RENDU TD2 BD TRAN FLORIAN GRP3

Exercice	Nom Trigger	Type : before ou after	Insert, delete, update	Nom Table	For each row :oui ou non
Ex 1 A	Modif_salaire	After	Update	Employe	oui
Ex 1 B	Modif_heure_hebdo	After	Updtae	Employe	oui
Ex 2 A	Supprimer_employe	Before	Delete	Employe	non
Ex 2 B	Supprimer_proj	Before	Delete	Projet	non
Ex 3 A	Maj_hebdo	After	Insert, update	Travail	non
Ex 3 B	Maj_resp_proj	After	Insert, update	Projet	non
Ex 3 C	Maj_concerne	After	Insert, update	Concerne	non
Ex 3 D	Maj_salaire_employevschef	After	Insert, update	Employe	non
Ex 3 E	Maj_salaire_chef_resp	After	Insert, update	Employe	non
Ex 3 F	Maj_salaire_chef_resp	After	Insert, update	Employe	non
Ex 4	Employe_alerte	After	Insert, update	Employe	oui

EX 1

A)

Le trigger modif_salaire interdit la diminution du salaire d'un employé. Avec la condition on regarde si le après l'update de salaire ce même salaire est moins grand que l'ancien sur la table employe grâce à :new.salaire(le nouveau) et :old.salaire(l'ancien). Si c'est vrai alors on lève l'exception 20001.

```
create or replace TRIGGER modif_salaire after update of salaire on employe for each row BEGIN
if :new.salaire<:old.salaire then
raise_application_error
(-20001,'viol de la règle :un salaire ne peut etre diminué');
end if;
END;
```

On test de mettre le salaire de l'employé numéro 20 à 1500 donc moins élevé que son salaire initial qui est de 2000.

```
update employe set salaire = 1500 where employe.nuempl = 20;
```

Donc on reçoit ce message d'erreur au numéro 20001 donc notre trigger à bien marché.

```
Erreur commençant à la ligne: 92 de la commande -
update employe set salaire = 1500 where employe.nuempl = 20
Rapport d'erreur -
ORA-20001: viol de la règle :un salaire ne peut etre diminué
ORA-06512: à "I2C08A.MODIF_SALAIRE", ligne 3
ORA-04088: erreur lors d'exécution du déclencheur 'I2C08A.MODIF_SALAIRE'
```

B)
Le trigger modfi_heure_hebdo interdit que la durée hebdomadaire d'un employé augmente. Donc on utilise un condition pour vérifié cela sur la table employe : si :newhebdo(la durée hebdomadaire après l'update) est supérieur à :old.hebdo (l'ancienne durée hebdomadaire). Si la condition est vérifié on lève l'exception 20002

```
create or replace TRIGGER modif_heure_hebdo after update of hebdo on employe for each row begin
if :new.hebdo>:old.hebdo then raise_application_error
(-20002,'viol de la règle: la durée hebdo dun employé ne peut pas augmenter');
end if;
end;
```

On test de mettre la durée hebdo de l'employé numéro 20 à 40 heures ce qui est plus grand que sa durée hebdo initial qui est de 20h.

```
update employe set hebdo = 40 where employe.nuempl =20;
```

Comme la durée hebdo augmente on reçoit le message d'erreur 20002 donc notre trigger à bien fonctionné.

```
Erreur commençant à la ligne: 104 de la commande -
update employe set hebdo = 40 where employe.nuempl =20
Rapport d'erreur -
ORA-20002: viol de la règle: la durée hebdo dun employé ne peut pas augmenter
ORA-06512: à "I2C08A.MODIF_HEURE_HEBDO", ligne 2
ORA-04088: erreur lors d'exécution du déclencheur 'I2C08A.MODIF_HEURE_HEBDO'
```

EX 2
A)
Supprimer_employe fait en sorte que la suppression d'un employé est accompagnée de la suppression des lignes de travail correspondantes. D'abord on doit changer la contrainte FK_employe en 'deferred' . Ensuite dans le trigger on chercher les lignes de travail de ce même employé pour les supprimées.

```
create or replace trigger supprimer_employe before
delete on employe
begin
delete from travail where nuempl not in (select nuempl from employe);
end;
```

Pour tester on supprime un employé

```
delete employe where nuempl = 42;
select * from travail where nuempl = 42;
```

Ensuite on vérifie que l'employé n'existe plus
B)
Même principe pour supprimer un projet Supprimer_proj fait en sorte que si on supprime un projet, on supprime toutes les autres lignes de travail correspondantes.

```
create or replace trigger supprimer_proj before
delete on projet
begin
delete from concerne where nuproj not in(select nuproj from employe);
delete from Travail where nuproj not in (SELECT nuproj from Travail);
end;
```

Et on test de supprimer un projet.

```
delete projet where nuproj = 103;
```

EX 3

A)

Le trigger Maj_hebdo fait en sorte que la somme de temps de travail d'un employé ne dépasse pas son temps de travail hebdomadaire. Pour cela on va faire une MAJ de la base de donnés qui va déclenché une erreur. On va utilisé la table travail. Pour trouver la somme de temps de travail d'un employé on va se servir de sum(duree). On va créer 2 message d'erreur une pour si il y a que 1 employé qui lève l'erreur ou plusieurs.

```
create or replace trigger maj_hebdo after insert or update on travail
declare
E_REC employe%ROWTYPE;
begin
select * INTO E_REC from employe e
where (select sum(duree) from travail t where e.nuempl=t.nuempl)> hebdo;
RAISE_APPLICATION_ERROR (-20003,'Il y a un employé ou la somme du temps dépasse les 35h');
EXCEPTION
WHEN NO_DATA_FOUND THEN NULL;
WHEN TOO_MANY_ROWS THEN
RAISE_APPLICATION_ERROR (-20004,'Il y a plusieurs employés ou la somme du temps dépasse les 35h');
END;
```

Ensuite on effectue un test pour lever l'exception

```
update travail set duree = 20 where nuempl = 20 and nuproj = 175;
```

Et on à réussi à avoir le message d'erreur numéro 20004 donc notre trigger à bien fonctionner

```
Erreur commençant à la ligne: 145 de la commande -
update travail set duree = 20 where nuempl = 20 and nuproj = 175
Rapport d'erreur -
ORA-20004: Il y a plusieurs employés ou la somme du temps dépasse les 35h
ORA-06512: à "I2C08A.MAJ_HEBDO", ligne 10
ORA-04088: erreur lors d'exécution du déclencheur 'I2C08A.MAJ_HEBDO'
```

B)

On fait pareil pour Maj_resp_proj qui fait en sorte qu'un employé n'ai pas plus de 3 projet. Cette fois ci on va utiliser le select(count(*)...

```
create or replace trigger maj_resp_proj after insert or update on projet
declare
T1_REC employe%ROWTYPE;
begin
Select * into T1_REC from employe e where (select count(*) from projet p where e.nuempl = p.resp )> 3;
RAISE_APPLICATION_ERROR (-20005, 'Un employé ne peut pas être responsable de plus de 3 projets en même temps');
EXCEPTION
WHEN NO_DATA_FOUND THEN NULL;
WHEN TOO_MANY_ROWS THEN
RAISE_APPLICATION_ERROR (-20007, 'Too many rows');
END;
```

On test :

```
insert into projet values (2,'test2',57);
update projet set resp = 57 where nuproj = 160;
```

On a bien le bon message d'erreur.

```
Erreur commençant à la ligne: 163 de la commande -
update projet set resp = 57 where nuproj = 160
Rapport d'erreur -
ORA-20005: Un employé ne peut pas être responsable de plus de 3 projets en même temps
ORA-06512: à "I2C08A.MAJ_RESP_PROJ", ligne 5
ORA-04088: erreur lors d'exécution du déclencheur 'I2C08A.MAJ_RESP_PROJ'
```

C)
On fait parreil pour Maj_concerne qui empêche que des services soit concernés par plus de 3 projets.

```
create or replace trigger maj_concerne after insert or update on concerne
declare
T2_REC service%ROWTYPE;
begin
select * into T2_REC from service s where (select count (*) from concerne c where c.nuserv=s.nuserv)>3;
raise_application_error(-20008, 'Une service ne peut être concerné par plus de 3 projets');
exception
when no_data_found then null;
when too_many_rows then
raise_application_error(-20009,'Il y a plusieurs services qui sont concernés par plus de 3 projets');
end;
```

On test en insérant une donnée dans concerne

```
insert into concerne values (1,237);
```

Et on obtient le bon message d'erreur 20009

```
Erreur commençant à la ligne: 179 de la commande -
insert into concerne values (1,237)
Rapport d'erreur -
ORA-20009: Il y a plusieurs services qui sont concernés par plus de 3 projets
ORA-06512: à "I2C08A.MAJ_CONCERNE", ligne 9
ORA-04088: erreur lors d'exécution du déclencheur 'I2C08A.MAJ_CONCERNE'
```

D)
Maj_salaire_employevschef empêche un salarié d'avoir un salaire supérieur à son chef de service.

```
create or replace trigger maj_salaire_employevschef after insert or update on employe
declare
T3_REC employe%ROWTYPE;
begin
select * into T3_REC from employe e1
where e1.nuempl In (Select chef from service)
and e1.salaire<(select max(e2.salaire) from employe e2 where e1.affect=e2.affect);
raise_application_error(-200010, 'Un employé ne peut pas avoir un plus grand salaire que son
chef');
exception
when no_data_found then null;
when too_many_rows then
raise_application_error(-200011,'Il y a des employé qui ont un plus grand saliare que leurs chef');
end;
```

On test de mettre un salaire supérieur à son chef de service.

```
update employe set salaire = 5000 where nuempl = 20;
```

E)
Maj_salaire_chef_resp fait en sorte que le chef de service gagne plus que le responsable de projet

```
CREATE OR REPLACE TRIGGER maj_salaire_chef_resp AFTER UPDATE OR INSERT ON
EMPLOYE
declare
T_REC4 EMPLOYE%rowtype;
BEGIN
select * into T_REC4 from EMPLOYE e
where e.NUEMPL in (select CHEF from SERVICE)
and e.SALAIRE < (select max(:NEW.salaire)
from EMPLOYE e1  where e1.NUEMPL in (select RESP from PROJET));
```

RAISE_APPLICATION_ERROR(-200012,'Le chef de service doit gagné plus que les responsable de projet');
EXCEPTION
WHEN NO_DATA_FOUND
THEN NULL;
WHEN TOO_MANY_ROWS
THEN RAISE_APPLICATION_ERROR(-20013,'Plusieurs chefs de service gagne moins que les responsable');
end;

On test avec :

```
update employe set salaire = 5000 where nuempl = 30;
```

F)
Oui c'est possible :

CREATE OR REPLACE TRIGGER maj_salaire_chef_resp AFTER UPDATE OR INSERT ON EMPLOYE
declare
T_REC5 EMPLOYE%rowtype;
BEGIN
select * into T_REC5 from EMPLOYE e
where e.NUEMPL in (select CHEF from SERVICE)
and e.SALAIRE < (select max(e1.salaire) from EMPLOYE e1
where e1.NUEMPL in
(select RESP from PROJET)
or e1.AFFECT = e.AFFECT);
RAISE_APPLICATION_ERROR(-200014,'Viol de la règle : le chef de service doit gagné plus que les responsable de projet ou que les employé de son service');
EXCEPTION
WHEN NO_DATA_FOUND
THEN NULL;
WHEN TOO_MANY_ROWS
THEN
RAISE_APPLICATION_ERROR(-20015,'Plusieurs chefs de service gagne moins que les responsable ou les employé de leur service');
end;

On le test :

```
update employe set salaire = 3500 where nuempl = 30;
```

<u>EX 4</u>
On veut créer un table employe_alerte qui enregistre tout les employés qui ont un salaire >5000€

CREATE OR REPLACE TRIGGER emp_alerte AFTER UPDATE OR INSERT ON EMPLOYE
for each row when (NEW.SALAIRE > 5000)
BEGIN
INSERT INTO EMPLOYE_ALERTE VALUES
(:NEW.NUEMPL,:NEW.NOMEMPL,:NEW.HEBDO,:NEW.AFFECT,:NEW.SALAIRE);
end;

on test en donnant à des employés des salaires >5000€
ensuite on faite un select * sur notre nouvelle table