1 PL (ProceduraL) /SQL

1.1 Préliminaires

1.1.1 Tables

On considère ici les tables suivantes :

- ETUDIANT=(ID_ETUDIANT, NOM_ETUDIANT, TUTEUR) qui, pour un étudiant identifié de manière unique par ID_ETUDIANT donne, quand il est désigné, le nom de l'enseignant qui est son tuteur de stage.
- ENS=(NOM_ENS, NB_ETU) qui donne pour chaque enseignant le nombre d'étudiants qu'il encadre en stage.

1. Créez ces tables, avec les contraintes d'intégrité qui vous paraissent appropriées, ainsi qu'une séquence qui sera utilisée pour générer des identifiants pour les étudiants.

Correction :

```
drop table ETUDIANT;
drop table ENS;
drop sequence seq_etu;

create table ENS( NOM_ENS VARCHAR2(50) PRIMARY KEY,
  NB_ETU NUMBER(2) default 0 check (nb_etu >=0));

create table ETUDIANT( ID_ETUDIANT integer PRIMARY KEY,
                       NOM_ETUDIANT VARCHAR2(50),
                       TUTEUR varchar2(50) REFERENCES ENS(NOM_ENS));

create sequence seq_etu;
```

□

1.1.2 PL/SQL et SQL*Plus

Dans SQL*Plus, vous devez commencer par executer la commande suivante :

`SET serveroutput ON`

Ceci vous permettra d'afficher des informations à l'écran depuis vos programmes PL/SQL, en utilisant :

`DBMS_OUTPUT.PUT_LINE`

1. Ecrivez un bloc PL/SQL qui *declare* une variable x, lui *affecte* la valeur '**Hello World!**' et *affiche* son contenu à l'écran.
   Correction :

```
set serveroutput on

declare
  x varchar2(50);
begin
  x:= 'Hello World';
  dbms_output.put_line(x);
end;
/
```

□

2. Executez-le dans SQL*Plus. Comment le réexecuter sans avoir à le redéfinir ?
   Correction :

```
-- Pour ré-exécuter:
-- terminer par un point et taper run
run
```

□

3. En cas d'erreur, utilisez l'ordre SQL*Plus : SHOW ERRORS.
   Correction :

```
-- S'il y a des erreurs:
show errors
```

□

1.2 Blocs, fonctions et procédures stockées.

Nous testons ici la possibilité offerte par le language PL/SQL de définir des fonctions et des procédures stockées et de les appeler dans des blocs PL/SQL.

1. Ecrivez une *procédure stockée* PL/SQL **Nouvel_Etudiant** prenant comme paramètre le nom d'un étudiant et l'insérant dans la table ETUDIANT, mais sans lui attribuer de tuteur. Quel est le type le plus adapté pour la déclaration du paramètre ? Testez cette procédure dans un bloc PL/SQL.
   Correction :

```
create or replace procedure Nouvel_Etudiant(nom_etu in etudiant.nom_etudiant%type) as
  begin
    insert into etudiant values(seq_etu.nextval, nom_etu, null);
  end;
/

-- test:
begin
  nouvel_etudiant('Omar');
end;
/
```

□

2. Ecrivez et testez de même une *fonction stockée* PL/SQL **Nb_Etudiants** prenant comme paramètre un nom d'enseignant et calculant à partir des données de la relation ETUDIANT le nombre d'étudiants qu'il encadre.
   Correction :

```
create or replace function Nb_Etudiants(nom_ens in ens.nom_ens%type) RETURN number as
     nb_etu number(2);
  begin
    select count(*) into nb_etu
    from etudiant
    where tuteur = nom_ens;
    return nb_etu;
  end;
/
```

□

3. Manuellement, insérez des enseignants et affectez-leur des étudiants. Testez la fonction Nb_Etudiants dans un bloc PL/SQL, en affichant ses résultats à l'écran.
   Correction :

```
insert into ens values('Abiteboul', null);
insert into ens values('Vianu', null);
insert into etudiant values(seq_etu.nextval, 'Omar', 'Abiteboul');
insert into etudiant values(seq_etu.nextval, 'Manu', 'Abiteboul');


begin
  nouvel_etudiant('Benjie');
  dbms_output.put_line('Serge a ' || nb_etudiants('Abiteboul') || ' étudiants.');
end;
/
```

□

4. Déconnectez-vous de la base de données, reconnectez-vous et renouvelez les tests des questions précédentes. Qu'en concluez-vous ?
   Correction :

□

5. Vous pouvez consulter à partir de maintenant les tables systèmes concernées :

```
USER_SOURCE(NAME, TYPE, LINE, TEXT)

USER_ERRORS
```

Correction :

□

1.3 Triggers et contraintes

Nous allons voir maintenant comment utiliser les triggers pour s'assurer que des contraintes complexes restent vérifiées par la base. C'est d'ailleurs l'une des utilisation principales des triggers. Ici, on veut s'assurer que le nombre d'étudiants encadrés par un tuteur de la table ENS est cohérent avec les données de la table ETUDIANT.

1. Créez un trigger qui, lors de l'enregistrement du tuteur d'un étudiant (jusqu'alors sans tuteur) dans la table ETUDIANT, met à jour automatiquement dans la table ENS le nombre d'étudiants que cet enseignant encadre en stage.

   Correction :

```
create or replace trigger t1
  after insert of TUTEUR on ETUDIANT
for each row
  begin
    update ens
     set nb_etu = nb_etu+1
     where nom_ens = :new.tuteur;
  end;
```

□

2. Créez plus généralement un trigger qui, lors de toute insertion / modification / suppression d'affectation de tuteur dans la table ETUDIANT, met à jour l'attribut NB_ETU dans la table ENS.

   Correction :

```
create or replace trigger t1
  after insert or delete or update of TUTEUR on ETUDIANT
for each row
  begin
    -- on rajoute un étudiant pour le nouveau tuteur
    if (:new.tuteur is not null) then
      update ens
      set nb_etu = nb_etu+1
      where nom_ens = :new.tuteur;
    end if;
    -- on en enlève un pour l'ancien
    if (:old.tuteur is not null) then
      update ens
      set nb_etu = nb_etu-1
      where nom_ens = :old.tuteur;
    end if;
  end;
/
```

□

3. Testez tous les cas de figure possibles et assurez-vous que votre trigger remplit bien sa fonction.

   Correction :

```
--test:
-- On initialise Abiteboul et Vianu
update ens set nb_etu = 0 where nom_ens = 'Vianu';
update ens set nb_etu= 2 where nom_ens='Abiteboul';
-- puis on lui rajoute un étudiant
update etudiant set tuteur='Abiteboul' where nom_etudiant='Benjie';
-- on peut même changer le tuteur de Benjie
update etudiant set tuteur='Vianu' where nom_etudiant='Benjie';
-- et supprimer Omar
delete from etudiant where nom_etudiant='Omar';

--A chaque fois, on vérifie que la table ens suit bien...
```

□

1.4 Curseurs

Les curseurs permettent de gérer des opérations (sélections, mises à jour) sur des ensembles de n-uplets.

1. Créer une procédure stockée PL/SQL qui affiche à l'écran la liste des étudiants encadrés par un tuteur donné.

   Correction :

```
create or replace procedure Liste_Etudiants(nom_tut etudiant.tuteur%type) as

  nom_etu etudiant.nom_etudiant%type;
  -- le curseur utilise un paramètre
  CURSOR tampon_etu(the_tuteur etudiant.tuteur%type) is
select nom_etudiant
from etudiant
where tuteur = the_tuteur;

begin
  -- paramètre passé à l'ouverture du curseur
  OPEN tampon_etu(nom_tut);
  DBMS_OUTPUT.PUT_LINE('Les étudiants encadrés par '||nom_tut||' sont :');
  LOOP
    FETCH tampon_etu INTO nom_etu;

    EXIT WHEN tampon_etu%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE(nom_etu);

  END LOOP;
end;
/

execute liste_etudiants('Abiteboul');
execute liste_etudiants('Vianu');
```

□

2. Créer une procédure stockée PL/SQL de "consolidation", dont le rôle est de s'assurer que chaque tuteur a un nombre d'étudiants cohérent avec la table ETUDIANT.

Correction :

```
create or replace procedure consolidation as

  -- on peut utiliser des vaibles de n-uplet:
  the_ens ens%rowtype;
  the_nb ens.nb_etu%type;

  -- un tampon pour miseăà jour
  CURSOR tampon_ens is
 select * from ens
  FOR UPDATE;

begin
  OPEN tampon_ens;
  LOOP
     FETCH tampon_ens into the_ens;

     EXIT WHEN tampon_ens%NOTFOUND;

     select count(*) into the_nb
     from etudiant
     where tuteur = the_ens.nom_ens;

     -- mise à jour du n-uplet en cours du tampon
     update ens
set nb_etu = the_nb
      where CURRENT OF tampon_ens;
  END LOOP;
end;
/

-- test
execute consolidation;
```

□

1.5 Packages

Les packages permettent de regrouper des fonctions et procédures apparentées dans une seule unité et de contrôler leur visibilité. Par exemple DBMS_OUTPUT est un package.

1. Vous allez créer un package permettant de gérer les étudiants et leurs tuteurs, en proposant les opérations suivantes :
   (a) Ajouter un nouvel enseignant/un nouvel étudiant (Vous savez déjà faire). Gérez les erreurs qui peuvent survenir.
   (b) Affecter un enseignant sans stagiaire comme tuteur d'un étudiant sans tuteur.
   (c) Donner le tuteur d'un étudiant dont on donne le nom ou le numéro.
   (d) Incrémenter ou décrémenter l'attribut NB_ETU d'un enseignant

Correction :

```
create or replace package etu_ens as

procedure Nouvel_Etudiant(nom_etu in etudiant.nom_etudiant%type) ;
procedure Nouvel_Enseignant(nom_ens in ens.nom_ens%type);
procedure Affecter(nom_ens in ens.nom_ens%type, nom_etu in etudiant.nom_etudiant%type);
function tuteur(nom_etu in etudiant.nom_etudiant%type) return ens.nom_ens%type;
function tuteur(id_etu in etudiant.id_etudiant%type) return ens.nom_ens%type;
procedure dec_etu(nom ens.nom_ens%type);
procedure inc_etu(nom ens.nom_ens%type);

end;
/

create or replace package body etu_ens as

procedure Nouvel_Etudiant(nom_etu in etudiant.nom_etudiant%type)
as
  begin
    insert into etudiant values(seq_etu.nextval, nom_etu, null);
  end;


procedure Nouvel_Enseignant(nom_ens in ens.nom_ens%type)
as
  existe_deja exception;
  pragma exception_init(existe_deja, -1);
  begin
    insert into ens values(nom_ens, null);
  exception
    when existe_deja then
    dbms_output.put_line('L''enseignant '||  nom_ens || ' existe deja');
  end;


-- Il se peut que l'enseignat n'existe pas, ou l'étudiant
--  On peut vouloir les insérer si c'est le cas
procedure Affecter(nom_ens in ens.nom_ens%type, nom_etu in etudiant.nom_etudiant%type)
as
 begin
   update etudiant
   set tuteur = nom_ens
   where nom_etudiant = nom_etu;
 end;

function tuteur(nom_etu in etudiant.nom_etudiant%type) return ens.nom_ens%type
as
  le_nom ens.nom_ens%type;
begin
  select tuteur into le_nom
  from etudiant
  where nom_etudiant = nom_etu;
  return le_nom;
end;


function tuteur(id_etu in etudiant.id_etudiant%type) return ens.nom_ens%type
as
  le_nom ens.nom_ens%type;
begin
  select tuteur into le_nom
  from etudiant
  where id_etudiant = id_etu;
  return le_nom;
end;


-- On pourra gérer les erreurs dues au check >=0  ici
procedure dec_etu(nom ens.nom_ens%type)
as
begin
  update ens
  set nb_etu = nb_etu - 1
  where nom_ens = nom;
end;

procedure inc_etu(nom ens.nom_ens%type)
as
begin
  update ens
  set nb_etu = nb_etu + 1
  where nom_ens = nom;
end;

end;
/
```

□

2. Modifier le trigger de la question 2 en réutilisant les opérations de votre package.

Correction :

□