TP1 : correction

3.2)Manipulation des curseurs:

a) Créer la procédure stockée « upd_adresse » pour mettre à jour l'adresse d'un abonné donné. Passer en paramètre le code et la nouvelle adresse de l'abonné.
Prévoir le traitement des exceptions requis si aucune mise à jour n'est effectuée.

```
Procedure upd_adresse (code_ab number(4), adresse_ab varchar(50)) is
Begin
Update abonne set adresse=adresse_ab where matricule =code_ab ;
Exception
When NO_DATA_FOUND then dbms_output.put_line('matricule non trouve');
End ;
```

b) 0n veut afficher les trois auteurs qui ont écrit le maximum de titres ainsi que le nombre de titres de chacun. Ecrire le traitement SQL qui permet de le faire en utilisant deux solutions :

1) Par utilisation des curseurs.

```
Declare
Curseur c is select cod_auteur,nom_auteur, count(cod_livre)  from auteur,ecrit where auteur.cod_auteur=ecrit.cod_livre group by cod_auteur order by count(cod_livre) desc ;
Code auteur.cod_auteur%TYPE;
Nom auteur.nom_auteur%TYPE;
nombre number ;
begin
open c ;
fetch c into code,nom,nombre ;
for i in 1..3 loop
Dbms_output.put_line(code,nom,nombre);
end loop
close c ;
end ;
```

2) Par utilisation des vues en ligne

```
Create view max_titre as
select cod_auteur,nom_auteur, count(cod_livre)  from auteur,ecrit where auteur.cod_auteur=ecrit.cod_livre and rownum<=3 group by cod_auteur order by count(cod_livre) desc ;
```

c) Ecrire un bloc PL/SQL qui affiche la valeur moyenne des cotisations des étudiants. Extraire les cotisations des enseignants dans un curseur.

```
Declare
Curseur c is select avg(cotisation)  from abonne where categorie=’etudiant’ ;
Curseur e is select cotisation rom abonne where categorie=’enseignant’ ;
nombre number ;
begin
open c ;
fetch c into code,nom,nombre ;
Dbms_output.put_line(nombre);
close c ;
open e ;
While e%FOUND loop
Fetch e into nombre ;
Dbms_output.put_line(nombre);
End loop ;
Close e ;
end ;
```

3.3) Manipulation des triggers :

a) Créer un trigger sur ligne qui permet d’auditer les mises à jour sur la table livre. Pour cela créer une table « audit_livre » qui va contenir pour chaque modification d’une ligne de la table livre les informations suivantes : le nom de l’utilisateur, la date exacte de la maj, les anciennes valeurs du titre, thème et nombre de pages ainsi que les nouvelles valeurs.

```
Create trigger maj_livre
After UPDATE ON livre ;
FOR EACH ROW
Begin
Insert into audit_livre values(user,sysdate, :old.titre, :old.theme, :old.nbr_page) ;
End ;
```

b) Définir une contrainte qui stipule que la cotisation d’un abonné ne peut jamais augmenter. Faire appel à la procédure RAISE_APPLICATION_ERROR.

```
Create trigger maj_abonne
before UPDATE ON abonne ;
FOR EACH ROW
Begin
If :new.cotisation> :old.cotisation then
raise_application_error(-20101, ' la cotisation ne peut jamais augmenter ');
```

end if ;

End ;

c) Créer un trigger qui permet de préserver l'intégrité référentielle dans la table Ecrit. Le trigger devrait s'exécuter après chaque modification du code d'un auteur dans la table auteur (cod_auteur étant la clé primaire de la table auteur) pour modifier dans la table Ecrit le cod_auteur de tous les livres écrits par cet auteur.
(ce trigger simule en fait la clause on update de la contrainte d'intégrité référentielle)

```
Create trigger maj_auteur
After UPDATE ON auteur ;
FOR EACH ROW
Begin
Update ecrit set cod_auteur= :new.cod_auteur where cod_auteur= :old.cod_auteur ;
End ;
```

4)Manipulation des vues :

1) créer les vues suivantes :
   a) La vue ROMANS (cod_roman, titre, theme) des livres qui sont des romans.
      Create view ROMANS (cod_roman,titre,theme) AS select cod_livre,titre,theme from livre where theme='roman' ;
   b) La vue NOMBRE _THEME qui donne pour chaque thème le nombre de titres de ce thème.
      Create view NOMBRE_THEME (theme,nbr_titre) AS select theme,count(titre) from livre where theme='roman' group by theme ;

2) Répondez à la requête suivante « trouver le nombre de romans » en utilisant :
   a) La vue ROMANS
      Select count(*) from ROMANS ;
   b) LA VUE NOMBRE_THEME
      Select sum(nbr_titre) from NOMBRE_THEME ;
3) Afficher les titres et les nombres de pages des livres qui ont les 4 plus grands nombres de pages en utilisant les requêtes de type n premiers.
4) Insérer la ligne (11, 'les réseaux',420, 'scientifique') dans la vue ROMANS
   Insert into ROMANS values(11, 'les réseaux',420, 'scientifique') ;
5) Afficher la vue et la table d'origine. Que constatez- vous ?
6) Rajouter à la vue ROMANS la clause « with check option ».
   Replace view ROMANS (cod_roman,titre,theme) AS select cod_livre,titre,theme from livre where theme='roman' ; with check option
7) Insérer la ligne (12,'sciences et vie','revue') dans la vue Romans.
   Insert into ROMANS values(12,'sciences et vie','revue') ;
8) Afficher la vue et la table d'origine. Que constatez- vous ?
9) Supprimer la dernière ligne rajoutée. Puis afficher la vue roman et la table d'origine. Que constatez-vous ?

10) En utilisant une vue en ligne écrire la requête qui permet d'afficher le titre et la catégorie des livres qui ont un nombre de pages > à la moyenne des nombres de pages des livres scientifiques. Est-il possible d'écrire une telle requête ? non on peut pas par ce que cette requete contient la clause group by