TP 5: Les Triggers SQL/PSM

Exercice 1 :

Soit le modèle relationnel suivant :

Employé (**Matricule**, Nom_Emp, Prénom_Emp, DateNaissance_Emp, Salaire_Emp, *#Num_Dep,* Total_vente)

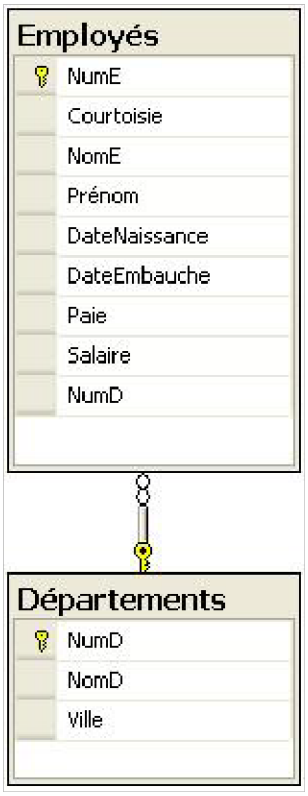
Département (**Num_Dep**, Nom_Dep, Ville_Dep)

Vente (**Num_Vente**, *#Matricule,* Date_Vente, Montant)

1. Créez un trigger "BEFORE INSERT" sur la table "Employé". Le trigger doit s'assurer que le salaire de chaque employé est supérieur ou égal à 1000 lors de l'insertion. Si le salaire est inférieur à 1000, le trigger doit générer une erreur et empêcher l'insertion.
2. Créez un trigger "BEFORE INSERT" sur la table "Employé" pour vous assurer que la date de naissance de chaque nouvel employé est telle que son âge est supérieur ou égal à 18 ans. Si l'âge est inférieur à 18 ans, le trigger doit empêcher l'insertion.
3. Créez un trigger "BEFORE DELETE" sur la table "Employé". Le trigger doit empêcher la suppression d'un employé si cet employé a effectué des ventes (c'est-à-dire, si des enregistrements existent dans la table "Vente" pour cet employé). Si des ventes sont associées à l'employé, le trigger doit générer une erreur et empêcher la suppression.
4. Créez un trigger "BEFORE UPDATE" sur la table "Département" pour empêcher la modification de l'ID du département. Si une tentative est faite pour mettre à jour l'ID du département, le trigger doit générer une erreur et empêcher la mise à jour.
5. Créez un trigger "AFTER INSERT" sur la table "Vente". Le trigger devrait mettre à jour le champ "Montant" dans la table "Employé" en ajoutant le montant de la nouvelle vente à son total de ventes.
6. Créez un trigger "AFTER INSERT" sur la table "Vente" qui enregistre chaque nouvelle vente dans une table de journal des ventes. La table de journal des ventes doit inclure les informations suivantes : ID de la vente, ID de l'employé (Matricule), date de la vente et montant de la vente.

Exercice 2 :

Soit le schéma relationnel suivant :



1. Créez un trigger qui annule toute mise à jour de la table Employés effectuée durant un jour non ouvrable, pour simplifier on se contente juste du weekend.
2. Créez un trigger qui vérifie que le nom de département est unique et non NULL lors de l'insertion d'une ligne dans la table Département.
3. Créer un trigger qui vérifie que le salaire des employés dont le mode de paie est 'Par heure' est supérieure à 75 et à 10000 pour les autres, le trigger se déclenche à la suite d'une opération d'insertion ou de mise à jour d'une ligne dans la table Employés.
4. Créez un trigger qui garde les traces des changements des salaires des employé.
5. Ajoutez à la table Départements le champ NbrEmployés, créez le trigger qui permet de maintenir cette colonne à la suite d'une opération d'insertion.
6. Implémentez, par les triggers, la suppression en cascade d'un département.

Exercice 3 :

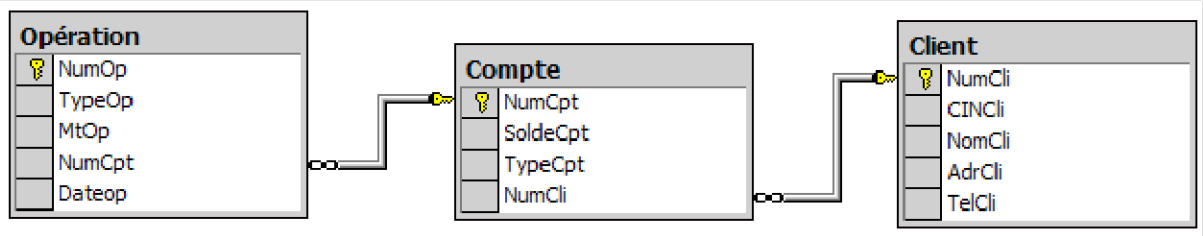
Soit le modèle relationnel suivant :

CLIENT (**Codeclt**, nom, prenom, tel)
PRODUIT (**Codeprd**, designation, prix, qte_dispo)
COMMANDE (**numero**, *#codeclt*, datecmd)
DETAIL (#numero, #codeprd, qte_cmd)

1. Créer la base de données et les tables en précisant les clés primaires et étrangères.
2. Créer un trigger qui interdit l'insertion d'un produit avec une quantité disponible négative en générant une exception avec le message 'Quantité non acceptée !'.
3. Créer un trigger qui interdit toute modification au niveau de la table DETAIL en générant une exception avec le message 'La table DETAIL ne peut être modifié !'.
4. Créer un trigger qui évite l'exception d'insertion au niveau de la table commande dans le cas où le numéro donné est déjà utilisé en incrémentant ce dernier jusqu'à l'obtention d'une nouvelle valeur.
5. Ajouter une colonne 'Montant' dans le table COMMANDE, puis créer un trigger qui, après insertion dans la table DETAIL :
   - Met à jour le montant de la commande associée en calculant et modifiant le total.
   - Met à jour la quantité disponible du produit concerné en retirant la quantité commandée. (Attention si la quantité disponible n'est pas suffisante, une exception est lancée et toute l'opération sera annulée !).
6. Créer une table Historique_Client(numero(Auto), Codeclt, ancien_tel,nouv_tel, date_changement) , puis créer un trigger qui à chaque changement du numéro du téléphone de client insère une ligne dans la table Historique_Client contenant le code du client, l'ancien téléphone, le nouveau téléphone et la date de changement.
7. Créer un trigger à la suppression d'une ligne de la table DETAIL permettant de remettre la quantité disponible du produit à sa valeur initiale (avant d'être commandée) et de recalculer et mettre à jour le montant de la commande.

Exercice 4 :

Soit la base de données suivante :



Les opérations consistent en des opérations de retrait ou de dépôt d'argent (TypeOp=D si le client a déposé de l'argent sur son compte et TypeOp=R si le client a retiré de l'argent sur son compte)

Un client ne peut avoir qu'un seul compte courant (TypeCpt="CC") et qu'un seul compte sur carnet (TypeCpt="CN")

Le numéro d'opération est automatique

La date de l'opération prend par défaut la date du jour

Créer les triggers suivants :

1. Qui à l'ajout ou à la modification de clients dans la table client vérifie si les Numéros de CIN saisies n'existent pas pour d'autres clients
2. Qui à la création de comptes, vérifie si :
   - Les soldes sont supérieurs à 1500 DH ;
   - Les types de compte sont CC ou CN et aucune autre valeur n'est acceptée ;
   - Les clients n'ont pas déjà des comptes du même type.
3. Qui interdit la suppression de comptes dont le solde est > 0 ou de comptes pour lesquels la dernière opération date de moins de 3 mois même s'ils sont vides (solde=0).
4. Qui :
   - Interdit la modification des numéros de comptes ;
   - Interdit la modification du solde de comptes auxquels sont associées des opérations
   - Ne permet pas de faire passer des comptes sur carnet en comptes courants, le contraire étant possible ;
   - A la modification de numéros de clients vérifie si les nouveaux clients n'ont pas de comptes associés du même type.