

Rapport du TP n°5 Les déclencheurs -Triggers-

Rougab Imene
M1 SII Groupe 01
Année scolaire : 2016/2017

1. **Créons le trigger « Mise A Jour Hotel » :** Ce trigger affiche un message après chaque insertion, mise à jour ou la suppression d'un hôtel effectuée dans la table « HOTEL »

Requête :

```
Create or replace trigger Mise_A_Jour_hotel
after insert or update or delete on hotel
BEGIN
  if (inserting) then
    dbms_output.put_line('Un nouvel Hotel a été ajouté');
  end if;
  if (updating) then
    dbms_output.put_line('L hotel a été mis à jour');
  end if;
  if (deleting) then
    dbms_output.put_line('L hotel a été supprimé');
  end if;
END;
/
```

Résultat de création et du test (insertion, mise à jour puis suppression d'un hôtel):

```
SQL> /* *****1***** */
SQL> Create or replace trigger Mise_A_Jour_hotel
  2  after insert or update or delete on hotel
  3  BEGIN
  4  if (inserting) then
  5  dbms_output.put_line('Un nouvel Hotel a été ajouté');
  6  end if;
  7  if (updating) then
  8  dbms_output.put_line('L hotel a été mis à jour');
  9  end if;
 10  if (deleting) then
 11  dbms_output.put_line('L hotel a été supprimé');
 12  end if;
 13  END;
 14  /

Déclencheur créé.

SQL> /* *****test***** */
SQL> insert into hotel values (15,'Mercure','Alger',5);
Un nouvel Hotel a été ajouté

1 ligne créée.

SQL> update hotel set nom = 'Hotel Suisse' where numhotel = 15;
L hotel a été mis à jour

1 ligne mise à jour.

SQL> delete from hotel where numhotel = 15;
L hotel a été supprimé

1 ligne supprimée.
```

2. Créons le trigger « Nouvelle Reservation » : Ce trigger affiche un message après chaque insertion d'une nouvelle réservation, avec le nom de l'hôtel dans lequel cette réservation a été effectuée.

Requête :

```
Create or replace trigger nouvelle_reservation
After insert on reservation
For each row
DECLARE nomhotel hotel.nom%type;
BEGIN
    select nom
    into nomhotel
    from hotel
    where numhotel= :new.numhotel;
    dbms_output.put_line('Une nouvelle reservation dans 1 hotel'
||nomhotel|| ' enregistrée');
END;
/
```

Résultat de la création et vérification :

```
SQL> Create or replace trigger nouvelle_reservation
  2  After insert on reservation
  3  For each row
  4  DECLARE nomhotel hotel.nom%type;
  5  BEGIN
  6  select nom
  7  into nomhotel
  8  from hotel
  9  where numhotel= :new.numhotel;
 10  dbms_output.put_line('Une nouvelle reservation dans 1 hotel' ||nomhotel|| ' enregistrée');
 11  END;
 12  /

Déclencheur créé.

SQL> Insert into Reservation values (1,2,'02/02/2016','10/02/2016',1);
Une nouvelle reservation dans 1 hotelSeybouse enregistrée

1 ligne créée.
```

3. **Créons le trigger « Vérification Prix Chambre » :** Ce trigger vérifie lors de la mise à jour du prix d'une nuit d'une chambre que le nouveau prix est supérieur à l'ancien, et rejette la requête sinon. On va utiliser pour cela « Raise_Application_Errors ».

Requête :

```
Create or replace trigger Verifcation_Prix_Chambre
Before update of PRIXNUIT on CHAMBRE
For each row
BEGIN
  if (:new.prixnuit < :old.prixnuit) then
    raise_application_error(-20010,'Attention, le nouveau prix doit être
    supérieur à l ancien prix.');
```

Résultat :

```
SQL> /* *****trigger 3***** */
SQL> Create or replace trigger Verifcation_Prix_Chambre
  2  Before update of PRIXNUIT on CHAMBRE
  3  For each row
  4  BEGIN
  5  if (:new.prixnuit < :old.prixnuit) then
  6    raise_application_error(-20010,'Attention, le nouveau prix doit être supérieur à l ancien prix.');
```

Insérons à présent un tuple dans la table chambre puis mettons à jour le prix, en choisissant un prix inférieur à celui inséré au début.

```
SQL> insert into chambre values (9,12,3,'double',10500);
1 ligne créée.

SQL> update chambre set prixnuit = 1 where numchambre = 9 and numHotel = 12;
update chambre set prixnuit = 1 where numchambre = 9 and numHotel = 12
*
ERREUR Ó la ligne 1 :
ORA-20010: Attention, le nouveau prix doit être supérieur à l ancien prix.
ORA-06512: Ó "TP5.VERIFCATION_PRIX_CHAMBRE", ligne 3
ORA-04088: erreur lors d'exécution du déclencheur 'TP5.VERIFCATION_PRIX_CHAMBRE'
```

4. L'administrateur veut avoir le total du gain de chaque hôtel :

4.1. Ajoutons l'attribut Total_Gains qui va contenir le total du gain de chaque hôtel à la table hôtel :

Requêtes:

```
--Ajout de l'attribut
Alter table Hotel add Total_Gains number(10);
--Initialisation pour pouvoir additionner après
update hotel set total_gains=0;
```

Résultats:

```
SQL> Alter table Hotel add Total_Gains number(10);
Table modifiée.
```

```
SQL> --Initialisation
SQL> update hotel set total_gains=0;
L hotel a été mis à jour
```

4.2. Créons le trigger « TOTAL_GAINS_HOTEL » qui va mettre à jour le total gain après chaque nouvelle réservation effectuée dans un hôtel :

Requêtes:

```
Create or replace trigger TOTAL_GAINS
After insert on reservation
For each row
DECLARE
prix_nuit number;
duree_reservation number;
montant_reservation number;
BEGIN
-- calcul de la durée de la nouvelle reservation
duree_reservation := :new.datedepart - :new.datearrivee;
dbms_output.put_line('Durée de la reservation: '||duree_reservation);

-- récupération du prix de la chambre réservée
select prixnuit into prix_nuit from chambre
where numchambre = :new.numchambre
and numhotel = :new.numhotel ;
dbms_output.put_line('Prix de la chambre réservé: '||prix_nuit);

-- calcul du prix total (montant) de la reservation
montant_reservation := duree_reservation * prix_nuit;
dbms_output.put_line('Montant total: '||montant_reservation);

-- Mise à jour du total de gains dans la table hotel
update hotel set total_gains =total_gains + montant_reservation
where numHotel = :new.numHotel;
dbms_output.put_line('Mise à jour du gain total effectuée ');
END;
/
```


Résultats:

```
SQL> --4.2*****
SQL> Create or replace trigger  TOTAL_GAINS
  2 After insert on reservation
  3 For each row
  4 DECLARE
  5 prix_nuit number;
  6 duree_reservation number;
  7 montant_reservation number;
  8 BEGIN
  9 -- calcul de la durée de la nouvelle reservation
 10 duree_reservation := :new.datedepart - :new.datearrivee;
 11 dbms_output.put_line('Durée de la reservation: '||duree_reservation);
 12
 13 -- récupération du prix de la chambre reservée
 14 select prixnuit
 15 into prix_nuit
 16 from chambre
 17 where numchambre = :new.numchambre
 18 and numhotel = :new.numhotel ;
 19 dbms_output.put_line('Prix de la chambre réservé: '||prix_nuit);
 20
 21 -- calcul du prix total (motant) de la reservation
 22 montant_reservation := duree_reservation * prix_nuit;
 23 dbms_output.put_line('Montant total: '||montant_reservation);
 24
 25 -- Mise à jour du total de gains dans la table hotel
 26 update hotel
 27 set total_gains =total_gains + montant_reservation
 28 where numHotel = :new.numHotel;
 29 dbms_output.put_line('Mise à jour du gain total effectuée ');
 30
 31 END;
 32 /

DÚclencheur crÚÚ.
```

Vérification et test du trigger : on ajoute une réservation « d'une durée de 10 jours », pour une chambre dont le prix d'une nuit est de « 6000 ». le total gain devrait donc être égal à « 60000 »

```
SQL> /* *****test*****/
SQL> Insert into Reservation values (4,6,'10/02/2016','20/02/2016',1);
Durée de la reservation: 10
Prix de la chambre réservé: 6000
Montant total: 60000
L hotel a été mis à jour
Mise à jour du gain total effectuée
Une nouvelle reservation dans l hotelEl Mountazah Annaba enregistrée

1 ligne crÚÚe.
```

Affichons à présent l'attribut Total_Gains de la table Hôtel pour vérifier que le gain total de l'hôtel N°6, El mountazah Annaba, a bien été mise à jour et que la valeur actuelle est de 60000 :

```
SQL> select * from hotel;
```

NUMHOTEL	NOM	VILLE	ETOILES	TOTAL_GAINS
1	Renaissance	Tlemcen	5	0
2	Seybouse	Annaba	3	0
3	Hôtel Novotel	Constantine	4	0
4	Saint George d'Alger	Alger	5	0
5	Ibis Alger Aéroport	Alger	2	0
6	El Mountazah Annaba	Annaba	3	60000
7	Hôtel Albert 1er	Alger	3	0
8	Chems	Oran	2	0
9	Colombe	Oran	3	0
10	Mercure	Alger	4	0
11	Le Méridien	Oran	5	0
12	Hôtel Sofitel	Alger	5	0

12 ligne(s) sélectionnée(s).

5. L'administrateur veut connaître à chaque instant le nombre de chambres libres de chaque hôtel :

5.1. Ajoutons l'attribut NBCHDISP qui va contenir le nombre de chambres disponibles de chaque hôtel à la table HOTEL :

Après l'ajout de l'attribut, on va calculer le nombre de chambre disponibles pour chaque hôtel pour le mettre à jour.

Pour ce fait, on va d'abord le nombre total des chambres de chaque hôtel (disponibles ou réservées). Puis on va calculer le nombre des chambres réservées, c'est-à-dire celles qui sont actuellement occupées, ou celles qui sont réservées prochainement. On fera ensuite la différence pour connaître le nombre de chambres disponibles.

Requêtes:

```
--Ajout de l'attribut et initialisation à 0
Alter table Hotel add NBCHDISP number(3);
Update hotel set NBCHDISP = 0 ;
--Création d'une procédure qui va calculer le nombre de chambres
disponibles pour mettre à jour NBCHDISP
create or replace procedure Init_nb_chambres_hotel as
cursor cr1 is select numhotel , count(numhotel) as nombre_chambres
              from chambre group by numHotel;
cursor cr2 is select numchambre, numhotel
              from reservation;
r1 cr1%rowtype;
r1 cr2%rowtype;
BEGIN

--Initialiser le nombre de chambre de chaque hotel au nombre total de
chambres qu'il contient
for r1 in cr1 loop
    update hotel
    set NBCHDISP = NBCHDISP + r1.nombre_chambres
    where numHotel = r1.numHotel ;
end loop ;

--Enlever le nombre de chambres réservés dans les jours à venir (si
datedepart>dateActuelle alors la chambre est résevrée)

for r2 in cr2 loop
    if(r2.datedepart>sysdate) then
        update hotel
        set NBCHDISP = NBCHDISP -1
        where numHotel = r2.numHotel ;
    end if;
end loop ;

END;
/
```


Résultats:

```
SQL> Alter table Hotel add NBCHDISP number(3);
```

Table modifiée.

```
SQL> update hotel set NBCHDISP = 0 ;
```

L hotel a été mis à jour

12 ligne(s) mise(s) à jour.

```
SQL> create or replace procedure Init_nb_chambres_hotel
```

```
2 as
```

```
3 cursor cr1 is select numhotel , count(numhotel) as nombre_chambres from chambre group by numHotel;
```

```
4 cursor cr2 is select * from reservation;
```

```
5 r1 cr1%rowtype;
```

```
6 r2 cr2%rowtype;
```

```
7 BEGIN
```

```
8 --Initialiser le nombre de chambre de chaque hotel au nombre total de chambres qu'il contient
```

```
9 for r1 in cr1 loop
```

```
10   update hotel
```

```
11     set NBCHDISP = NBCHDISP + r1.nombre_chambres
```

```
12     where numHotel = r1.numHotel ;
```

```
13 end loop ;
```

```
14 --Enlever le nombre de chambres réservés dans les jours à venir (si datedepart>dateActuelle alors la chambre est réservée)
```

```
15 for r2 in cr2 loop
```

```
16   if(r2.datedepart>sysdate) then
```

```
17     update hotel
```

```
18     set NBCHDISP = NBCHDISP -1
```

```
19     where numHotel = r2.numHotel ;
```

```
20   end if;
```

```
21 end loop ;
```

```
22 END;
```

```
23 /
```

Procédure créée.

```
SQL> execute Init_nb_chambres_hotel;
```

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

L hotel a été mis à jour

Procédure PL/SQL terminée avec succès.

Vérifions à présent que le nombre de chambres disponibles a bien été mis à jour dans la table HOTEL :

```
SQL> select numhotel, NBCHDISP from hotel;
```

NUMHOTEL	NBCHDISP
1	22
2	26
3	33
4	25
5	3
6	8
7	34
8	29
9	12
10	13
11	80
12	48

12 ligne(s) sélectionné(s).

5.2. Créons le trigger « Mise_A_Jour_NbChambres » qui va mettre à jour le nombre de chambre disponibles après chaque nouvelle réservation ou annulation d'une réservation :

Requêtes:

```
create or replace trigger Mise_A_Jour_NbChambres
After insert or delete on reservation
For each row
BEGIN
--Si une nouvelle réservation alors on décrémente le nombre de chambres
disponibles
if (inserting) then
  update hotel
    set NBCHDISP = NBCHDISP - 1
    where numHotel = :new.numHotel ;
end if;
--Si annulation d'une réservation alors on incrémente
if (deleting) then
  update hotel
    set NBCHDISP = NBCHDISP +1
    where numHotel = :old.numHotel ;
end if;
END;
/
```

Résultat de l'exécution et vérification :

Voici le nombre de chambres disponibles avant annulation ou ajout d'une réservation :

```
SQL> select numhotel, NBCHDISP from hotel;
```

NUMHOTEL	NBCHDISP
1	22
2	26
3	33
4	25
5	3
6	8
7	34
8	29
9	12
10	13
11	80

NUMHOTEL	NBCHDISP
12	48

12 ligne(s) sélectionné(s).

Supprimons une réservation d'une chambre de l'hôtel N°6, le nombre de chambres disponibles dans cet hôtel va augmenter:

```
SQL> Delete from Reservation where numhotel=6 and numclient=4 and datearrivee='30/11/2016';
L hotel a été mis à jour

1 ligne supprimée.

SQL> select numhotel, NBCHDISP from hotel;
```

NUMHOTEL	NBCHDISP
1	22
2	26
3	33
4	25
5	3
6	9
7	34
8	29
9	12
10	13
11	80

NUMHOTEL	NBCHDISP
12	48

12 ligne(s) sélectionné(s).

Ajoutons à nouveau cette réservation : Le nombre va être décrémenter de 1

```
SQL> Insert into Reservation values (4,6,'30/11/2016','15/12/2016',1);
L hotel a été mis à jour
Durée de la reservation: 15
Prix de la chambre réservé: 6000
Montant total: 90000
L hotel a été mis à jour
Mise à jour du gain total effectuée
Une nouvelle reservation dans 1 hotelEl Mountazah Annaba enregistrée

1 ligne cr    e.

SQL> select numhotel, NBCHDISP from hotel;
```

NUMHOTEL	NBCHDISP
1	22
2	26
3	33
4	25
5	3
6	8
7	34
8	29
9	12
10	13
11	80
12	48

```
12 ligne(s) s  lectionn  e(s).
```

6. L'administrateur souhaite garder un historique en temps des r  servations effectu  es :

Pour ce fait on va cr  er une table historique et un trigger qui va,    chaque fois qu'une r  servation est effectu  e, ins  rer cette r  servation dans la table historique en mettant la date    laquelle elle a   t   ins  r  e

Requ  te :

```
--Cr  ation de la table avec les m  mes attributs que la table reservation
-- + un attribut qui va contenir la date de l'ajout
Create table Historique_Reserv (
  NumClient number(5),
  NumHotel number(3),
  DateArrivee date ,
  DateDepart date ,
  NumChambre number(3),
  DateAjoutReserv date,
  constraint pk_H_Reserv primary key(numClient,numHotel,DateArrivee),
  constraint fk_H_Chambre foreign key (numHotel,numChambre) references
Chambre(numHotel,numChambre) on delete cascade,
  constraint fk_H_Client foreign key (numClient) references Client
(numClient) on delete cascade
);
```

```
-- Création du trigger qui va inserer dans la table historique lors de
-- l'insertion dans la table reservation
Create or replace trigger Insert_historique_reserv
After insert on reservation
For each row
BEGIN
insert into Historique_Reserv values
(:new.numClient , :new.numhotel , :new.DateArrivee , :new.DateDepart ,
:new.numChambre, sysdate );
END;
/
```

Résultats et vérification :

```
SQL> Create table Historique_Reserv (
2   NumClient number(5),
3   NumHotel number(3),
4   DateArrivee date ,
5   DateDepart date ,
6   NumChambre number(3),
7   DateAjoutReserv date,
8   constraint pk_H_Reserv primary key(numClient,numHotel,DateArrivee),
9   constraint fk_H_Chambre foreign key (numHotel,numChambre) references Chambre(numHotel,numChambre) on delete cascade,
10  constraint fk_H_Client foreign key (numClient) references Client (numClient) on delete cascade
11 );
```

Table cr   e.

```
SQL> Create or replace trigger Insert_historique_reserv
2   After insert on reservation
3   For each row
4   BEGIN
5   insert into Historique_Reserv values
6   (:new.numClient , :new.numhotel , :new.DateArrivee , :new.DateDepart , :new.numChambre, sysdate );
7   END;
8   /
```

D   clencheur cr    .

Ins  rons    pr  sent une nouvelle r  servation : on a bien la r  servation avec la date de son ajout

```
SQL> select * from Historique_Reserv;
```

NUMCLIENT	NUMHOTEL	DATEARRI	DATEDIPA	NUMCHAMBRE	DATEAJOU
4	11	20/10/16	30/10/16	7	11/11/16