Rapport : Triggers (Gestion des comptes bancaires)

Module : Base de Données MySQL

Nom: Aya ouazzani akkar hammoudi

SOMMAIRE :

1. Le script SQL complet :

1)Création de la base et des tables :

Création de la base

```sql
CREATE DATABASE IF NOT EXISTS banque_horizon;
USE banque_horizon;
```

Création des tables

```sql
-- Table clients
CREATE TABLE clients (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50),
    prenom VARCHAR(50)
);

-- Table comptes
CREATE TABLE comptes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    client_id INT,
    type_compte ENUM('Courant','Épargne','Entreprise'),
    solde DECIMAL(10,2),
    date_creation DATE DEFAULT (CURRENT_DATE),
    FOREIGN KEY (client_id) REFERENCES clients(id)
);
```

```sql
-- Table pour logs des erreurs (optionnel pour aller plus loin)
CREATE TABLE logs_erreurs (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    message VARCHAR(255),
    date_log TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2)Insertion des données initiales :

```sql
INSERT INTO clients (nom, prenom)
VALUES ('Dupont', 'Alice'),
       ('Martin', 'Bob'),
       ('Leroy', 'Claire');

-- Comptes existants
INSERT INTO comptes (client_id, type_compte, solde)
VALUES (1, 'Courant', 1200.50),
       (2, 'Épargne', 5000.00);

```

3)Création du Trigger pour validation :

```sql
DELIMITER $$

CREATE TRIGGER avant_insertion_compte
BEFORE INSERT ON comptes
FOR EACH ROW
BEGIN
    -- Vérification si le client a déjà un compte courant
    IF (SELECT COUNT(*) FROM comptes
        WHERE client_id = NEW.client_id
        AND type_compte = 'Courant') > 0 THEN
        -- Bloquer l'opération et afficher un message
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erreur : ce client possède déjà un compte courant.';
    END IF;
END $$

DELIMITER ;
```

4) Phase de test :

Cas 1 : Insertion valide

```sql
INSERT INTO comptes (client_id, type_compte, solde)
VALUES (3, 'Courant', 800.00);
```

Cas 2 : Insertion refusée

```sql
INSERT INTO comptes (client_id, type_compte, solde)
VALUES (1, 'Courant', 2500.00);
```

Cas 3 : Insertion d'un autre type de compte

```sql
INSERT INTO comptes (client_id, type_compte, solde)
VALUES (1, 'Épargne', 1500.00);
```

5) Vérification du Trigger:

```sql
SELECT TRIGGER_NAME, EVENT_MANIPULATION, EVENT_OBJECT_TABLE, ACTION_TIMING
FROM information_schema.TRIGGERS
WHERE TRIGGER_SCHEMA = 'banque_horizon';
```

6. Pour aller plus loin:

Trigger pour enregistrer les erreurs dans logs_erreurs au lieu de SIGNAL

Création de la table logs_erreurs :

```sql
CREATE TABLE IF NOT EXISTS logs_erreurs (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    message VARCHAR(255),
    date_log TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Trigger BEFORE INSERT pour compte courant

```sql
DELIMITER $$

CREATE TRIGGER avant_insertion_compte_log
BEFORE INSERT ON comptes
FOR EACH ROW
BEGIN
    IF NEW.type_compte = 'Courant' AND
       (SELECT COUNT(*) FROM comptes
        WHERE client_id = NEW.client_id
        AND type_compte = 'Courant') > 0
    THEN
        INSERT INTO logs_erreurs(message)
        VALUES (CONCAT('Erreur : client ', NEW.client_id, ' possède déjà un compte courant.'));
    END IF;
END $$

DELIMITER ;
```

Trigger AFTER DELETE pour archiver les comptes supprimés:

Création de la table archives_comptes :

```sql
CREATE TABLE IF NOT EXISTS archives_comptes (
    id_archive INT AUTO_INCREMENT PRIMARY KEY,
    client_id INT,
    type_compte ENUM('Courant','Épargne','Entreprise'),
    solde DECIMAL(10,2),
    date_suppression TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Trigger AFTER DELETE:

```sql
DELIMITER $$

CREATE TRIGGER apres_suppression_compte
AFTER DELETE ON comptes
FOR EACH ROW
BEGIN
    INSERT INTO archives_comptes(client_id, type_compte, solde)
    VALUES (OLD.client_id, OLD.type_compte, OLD.solde);
END $$

DELIMITER ;
```

La liste des triggers dans information_schema:

```sql
SELECT
    TRIGGER_NAME,
    EVENT_MANIPULATION,
    EVENT_OBJECT_TABLE,
    ACTION_TIMING,
    ACTION_STATEMENT
FROM information_schema.TRIGGERS
WHERE TRIGGER_SCHEMA = 'banque_horizon';
```

2. Captures d'écran :

Création du Trigger pour validation :

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0266 seconde(s).)

```sql
CREATE TRIGGER avant_insertion_compte BEFORE INSERT ON comptes FOR EACH ROW BEGIN -- Vérification si le client a déjà un compte courant
IF (SELECT COUNT(*) FROM comptes WHERE client_id = NEW.client_id AND type_compte = 'Courant') > 0 THEN -- Bloquer l'opération et afficher
un message SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Erreur : ce client possède déjà un compte courant.'; END IF; END;
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

4) Phase de test :

Cas 1 : Insertion valide

1 ligne insérée.
Identifiant de la ligne insérée : 3 (traitement en 0,0006 seconde(s).)

```sql
INSERT INTO comptes (client_id, type_compte, solde) VALUES (3, 'Courant', 800.00);
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0003 seconde(s).)

```sql
-- Résultat attendu : ligne insérée avec succès;
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

Cas 2 : Insertion refusée

Erreur

Requête SQL : Copier

```
INSERT INTO comptes (client_id, type_compte, solde)
VALUES (1, 'Courant', 2500.00);
```

MySQL a répondu :

```
#1644 - Erreur : ce client possède déjà un compte courant.
```

Cas 3 : Insertion d'un autre type de compte

1 ligne insérée.
Identifiant de la ligne insérée : 4 (traitement en 0,0012 seconde(s).)

```
INSERT INTO comptes (client_id, type_compte, solde) VALUES (1, 'Épargne', 1500.00);
```

[Éditer en ligne] [Éditer] [Créer le code source PHP]

La liste des triggers dans information_schema:

Affichage des lignes 0 - 2 (total de 3, traitement en 0,0354 seconde(s).)

```
SELECT TRIGGER_NAME, EVENT_MANIPULATION, EVENT_OBJECT_TABLE, ACTION_TIMING, ACTION_STATEMENT FROM information_schema.TRIGGERS WHERE TRIGGER_SCHEMA = 'banque_horizon';
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

TRIGGER_NAME	EVENT_MANIPULATION	EVENT_OBJECT_TABLE	ACTION_TIMING	ACTION_STATEMENT
avant_insertion_compte	INSERT	comptes	BEFORE	BEGIN -- Vérifie uniquement si le nouveau com...
avant_insertion_compte_log	INSERT	comptes	BEFORE	BEGIN IF NEW .type_compte
apres_suppression_compte	DELETE	comptes	AFTER	INTO archives_comptes (client_id ,...

3. Une courte explication écrite :

Quelle est la différence entre BEFORE et AFTER ?:

Le trigger BEFORE s'exécute avant l'opération (INSERT, UPDATE, DELETE) et sert à vérifier ou modifier les données avant leur enregistrement.

Le trigger AFTER s'exécute après l'opération et permet souvent d'enregistrer une action dans une autre table (comme une table historique).

Pourquoi utiliser SIGNAL ?

La commande SIGNAL permet de générer une erreur personnalisée afin d'interrompre une opération qui viole une règle métier (ex. : un client ne peut pas avoir deux comptes courants).

Que se passe-t-il si le trigger contient une erreur de logique ?

Si le trigger contient une erreur logique, l'exécution s'arrête immédiatement et l'opération principale est annulée, empêchant ainsi toute modification incorrecte dans la base.