

Connectez-vous au compte exemple SCOTT.

Exercice 1-

1- écrire une fonction qui permet de retourner le nom du département, en lui passant comme paramètre le numéro du département.

2- refaire la même question en utilisant cette une procédure, en lui passant comme paramètre le numéro du département et le nom du département.

Exercice 2- écrire une procédure (en utilisant les curseurs) qui permet d'afficher les noms des employés sous la forme suivante :

<<Nom employé>> || <<Profession>> || <<Nom département >>

Exercice 3- écrire une procédure qui permet d'augmenter les salaires des employés selon leurs grades.

- 10% pour le grade 1 et grade 2.
- 15% pour le grade 3.
- 20% pour le grade 4 et grade 5.

Exercice 4- Créer un trigger qui -avant de mettre à jour l'âge- d'un étudiant donné, le nouvel âge saisi doit être supérieur à l'ancien.

Exercice 5-Définir un trigger en insertion permettant d'implémenter une numérotation automatique de la **cléNumEt** de la table Etudiant. Le premier numéro doit être égal à 1.

Exercice 6-Il s'agit de réaliser la gestion d'une base de données commerciale des produits pharmaceutiques. Les tables de la base de données sont définies ci-dessous :

Produit(**Nprod**, design, date_p, PU, qte_stock), ligne_Commande(**Ncmd**,**Nprod**,qte,prix_unité), commande(**Ncmd**,date_cmd,montant).

- 1- Connectez-vous à USER1, et exécuter le script (donné au cours de la séance) pour créer le schéma précédent, et remplir les tables.
- 2- Définir un trigger qui doit mettre à jour automatiquement les montants de la table commande, ainsi que les quantités en stock de la table produit.

NB : On ne pourra de plus pas enregistrer une ligne de commande que l'on ne pourrait satisfaire pour cause de stock insuffisant.

Exercice 1 :

```
create or replace trigger alert after insert or update or delete on etudiant
begin
dbms_output.put_line('Operation termine');
end;
/
```

Exercice 2 :

```
create or replace trigger verifeage
before insert or update on etudiant
for each row
declare
exc exception;
begin
if :new.ageet<18 or :new.ageet>100
then
```

```

        raise exc;
    end if;
    exception
when exc then
Raise_Application_error(-20001, 'l age d un etudiant doit etre compris entre 18 et 100');
end;
/

```

Exercice 3 :

Sql > select trigger_name from user_triggers ;

Exercice 4 :

create or replace trigger update_age before update on etudiant
for each row

```

declare
    exc exception;
begin
    if :new.ageet<:old.ageet
    then

        raise exc;
    end if;
    exception
when exc then
Raise_Application_error(-20001, 'le ouveau age doit etre superieur a l ancienne');
end;
/

```

Exercise 5 :

**create or replace trigger inc before insert on etudiant
for each row**

declare

id number;

begin

select count(*) into id from etudiant;

if id = 0

then

:new.numet := 1 ;

else

select max(numet) into id from etudiant;

:new.numet := id + 1;

end if;

end;

/

Exercice 6 :

CREATE or replace trigger updateCOMPRO

after insert or update or delete on ligne_commande

for each row

declare

MT float;

qt Integer;

stock1 Integer;

NC Integer;

NP Integer;

begin

IF Inserting then

MT:=:new.qte * :new.prix_unité;

qt:=:new.qte;

NC:=:new.no_cmd;

NP:=:new.no_prod;

END IF;

IF deleting then

MT:=-:old.qte * :old.prix_unité;

qt:=-:old.qte;

NC:=:old.no_cmd;

NP:=:old.no_prod;

END IF;

IF updating then

MT:=:new.qte * :new.prix_unité - :old.qte*:old.prix_unité;

qt:=:new.qte - :old.qte;

NC:=:new.no_cmd;

NP:=:new.no_prod;

END IF;

select stock into stock1 from produit where no-prod=NP;

```
update commande set montant=MT where no_cmd=NC;  
update produit set stock=stock1 where no_prod=NP;  
end;  
/
```

Connectez-vous au compte exemple SCOTT.

Exercice 1

- 1- Ecrire une fonction qui permet de retourner le nom du département, prenant en paramètre le numéro du département.
- 2- Refaire la même question en utilisant une procédure, prenant en paramètre le numéro du département et le nom du département.

Exercice 2

1- Ecrire une procédure (en utilisant les curseurs et la fonction de l'exercice 1) qui permet d'afficher les noms des employés sous la forme suivante:

L'employé <<Nom employé>> a la profession <<Profession>> dans le département <<Nom département >>.

2- Ecrire une fonction, prenant en paramètre le numéro de l'employé, qui permet d'afficher et de retourner le nom de l'employé (l'affichage est de même forme que la question 1).

Exercice 3

Ecrire une fonction prenant en paramètre le numéro de l'employé et retournant le nom de son chef, si il s'agit du chef, la fonction retourne "Aucun".

NB: utiliser la question 2 de l'exercice 2.

Exercice 4

Ecrire une procédure qui permet d'augmenter les salaires (sans tenir compte la commission) des employés selon leurs grades.

- 10% pour le grade 1 et grade 2.
- 15% pour le grade 3.

- 20% pour le grade 4 et grade 5.

Exercice 5

DEVOIR à rendre par email avant le Mardi 19/05/2015.

Ecrire un package gestion_emp permettant de regrouper les procédures et les fonctions créées dans les quatre exercices.

C. FONCTIONS/ PROCEDURES/PACKAGES

Exercice1

1-

```
create or replace function nom_dept(numero dept.deptno%type) return  
VARCHAR2 is  
nom dept.dname%type ;  
begin  
select dname into nom from dept where deptno = numero;  
return nom;  
exception
```

```
when no_data_found then  
DBMS_OUTPUT.PUT_LINE('le département n'existe pas');  
end nom_dept;  
/
```

Appel : sql> select nom_dept(20) from dept;

2- PROCEDURE

```
create or replace procedure nom_dept2(numero IN dept.deptno%type, nom OUT  
dept.dname%type) is  
begin  
select dname into nom from dept where deptno = numero ;
```

```
exception  
  when no_data_found then  
    DBMS_OUTPUT.PUT_LINE('le département n'existe pas');  
  end nom_dept2;
```

```
/
```

```
--appel de la procédure
```

```
declare  
  pnom VARCHAR2(100);  
begin  
  nom_dept2(1,pnom); -- appel de la procedure  
  dbms_output.put_line(pnom);  
end;
```

```
/
```

Exercice2

1-

create or replace procedure affiche is

cursor S is select ENAME, JOB, DEPTNO from EMP;

v_ename emp.ename%type;
v_job emp.job%type;
v_deptno emp.deptno%TYPE;

ligne s%rowtype;

begin

open s;

loop

fetch s into v_ename, v_job, v_deptno ligne;

dbms_output.put_line('L'employé '|| ligne.ename|| ' a la profession ' || ligne.job || '
dans le département ' || nom_dept(ligne.deptno));

```
exit when s%notfound;  
end loop;  
close s;  
end affiche;  
/
```

2-

```
create or replace function rname(nemp in emp.empno%type) return emp.ename%type  
is  
  nom emp.ename%type;  
  fonction emp.job%type;  
  no emp.deptno%type;
```

```

begin
select ename,job,deptno into nom, fonction,no from emp where empno=nemp;
dbms_output.put_line('L'employé '|| nom || ' a la profession ' || fonction|| ' dans le
département ' || nom_dept(no));
return nom;
exception
    when no_data_found then
dbms_output.put_line('cet employé n'existe pas');
end rname;
/

```

Exercice3

```

create or replace function nchef(no emp.empno%type) return emp.ename%type
is
nc emp.empno%type;
nch emp.ename%type;
begin
select mgr into nc from emp where empno=no;
dbms_output.put_line(nc);
if nc is null then

```

```

nch:='Aucun';
else
nch:=rname(nc);
end if;
return nch;
exception
when no_data_found then
dbms_output.put_line('cet employé n''existe pas');

return null;
end nchef;
/

```

Exercice4

```

create or replace procedure augmentation is
cursor c is
select empno,sal,grade from emp, SalGrade where
EMP.SAL between SalGrade.LOSAL and SalGrade.HISAL FOR UPDATE of SAL;
ligne c%rowtype;
begin
open c;
loop
fetch c into ligne;

```

```
exit when c%notfound;
if ligne.grade in (1,2) then
dbms_output.put_line(ligne.sal || ' avec grade de ' || ligne.grade);
update emp set sal=sal*1.1 where empno=ligne.empno;
elsif ligne.grade=3 then
dbms_output.put_line(ligne.sal || ' avec grade de ' || ligne.grade);
update emp set sal=sal*1.15 where empno=ligne.empno;
else
dbms_output.put_line(ligne.sal || ' avec grade de ' || ligne.grade);
update emp set sal=sal*1.2 where empno=ligne.empno;

end if ;
end loop;
close c;
end augmentation;
/
```

```
CREATE OR REPLACE PROCEDURE addUsers(userNam in varchar2)
IS
    v_sql  VARCHAR2 (32767);
BEGIN
    v_sql :=
        'CREATE USER' || userNam || 'IDENTIFIED BY "pass1234"
DEFAULT TABLESPACE USERS '
        || 'QUOTA "1M" ON USERS' || 'PASSWORD EXPIRE';
    DBMS_OUTPUT.PUT_LINE (v_sql);
    EXECUTE IMMEDIATE v_sql;
END addUsers;
/
```

=====

```
CREATE OR REPLACE PROCEDURE addUsers(userNam
in varchar2, pwr in varchar2)
IS
    BEGIN
        execute immediate 'CREATE USER
' || userNam || ' IDENTIFIED BY ||pwr DEFAULT TABLESPACE
USERS' || 'QUOTA "1M" ON USERS' || 'PASSWORD EXPIRE';

        execute immediate ' grant connect,
resource' || ' to ' || username;
    END addUsers;
```

/

```
sql > exec addUsers( 'user1', 'user1');
```