal AND salaire < hsal;

RETURN (cpt);
End sal_ok;
/

VARIABLE moyenne NUMBER
EXECUTE :moyenne:=sal_ok_bian('PRESEDENT',1000);|
PRINT moyenne;
```

Figure 22 – Fonction

Pour les paramètres job='PRESEDENT' et salaire=1000, nous avons obtenu la valeur 0 :

```
BEGIN :moyenne:=sal_ok_bian('PRESEDENT',1000); END;

*
ERREUR a la ligne 1 :
ORA-06550: Ligne 1, colonne 18 :
PLS-00905: l'objet XIBIAN.SAL_OK_BIAN n'est pas valide
ORA-06550: Ligne 1, colonne 7 :
PL/SQL: Statement ignored

MOYENNE
-----
```

Figure 23 – Resultat de fonction

- **1.3 Procédure raisesalary_ votrenom (emp_id, amount) : permettant d'augmenter le salaire d'un employé d'un montant donné si le salaire augmenté est dans l'intervalle concerné (la fonction salok_votrenom). Sinon, afficher un message d'erreur.**

```
Create or replace procedure raisesalary_bian(emp_id number, amount number)
Is
cpt number;
Begin
    SELECT COUNT(*) into cpt FROM Emp E, SalIntervalle S Where emp_id = E.empno
    AND S.job = E.job
    AND (E.sal + amount) < S.hsal
    AND (E.sal + amount) > S.lsal;
    if cpt = 1 THEN
        update Emp set sal = sal + amount
        where emp_id=empno;
    END IF;
END raisesalary;
/
```

Figure 24 – Procedure

2. Bloc PL/SQL

- Implémenter un bloc PL/SQL qui fait un backup de toutes les tables de l'utilisateur connecté en les copiant

```
Declare
CURSOR curDelete IS SELECT table_name
                      FROM user_tables WHERE table_name LIKE '%_old' ORDER BY sal DESC;
CURSOR curSave IS SELECT table_name
                  FROM user_tables WHERE table_name NOT LIKE '%_old' ORDER BY sal DESC;|
name_table varchar2(20);
requete varchar2(80);
BEGIN
open curDelete;
LOOP
    FETCH curDelete INTO name_table;
    EXIT WHEN cur%NOTFOUND;
    execute immediate ('DROP table name_table;');
END LOOP;
close curDelete;
open curSave;
LOOP
    FETCH curSave INTO name_table;
    EXIT WHEN cur%NOTFOUND;
    //gerer la generation de chaine via des concat etc
    create table CONCAT(name_table,'_old') as Select * from nametable;
END LOOP;
close curSave;
END;
```

Figure 25 – Backup

```
PK_VIOLATION
AUTEURS
AUTEUROUVRAGE
OUVRAGE
DEPT
EMP
VOTRENOM
TEMP
SALINTERVALLE_F2_OLD
PK_VIOLATION_OLD

TABLE_NAME
-----
AUTEURS_OLD
AUTEUROUVRAGE_OLD
OUVRAGE_OLD
DEPT_OLD
EMP_OLD
VOTRENOM_OLD
TEMP_OLD

18 lignes selectionnees.
```

Figure 26 – Backup - Résultat

TP4 : Dev. BD

1. Package

1.1 Partie spécification

```
Create or replace Package bian AS
Type EmpType is record (emp_no NUMBER, ename VARCHAR2(100));
cursor emp_par_dep_bian(dept_no NUMBER) RETURN EmpType;
function salok_bian(v_job SalIntervalle, job%type, salaire NUMBER) return number;
procedure raise_salary_bian(emp_id number, amount number);
procedure afficher_emp_bian(dept_n number);
end bian;
```

Figure 27 – Package - Partie specification

1.2 Partie corps

```
CREATE OR REPLACE PACKAGE BODY bian IS
    CURSOR emp_par_dep_bian(dept_no NUMBER) RETURN EmpType IS
        SELECT empno, ename FROM emp where emp.deptno=dept_no;
    FUNCTION salok_bian (v_job SalIntervalle, job%type, salaire NUMBER) RETURN NUMBER IS
        v_lsal NUMBER;
        v_hsal NUMBER;
    BEGIN
        Select lsal,hsal into v_lsal,v_hsal from SalIntervalle where
        SalIntervalle,job=v_job;

        if salaire <= v_hsal then
            return 1;
        else
            return 0;
        end if;
        exception when NO_DATA_FOUND THEN
            return 2;
    END;

    PROCEDURE raise_salary_bian (emp_id NUMBER, amount NUMBER) IS
        v_empno NUMBER;
        v_sal NUMBER;
        v_job VARCHAR2(100);
        v_return NUMBER;
        e EXCEPTION;
    BEGIN
        SELECT empno,job,sal into v_empno,v_job,v_sal FROM emp WHERE empno=emp_id;
        v_return := salok_bian(v_job, v_sal +amount);
        IF v_return = 1 THEN
            DBMS_OUTPUT.PUT('Le salaire est bien modifié');
            UPDATE emp SET sal=v_sal+amount WHERE empno = v_empno;
        ELSIF v_return = 0 THEN
            raise e;
        ELSE
            DBMS_OUTPUT.PUT('introuvable');
        END IF;
    EXCEPTION
        when e then
            DBMS_OUTPUT.PUT('Le salaire a deja atteind le maximum');
    END;
```

```

PROCEDURE afficher_emp_bian(dept_n NUMBER) IS
v_ename VARCHAR2 (100);
v_empno NUMBER;
BEGIN
    OPEN emp_par_dep_bian(dept_n);
    LOOP
        FETCH emp_par_dep_bian INTO v_empno,v_ename;
        EXIT WHEN emp_par_dep_bian%NOTFOUND;
        DBMS_OUTPUT.PUT('empno=' || v_empno || 'ename=' || v_ename);
    END LOOP;
    CLOSE emp_par_dep_bian;
END;
END bian;
|

```

Figure 28 Partie corps

```

81 declare
82     v number;
83 begin
84     bian.afficher_emp_bian(20);
85 end;
86

```

Figure 29 – Appel de la fonction

EMPNO	ENAME

7876	ADAMS
7499	ALLEN
7000	BIAN
7698	BLAKE
7782	CLARK
7902	FORD
7900	JAMES
7566	JONES
7839	KING
7654	MARTIN
7934	MILLER
EMPNO	ENAME

7788	SCOTT
7369	SMITH
7844	TURNER
7521	WARD

Figure 30 – Appel de la fonction raise_salary_bian

2. Triggers

- 2.1 (Nom du trigger : raise_votrenom) Le salaire d'un employé ne diminue jamais.

```
CREATE OR REPLACE Trigger raise_bian
BEFORE UPDATE of sal on emp
For each row
Begin
    if (:old.sal > :new.sal) then
        raise_application_error(-20120, 'le salaire ne peut pas diminuer');
    End if;
End;
/
```

Declencheur cree.

```
SQL> UPDATE emp SET sal = sal - 100 WHERE ename = 'BIAN';
      *
```

ERREUR a la ligne 1 :

ORA-20120: le salaire ne peut pas diminuer

ORA-06512: a "XIBIAN.RAISE_LY", ligne 3

ORA-04088: erreur lors d'execution du declencheur 'XIBIAN.RAISE_BIAN'

Figure 31 – Trigger - raise_bian

- 2.2 (Nom du trigger : numdept_votrenom)

```
CREATE OR REPLACE Trigger numdept_bian
BEFORE UPDATE of deptno on emp
For each row
Begin
    if (:new.deptno > 69 OR :new.deptno < 61) then
        raise_application_error(-20121, 'le departement doit etre dans [61,69]');
    End if;
End;
/
```

Declencheur cree.

```
SQL> UPDATE emp SET deptno = 70 WHERE ename = 'BIAN';
      *
```

ERREUR a la ligne 1 :

ORA-20121: le departement doit etre dans [61,69]

ORA-06512: a "XIBIAN.NUMDEPT_BIAN", ligne 3

Figure 32 – Trigger - numdept_bian

- 2.3 (Nom du trigger : dept_votrenom)

```
CREATE OR REPLACE Trigger dept_bian
BEFORE INSERT OR UPDATE of deptno on emp
For each row
DECLARE
    c number;
Begin
    SELECT COUNT(*) INTO c FROM DEPT WHERE DEPTNO = :new.deptno;
    if c = 0 then
        INSERT INTO DEPT VALUES(:new.deptno, 'A SAISIR', 'A SAISIR');
    End if;
End;
/
```

Declencheur cree.

SQL> INSERT INTO EMP VALUES(6969, 'Longinus', 'CLERK', 7839,
TO_DATE('20/12/2020', 'DD/MM/YYYY'), 1000, 200, 48);

1 ligne creee

SQL> SELECT * FROM DEPT;

DEPTNO	DNAME	LOC
1	WUHAN	CHINA
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
48 A SAISIR	A SAISIR	

Figure 33 – Trigger - dept_bian

- **2.4 (Nom du trigger : noweek_votrenom) Pour des raisons de sécurité, on souhaite interdire toute modification de la relation employé pendant le week-end (samedi et dimanche).**

```
CREATE OR REPLACE Trigger noweek_bian
BEFORE UPDATE OR INSERT OR DELETE on emp
For each row
Begin
    if (TO_CHAR(SYSDATE, 'D') = '7' OR TO_CHAR(SYSDATE, 'D') = '6') then
        RAISE_APPLICATION_ERROR(-20021, 'Cannot insert record on weekends');
    End if;
End;
/
```

Declencheur cree.

SQL> UPDATE emp SET sal = sal + 100 WHERE ename = 'BIAN';

UPDATE emp SET sal = sal + 100 WHERE ename = 'BIAN'

*

ERREUR a la ligne 1 :

ORA-20021: Cannot insert record on weekends

ORA-06512: a "XIBIAN.NOWEEK_BIAN", ligne 3

ORA-04088: erreur lors d'execution du declencheur 'XIBIAN.NOWEEK_BIAN'

Figure 34 – Trigger - noweek_bian

- **2.5 Désactiver le trigger de la question précédente et tester le.**

```
ALTER TRIGGER nowweek_bian DISABLE;
```

Pour désactiver le trigger, j'ai utilisé DISABLE.

Declencheur modifie.

```
SQL> UPDATE emp SET sal = sal + 100 WHERE ename = 'BIAN';
```

1 ligne mise a jour.

Figure 35 – Trigger – disable

- **2.6 Réactiver le trigger la question précédente.**

```
ALTER TRIGGER nowweek_bian ENABLE;
```

Declencheur modifie.

```
SQL> UPDATE emp SET sal = sal + 100 WHERE ename = 'BIAN';
```

```
UPDATE emp SET sal = sal + 100 WHERE ename = 'BIAN'
```

*

ERREUR a la ligne 1 :

ORA-20021: Cannot insert record on weekends

ORA-06512: a "XIBIAN.NOWEEK_BIAN", ligne 3

ORA-04088: erreur lors d'execution du declencheur 'XIBIAN.NOWEEK_BIAN'

Figure 36 – Désactiver et réactiver le trigger nowweek_bian

- 2.7 (Nom du trigger : stat_votrenom)

A)

```
CREATE TABLE STATS_bian (TypeMaj varchar2(10), NbMaj number, Date_derniere_Maj date);

insert into STATS_bian values ('INSERT', 0, NULL);
insert into STATS_bian values ('DELETE', 0, NULL);
insert into STATS_bian values ('UPDATE', 0, NULL);

CREATE OR REPLACE Trigger update_stats_bian
AFTER UPDATE OR INSERT OR DELETE on emp
For each row
Begin
    if INSERTING then
        UPDATE STATS_bian SET NbMaj = NbMaj+1, Date_derniere_Maj = SYSDATE WHERE TypeMaj='INSERT';
    End if;
    if UPDATING then
        UPDATE STATS_bian SET NbMaj = NbMaj+1, Date_derniere_Maj = SYSDATE WHERE TypeMaj='UPDATE';
    End if;
    if DELETING then
        UPDATE STATS_bian SET NbMaj = NbMaj+1, Date_derniere_Maj = SYSDATE WHERE TypeMaj='DELETE';
    End if;
End;
/
```

Figure 37 – A

Declencheur modifie.

SQL> UPDATE emp SET sal = sal + 100 WHERE ename = 'BIAN';

UPDATE emp SET sal = sal + 100 WHERE ename = 'BIAN'

*

ERREUR a la ligne 1 :

ORA-20021: Cannot insert record on weekends

ORA-06512: a "XIBIAN.NOWEEK_BIAN", ligne 3

ORA-04088: erreur lors d'execution du declencheur 'XIBIAN.NOWEEK_BIAN'

SQL> UPDATE emp SET sal = sal + 100 WHERE ename = 'Longinus';

1 ligne mise a jour.

SQL> select * from stats_bian;

TYPEMAJ	NBMAJ	DATE_DER
INSERT	1	20/03/22
DELETE	0	
UPDATE	1	20/03/22

SQL> DELETE FROM emp WHERE ename = 'Longinus';

1 ligne supprimee.

SQL> select * from stats_bian;

TYPEMAJ	NBMAJ	DATE_DER
INSERT	1	20/03/22
DELETE	1	20/03/22
UPDATE	1	20/03/22

B)

On constate en utilisant par exemple `UPDATE emp SET sal = sal * 1.05;` le résultat suivant :

SQL> select * from stats_bian;		
TYPEMAJ	NBMAJ	DATE_DER

INSERT	0	
DELETE	0	
UPDATE	15	20/03/22

- 2.8 (Nom du trigeur : `checksal_votrenom`) Chaque fois qu'un employé change de « job », on lui accorde une augmentation de salaire d'un montant de 100 euros.

```
CREATE OR REPLACE Trigger checksal_bian
BEFORE UPDATE of job on emp
For each row
DECLARE
    min_sal emp.sal%type;
    max_sal emp.sal%type;
Begin
    if (:old.job != 'PRESIDENT') then
        SELECT lsal, hsal INTO min_sal, max_sal FROM SalIntervalle WHERE job = :new.job;
        :new.sal := GREATEST(min_sal, LEAST(max_sal, :old.sal+100));
    End if;
End;
```

SQL> select * from emp where empno = 7499;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

7499	ALLEN	SALESMAN	7698	20/02/81	1680	300	30

SQL> update emp set job='MANAGER' WHERE empno = 7499;

1 ligne mise a jour.

SQL> select * from emp where empno = 7499;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

7499	ALLEN	MANAGER	7698	20/02/81	2400	300	30

Figure 38 – Trigger - `checksal_bian`