ASGBD

Master 1 IL
Groupe 2
2018

Rapport de TP N°5 ASGBD : TRIGGERS

BOUDOUR Mehdi / 201500008386/ TP: ORACLE (5) TRIGGERS

[ARCHITECTURE DES S.G.B.D. RELATIONNELS]

E-mail : geronimotoutcourt@gmail.com

1) Créez un trigger qui affiche « **un nouveau employé de type infirmier est ajouté**» après chaque insertion d'un infirmier. Répétez la même chose pour la modification ou la suppression.

```
--1)
connect SYSTEM/pwd

connect DBHOPITAL/pwd;
CREATE OR REPLACE TRIGGER AFFICHER_MAJ_INFIRMIER
AFTER INSERT OR UPDATE OR DELETE ON INFIRMIER
FOR EACH ROW

BEGIN
      IF INSERTING THEN DBMS_OUTPUT.PUT_LINE('un nouveau employé de type
infirmier est ajouté (N° '||:NEW.NUM_INF||').'); END IF;
      IF UPDATING THEN DBMS_OUTPUT.PUT_LINE('un employé de type infirmier a ete
modifié (N° '||:NEW.NUM_INF||').'); END IF;
      IF DELETING THEN DBMS_OUTPUT.PUT_LINE('un employé de type infirmier a ete
supprimé (N° '||:OLD.NUM_INF||').'); END IF;
END;
/
```

```
SQL> connect SYSTEM/pwd
Connected.
SQL>
SQL> connect DBHOPITAL/pwd;
Connected.
SQL> --1)
SQL>
SQL> CREATE OR REPLACE TRIGGER AFFICHER_MAJ_INFIRMIER
  2  AFTER INSERT OR UPDATE OR DELETE ON INFIRMIER
  3  FOR EACH ROW
  4
  5  BEGIN
  6  IF INSERTING THEN DBMS_OUTPUT.PUT_LINE('un nouveau employé de type infirmier est ajouté (N° '||:NEW.NUM_INF||').'); END IF;

  7  IF UPDATING THEN DBMS_OUTPUT.PUT_LINE('un employé de type infirmier a ete modifié (N° '||:NEW.NUM_INF||').'); END IF;
  8  IF DELETING THEN DBMS_OUTPUT.PUT_LINE('un employé de type infirmier a ete supprimé (N° '||:OLD.NUM_INF||').'); END IF;
  9  END;
 10  /

Trigger created.
```

REMARQUE :
Testons le trigger avec 3 requêtes **DML** sur **INFIRMIER** : (1 insertion, une modification et une suppression).

```
SET SERVEROUTPUT ON;
      INSERT INTO INFIRMIER VALUES  (99,'REA','JOUR',12560.78);
      UPDATE INFIRMIER SET ROTATION = 'NUIT' WHERE NUM_INF = 99;
      DELETE FROM INFIRMIER WHERE NUM_INF = 99;
```

```
SQL> SET SERVEROUTPUT ON;
SQL> INSERT INTO INFIRMIER VALUES  (99,'REA','JOUR',12560.78);
un nouveau employé de type infirmier est ajouté (N° 99).

1 row created.

SQL> UPDATE INFIRMIER SET ROTATION = 'NUIT' WHERE NUM_INF = 99;
un employé de type infirmier a ete modifié (N° 99).

1 row updated.

SQL> DELETE FROM INFIRMIER WHERE NUM_INF = 99;
un employé de type infirmier a ete supprimé (N° 99).

1 row deleted.
```

2) Créez un trigger qui affiche « un nouveau infirmier est affecté à un [Nom de service] » après chaque insertion d'un infirmier.

```
--2)
CREATE OR REPLACE TRIGGER AFFICHER_AFFECTATION_INFIRMIER
AFTER INSERT  ON INFIRMIER
FOR EACH ROW
DECLARE
s SERVICE.NOM_SERVICE%TYPE;
BEGIN
      SELECT NOM_SERVICE INTO s
      FROM SERVICE
      WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
      DBMS_OUTPUT.PUT_LINE('un nouveau infirmier est affecté au service :
'||s||'.');
END;
/
```

```
SQL> --2)
SQL>
SQL> CREATE OR REPLACE TRIGGER AFFICHER_AFFECTATION_INFIRMIER
  2  AFTER INSERT  ON INFIRMIER
  3  FOR EACH ROW
  4  DECLARE
  5  s SERVICE.NOM_SERVICE%TYPE;
  6  BEGIN
  7  SELECT NOM_SERVICE INTO s
  8  FROM SERVICE
  9  WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
 10  DBMS_OUTPUT.PUT_LINE('un nouveau infirmier est affecté au service : '||s||'.');
 11  END;
 12  /

Trigger created.
```

REMARQUE :
Testons le trigger avec une insertion dans la table **INFIRMIER**.

```
SET SERVEROUTPUT ON;
INSERT INTO INFIRMIER VALUES  (99,'REA','JOUR',12560.78);
```

```
SQL> SET SERVEROUTPUT ON;
SQL> INSERT INTO INFIRMIER VALUES  (99,'REA','JOUR',12560.78);
un nouveau infirmier est affecté au service : RÚanimation et Traumatologie.
un nouveau employé de type infirmier est ajouté (N° 99).

1 row created.
```

3) Créer un triggers qui vérifie avant modification du code service dans la table infirmier que la nouvelle valeur existe réellement, sinon, il refuse l'opération.

REMARQUE :
Pour ce qui est du refus de l'opération on utilise la procédure introduite dans l'énoncé du TP **RAISE_APPLICATION_ERROR (**-Num_Message, 'Message à Afficher'**)** .

```
--3)
CREATE OR REPLACE TRIGGER VERIF_CODE_SERVICE_INF
BEFORE UPDATE OF CODE_SERVICE ON INFIRMIER
FOR EACH ROW

DECLARE
NB_SELECTION INTEGER;
BEGIN
      SELECT COUNT(*) INTO NB_SELECTION
```

```
      FROM SERVICE WHERE CODE_SERVICE = :NEW.CODE_SERVICE;

      IF(NB_SELECTION <> 1) THEN
            RAISE_APPLICATION_ERROR(-20000, 'Erreur : CODE_SERVICE =
'||:NEW.CODE_SERVICE||' nexiste pas dans la table SERVICE.') ;
      END IF;
END;
/
```

```
SQL> --3)
SQL> CREATE OR REPLACE TRIGGER VERIF_CODE_SERVICE_INF
  2  BEFORE UPDATE OF CODE_SERVICE ON INFIRMIER
  3  FOR EACH ROW
  4
  5  DECLARE
  6  NB_SELECTION INTEGER;
  7  BEGIN
  8  SELECT COUNT(*) INTO NB_SELECTION
  9  FROM SERVICE WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
 10
 11  IF(NB_SELECTION <> 1) THEN
 12  RAISE_APPLICATION_ERROR(-20000, 'Erreur : CODE_SERVICE = '||:NEW.CODE_SERVICE||' nexiste pas dans la table SERVICE.') ;
 13  END IF;
 14
 15  END;
 16  /

Trigger created.
```

REMARQUE :
Testons le trigger en tentant une modification du **CODE_SERVICE** d'un **INFIRMIER** avec une valeur qui n'existe pas dans la table **SERVICE**.

```
SET SERVEROUTPU ON;
UPDATE INFIRMIER SET CODE_SERVICE = 'PED' WHERE NUM_INF = 12;
```

```
SQL> SET SERVEROUTPU ON;
SQL> UPDATE INFIRMIER SET CODE_SERVICE = 'PED' WHERE NUM_INF = 12;
UPDATE INFIRMIER SET CODE_SERVICE = 'PED' WHERE NUM_INF = 12
       *
ERROR at line 1:
ORA-20000: Erreur : CODE_SERVICE = PED nexiste pas dans la table SERVICE.
ORA-06512: at "DBHOPITAL.VERIF_CODE_SERVICE_INF", line 8
ORA-04088: error during execution of trigger 'DBHOPITAL.VERIF_CODE_SERVICE_INF'
```

REMARQUE :
Le message d'erreur définit au niveau du trigger s'est bel et bien affiché.

4) Créer un trigger qui vérifie que lors de la modification du salaire d'un infirmier, la nouvelle valeur ne peut jamais être inférieure à la précédente.

REMARQUE :
L'idée ici est de réutiliser **RAISE_APPLICATION_ERROR** afin d'interrompre l'opération de mise à jour dans le cas où le **SALAIRE** d'un **INFIRMIER** venait à diminuer.

```
--4)
CREATE OR REPLACE TRIGGER VERIF_SALAIRE_INF
BEFORE UPDATE OF SALAIRE ON INFIRMIER
FOR EACH ROW

BEGIN
      IF(:OLD.SALAIRE > :NEW.SALAIRE) THEN
            RAISE_APPLICATION_ERROR(-20001, 'la nouvelle valeur
('||:NEW.SALAIRE||') ne peut jamais être inférieure à la précédente
('||:OLD.SALAIRE||').') ;
      END IF;
END;
```

```
/
```

```
SQL> --4)
SQL> CREATE OR REPLACE TRIGGER VERIF_SALAIRE_INF
  2  BEFORE UPDATE OF SALAIRE ON INFIRMIER
  3  FOR EACH ROW
  4
  5  BEGIN
  6  IF(:OLD.SALAIRE > :NEW.SALAIRE) THEN
  7  RAISE_APPLICATION_ERROR(-20001, 'la nouvelle valeur ('||:NEW.SALAIRE||') ne peut jamais être inférieure à la précédente ('|
|:OLD.SALAIRE||').') ;
  8  END IF;
  9  END;
 10  /

Trigger created.
```

REMARQUE :
Testons le trigger avec une tentative de **UPDATE** du salaire d'un **INFIRMIER** en y affectant une valeur inférieur à la précédente.

```
SET SERVEROUTPUT ON;
UPDATE INFIRMIER SET SALAIRE = SALAIRE - 2000 ;
```

```
SQL> SET SERVEROUTPUT ON;
SQL> UPDATE INFIRMIER SET SALAIRE = SALAIRE - 2000 ;
UPDATE INFIRMIER SET SALAIRE = SALAIRE - 2000
       *
ERROR at line 1:
ORA-20001: la nouvelle valeur (10560,78) ne peut jamais être inférieure à la
précédente (12560,78).
ORA-06512: at "DBHOPITAL.VERIF_SALAIRE_INF", line 3
ORA-04088: error during execution of trigger 'DBHOPITAL.VERIF_SALAIRE_INF'
```

REMARQUE :
Une erreur est levée et le message définit au niveau du trigger est affiché, preuve de sa bonne exécution.

5) L'administrateur veut, pour un besoin interne, avoir le total des salaires infirmiers pour chaque service. Pour cela, il ajoute un attribut : TOTAL_SALAIRE_SERVICE dans la table service.

a) Ajoutez l'attribut.

```
--5)
--a)
ALTER TABLE SERVICE
ADD TOTAL_SALAIRE_SERVICE FLOAT ;
```

```
SQL> --5)
SQL> --a)
SQL> ALTER TABLE SERVICE
  2  ADD TOTAL_SALAIRE_SERVICE FLOAT ;

Table altered.
```

REMARQUE :
Suite à la création de l'attribut, il faut mettre à jour les services afin que les valeurs contenues dans **TOTAL_SALAIRE_SERVICE** soit cohérentes au départ.

```
UPDATE SERVICE
SET TOTAL_SALAIRE_SERVICE = (
SELECT SUM(SALAIRE)
FROM INFIRMIER
WHERE CODE_SERVICE = SERVICE.CODE_SERVICE);
```

```
SQL> UPDATE SERVICE
  2  SET TOTAL_SALAIRE_SERVICE = (
  3  SELECT SUM(SALAIRE)
  4  FROM INFIRMIER
  5  WHERE CODE_SERVICE = SERVICE.CODE_SERVICE);

3 rows updated.
```

b) Créez un trigger TOTALSALAIRE_SERIVCE_TRIGGER qui met à jour l'attribut TOTAL_SALAIRE_SERVICE après l'insertion d'un infirmier.

```
--b)
CREATE OR REPLACE TRIGGER TotalSalaire_Serivce_trigger
AFTER INSERT ON INFIRMIER
FOR EACH ROW

BEGIN
      UPDATE SERVICE
      SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE + :NEW.SALAIRE
      WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
END;
/
```

```
SQL> --b)
SQL> CREATE OR REPLACE TRIGGER TotalSalaire_Serivce_trigger
  2  AFTER INSERT ON INFIRMIER
  3  FOR EACH ROW
  4
  5  BEGIN
  6  UPDATE SERVICE
  7  SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE + :NEW.SALAIRE
  8  WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
  9  END;
 10  /

Trigger created.
```

REMARQUE :
Testons le triggers en effectuant une insertion dans **INFIRMIER** et observons le changement de valeurs des **TOTAL_SALAIRE_SEVICE**.

/*Valeurs de TOTAL_SALAIRE_SERVICE avant l'insertion*/

```
SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;
```

```
SQL> SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;

CODE_SERVICE         TOTAL_SALAIRE_SERVICE
-------------------- ---------------------
CAR                              120548,33
CHG                              192028,88
REA                               96517,07
```

REMARQUE :
Suite à la création de l'attribut, il faut mettre à jour les services afin que les valeurs contenues dans **TOTAL_SALAIRE_SERVICE** soit cohérentes au départ.

/*INSERTION d'un infirmier dans le service 'REA'*/

```
INSERT INTO INFIRMIER VALUES  (99,'REA','JOUR',10060.78);
```

```
SQL> INSERT INTO INFIRMIER VALUES  (99,'REA','JOUR',10060.78);
un nouveau infirmier est affecté au service : RUanimation et Traumatologie.
un nouveau employé de type infirmier est ajouté (N° 99).

1 row created.
```

/*Valeurs de TOTAL_SALAIRE_SERVICE après l'insertion*/

```
SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;
```

```
SQL> SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;

CODE_SERVICE         TOTAL_SALAIRE_SERVICE
-------------------- ---------------------
CAR                                120548,33
CHG                                192028,88
REA                                106577,85
```

REMARQUE :
Aavant : Total_Salaire_Service(REA) = 96517,07 . Après : Total_Salaire_Service(REA) = 106577,85 – Sa valeur a augmenté de 10060.

c) Créez un trigger TOTALSALAIREUPDATE_TRIGGER qui met à jour l'attribut TOTAL_SALAIRE_SERVICE après la mise à jour d'un salaire.

```
--c)
CREATE OR REPLACE TRIGGER TotalSalaireUpdate_trigger
AFTER UPDATE OF SALAIRE ON INFIRMIER
FOR EACH ROW

BEGIN
      UPDATE SERVICE
      SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE - :OLD.SALAIRE
+ :NEW.SALAIRE
      WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
END;
/
```

```
SQL> --c)
SQL> CREATE OR REPLACE TRIGGER TotalSalaireUpdate_trigger
  2  AFTER UPDATE OF SALAIRE ON INFIRMIER
  3  FOR EACH ROW
  4
  5  BEGIN
  6  UPDATE SERVICE
  7  SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE - :OLD.SALAIRE + :NEW.SALAIRE
  8  WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
  9  END;
 10  /

Trigger created.
```

REMARQUE :
Testons le triggers en effectuant une modification du **SALAIRE** d'un **INFIRMIER** et observons le changement de valeurs des **TOTAL_SALAIRE_SEVICE**.

/*Valeurs de TOTAL_SALAIRE_SERVICE avant la modification*/

```
SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;
```

```
SQL> SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;

CODE_SERVICE         TOTAL_SALAIRE_SERVICE
-------------------- ---------------------
CAR                                120548,33
CHG                                192028,88
REA                                116638,63
```

/*MODIFICATION*/

```
UPDATE INFIRMIER SET SALAIRE = SALAIRE + 500 WHERE NUM_INF = 99;
```

```
SQL> UPDATE INFIRMIER SET SALAIRE = SALAIRE + 500 WHERE NUM_INF = 99;
un employé de type infirmier a ete modifié (N° 99).

1 row updated.
```

/*Valeurs de TOTAL_SALAIRE_SERVICE après la modification*/

```
SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;
```

```
SQL> SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;

CODE_SERVICE         TOTAL_SALAIRE_SERVICE
-------------------- ---------------------
CAR                                120548,33
CHG                                192028,88
REA                                117138,63
```

REMARQUE :
Aavant : Total_Salaire_Service(REA) = 116638.63 . Après : Total_Salaire_Service(REA) =117138.63 – Sa valeur à augmenter de 500.

6) Un infirmier peut changer de service. Créer un trigger qui met à jour l'attribut TOTAL_SALAIRE_SERVICE des deux services.

REMARQUE :
Le changement de service d'un **INFIRMIER** revient à effectuer une modification de son attribut **CODE_SERVICE** référençant le service auquel il est affecté, donc le trigger qu'il faut créer doit se déclencher suite à la modification (**UPDATE**) de l'attribut **CODE_SERVICE** dans la table **INFIRMIER**.

```
--6)
CREATE OR REPLACE TRIGGER TotalSalChangerService_trigger
AFTER UPDATE OF CODE_SERVICE ON INFIRMIER
FOR EACH ROW

BEGIN
      -- Ici :NEW.SALAIRE = :OLD.SALAIRE puisque SALAIRE n'est pas modifié
      UPDATE SERVICE
      SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE + :NEW.SALAIRE
      WHERE CODE_SERVICE = :NEW.CODE_SERVICE;

      UPDATE SERVICE
      SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE - :NEW.SALAIRE
      WHERE CODE_SERVICE = :OLD.CODE_SERVICE;
END;
/
```

```
SQL> --6)
SQL> CREATE OR REPLACE TRIGGER TotalSalChangerService_trigger
  2  AFTER UPDATE OF CODE_SERVICE ON INFIRMIER
  3  FOR EACH ROW
  4
  5  BEGIN
  6  UPDATE SERVICE
  7  SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE + :NEW.SALAIRE
  8  WHERE CODE_SERVICE = :NEW.CODE_SERVICE;
  9
 10  UPDATE SERVICE
 11  SET TOTAL_SALAIRE_SERVICE = TOTAL_SALAIRE_SERVICE - :NEW.SALAIRE
 12  WHERE CODE_SERVICE = :OLD.CODE_SERVICE;
 13  END;
 14  /

Trigger created.
```

REMARQUE :
Testons le triggers en effectuant une modification du **CODE_SERVICE** d'un **INFIRMIER** et observons le changement de valeurs des **TOTAL_SALAIRE_SEVICE**.

/*Valeurs de TOTAL_SALAIRE_SERVICE avant la modification*/

```
SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;
```

```
SQL> SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;

CODE_SERVICE         TOTAL_SALAIRE_SERVICE
-------------------- ---------------------
CAR                              120548,33
CHG                              192028,88
REA                              127199,41
```

```
UPDATE INFIRMIER
SET CODE_SERVICE = 'CAR'
WHERE NUM_INF = 99;
```

```
SQL> UPDATE INFIRMIER
  2  SET CODE_SERVICE = 'CAR'
  3  WHERE NUM_INF = 99;
un employé de type infirmier a ete modifié (N° 99).

1 row updated.
```

```
SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;
```

```
SQL> SELECT CODE_SERVICE , TOTAL_SALAIRE_SERVICE FROM SERVICE;

CODE_SERVICE         TOTAL_SALAIRE_SERVICE
-------------------- ---------------------
CAR                              130609,11
CHG                              192028,88
REA                              117138,63
```

REMARQUE :
Total_Salaire_Service(**CAR**) a augmenté et Total_Salaire_Service(**REA**) a diminué.

7)L'administrateur veut sauvegarder toutes les hospitalisations des patients dans le temps. A chaque fois un patient est hospitalisé une ligne sur les informations de l'hospitalisation est sauvegardée dans une autre table « Hist_Hospit ». La table « Hist_Hospit » est définie par Hist_Hospit (date_hospit,num_patient code_service*). Où date_hospit est la date d'hospitalisation.

REMARQUE :
Commençons pas créer la table **HIST_HOSPIT**.

```
--7)
CREATE TABLE HIST_HOSPIT (
      DATE_HOSPIT DATE,
      NUM_PATIENT INTEGER,
      CODE_SERVICE VARCHAR2(20),
      CONSTRAINT PK_HIST_HOSPIT PRIMARY KEY (DATE_HOSPIT,NUM_PATIENT),
      CONSTRAINT FK_HIST_HOSPIT_SERVICE FOREIGN KEY (CODE_SERVICE)
      REFERENCES SERVICE (CODE_SERVICE),
      CONSTRAINT FK_HIST_HOSPIT_PATIENT FOREIGN KEY (NUM_PATIENT)
      REFERENCES PATIENT (NUM_PATIENT)
);
```

```
SQL> --7)
SQL> CREATE TABLE HIST_HOSPIT (
  2  DATE_HOSPIT DATE,
  3  NUM_PATIENT INTEGER,
  4  CODE_SERVICE VARCHAR2(20),
  5  CONSTRAINT PK_HIST_HOSPIT PRIMARY KEY (DATE_HOSPIT,NUM_PATIENT),
  6  CONSTRAINT FK_HIST_HOSPIT_SERVICE FOREIGN KEY (CODE_SERVICE)
  7  REFERENCES SERVICE (CODE_SERVICE),
  8  CONSTRAINT FK_HIST_HOSPIT_PATIENT FOREIGN KEY (NUM_PATIENT)
  9  REFERENCES PATIENT (NUM_PATIENT)
 10  );

Table created.
```

/*Création du Trigger*/

```
CREATE OR REPLACE TRIGGER HIST_HOSPITALISATION_TRIGG
AFTER INSERT ON HOSPITALISATION
FOR EACH ROW
BEGIN
      INSERT INTO HIST_HOSPIT VALUES
(SYSDATE,:NEW.NUM_PATIENT,:NEW.CODE_SERVICE);
--SYSDATE = Date Courantes.
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER HIST_HOSPITALISATION_TRIGG
  2  AFTER INSERT ON HOSPITALISATION
  3  FOR EACH ROW
  4  BEGIN
  5  INSERT INTO HIST_HOSPIT VALUES (SYSDATE,:NEW.NUM_PATIENT,:NEW.CODE_SERVICE);
  6  END;
  7  /

Trigger created.
```

REMARQUE :
Testons le trigger en insérant un tuple dans la table **HOSPITALISATION** et observant le contenu de la tables **HIST_HOSPIT**.

```
SELECT * FROM HIST_HOSPIT;
```

```
SQL> SELECT * FROM HIST_HOSPIT;

no rows selected
```

/*Insertion*/

```
INSERT INTO HOSPITALISATION VALUES (81,'CHG',401,1);
```

```
SQL> INSERT INTO HOSPITALISATION VALUES (81,'CHG',401,1);

1 row created.
```

/*Contenu de HIST_HOSPIT après l'insertion dans HOSPITALISATION*/

```
SELECT * FROM HIST_HOSPIT;
```

```
SQL> SELECT * FROM HIST_HOSPIT;

DATE_HOS NUM_PATIENT CODE_SERVICE
-------- ----------- --------------------
26/10/18          81 CHG
```

REMARQUE :
1 tuple à été inséré dans **HIST_HOSPIT** correspondant à la date courante et contenant les informations de l'**HOSPITALISATION** récemment inséré, donc l'effet de **HIST_HOSPITALISATION_TRIGG** est observé.