Université des Sciences et de la Technologie Houari Boumediene

Architectures et Systèmes des Base de Données

Rapport de Travaux Pratiques N°5
TRIGGERS

Binôme
MOHAMMEDI Haroune
HOUACINE Naila Aziza

Professeur
Pr. BOUKHALFA Kamel

15 novembre 2017

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

1 Création d'un trigger qui affiche "un nouveau client est ajouté" après chaque insertion d'un client, Répétez la même chose pour la modification ou la suppression.

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

```
CREATE OR REPLACE TRIGGER INSERT_CLIENT
AFTER INSERT ON CLIENT
FOR EACH ROW
BEGIN
    dbms_output.put_line('un nouveau client est ajouté');
END;
/


CREATE OR REPLACE TRIGGER UPDATE_CLIENT
AFTER UPDATE ON CLIENT
FOR EACH ROW
BEGIN
    dbms_output.put_line('Les informations d un client sont mises a jours');
END;
/


CREATE OR REPLACE TRIGGER DELETE_CLIENT
AFTER DELETE ON CLIENT
FOR EACH ROW
BEGIN
    dbms_output.put_line('un client est supprimé');
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

```
Trigger created.

Trigger created.

Trigger created.
```

2 Tester les trigger

2.1 Requête

```
/** l'insertion **/
INSERT INTO CLIENT
VALUES(40,'MME','Adjaout','feriel',to_date('21/01/1995', 'DD/MM/YYYY'),'cite 2110 logts
    ↪ bt 5 alger','0561381877','0565568716','');

/** la modification **/
UPDATE CLIENT
```

```
SET PRENOMCLIENT = 'ADJAOUTE'
WHERE NUMCLIENT = 40 ;

/** la suppression **/
DELETE FROM CLIENT
WHERE NUMCLIENT = 40 ;
```

2.2 Résultat

```
un nouveau client est ajouté
1 row created.

Les informations d un client sont mises a jours
1 row updated.

un client est supprimé
1 row deleted.
```

3 Création du trigger qui afficher "un nouveau modèle est ajouté à la marque [Nom de la marque]" après chaque insertion d'un modèle.

- La clause `INTO` dans la requête `SELECT` nous permet de récupérer la valeur de l'attribut en question dans une variable
- Le tuple spécial `:NEW` générer par le SGBD dans le corps les triggers des insertions sauvegarde le tuple qui viens d'être insérer dans la base
- En utilisant les concepts `INTO` et `:NEW`, on écrit la requête qui récupère la marque du modèle en question

3.1 Requête

```
CREATE OR REPLACE TRIGGER NOUVEAU_MODELE
AFTER INSERT ON MODELE
FOR EACH ROW
DECLARE
NOM_MARQUE MARQUE.MARQUE%TYPE;
NUM MARQUE.NUMMARQUE%TYPE;

BEGIN
    NUM := :NEW.NUMMARQUE ;

    SELECT MARQUE INTO NOM_MARQUE
    FROM MARQUE WHERE NUMMARQUE = NUM;


    DBMS_OUTPUT.PUT_LINE('un nouveau modele est ajoute a la marque' || NOM_MARQUE);
END;
/
```

3.2 Résultat

```
Trigger created.
```

4 Tester les triggers

4.1 Requête

```
INSERT INTO MODELE VALUES (99,15,'504');
```

4.2 Résultat

```
un nouveau modele est ajoute a la marque peugeot
1 row inserted.
```

5 Création d'un trigger qui vérifie que lors de la modification du salaire d'un employé, la nouvelle valeur ne peut jamais être inférieure à la précédente.

— Le `BEFORE` permet de remettre en cause la mise a jours contenu dans le trigger
— Pour cela l'on doit arrêter la modification avant son exécution , donc utiliser le `BEFORE` puis l'on compare la nouvelle valeur du `SALAIRE` que l'on souhaite mettre (`NEW`) avec l'ancienne existante (`OLD`) et l'on génère une interruption (erreur) avec la fonction `raise_application_error` pour empêcher la modification dans le cas ou `NEW < OLD`

5.1 Requête

```
CREATE OR REPLACE TRIGGER UPDATE_SALAIRE
BEFORE UPDATE OF SALAIRE ON EMPLOYE
FOR EACH ROW
BEGIN
    IF :NEW.SALAIRE < :OLD.SALAIRE
    THEN raise_application_error(-20001, 'Le nouveau salaire ne doit pas etre inférieure
        ↪  a l'ancien!') ;
    END IF;
END;
/
```

5.2 Résultat

```
Trigger created.
```

6 Tester les triggers

Pour pouvoir s'assurer que le trigger `UPDATE_SALAIRE` est fonctionnelle on va insérer un tuple qui génère une exception

6.1 Requête

```
UPDATE EMPLOYE SET SALAIRE = SALAIRE - 100 WHERE NUMEMPLOYE = 53 ;
```

6.2 Résultat

```
ERROR at line 1:
ORA-20001: Le nouveau salaire ne doit pas etre inférieure a l'ancien!
ORA-06512: at "SYSTEM.UPDATE_SALAIRE", line 3
ORA-04088: error during execution of trigger 'SYSTEM.UPDATE_SALAIRE'
```

Le trigger a arrêté le programme et invalidé la mise a jours, Il a ainsi protégé l'intégrité de la base de données

7 Création du trigger pour vérifier la période des interventions

Pour pouvoir s'assurer que les périodes sont correctes il faut déclarer un tirgger qui va être exécuté a chaque insertion ou mise à jour des dates (`DATEDEBUT`,`DATEFIN`) Ce trigger a pour but de comparer les valeurs des dates en questions avec les valeurs de l'intervention correspondante et dans le cas où la période de l'intervention de l'employé n'est pas comprise entre la période de l'intervention correspondante une exception est levé a l'aide de la fonction `RAISE_APPLICATION_ERROR`

7.1 Requête

```
CREATE OR REPLACE TRIGGER CHECK_PERIODE_INTER
BEFORE INSERT OR UPDATE OF DATEDEBUT,DATEFIN ON INTERVENANT
FOR EACH ROW

DECLARE
INTER_ROW INTERVENTIONS%ROWTYPE;

BEGIN

IF(INSERTING) THEN
  SELECT * INTO INTER_ROW FROM INTERVENTIONS WHERE NUMINTERVENTION = :NEW.
     ↪ NUMINTERVENTION;
  IF((INTER_ROW.DATEDEBINTERV <= :NEW.DATEDEBUT) AND (INTER_ROW.DATEFININTERV >= :NEW.
     ↪ DATEFIN ))
    THEN
        DBMS_OUTPUT.PUT_LINE('PERIODE VALIDE POUR L EMPLOYE :'||:NEW.NUMEMPLOYE ||'
           ↪ POUR L INTERVENTION :'|| INTER_ROW.NUMINTERVENTION);

    ELSE RAISE_APPLICATION_ERROR(-20002,'PERIODE INVALIDE POUR L EMPLOYE INTERVENANT :'
       ↪ ||:NEW.NUMEMPLOYE ||'  POUR L INTERVENTION :'|| INTER_ROW.NUMINTERVENTION);
  END IF;

  ELSE

    SELECT * INTO INTER_ROW FROM INTERVENTIONS WHERE NUMINTERVENTION = :NEW.
       ↪ NUMINTERVENTION;
  IF((INTER_ROW.DATEDEBINTERV <= :NEW.DATEDEBUT) AND (INTER_ROW.DATEFININTERV >= :NEW.
     ↪ DATEFIN ))
    THEN
        DBMS_OUTPUT.PUT_LINE('PERIODE VALIDE POUR L EMPLOYE :'||:NEW.NUMEMPLOYE ||'
           ↪ POUR L INTERVENTION :'|| INTER_ROW.NUMINTERVENTION);

    ELSE RAISE_APPLICATION_ERROR(-20002,'PERIODE INVALIDE POUR L EMPLOYE INTERVENANT :'
       ↪ ||:NEW.NUMEMPLOYE ||'  POUR L INTERVENTION :'|| INTER_ROW.NUMINTERVENTION);
  END IF;


  END IF;

END;
/
```

7.2 Résultat

```
Trigger created.
```

8 Tester le trigger

Pour pouvoir s'assurer que le trigger CHECK_PERIODE_INTER est fonctionnelle. on insert deux intervenants avec des périodes différentes (une valide et l'autre invalide)

8.1 Cas Invalide

```
INSERT INTO INTERVENTIONS VALUES (60,8,'réparation systeme',to_date('2006-05-11 09:00:00
   ↪ ','YYYY-MM-DD HH24:MI:SS' ),to_date('2006-05-12 12:00:00','YYYY-MM-DD HH24:MI:SS'
   ↪  ),17846);
```

```
1 row created.

INSERT INTO INTERVENANT VALUES (60,54,to_date('2006-02-12 09:00:00','YYYY-MM-DD HH24:MI:
    ↪ SS' ),to_date('2006-05-12 12:00:00','YYYY-MM-DD HH24:MI:SS' ));

ERROR at line 1:
ORA-20002: PERIODE INVALIDE POUR L EMPLOYE INTERVENANT :54  POUR L INTERVENTION
:60
ORA-06512: at "SYSTEM.CHECK_PERIODE_INTER", line 12
ORA-04088: error during execution of trigger 'SYSTEM.CHECK_PERIODE_INTER'
```

8.2 Cas Valide

```
INSERT INTO INTERVENTIONS VALUES (59,8,'réparation systeme',to_date('2006-05-12 09:00:00
    ↪ ','YYYY-MM-DD HH24:MI:SS' ),to_date('2006-05-12 12:00:00','YYYY-MM-DD HH24:MI:SS'
    ↪  ),17846);

1 row created.

INSERT INTO INTERVENANT VALUES (59,54,to_date('2006-05-12 09:00:00','YYYY-MM-DD HH24:MI:
    ↪ SS' ),to_date('2006-05-12 12:00:00','YYYY-MM-DD HH24:MI:SS' ));

PERIODE VALIDE POUR L EMPLOYE :54  POUR L INTERVENTION :59

1 row created.
```

Effectivement après l'insertion d'une intervention avec une période qui ne correspond pas à l'intervention le trigger lance une exception.

9 Création du trigger TOTAL_INTERVENTIONS_TRIGGER Afin que l'administrateur puisse connaitre le nombre total des interventions pour chaque employé.

Pour cela, nous ajoutons l'attribut TOTAL_INTERVENTIONS dans la table employé. Puis nous créons le trigger TOTAL_INTERVENTIONS_TRIGGER qui met à jour cet attribut TOTAL_ INTERVENTIONS.

9.1 Requête

```
ALTER TABLE EMPLOYE ADD (TOTAL_INTERVENTIONSINTEGER DEFAULT 0);
```

9.2 Résultat

```
Table altered.
```

le trigger suivant incrémente la valeur de l'attribut TOTAL_INTERVENTIONS lorsqu'un employé participe a une nouvelle intervention, c'est à dire lorsque un élément (tuple) est inséré dans la table INTERVENANT

9.3 Requête

```
CREATE OR REPLACE TRIGGER TOTAL_INTERVENTIONS_TRIGGER
AFTER INSERT ON INTERVENANT
FOR EACH ROW
BEGIN
    UPDATE EMPLOYE E
    SET TOTAL_INTERVENTIONS = TOTAL_INTERVENTIONS + 1
    WHERE E.NUMEMPLOYE = :NEW.NUMEMPLOYE ;
END;
/
```

9.4 Résultat

```
Trigger created.
```

10 Tester les triggers

Pour pouvoir s'assurer que le trigger UPDATE_SALAIRE est fonctionnelle on va afficher le nombre d'intervention avant et après l'insertion d'une intervention d'un employé donné

10.1 Requête

```
SELECT TOTAL_INTERVENTIONS FROM EMPLOYE WHERE NUMEMPLOYE = 53;
INSERT INTO INTERVENANT VALUES (1,53,to_date('2017-01-27 08:00:00','YYYY-MM-DD HH24:MI:
    ↪ SS' ),to_date('2017-02-09 14:00:00','YYYY-MM-DD HH24:MI:SS' ));
SELECT TOTAL_INTERVENTIONS FROM EMPLOYE WHERE NUMEMPLOYE = 53;
```

10.2 Résultat

```
TOTAL_INTERVENTIONS
-------------------
                  0

1 row created.

TOTAL_INTERVENTIONS
-------------------
                  1
```

Effectivement après l'insertion le nombre totale d'intervention à été incrémenté

11 Création du trigger pour sauvegarder le total de gains de toutes les interventions

La suite de requêtes qui permet de répondre au besoin posé dans la question est */

- Création de la table CHIFRE AFFAIRE avec une requête CREATE TABLE
- Compter le nombre de tuples pour la date (le moi et l'année) en question avec une SELECT
- Si le nombre est nulle créer un nouveau tuple avec la requête INSERT
- Sinon mise à jour du nombre de gain actuelle avec la requête UPDATE

11.1 Requête

```
CREATE TABLE CHIFFRE_AFFAIRE
(
     MOIS INTEGER,
     ANNEE INTEGER ,
     TOTAL_GAINS INTEGER,
CONSTRAINT PK_CHIFFRE_AFFAIRE PRIMARY KEY (mois,annee),
CONSTRAINT CK_MOIS CHECK (Mois BETWEEN 1 AND 12)
);

CREATE OR REPLACE TRIGGER TRIGGER_GAINS
AFTER INSERT  ON INTERVENTIONS
FOR EACH ROW
DECLARE
CURSOR CR IS SELECT * FROM CHIFFRE_AFFAIRE ;
CR_GAIN CR%ROWTYPE;
I BINARY_INTEGER;
MOIS BINARY_INTEGER;
```

```
ANNE BINARY_INTEGER;
BEGIN
I:=0;
MOIS:=EXTRACT( MONTH FROM :NEW.DATEDEBINTERV );
ANNE:=EXTRACT(YEAR FROM :NEW.DATEDEBINTERV);
  FOR CR_GAIN IN CR LOOP
  EXIT WHEN  CR%NOTFOUND;

  IF(CR_GAIN.MOIS = MOIS AND ANNE=CR_GAIN.ANNEE ) THEN
  UPDATE CHIFFRE_AFFAIRE SET TOTAL_GAINS=(TOTAL_GAINS +:NEW.COUTINTERV)  WHERE(CR_GAIN.
      ↪ MOIS LIKE MOIS);
  DBMS_OUTPUT.PUT_LINE('TOTAL_GAINS DU MOIS DE : '|| CR_GAIN.MOIS ||' A ETE INCREMENTE
      ↪ DE :'|| :NEW.COUTINTERV ); I:=5;
  END IF;
  END LOOP;
  IF(I=0) THEN
  INSERT INTO CHIFFRE_AFFAIRE VALUES(MOIS,TO_CHAR(:NEW.DATEDEBINTERV ,'YYYY'),:NEW.
      ↪ COUTINTERV);
  DBMS_OUTPUT.PUT_LINE('UN NOUVEAU CHIFFRE_AFFAIRE A ETE AJOUTE ');
  DBMS_OUTPUT.PUT_LINE('DATE : MOIS : '|| MOIS ||' ,ANNEE:' || TO_CHAR(:NEW.
      ↪ DATEDEBINTERV ,'YYYY')||' TOTAL GAIN :'|| :NEW.COUTINTERV );
END IF;
END;
/
```

11.2 Résultat

```
Trigger created.
```

12 Tester les triggers

12.1 Requête et Résultat

```
SQL> INSERT INTO INTERVENTIONS VALUES (1000,3,'réparation',to_date('2006/02/25 0
9:00:00','YYYY-MM-DD HH24:MI:SS' ),to_date('2006/02/26 12:00:00','YYYY-MM-DD HH2
4:MI:SS'),30000);

UN NOUVEAU CHIFFRE_AFFAIRE A ETE AJOUTE
DATE : MOIS : 2 ,ANNEE:2006 TOTAL GAIN :30000

1 row created.

SQL> INSERT INTO INTERVENTIONS VALUES (2000,21,'réparation',to_date('2006-02-23
09:00:00','YYYY-MM-DD HH24:MI:SS' ),to_date('2006-02-24 18:00:00','YYYY-MM-DD HH
24:MI:SS' ),10000);

TOTAL_GAINS DU MOIS DE : 2 A ETE INCREMENTE DE :10000

1 row created.


SQL> SELECT * FROM CHIFFRE_AFFAIRE;

      MOIS      ANNEE TOTAL_GAINS
---------- ---------- -----------
         2       2006       80000
         4       2006       42000
         5       2006       10000
```

Effectivement après l'insertion de quelques interventions on voit clairement la mise à jour de la table CHIFFRE_AFFAIRE