Exercices chap. 8

8.1 Créer le package suivant et son body

Exemples de création de la partie spécification

```
CREATE OR REPLACE       PACKAGE gestion_employes  IS
            PROCEDURE pnouveau_sal (empid IN NUMBER, taux IN NUMBER);
            FUNCTION pemp_info(empid IN number) RETURN emp%ROWTYPE;

END gestion_employes ;
/
CREATE OR REPLACE PACKAGE BODY gestion_employes IS
PROCEDURE pnouveau_sal (empid IN NUMBER, taux IN NUMBER) IS
NO_ROW_EXECPT EXCEPTION;
BEGIN
            UPDATE emp  SET sal = sal* (1 + taux)  WHERE empno = empid ;
    IF SQL%ROWCOUNT = 0 THEN
    RAISE  NO_ROW_EXECPT;
    END IF;
    EXCEPTION
                WHEN  NO_ROW_EXECPT THEN
                DBMS_OUTPUT.PUT_LINE('Employe inexistant');
                RAISE_APPLICATION_ERROR(-20012, 'Employe Nr. ' ||
            to_char(empid) || 'Inexistant');
END pnouveau_sal;

          FUNCTION pemp_info(empid IN number)
            RETURN emp%ROWTYPE AS
             emprow emp%rowtype ;
            BEGIN
                SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno
                 INTO emprow FROM emp
                WHERE empno = empid;
        RETURN (emprow);
        EXCEPTION
        WHEN NO_DATA_FOUND THEN
         DBMS_OUTPUT.PUT_LINE('Employe inexistant');
        return null;
    END pemp_info    ;
END gestion_employes;
/
```

Exercices chap. 8

8.2 Option procédurale : écrire les procédures stockées suivantes

/*
1. Exercice procédure stockée 1

Ecrire une procédure stockée PL/SQL qui permet
d'afficher le numéro, le nom, la date de naissance,
l'adresse, le salaire et le téléphone.

Elle reçoit en paramètre un numéro de Pilote.

Elle renvoie la structure d'un pilote.

Voici le prototype de la fonction :
- Function getPiloteById(piloteId
IN number) return pilote%rowtype ;

Si le pilote n'existe lever une erreur avec
le message suivant : Le Pilote nr. X est inexistant

Tester la procédure stockée avec
les numéros suivants : 1 puis 100

*/

Exercices chap. 8

8.2 Option procédurale : écrire les procédures stockées suivantes

/*

2. Exercice procédure stockée 2

Ecrire une procédure stockée PL/SQL qui permet
d'insérer un nouveau pilote dans la base.
La fonction reçoit en paramètre la structure
d'un pilote (pilote%rowtype) et ne renvoie rien.

Voici le prototype de la fonction :

- Procedure insertPilote(lignePilote
  IN PILOTE%ROWTYPE);

Vous devez effectuer un maximum de contrôle
avant d'insérer le pilote dans la base :

1) Le numéro, le nom, l'adresse et le salaire
   du pilote sont obligatoires. Pas de valeur nulle.
   Lever une exception en cas de violation de cette contrainte.
2) Le salaire doit toujours être inférieur à 70000. Lever une exception en cas de violation de cette erreur.
3) Lever une exception en cas de duplication du numéro ou du nom du pilote
4) Lever une exception pour toutes autres erreurs non identifiées

Tester le programme avec un pilote de nom 'Bill'.
Provoquer la levée de chacune des erreurs définies plus haut.
*/

Exercices chap. 8

8.2 Option procédurale : écrire les procédures stockées suivantes

/*
3. Exercice procédure stockée 3

Ecrire une procédure stockée PL/SQL qui permet de
modifier le salaire d'un pilote connaissant son numéro.

Voici le prototype de la fonction :
- Procedure updateSalairePilote(piloteId IN number,
  nouveauSal IN number);

Si le pilote n'existe pas, lever une erreur avec
le message : Le pilote nr. X n'existe pas.

Tester le programme avec les numéros suivants :
Pl#= 1 puis pl#= 200
*/

4. Exercice procédure stockée 4

Ecrire une procédure stockée qui permet de renvoyer
la référence vers un curseur contenant le numéro,
le nom, la date de naissance, l'adresse, le salaire
et le téléphone des pilotes habitant une adresse donnée.

Créer pour cela un package contenant un type REF CURSOR générique :

```
CREATE OR REPLACE PACKAGE Pk_refCursType AS
TYPE REFCURSTYPE IS REF CURSOR;
End;
/
```

Voici le prototype de la fonction :
- function getPiloteByAdr(adresse IN VARCHAR2)
  return Pk_refCursType.REFCURSTYPE ;

Tester le programme avec les villes de villes de : Paris puis Lilles.
S'il n'ya aucun pilote dans la dite ville, afficher un message d'erreur.
Pas de Pilote dans cette ville.

Exercices chap. 8

8.2 Option procédurale : écrire les procédures stockées suivantes

5. Exercice procédure stockée 5

Regrouper les fonctions et les procédures définies dans
les exercices de 1 à 4 dans un package que vous appellerez PK_PILOTE.

Tester les fonctions comme indiquées dans les exercices de 1 à 4.

Exercices chap. 8

8.3 Option procédurale : Utilisation de trigger pour l'audit ou d'historique

Ecrire un trigger qui mémorise dans une table d'historique HistPilote
les mises à effectuées sur la table PILOTE.

La table HistPilote a la même structure que la table PILOTE.

```
create table histPilote as  select * from pilote where pl#<0;
```

Ajouter dans cette table les TROIS colonnes
(username : nom utilisateur, majDate : date mise à jour, action: insert, update ou delete)
COMME suit :

```
ALTER TABLE histPilote Add  username varchar2(30) ;
ALTER TABLE histPilote Add  majDate date ;
ALTER TABLE histPilote
    Add  ACTION VARCHAR2(20)
    CONSTRAINT CHK_histpiloteaction
    check(action in ('INSERT', 'UPDATE', 'DELETE')) ;

CREATE or replace TRIGGER audit_PILOTE
AFTER INSERT OR DELETE OR UPDATE ON pilote FOR EACH ROW
    declare
    ACTION   VARCHAR2(20);
    BEGIN
    IF INSERTING THEN
        INSERT INTO histPilote VALUES
        (:NEW.pl#, :NEW.plnom, :NEW.dnaiss, :NEW.adr, :NEW.tel,
        :NEW.sal, :NEW.age, user, sysdate, 'INSERT');
    END IF;
    IF DELETING THEN
        INSERT INTO histPilote VALUES
        (:old.pl#, :old.plnom, :old.dnaiss, :old.adr, :old.tel,
        :old.sal, :old.age, user, sysdate, 'DELETE');
    END IF;
    IF UPDATING THEN
        INSERT INTO histPilote VALUES
        (:old.pl#, :old.plnom, :old.dnaiss, :old.adr, :old.tel,
        :old.sal, :old.age, user, sysdate, 'UPDATE');
    END IF;
end;
/
```

Effectuer des mises à jour dans la table PILOTE. Vérifier l'effet dans la table histPilote

```
insert into  pilote values(40, 'Louis', TO_date('04-11-1966','DD-MM-YYYY' ), 'Paris', NULL, 21000, 45);
```