Bases de données Avancées
TP n° 6 : Les triggers

I) Création des tables

Les exercices portent sur une table qui contient des comptes de banque et leurs soldes.

```
CREATE TABLE comptes (
     numcompte integer PRIMARY KEY,
     nomclient varchar(20) NOT NULL,
     solde numeric(10,2) NOT NULL,
     decouvert_autorise numeric(10,2) NOT NULL DEFAULT 0
          CHECK (decouvert_autorise <=0)
);
```

Le but du TP est de mettre en place des contraintes d'intégrité sur les données et un système d'audit sur les transactions faites sur les comptes.
Créez la table ci-dessus et peuplez-la de quelques lignes de données.

II) Mise en place des contraintes d'intégrité

Pour chacune des contraintes suivantes, déterminez s'il est possible de l'implémenter avec une contrainte sur la table, ou s'il est nécesaire de créer un trigger. Implémentez la solution selon le cas.

1. Le solde d'un compte ne doit jamais être inférieur au découvert autorisé.
2. Si le solde d'un compte devient négatif, l'utilisateur doit être averti.
3. Un compte ne peut être fermé (supprimé de la table) que si le solde est 0.

III) Mise en place de règles de gestion

La banque a décidé d'appliquer des frais de 5% à toutes les transactions de retrait. Implémentez un trigger qui met en place cette nouvelle règle de gestion. Pour vérifier que le nouveau trigger et les triggers implémentés à la section précédente se déclenchent dans le bon ordre, testez les cas de figure suivants. Modifiez au besoin pour obtenir les bons comportements.

1. Un utilisateur a un solde 100 Euros, un découvert autorisé de 100 Euros, et effectue un retrait de 100 Euros de son compte. Son compte doit être débité de 105 Euros, ce qui rend son solde négatif et doit déclencher un avertissement.
2. Un utilisateur a un solde de 2 Euros, un découvert autorisé de 100 Euros, et effectue un retrait de 100 Euros. Son compte doit être débité de 105 Euros, ce qui ferait passer son solde sous le découvert autorisé. Le retrait doit être refusé.

IV) Mise en place d'une table audit

La banque doit maintenir une trace de toutes les opérations portées sur les comptes bancaires. La table ci-dessous maintient l'historique des transactions.

```
CREATE TABLE audit (
     numoperation SERIAL PRIMARY KEY,
     numcompte INTEGER NOT NULL REFERENCES comptes,
     date_operation DATE NOT NULL,
     operation VARCHAR(10) NOT NULL
          CHECK (operation in ('RETRAIT', 'OUVERTURE', 'DEPOT', 'FERMETURE')),
     montant numeric(10,2)
);
```

Écrivez un trigger qui peuple la table `audit` à chaque transaction.
Prenez soin de faire en sorte que seules les opérations qui ont été complétées apparaissent dans la table d'audit. Par exemple, si un utilisateur tente de retirer un montant et que le solde devient inférieur au découvert autorisé, l'opération est refusée et elle ne devra donc pas apparaître dans la table `audit`.
Que se passe-t-il maintenant si on tente de fermer un compte ? Dans quel ordre les triggers sont-ils exécutés ? Si vous rencontrez d'autres problèmes, proposez des solutions et implémentez-les.

V) Une règle de gestion plus complexe

La banque met en place un plafond de 1000 Euro sur les retraits qui ont lieu sur une journée. Sans modifier le schéma de la base, écrivez un trigger qui interdit les retraits qui violeraient cette contrainte. Vérifiez que les triggers se déclenchent dans le bon ordre en testant les cas suivants.

1. Un client a un solde de 1000 Euros, un découvert autorisé de 100 Euros. Il effectue deux retraits de 500 Euros, donc il est débité deux fois de 525 Euros. La seconde transaction devrait être refusée pour dépassement du plafond quotidien de retrait.
2. Un client a un solde de 1000 Euros, un découvert autorisé de 10 Euros. Il effectue deux retraits de 500 Euros, donc il est débité deux fois de 525 Euros. La transaction est-elle bloquée pour raison de découvert ou de plafond de retrait ? Expliquez en déterminant l'ordre dans lequel les triggers sont déclenchés.