TP N° 3 : Contraintes d'intégrité dynamiques : Triggers

-Se connecter avec un compte **system.**
-Créer un compte utilisateur appelé **userTP3**
-Affecter tous les privilèges à **userTP3**.
-Se connecter avec l'utilisateur userTP3.

-Créer la table Service en définissant la clé primaire puis insérer les tuples de Service.

1- Ajouter dans la table Service l'information sur le nombre d'opérations réalisées par un service (NbOp) ainsi que son nombre d'employés (NbEmp) **(Valeurs par défaut 0)**.

-Créer les tables Employe et Operation, en définissant les clés primaires, **sans insérer les tuples**.

2- Définir deux déclencheurs (triggers) **Increm_NbOP** et **decrem_NbOP**. **Increm_NbOP** (resp. **decrem_NbOP**) permet à chaque insertion (resp. Suppression) d'un tuple dans la table OPERATION, d'incrémenter (resp. Décrémenter) NbOp dans la table SERVICE. Afficher la table Service.
3- Insérer les 3 premiers tuples de la table OPERATION puis afficher la table SERVICE pour vérifier que NbOp a été correctement mis à jour.
4- Insérer les autres tuples de la table OPERATION puis Afficher la table SERVICE.
5- Supprimer le dernier tuple de la table OPERATION puis afficher la table SERVICE pour vérifier que NbOp a été correctement mis à jour.
6- Définir un trigger **MAJ_NbEmp** qui permet à chaque insertion/supression dans la table Employe, d'incrémenter/décrémenter NbEmp dans la table Service. Afficher la table Service.
7- Insérer tous les tuples de la table Employe puis afficher la table Service.
8- Supprimer l'employé N° 14 puis afficher la table Service.
9- Ajouter dans la table OPERATION l'information sur le nombre de membres (NbM) participant à la réalisation d'une opération (avec la valeur par défaut 0).

-Créer la table Membre, en définissant la clé primaire, **sans insérer les tuples**.

10- Définir un trigger qui permet à chaque insertion/suppression d'un tuple dans Membre, d'incrémenter/ décrémenter NbM dans OPERATION.
11- Insérer les tuples de Membre puis afficher la table Opération.
12- Supprimer de Membre le tuple (5, OP002) **[(3, OP002)]** puis afficher Opération (OP002) pour vérifier la màj de NbE.
13- On veut garder l'information nbchef : le nombre de fois qu'un employé a été chef d'opérations.
   a) Donner la commande qui permet l'ajout de cette information.
   b) Remarquons que la colonne chef dans opération est remplie. Donner la commande permettant d'initialiser nbchef par le nombre d'op dont l'employé a été chef.
   c) Afficher la table Employe.
14- La mise à jour de nbchef se fera maintenant, à travers le trigger MAJ_Nbchef.
   a) Quel est le ou les évènements du trigger ?
   b) Donner la commande Création du trigger.
   c) Mettre à jour la table Operation selon les tuples suivants :

CodeOP	Chef
OP003	3
OP005	8
OP006	2
OP007	4

OP010	36	3	04/01/2021	200000.00	S03

   d) Afficher Employé.

Manuel TP : Les déclencheurs (TRIGGERS)

Valeur par défaut : On peut définir des valeurs par défaut lors de la création de table ou la modification de sa structure :

CREATE TABLE nomTable (.... nomAttribut typeAtt **DEFAULT** valeur .....) ;
ALTER TABLE nomTable ADD (nomattribut [typeatt] **DEFAULT** valeur) ;

Les déclencheurs

a) Syntaxe de création de trigger dans oracle :

```
CREATE [OR REPLACE] TRIGGER nom-trigger
   {BEFORE | AFTER}
   {DELETE | INSERT | UPDATE [OF column ...]}
   [OR {DELETE | INSERT | UPDATE [OF column ...]}]..
   ON nom-table
   [ [REFERENCING {OLD [AS] old [NEW [AS] new]}]
   [ FOR EACH ROW]
    [WHEN (condition)]]

 PL/SQL_block ;
/
```

Où :
OR REPLACE : modifier un trigger existant.
nom-trigger : nom du trigger créé.
BEFORE : le trigger est déclenché avant d'exécuter l'événement.
AFTER : le trigger est déclenché après l'exécution de l'événement.
DELETE, INSERT, UPDATE...OF : événement déclenchant le trigger
ON nom-table : nom de la table où s'est produit l'événement.
REFERENCING: pour renommer OLD et NEW
FOR EACH ROW : le trigger est exécuté autant de fois qu'il y a d'enregistrements touchés par l'événement (**row trigger**). Si cette clause est omise alors une seule exécution sera réalisée pour tout le groupe d'enregistrements concernés (**statement trigger**).
WHEN : spécifie une condition qui doit être vérifiée pour déclencher le trigger (valable uniquement pour un row trigger. Oracle évalue la condition pour chaque ligne touchée par le trigger)
PL/SQL_block : Bloc PL/SQL que Oracle exécute pour le trigger.

b) Trigger avec plusieurs évènements :

```
CREATE OR REPLACE TRIGGER      nom-trigger
{BEFORE | AFTER} INSERT   OR   DELETE   OR   UPDATE OF nomAttribut
ON   nom-table
FOR EACH ROW
BEGIN
        IF INSERTING   THEN   .....instructions.....   END IF;
        IF DELETING   THEN    ..... instructions.....   END IF;
        IF UPDATING   THEN    ..... instructions.....   END IF;
END;
/
```