Université des Sciences et de la Technologie Houari Boumediene

Architectures et Systèmes des Base de Données

Rapport de Travaux Pratiques N°4
PLSQL

Binôme
MOHAMMEDI Haroune
HOUACINE Naila Aziza

Professeur
Pr. BOUKHALFA Kamel

9 novembre 2017

Avant de commencer à répondre aux questions demandées, il faut exécuter la commande `SERVEROUTPUT ON`, elle permet d'activer les affichages résultants de la fonction `dbms_output.put_line();` .

1 Écriture d'un code PLSQL qui permet l'affichage pour chaque marque du nombre de modèles.

Dans ce cas la, nous avons a sélectionner plusieurs lignes (tuples), donc nous utiliserons un curseur afin de parcourir les éléments sélectionnés un par un et puis les traiter. Aussi si la table `MARQUE` est vide, une exception doit être généré, affichant un message correspondant à l'erreur rencontrée.

1.1 Requête

```
DECLARE
cursor cr is SELECT marque , COUNT(*) AS nbr FROM MARQUE MA, MODELE MO WHERE MA.
    ↪ NUMMARQUE = MO.NUMMARQUE GROUP BY MA.MARQUE ;
c cr%rowtype;
vide exception;

BEGIN
for c in cr LOOP
    dbms_output.put_line('La marque : '|| c.marque || ' posséde ' || c.nbr || ' modeles
        ↪ .' );
    exit when cr%notfound;
END LOOP;

EXCEPTION WHEN vide THEN
    dbms_output.put_line('Aucune marque n est trouvé ! ');

END;
/
```

Ainsi chaque ligne affichée en sortie donne le nom de la marque ainsi que le nombre de modèles qu'elle possède.

1.2 Résultat

```
La marque : maserati posséde 1 modeles .
La marque : cadilac posséde 1 modeles .
La marque : lotus posséde 1 modeles .
La marque : peugeot posséde 2 modeles .
La marque : saab posséde 1 modeles .
La marque : lamborghini posséde 1 modeles .
La marque : mercedes posséde 2 modeles .
La marque : rolls-royce posséde 1 modeles .
La marque : ferrari posséde 2 modeles .
La marque : porsche posséde 2 modeles .
La marque : chrysler posséde 1 modeles .
La marque : audi posséde 2 modeles .
La marque : renault posséde 1 modeles .
La marque : venturi posséde 2 modeles .
La marque : volvo posséde 1 modeles .
La marque : bmw posséde 1 modeles .
La marque : alfa-romeo posséde 2 modeles .
La marque : toyota posséde 1 modeles .
La marque : jaguar posséde 1 modeles .
La marque : lexus posséde 1 modeles .

PL/SQL procedure successfully completed.
```

2 Mise à jours et affichage pour chaque employé du nouveau salaire selon spécifications.

Ajout de la contrainte d'intégrité vérifiant que le salaire d'un employé doit être compris entre 10000 DA et 30000 DA.

2.1 Requête

```
ALTER TABLE EMPLOYE ADD CONSTRAINT LIMIT_SALAIRE CHECK(SALAIRE >= 10000 AND SALAIRE <=
    ↪ 30000)
```

2.2 Résultat

```
Table altered.
```

Désactivation de la contrainte précédemment crée afin d'effectuer les mises à jours demandées.

2.3 Requête

```
ALTER TABLE EMPLOYE DISABLE CONSTRAINT LIMIT_SALAIRE;
```

2.4 Résultat

```
Table altered.
```

Écriture de la procédure qui augmente le salaire de chaque employé selon les pourcentage données.

2.5 Requête

```
CREATE OR REPLACE PROCEDURE Augmentation_salaire
AS

cursor cr is select NUMEMPLOYE,NOMEMP,PRENOMEMP,CATEGORIE,SALAIRE FROM EMPLOYE ;
s INTEGER ;
c cr%rowtype ;
vide exception ;

BEGIN

for c in cr LOOP
    if (c.CATEGORIE = 'MECANICIEN')
    then
        s:= c.SALAIRE * 1.50;
    else
        s:= c.SALAIRE * 1.30;
    end if;

    dbms_output.put_line('L employe : ' || c.NOMEMP || ' ' || c.PRENOMEMP || ' de caté
       ↪ gorie : ' || c.CATEGORIE || ' passe d un salaire de: ' || c.SALAIRE || ' a '
       ↪ || s || '.');
    UPDATE EMPLOYE E SET SALAIRE = s WHERE E.NUMEMPLOYE = c.NUMEMPLOYE ;

    exit when cr%notfound;
END LOOP;

EXCEPTION WHEN vide THEN
    dbms_output.put_line('Aucun employé n est trouvé ! ');
```

```
END;
/
```

2.6 Résultat

```
Procedure created.
```

Nous pouvons vérifier s'il y a des erreurs dans notre procédure.

2.7 Requête

```
show errors procedure Augmentation_salaire
```

2.8 Résultat

```
No errors.
```

Maintenant que nous sommes sure qu'il n'y a pas eu d'erreurs lors de la création de la procédure, nous passons à son exécution.

2.9 Requête

```
EXECUTE Augmentation_salaire;
```

2.10 Résultat

```
L employe:bouzid lachemi de catégorie:MECANICIEN passe d un salaire de: 25000 a 37500.
L employe:elias bouchemla de catégorie:ASSISTANT passe d un salaire de: 10000 a 13000.
L employe:zouhir hadj de catégorie:ASSISTANT passe d un salaire de: 12000 a 15600.
L employe:hakim oussedik de catégorie:MECANICIEN passe d un salaire de: 20000 a 30000.
L employe:abdelhamid abad de catégorie:ASSISTANT passe d un salaire de: 13000 a 16900.
L employe:tayeb babaci de catégorie:MECANICIEN passe d un salaire de: 21300 a 31950.
L employe:mourad belhamidi de catégorie:MECANICIEN passe d un salaire de: 19500 a 29250.
L employe:redouane igoudjil de catégorie:ASSISTANT passe d un salaire de: 15000 a 19500.
L employe:bahim koula de catégorie:MECANICIEN passe d un salaire de: 23100 a 34650.
L employe:ahcene rahali de catégorie:MECANICIEN passe d un salaire de: 24000 a 36000.
L employe:ismail chaoui de catégorie:ASSISTANT passe d un salaire de: 13000 a 16900.
L employe:hatem badi de catégorie:ASSISTANT passe d un salaire de: 14000 a 18200.
L employe:mustapha mohammedi de catégorie:MECANICIEN passe d un salaire de: 24000 a
    ↪ 36000.
L employe:abdelaziz fekar de catégorie:ASSISTANT passe d un salaire de: 13500 a 17550.
L employe:wahid saidouni de catégorie:MECANICIEN passe d un salaire de: 25000 a 37500.
L employe:farid boularas de catégorie:ASSISTANT passe d un salaire de: 14000 a 18200.
L employe:nassim chaker de catégorie:MECANICIEN passe d un salaire de: 26000 a 39000.
L employe:yacine terki de catégorie:MECANICIEN passe d un salaire de: 23000 a 34500.
L employe:ahmed tebibel de catégorie:ASSISTANT passe d un salaire de: 17000 a 22100.
L employe:karim lardjoune de catégorie:passe d un salaire de: 25000 a 32500.

PL/SQL procedure successfully completed.
```

Ainsi nous constatons que la mise à jours des salaires à bel et bien été faite selon les spécifications initiales.

3 Écriture une procédure Vérification (période intervention) qui affiche « vérification positive » si la date début d'intervention est inférieur à la date de fin d'intervention, et affiche « Vérification négative » sinon.

Dans cette procédure on commence par la création d'un curseur sur les véhicules de l'année donnée en paramètre ayant subit une intervention, afin de les vérifier une par une.
Puis pour chaque intervention nous vérifiant si la date de début d'intervention est inférieur à la date de fin de l'intervention.

3.1 Requête

```
CREATE OR REPLACE PROCEDURE Verification(AN VEHICULE.ANNEE%type)
AS
cursor cr is select V.ANNEE,V.NUMVEHICULE,DATEDEBINTERV,DATEFININTERV
FROM INTERVENTIONS I,VEHICULE V
WHERE I.NUMVEHICULE = V.NUMVEHICULE AND V.ANNEE =AN;

c cr%rowtype;
vide exception;

BEGIN

for c in cr LOOP
    if(c.DATEDEBINTERV < c.DATEFININTERV)
    then
        dbms_output.put_line('vérification positive !');
    else
        dbms_output.put_line('vérification négative !');
    end if;

    exit when cr%notfound;

END LOOP;

EXCEPTION WHEN vide THEN
    dbms_output.put_line('Aucun vehicule trouvé !!');

END;
/
```

La procédure est crée sans erreur de compilation.

3.2 Résultat

```
Procedure created.
```

Elle est donc prête à être exécuté.

Exécution pour l'année 1998, via la commende `EXECUTE`.

3.3 Requête

```
EXECUTE Verification('1998');
PL/SQL procedure successfully completed.
```

3.4 Résultat

```
vérification positive !
vérification négative !
```

Vérification que la procédure fonctionne correctement (les éléments affichés sont correctes).

3.5 Requête

```
select V.ANNEE,V.NUMVEHICULE,DATEDEBINTERV,DATEFININTERV
FROM INTERVENTIONS I,VEHICULE V
WHERE I.NUMVEHICULE = V.NUMVEHICULE AND V.ANNEE ='1998';
```

3.6 Résultat

```
ANNE NUMVEHICULE DATEDEBI DATEFINI
---- ----------- -------- --------
1998          25 06/04/06 09/04/06
1998          22 04/03/06 22/02/06
```

Effectivement on remarque que seule deux (2) éléments sont sélectionnés, le premier étant conforme à la vérification des dates et le deuxième possède une date de début supérieure à la date de fin.

4 Écriture d'une fonction qui retourne, pour chaque employé donné, le nombre d'interventions effectuées.

Pour cela la fonction devra prendre en paramètre le numéro de l'employé c'est à dire un `NUMEMPLOYE` puis retourner / afficher une phrase contenant le nom, prénom et nombre d'intervention de l'employé en question.

4.1 Requête

```
CREATE OR REPLACE FUNCTION Nombre_interv(num in EMPLOYE.NUMEMPLOYE%type )
return number
IS

cursor cr is SELECT I.NUMEMPLOYE, E.NOMEMP , E.PRENOMEMP, COUNT(*) AS NBR
FROM INTERVENANT I, EMPLOYE E
WHERE E.NUMEMPLOYE = num
AND I.NUMEMPLOYE = E.NUMEMPLOYE
GROUP BY I.NUMEMPLOYE, E.NOMEMP , E.PRENOMEMP;

c cr%rowtype;
vide exception;
i binary_integer;

BEGIN

for c in cr LOOP
    dbms_output.put_line('L employé : '|| c.NOMEMP || ' ' || c.PRENOMEMP || ' a fait '
        ↪ || c.NBR || ' interventions. ');

    return(c.NBR);
END LOOP;

END;
/
```

4.2 Résultat

```
Function created.
```

Maintenant nous pouvons tester le fonction, prenant les numéros d'employé 55 , 60 , 59 , 67

4.3 Requête

```
SELECT Nombre_interv(55) from dual;
```

4.4 Résultat

```
NOMBRE_INTERV(55)
-----------------
                2

L employé : zouhir hadj a fait 2 interventions.
```

4.5 Requête

```
SELECT Nombre_interv(60) from dual;
```

4.6 Résultat

```
NOMBRE_INTERV(60)
-----------------
                3

L employé : redouane igoudjil a fait 3 interventions.
```

4.7 Requête

```
SELECT Nombre_interv(59) from dual;
```

4.8 Résultat

```
NOMBRE_INTERV(59)
-----------------
                4

L employé : mourad belhamidi a fait 4 interventions.
```

4.9 Requête

```
SELECT Nombre_interv(67) from dual;
```

4.10 Résultat

```
NOMBRE_INTERV(67)
-----------------
                1

L employé : wahid saidouni a fait 1 interventions.
```

5 Création d'une procédure qui permet l'ajout d'une intervention à partir de tous les attributs nécessaires.

Afin d'ajouter une intervention nous devrons d'abord vérifier l'unicité de la clé primaire, se qui signifie parcourir tous les tuples de la table INTEVENTIONS pour vérifier si le nombre d'élément ayant la même clé primaire est supérieur à 0 (donc interdire l'insertion) ;
Puis nous allons parcourir les éléments de la table VEHICULE pour vérifier que la clé étrangère prise en paramètre existe réellement, si le nombre d'élément portant la même clé étrangère est inférieur à 1 alors on interdira l'ajout de l'intervention.

5.1 Requête

```
CREATE OR REPLACE PROCEDURE Ajout_interv(num INTERVENTIONS.NUMINTERVENTION%type,numv
    ↪ INTERVENTIONS.NUMVEHICULE%type , t INTERVENTIONS.TYPEINTERVENTION%type ,dd
    ↪ INTERVENTIONS.DATEDEBINTERV%type , df INTERVENTIONS.DATEFININTERV%type ,cout
    ↪ INTERVENTIONS.COUTINTERV%type )
AS
cursor cr1 is SELECT NUMINTERVENTION,COUNT(*) AS N1 FROM INTERVENTIONS WHERE
    ↪ NUMINTERVENTION = num GROUP BY NUMINTERVENTION;
cursor cr2 is SELECT NUMVEHICULE,COUNT(*) AS N2 FROM VEHICULE  WHERE NUMVEHICULE = numv
    ↪ GROUP BY NUMVEHICULE;

c1 cr1%rowtype;
c2 cr2%rowtype;

i binary_integer;
j binary_integer;

vide1 exception;
vide2 exception;

BEGIN
i := 0;
j := 0;

for c1 in cr1 LOOP
    i := i+1;
END LOOP;

if(i = 0)
then

    for c2 in cr2 LOOP
        j := j+1;
    END LOOP;

    if(j > 0)
    then
        INSERT INTO INTERVENTIONS (NUMINTERVENTION ,NUMVEHICULE ,TYPEINTERVENTION ,
            ↪ DATEDEBINTERV ,DATEFININTERV ,COUTINTERV ) VALUES (num , numv , t , dd ,
            ↪ df , cout);
        dbms_output.put_line('Intervention ajouté ! ');
    else
        dbms_output.put_line('ERREUR : violation de la clé étrangere, le numéro du vé
            ↪ hicule saisie n existe pas dans la table VEHICULE ! ');
    end if;

else
    close cr1;
    dbms_output.put_line('ERREUR : violation de la clé primaire, une intervention de
        ↪ meme numéro existe déja ! ');
```

```
end if;

END;
/
```

5.2 Résultat

```
Procedure created.
```

Testons maintenant l'insertion d'une intervention avec une clé primaire déjà existante.

5.3 Requête

```
EXECUTE Ajout_interv(13,8,'réparation systeme',to_date('2006-05-12 14:00:00','YYYY-MM-DD
    ↪  HH24:MI:SS' ),to_date('2006-05-12 18:00:00','YYYY-MM-DD HH24:MI:SS' ),17846);
```

Une erreur décrivant le problème rencontré est affiché.

5.4 Résultat

```
ERREUR : violation de la clé primaire, une intervention de meme numéro existe déja !

PL/SQL procedure successfully completed.
```

Testons maintenant l'insertion d'une intervention avec une clé étrangère inexistante dans la table VEHICULE.

5.5 Requête

```
EXECUTE Ajout_interv(20,52,'réparation systeme',to_date('2006-05-12 14:00:00','YYYY-MM-
    ↪ DD HH24:MI:SS' ),to_date('2006-05-12 18:00:00','YYYY-MM-DD HH24:MI:SS' ),17846);
```

Une erreur décrivant le problème rencontré est affiché.

5.6 Résultat

```
ERREUR : violation de la clé étrangere, le numéro du véhicule saisie n existe pas dans
    ↪ la table VEHICULE !

PL/SQL procedure successfully completed.
```

Testons maintenant l'insertion d'une intervention avec une clé primaire unique et clé étrangère existante dans la table VEHICULE.

5.7 Requête

```
EXECUTE Ajout_interv(25,8,'réparation systeme',to_date('2006-05-12 14:00:00','YYYY-MM-DD
    ↪  HH24:MI:SS' ),to_date('2006-05-12 18:00:00','YYYY-MM-DD HH24:MI:SS' ),200000);
```

Exécution réussit et ajout réussit.

5.8 Résultat

```
Intervention ajouté !

PL/SQL procedure successfully completed.
```

Vérifiant que l'insertion a bien été exécuté à travers la fonction.

5.9 Requête

```
SELECT * FROM interventions WHERE NUMINTERVENTION = 25;
```

5.10 Résultat

```
NUMINTERVENTION NUMVEHICULE TYPEINTERVENTION              DATEDEBI   DATEFINI COUTINTERV
--------------- ----------- ----------------------------- ---------- -------- ----------
             25           8 réparation systeme            12/05/06   12/05/06     200000
```

Enfin le tuple a été inséré avec succé dans la table `INTERVENTIONS`.