Rapport TP5 LES TRIGGERS
Module : ASGBD

Ferchichi Manel
SII M1
202031036637
groupe tp : 1

1. **Créez un trigger qui affiche: « un nouveau utilisateur est ajouté» après chaque insertion d'un utilisateur. Répétez la même chose pour la modification ou la suppression.**

Après l ajout d un user

```
CREATE OR REPLACE TRIGGER ajout_user
AFTER INSERT ON USERS
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('L''utilisateur ' || :NEW.iduser || ' a été ajouté à
la table users');
END;
/
```

Apres la suppression d un user

```
CREATE OR REPLACE TRIGGER delete_user
AFTER DELETE ON USERS
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('L''utilisateur ' || :NEW.iduser || ' a été supprimé
à la table users');
END;
/
```

Apres la modification d un user

```
CREATE OR REPLACE TRIGGER update_user
AFTER update ON USERS
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('L''utilisateur ' || :NEW.iduser || ' a été modifié
à la table users');
END;
/
```

Capture exécution des triggers code:

```
SQL> CREATE OR REPLACE TRIGGER ajout_user
  2  AFTER INSERT ON USERS
  3  FOR EACH ROW
  4  BEGIN
  5  DBMS_OUTPUT.PUT_LINE('L''utilisateur ' || :NEW.iduser || ' a été ajouté à la table users');
  6  END;
  7  /

DÚclencheur crÚÚ.

SQL> CREATE OR REPLACE TRIGGER delete_user
  2  AFTER DELETE ON USERS
  3  FOR EACH ROW
  4  BEGIN
  5  DBMS_OUTPUT.PUT_LINE('L''utilisateur ' || :NEW.iduser || ' a été supprimé à la table
  6  users');
  7  END;
  8  /

DÚclencheur crÚÚ.

SQL> CREATE OR REPLACE TRIGGER update_user
  2  AFTER update ON USERS
  3  FOR EACH ROW
  4  BEGIN
  5  DBMS_OUTPUT.PUT_LINE('L''utilisateur ' || :NEW.iduser || ' a été modifié à la table users');
  6  END;
  7  /

DÚclencheur crÚÚ.
```

Test insertion :

```
SQL> INSERT INTO users VALUES(11,'MERYEM', 'FERCHICHI', 'MERYEM.FERCHICHI@gmail.com');
L'utilisateur 11 a été ajouté à la table users

1 ligne crÚÚe.
```

Test suppression :

```
SQL> SET SERVEROUTPUT ON;
SQL> DELETE FROM users
  2  WHERE IDUSER = 11;
L'utilisateur 11 a été supprimé à la table users

1 ligne supprimÚe.
```

Test modification

```
SQL> SET SERVEROUTPUT ON;
SQL> UPDATE users
  2  SET lastname = 'Mebarki'
  3  WHERE IDUSER = 7;
L'utilisateur 7 a été modifié à la table users

1 ligne mise Ó jour.
```

2. **Créer un trigger qui, à la fois, affiche :**
   - **« L'objet d'adresse (MAC) est ajouté à l'utilisateur (Nom de l'utilisateur)» lorsqu'un nouvel objet est ajouté.**
   - **« L'objet d'adresse (MAC) est modifié » lorsqu'un objet est modifié.**
   - **« L'objet d'adresse (MAC) est supprimé » lorsqu'un objet est supprimé.**

```
CREATE OR REPLACE TRIGGER objet_user
AFTER INSERT OR UPDATE OR DELETE
ON thing
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    DBMS_OUTPUT.PUT_LINE('L''objet ' || :NEW.mac || ' a été ajouté à
l''utilisateur ' || :NEW.iduser);
  ELSIF DELETING THEN
    DBMS_OUTPUT.PUT_LINE('L''objet ' || :OLD.mac || ' est supprimé');
  ELSIF UPDATING THEN
    DBMS_OUTPUT.PUT_LINE('L''objet ' || :NEW.mac || ' est modifié');
  END IF;
END;
/
```

Capture Execution Triggers code :

```
SQL> CREATE OR REPLACE TRIGGER objet_user
  2  AFTER INSERT OR UPDATE OR DELETE
  3  ON thing
  4  FOR EACH ROW
  5  BEGIN
  6  IF INSERTING THEN
  7  DBMS_OUTPUT.PUT_LINE('L''objet ' || :NEW.mac || ' a été ajouté à l''utilisateur ' ||
  8  :NEW.iduser);
  9  ELSIF DELETING THEN
 10  DBMS_OUTPUT.PUT_LINE('L''objet ' || :OLD.mac || ' est supprimé');
 11  ELSIF UPDATING THEN
 12  DBMS_OUTPUT.PUT_LINE('L''objet ' || :NEW.mac || ' est modifié');
 13  END IF;
 14  END;
 15  /

DÚclencheur crÚÚ.
```

Test Insertion d' un objet :

```
SQL> set serveroutput on ;
SQL> INSERT INTO thing VALUES('f0:de:f1:39:7f:26',5, '', '');
L'objet f0:de:f1:39:7f:26 a été ajouté à l'utilisateur 5

1 ligne crÚÚe.
```

Test modification d' un objet :

```
SQL> UPDATE THING
  2  SET IDUSER =8
  3  WHERE MAC='f0:de:f1:39:7f:26';
L'objet f0:de:f1:39:7f:26 est modifié

1 ligne mise Ó jour.
```

Test suppression d' un objet :

```
SQL> DELETE FROM THING WHERE MAC ='f0:de:f1:39:7f:26';
L'objet f0:de:f1:39:7f:26 est supprimé

1 ligne supprimÚe.
```

3. **Créer un trigger qui vérifie avant la modification de iduser dans la table Thing que la nouvelle valeur existe réellement, sinon, il refuse l'opération.**

```
CREATE OR REPLACE TRIGGER exist_iduser
BEFORE UPDATE ON Thing
FOR EACH ROW
DECLARE
   exist_fois NUMBER;
BEGIN
   SELECT COUNT(*) INTO exist_fois
   FROM USERS
   WHERE IDUSER = :NEW.iduser;

   IF exist_fois = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'La nouvelle valeur de iduser
n''existe pas.');
   END IF;
END;
/
```

Capture Execution Trigger code :

```
SQL> CREATE OR REPLACE TRIGGER exist_iduser
  2  BEFORE UPDATE ON Thing
  3  FOR EACH ROW
  4  DECLARE
  5      exist_fois NUMBER;
  6  BEGIN
  7
  8      SELECT COUNT(*)
  9      INTO exist_fois
 10      FROM USERS
 11      WHERE IDUSER = :NEW.iduser;
 12
 13
 14      IF exist_fois= 0 THEN
 15          RAISE_APPLICATION_ERROR(-20001, 'La nouvelle valeur de iduser n''existe pas.');
 16      END IF;
 17  END;
 18  /
```

Capture Test :

```
SQL> SET SERVEROUTPUT ON;
SQL> UPDATE THING
  2  SET IDUSER =10
  3  WHERE MAC='f0:de:f1:39:7f:22';
UPDATE THING
       *
ERREUR Ó la ligne 1 :
ORA-20001: La nouvelle valeur de iduser n'existe pas.
ORA-06512: Ó "DBAIOT.EXIST_IDUSER", ligne 12
ORA-04088: erreur lors d'exÚcution du dÚclencheur 'DBAIOT.EXIST_IDUSER'
```

4. **L'administrateur veut avoir le nombre d'objets de chaque utilisateur sans avoir à le calculer à chaque fois. Pour cela, il ajoute un attribut : NbThing dans la table Users.**

```
Ajoutez  l'attribut.
ALTER TABLE Users
ADD (NbThing NUMBER DEFAULT 0);
```

Capture Execution requete :

```
SQL> ALTER TABLE Users
  2  ADD (NbThing NUMBER DEFAULT 0);

Table modifiÚe.

SQL> DESC USERS;
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ---------------
 IDUSER                                    NOT NULL NUMBER(10)
 LASTNAME                                  NOT NULL VARCHAR2(50)
 FIRSTNAME                                          VARCHAR2(50)
 EMAIL                                              VARCHAR2(100)
 NBTHING                                            NUMBER
```

Capture avant definition de la valeur nbthing

```
SQL> SELECT * FROM USERS WHERE IDUSER=4 ;

    IDUSER LASTNAME
---------- ----------------------------------------
FIRSTNAME
--------------------------------------------------
EMAIL
--------------------------------------------------
   NBTHING
----------
         4 Samia
OUALI
samia.ouali@gmail.com
         0
```

```
UPDATE Users u
SET NbThing = (SELECT COUNT(*) FROM Thing t WHERE t.iduser = u.iduser);
```

```
SQL> UPDATE Users u
  2  SET NbThing = (SELECT COUNT(*) FROM Thing t WHERE t.iduser = u.iduser);
L'utilisateur 1 a été modifié à la table users
L'utilisateur 2 a été modifié à la table users
L'utilisateur 3 a été modifié à la table users
L'utilisateur 4 a été modifié à la table users
L'utilisateur 5 a été modifié à la table users
L'utilisateur 6 a été modifié à la table users
L'utilisateur 7 a été modifié à la table users
L'utilisateur 8 a été modifié à la table users
L'utilisateur 9 a été modifié à la table users
```

Créez un trigger NbThing_trigger qui met à jour l'attribut NbThing après l'insertion d'un objet.

```
CREATE OR REPLACE TRIGGER NbThing_trigger
AFTER INSERT ON Thing
FOR EACH ROW
BEGIN
    UPDATE Users
    SET :NEW.NbThing = :OLD.NbThing + 1
    WHERE iduser = :NEW.iduser;
END;
/
```

capture execution trigger :

```
SQL> CREATE OR REPLACE TRIGGER NbThing_trigger
  2  AFTER INSERT ON Thing
  3  FOR EACH ROW
  4  BEGIN
  5  UPDATE Users
  6  SET NbThing = NbThing + 1
  7  WHERE iduser = :NEW.iduser;
  8  END;
  9  /

DÚclencheur crÚÚ.
```

Test :
NBthing avant une insertion :

```
SQL> SELECT NBTHING FROM USERS WHERE IDUSER=2 ;

   NBTHING
----------
         3
```

NBthing apres une insertion

```
SQL> INSERT INTO thing VALUES('f0:de:f1:39:7f:25',2,'thingtempo',1.5);
L'utilisateur 2 a été modifié à la table users
L'objet f0:de:f1:39:7f:25 a été ajouté à l'utilisateur 2

1 ligne crÚÚe.

SQL> SELECT NBTHING FROM USERS WHERE IDUSER=2 ;

   NBTHING
----------
         4
```

Créez un trigger NbThingUpdate_trigger qui met à jour l'attribut NbThing après la modification de iduser dans la table Thing. La mise à jour concerne l'ancien et le nouvel utilisateur.

```
CREATE OR REPLACE TRIGGER NbThingUpdate_trigger
AFTER UPDATE OF iduser ON Thing
FOR EACH ROW
BEGIN
    -- on enleve un objet pour l ancien user id
    UPDATE Users
    SET :NEW.NbThing = :OLD.NbThing - 1
    WHERE iduser = :OLD.iduser;

    -- et on ajoute cet objet au nouveau user id
    UPDATE Users
    SET :NEW.NbThing = :OLD.NbThing + 1
    WHERE iduser = :NEW.iduser;
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER NbThingUpdate_trigger
  2  AFTER UPDATE OF iduser ON Thing
  3  FOR EACH ROW
  4  BEGIN
  5  -- on enleve un objet pour l ancien user id
  6  UPDATE Users
  7  SET NbThing = NbThing - 1
  8  WHERE iduser = :OLD.iduser;
  9  -- et on ajoute cet objet au nouveau user id
 10  UPDATE Users
 11  SET NbThing = NbThing + 1
 12  WHERE iduser = :NEW.iduser;
 13  END;
 14  /
```

Test :
modification d un objet

```
SQL> UPDATE THING
  2  SET IDUSER =5
  3  WHERE MAC='f0:de:f1:39:7f:25';
L'utilisateur 2 a été modifié à la table users
L'utilisateur 5 a été modifié à la table users
L'objet f0:de:f1:39:7f:25 est modifié

1 ligne mise Ó jour.
```

Nbthing avant modification pour iduser =2 est 4 et iduser=5 est 0

Nbthing après modification pour iduser =2 est 3 et iduser=5 est 1

```
SQL> SELECT NBTHING FROM USERS WHERE IDUSER=2;

   NBTHING
----------
         3
```

```
SQL> SELECT NBTHING FROM USERS WHERE IDUSER=5;

   NBTHING
----------
         1
```