Direction de la formation continue

Session : Été 2020

Nom :	**Prénom :**
MEGHAR	**KARIM**
HAMDAOUI	**MENADE**

Groupe : 00413

420-A34-RO - Approfondissement des bases de données

Travail Pratique N° 1

Enseignant : Elena Smagina

Exercice 1 : Un programme PL/SQL non stocké (deleteDept.sql) qui supprime tous les départements qui n'ont aucun employé.

```
/*
Jeu de tests
Cas 1 : chefDepartement is not null (Table Départements) : Département non supprimé
Cas 2 : chefDepartement is null (Table Départements) et
      numeroDepartement existe dans (Table Employes) : Département non supprimé
Cas 3 : chefDepartement is null (Table Départements) et
      numeroDepartement n'existe pas dans (Table Employes) : Département supprimé
*/

set serveroutput on
declare
nbreDepartementSupprimes number(6);
begin
      delete from departements where chefdepartement is null and numerodepartement not in
      (select numerodepartement from employes where numerodepartement is not null);
       nbredepartementsupprimes:=SQL%ROWCOUNT;
   if (nbredepartementsupprimes=0) then
     dbms_output.put_line('Il n''y a pas de département sans employés');
  else
     dbms_output.put_line(nbredepartementsupprimes||' département(s) supprimé(s).');
  end if;
end;
/
```

Exercice 2 : Un programme PL/SQL non stocké (Affiche.sql) qui affiche les numéros, les noms, les dates d'embauche et les salaires de tous les employés d'un département dont le nom est entré au clavier. A la fin il doit afficher le nombre total des employés trouvés.

```
/*
Jeu de tests
Cas 1 : nom du département NULL - message d'erreur "Il faut saisir  un nom de département"
Cas 2 : nom du département incorrect - message d'erreur "Le département n'existe pas"
Cas 3 : nom du département correct trouvé - affichage des employés et le nombre total
Cas 4 : Autres cas - exception
*/
set serveroutput on
set verify off
accept nomDeptEntree prompt 'Veuillez entrer le nom du département : '
declare
nbreTotalEmployes employes.nomemploye%type :=0; -- nombre total des employés trouvés
nomDept departements.nomdepartement%type;
Cursor cur_EmplDept is
(select numeroEmploye,nomEmploye,dateEmbaucheemploye,
    salaireEmploye  from employes where numerodepartement=
```

```
   (select numerodepartement from departements where nomDepartement=nomDept));
begin
nomDept:='&nomDeptEntree';
if (nomDept is null) then
     dbms_output.put_line('Il faut saisir  un nom de département !');
else
      for EmplDept in cur_EmplDept loop
           nbreTotalEmployes:=nbretotalemployes+1;
      end loop;
   if (nbreTotalEmployes=0) then
    dbms_output.put_line('Le département "'||nomDept ||'" soit il n''existe pas, soit il n''a aucun employé');
   else
        DBMS_OUTPUT.PUT_LINE('Informations sur les salariés du département '||nomdept||'
                :'||CHR(10)||
                  '============================================================');
       for EmplDept in cur_EmplDept loop
                DBMS_OUTPUT.PUT_LINE('Employé numéro :
                '||EmplDept.numeroemploye||CHR(10)||CHR(09)||'Nom Employé : '||
                EmplDept.nomemploye||CHR(10)||CHR(09)||'Date d''embauche :
                '||EmplDept.dateEmbaucheemploye
                 ||CHR(10)||CHR(09)||'Salaire : '||to_char(EmplDept.salaireemploye,'999,999.99')||' $');
        end loop;
        dbms_output.put_line('=========================================');
        dbms_output.put_line('Le nombre total d''employés trouvés est : '||nbretotalemployes);
    end if;
end if;
Exception
when others then DBMS_OUTPUT.PUT_LINE('Erreur inconnue !');
end;
/
```

Exercice 3 : Une procédure PL/SQL stockée AugmenteSalaire (AugmenteSalaire_Proc.sql) qui augmente les salaires de tous les employés d'un département donné en paramètre à l'exception du chef à un taux donné en paramètre, si le total des tous les employés ne dépasse 150 000 et la tester dans un bloc anonyme (fichier Proc.sql).

- Procedure « AugmenteSalaire » (AugmenteSalaire_Proc.sql) :

```
Create or replace procedure AugmenteSalaire (numDeptInput in departements.nomdepartement%type,
                                tauxAugmentation in  number,nbreModif out number) is
begin
update employes set salaireemploye=salaireemploye*(1+tauxAugmentation/100)
where numerodepartement=numDeptInput and  (numeroemploye <>
 (select chefdepartement  from departements where numerodepartement=numDeptInput))
and (select sum(salaireemploye) from employes  where numerodepartement=numDeptInput) <=150000;
nbreModif:=SQL%ROWCOUNT;
exception
when others then nbreModif:=-1;
end;//
```

- Bloc anonyme Fichier test Procédure (Proc.sql) :

```
/*
Jeu de tests
Cas 1 : numéro département is NULL, message d'erreur "Le numéro de département est obligatoire !"
Cas 2 : taux d'augmentation is NULL, message d'erreur "Il faut saisir le taux d''augmentations"
Cas 3 : taux d'augmentation <=0,  message d'erreur "Le taux d''augmentation doit être > 0 !"
Cas 4 : numéro département incorrect, message d'erreur
Cas 5 : département qui contient seulement un chef - pas d'augmentation
Cas 6 : Montant Global > 150 000, pas d'augmentation
Cas 7 : variable qui retourne nombre de modifications dans la procédure retourne -1, message d'erreur
        "Erreur   inconnue  lors de l'exécution de la procédure"
Cas 8 : numéro département correct et taux d'augmentation correct et condition d'augmentation
          Vérifiées, message précisant le nombre de salariés augmentés.
Cas 9 : Exception, pour les autres situations avec un message "Erreur inconnue"
*/

set serveroutput on
set verify off
accept numDeptInput prompt 'Entrez le numéro du département : '
accept tauxAugmentation prompt 'Entrez le taux d''augmentation un chiffre en 0% et 100% :'
declare
numDept departements.numerodepartement%type;
tauxAugmt number(5,2);
nbreModif number(9);
begin
numDept:='&numDeptInput';
if (numDept is null) then
   dbms_output.put_line('Le numéro de département est obligatoire ! ');
else
     tauxAugmt:='&tauxAugmentation';
    if (tauxAugmt is null) then
          dbms_output.put_line('Il faut saisir le taux d''augmentation !');
   else if (tauxAugmt<=0) then
          dbms_output.put_line('Le taux d''augmentation doit être > 0 !');
       else
           augmentesalaire(numDept,tauxAugmt,nbreModif);
          if (nbreModif=-1) then
              dbms_output.put_line('Erreur inconnue qui a empêcher d''effectuer l''augmentation !');
         else if (nbreModif=0) then
                dbms_output.put_line('Le département n° "'||numDept||'" soit : '||CHR(10)||CHR(9)||
                '1) il n''existe pas,'||CHR(10)||CHR(9)||'2) soit il n''a aucun employé ou seulement le chef de
                département,'
                ||CHR(10)||CHR(9)||'2) soit le montant global des salaires des employés de ce département
                dépasse 150 000.');
             else
               dbms_output.put_line('Augmentation de salaire de '||nbremodif||' salarié(s)');
            end if;
```

```
        end if;
      end if;
    end if;
end if;
Exception
when others then dbms_output.put_line('Erreur inconnue !');
end;
/
```

Exercice 4 : Une fonction PL/SQL stockée MaxTotalSalaires (MaxTotalSalaires_Fonc.sql) qui retourne la valeur maximale des totaux des totaux des salaires des anciens employés par département les dates limites pour définir un ancien employé sont données en paramètres et la tester dans un bloc anonyme (fichier Fonc.sql).

- Fonction « MaxTotalSalaires » (MaxTotalSalaires_Fonc.sql) :

```
create or replace function MaxTotalSalaires (dateLimiteAnciennte in date)      return number is
MaxTotSal number (12,2);
begin
      select sum(salaireemploye) into MaxTotSal
      from employes
      where  dateembaucheemploye < dateLimiteAnciennte
      group by numerodepartement
      having sum(salaireemploye) >= all
         (select  sum(salaireemploye) from employes
         where  dateembaucheemploye <= dateLimiteAnciennte group by numerodepartement );
  return MaxTotSal;
Exception
when others then return -1;
end;
/
```

- Bloc anonyme Fichier test fonction (Fonc.sql) :

```
/*
Jeu de tests
Cas 1 : date limite d'ancienneté is NULL, message d'erreur "La date limite est obligatoire !"
Cas 2 : si date limite d'ancienneté < min dateEmbauche ou > date system, message d'erreur
    "La date limite doit être entre date min dateEmbauche et la date system au moment de l'exécution "
Cas 3 : Si la valeur du retour de la fonction est différent de  -1, affichage du résultat des totaux des
      salaires des ancien employés par rapport a la date donné en paramètres.
Cas 4 : Si la valeur du retour de la fonction est égale à -1, message d'erreur "Une erreur inconnue est
        survenue lors de l'exécution de la fonction"
Cas 5 : Exception,  pour les autres situations avec un message "Erreur inconnue"

*/
```

```
set serveroutput on
set verify off
accept dateLimiteAnciennte prompt 'Entrez la date limite d''ancienneté :'
declare
dateLimiteAncInput employes.dateembaucheemploye%type;
minDateEmbauche  employes.dateembaucheemploye%type;
begin
dateLimiteAncInput:='&dateLimiteAnciennte';
if   (dateLimiteAncInput is null) then
     dbms_output.put_line('La date limite est obligatoire pour déterminer l''ancienneté d''un employé !');
else
    select min(dateembaucheemploye)  into minDateEmbauche from employes ;
    if (dateLimiteAncInput<minDateEmbauche or dateLimiteAncInput> sysdate) then
      dbms_output.put_line ('La date limite doit être entre '||minDateEmbauche||' et '||sysdate);
    else
        if (MaxTotalSalaires(dateLimiteAncInput) <> -1)  then
          dbms_output.put_line('La valeur maximale des totaux des salaires des anciens employés est : '||
          MaxTotalSalaires(dateLimiteAncInput));
        else
         dbms_output.put_line ('Une erreur inconnue est survenue à l''execution de la fonction');
        end if;
    end if;
end if;
Exception
when others then dbms_output.put_line ('Erreur inconue !');
end;
/
```

Exercice 5 : Un déclencheur DateEmbauche de base de données qui vérifie si la date d'embauche d'un employé est ultérieur à la date de création du département dans lequel l'employé est embauché (au moment d'insertion ou de modification).

- Déclencheur « DateEmbauche » (DateEmbauche.sql) :

```
create or replace trigger DateEmbauche
before insert or update of dateEmbaucheEmploye on Employes
for each row
declare
dateEmbaucheValide date;
numeroDepartementValide  employes.numerodepartement%type;
dateCreatDepartConcerne  date;
begin
numeroDepartementValide:=:new.numeroDepartement;
if  numeroDepartementValide is not null then
     dateEmbaucheValide:=:new.dateEmbaucheEmploye;
   if dateEmbaucheValide is  not null  then
     select dateCreationDepartement into dateCreatDepartConcerne from departements
     where numeroDepartement=numeroDepartementValide;
      if dateCreatDepartConcerne > dateEmbaucheValide then
```

```
        RAISE_APPLICATION_ERROR(-20100, 'Transaction refusée, la date d''embauche doit être > date de création
du département!');
      end if;
   end if;
end if;
end;
/

/*
Jeu de Tests
Cas 1 : update employes set dateEmbaucheEmploye='01/01/2000' where numeroEmploye=1;
:new.numeroDepartement = :old.numeroDepartement =NULL, 1 ligne mis à jour.
(Condition satisfaite car l'employe n'appartient à aucun département)

Cas 2 : update employes set dateEmbaucheEmploye='01/01/1990' where numeroEmploye=2;
:new.numeroDepartement = :old.numeroDepartement =DP0001, :new.dateEmbaucheEmploye='01/01/1990',
message d'erreur ''Transaction refusée, la date d''embauche doit être > date de création du département''
(condition non satisfaite, car la date de création du département DEP001 est le 12/09/1992 )

Cas 3 : update employes set dateEmbaucheEmploye='01/01/1998' where numeroEmploye=2;
:new.numeroDepartement = :old.numeroDepartement =DP0001, 1 ligne mis à jour.
(condition satisfaite car la date de création du département DEP001 est le 12/09/1992 )

Cas 4 : INSERT INTO Employes VALUES (18, 'Heisenberg', 'Lisa', '8764, Rue du collège',
'Longueuil', '', 'H1A2C6', 5146325943, 'Assistant', 25127.64,
NULL, NULL,1, '01/01/1990', '26/02/1968');
:new.numeroDepartement=null, 1 ligne mis à jour.
(Condition satisfaite car l'employe n'appartient à aucun département)

Cas 5 : INSERT INTO Employes VALUES (18, 'Heisenberg', 'Lisa', '8764, Rue du collège',
'Longueuil', '', 'H1A2C6', 5146325943, 'Assistant', 25127.64,
NULL, 'DEP002',1, '01/01/1990', '26/02/1968');
:new.numeroDepartement=DEP002, message d'erreur ''Transaction refusée,
la date d''embauche doit être > date de création du département''
(condition non satisfaite, car la date de création du département DEP002 est le 13/03/1993 )

Cas 6 : INSERT INTO Employes VALUES (18, 'Heisenberg', 'Lisa', '8764, Rue du collège',
'Longueuil', '', 'H1A2C6', 5146325943, 'Assistant', 25127.64,
NULL, 'DEP002',1, '15/03/1993', '26/02/1968');
:new.numeroDepartement=DEP002, 1 ligne mis à jour.
(condition satisfaite car la date de création du département DEP002 est le 13/03/1993 )

Cas 7 : INSERT INTO Employes VALUES (19, 'Heisenberg', 'Lisa', '8764, Rue du collège',
'Longueuil', '', 'H1A2C6', 5146325943, 'Assistant', 25127.64,
NULL, NULL,1, null, '26/02/1968');
:new.numeroDepartement=NULL,:new.dateEmbaucheEmploye=NULL, 1 ligne mis à jour.
(Car dans la table employé le numéro de département et la date d'embauche peuvent être nuls).
*/
```