PROJET RÉALISÉ PAR JETHRO ROMÉLUS

<< Conception et implémentation d'une base de données en language SQL permettant la gestion des comptes bancaires et des transactions>>

Introduction

Dans un contexte où la gestion efficace des comptes bancaires et des transactions est essentielle pour assurer la sécurité et la fluidité des opérations financières, la mise en place d'une base de données robuste et bien structurée devient une nécessité. Ce projet vise à concevoir et implémenter une base de données relationnelle en langage SQL permettant de centraliser et d'optimiser la gestion des comptes bancaires, des clients et des transactions associées.

Ce document détaillera les principales étapes d'implémentation, incluant le Modèle Conceptuel des Données (MCD), qui définira les entités clés, les clés primaires et étrangères, les cardinalités et leurs relations, ainsi que le Modèle Physique des Données (MPD), qui traduira cette structure en tables SQL avec leurs contraintes et types de données. De plus, des requêtes SQL seront réalisées pour implémenter certaines fonctionnalités essentielles, telles que la gestion des transactions, la mise à jour des soldes et la sécurisation des opérations pour ne citer que ceux-là.

Modèle Conceptuel des données

Le Modèle Conceptuel des Données est présenté dans la figure se trouvant après les explications afin de faciliter une meilleure compréhension du travail. **À noter que les clés étrangères sont représentées par des cercles en pointillés.** Les principaux éléments du modèle incluent :

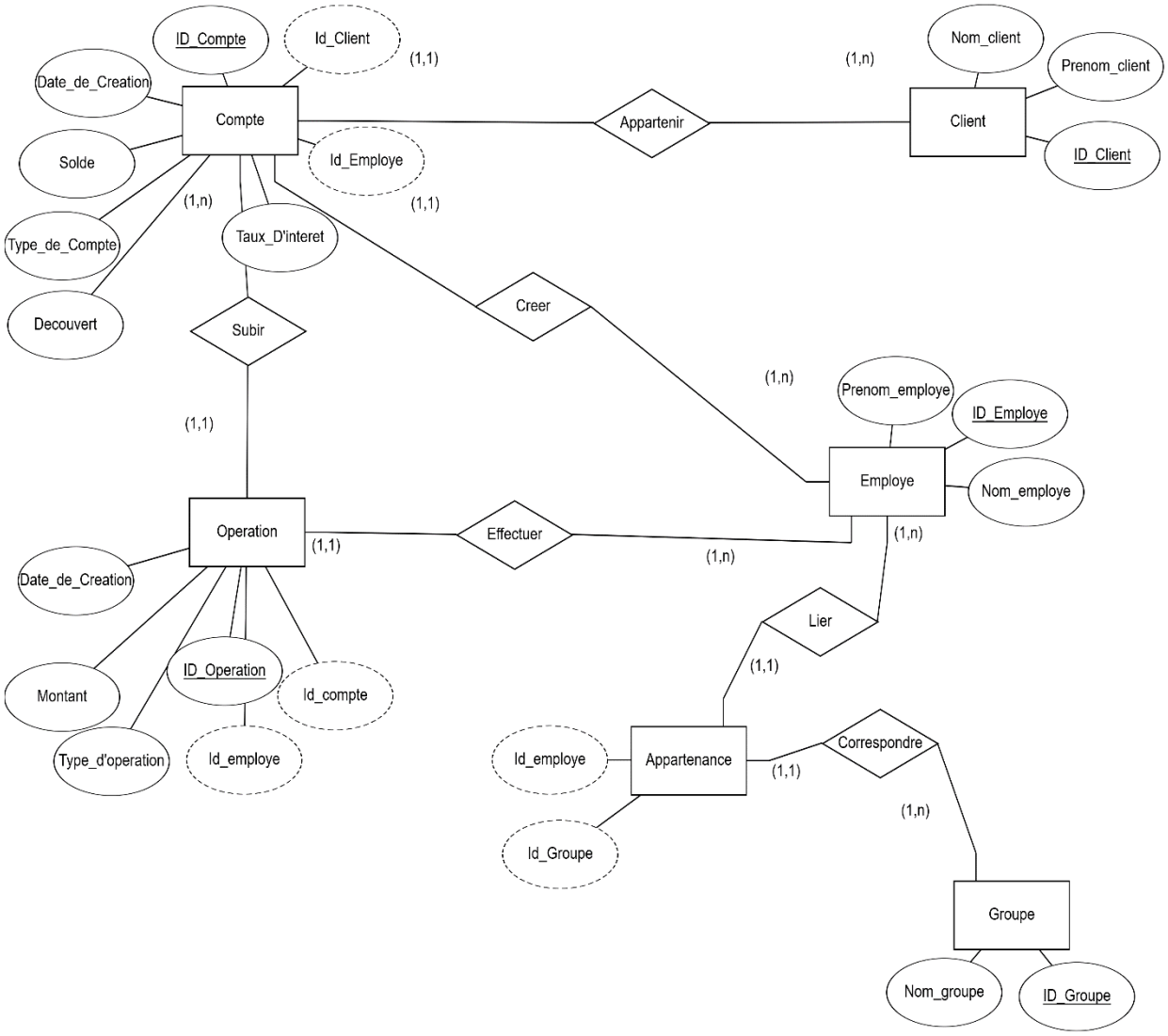
Entités et attributs:

- **Compte** : id_Compte, Date_de_Creation, Solde, Type_de_Compte, Découvert, Taux_D'Intérêt, id_employe (clé étrangère) et id_client (clé étrangère)
- **Client** : id_Client, Nom_client, Prénom_client.
- **Employé** : id_Employe, Nom_employe, Prénom_employe.
- **Opération** : id_Operation, Date_de_Creation, Montant, Type_d'operation, id_compte (clé étrangère) et id_employe (clé étrangère)
- **Groupe** : id_Groupe, Nom_groupe.
- **Appartenance** : id_employe (clé étrangère), id_groupe (clé étrangère)

Relations Principales:

- **Appartenir** : Un client (1, n) peut avoir plusieurs comptes, chaque compte appartient à un seul client (1,1).
- **Créer** : Un employé (1, n) peut créer plusieurs comptes, chaque compte étant créé par un seul employé (1,1).
- **Subir** : Un compte (1, n) peut subir plusieurs opérations, une opération est liée à un compte (1,1).
- **Effectuer** : Une opération (1,1) est effectuée par un employé mais un employé peut effectuer plusieurs opérations (1, n).
- **Lier** : Un employé (1, n) peut être lié à plusieurs appartenances, chaque appartenance concerne un employé (1,1).

- **Correspondre** : Une appartenance (1,1) est associée à un groupe mais un groupe peut-être lier à plusieurs appartenance (1, n).



Modèle physique des données

Dans un premier temps, nous allons créer la base de données avec le mot clé **Create database**, ainsi que les différentes tables associées, **en commençant par les tables périphériques, c'est-à-dire celles ne possédant aucune clé étrangère**. Voici les scripts correspondants :

Script de création de la base de données

Create database gestion_des_comptes_bancaires

Script permettant de créer les différentes tables :

```
-- Création de la table employée
Create table EMPLOYE
(
id_employe integer primary key,
nom_employe varchar(128),
prenom_employe varchar(128)
);
-- Création de la table client
Create table Client
(
id_client integer primary key,
nom_client varchar(128),
prenom_client varchar(128)
);
-- Création de la table groupe
Create table groupe
(
id_groupe integer primary key,
nom_groupe varchar(128)
);
-- Création de la table compte
 (
   id_compte INTEGER PRIMARY KEY,
   date_de_creation DATE NOT NULL,
   solde DECIMAL(15,2) DEFAULT 0 CHECK (solde >= 0), -- Restriction pour éviter un solde négatif
   type_de_compte ENUM('Epargne', 'Courant') NOT NULL,
```

```
    taux_dinteret DECIMAL(3,2) DEFAULT 0,
    decouvert DECIMAL(15,2) DEFAULT 0,
    id_client INTEGER REFERENCES client(id_client),
    id_employe INTEGER REFERENCES employe(id_employe)
);
-- Création de la table Opération
Create table operation
(
id_operation integer primary key,
date_de_creation date not null,
type_d_operation enum('depot', 'retrait', 'virement') not null,
id_compte integer
references compte (id_compte),
id_employe integer
references employe(id_employe),
montant decimal(15,2)
);
-- Création de la table Appartenance
CREATE TABLE Appartenance
(
id_employe integer,
id_groupe integer,
    PRIMARY KEY (ID_Employe, ID_Groupe),
    FOREIGN KEY (ID_Employe) REFERENCES Employe(ID_Employe),
    FOREIGN KEY (ID_Groupe) REFERENCES Groupe(ID_Groupe)
);
```

- Créer des requêtes d'insertion répondant aux fonctionnalités 1 à 4, et 9, c'est-à-dire ajouter dans la base un client, un employé, un groupe, un employé à un groupe, un compte, une opération, créer des comptes bancaires, effectuer un dépôt d'un montant dans un compte, effectuer un retrait d'un montant dans un compte et ajouter les intérêts pour les comptes épargnes à la fin du mois.

-- Ajouter dans la base un client, un employé, un groupe, un employé à un groupe, un compte

```
INSERT INTO Client (Nom_client, prenom_client, id_client)

VALUES ('ROMELUS','Jethro', 123);

INSERT INTO Employe (Nom_employe, prenom_employe, id_employe)

VALUES ('PAUL', 'Martin', 456);

INSERT INTO Groupe (Nom_groupe, id_groupe)

VALUES ('Service Client', 1);

INSERT INTO Appartenance (ID_Employe, ID_Groupe)

VALUES (456, 1);

INSERT INTO Compte (id_compte, Solde, Type_de_Compte, Taux_Dinteret, Decouvert,
Date_de_Creation, ID_Client, ID_Employe)

VALUES (15, 5000.00, 'Epargne', 2.50, 0.00, '2025-02-12', 123, 456);
```

-- Effectuer une opération de dépôt sur un compte

```
INSERT INTO Operation (id_operation, Type_d_operation, Montant, Date_de_Creation,
ID_Compte, ID_Employe)

VALUES (61, 'depot', 1000.00, '2025-02-12', 15, 456);

-- Mettre à jour le solde du compte concerné

UPDATE Compte

SET Solde = Solde + 1000.00

WHERE ID_Compte = 15;

-- verifier le solde mis à jour

select * from compte
```

-- Créer des comptes bancaires

```
START TRANSACTION;
-- Ajouter les clients correspondants aux comptes
INSERT INTO Client (id_client, nom_client, prenom_client) VALUES
   (124, 'DUPONT', 'Jean'),
   (125, 'MARTIN', 'Sophie'),
   (126, 'LAMBERT', 'Luc'),
   (127, 'BERTRAND', 'Alice'),
   (128, 'GIRAUD', 'Pierre');
-- Ajouter les employés associés à la création des comptes
INSERT INTO Employe (id_employe, nom_employe, prenom_employe) VALUES
   (457, 'ROBERT', 'Michel'),
   (458, 'LEGRAND', 'Nathalie'),
   (459, 'GONZALEZ', 'Patrick'),
   (460, 'FERNAND', 'Elodie'),
   (461, 'MOREAU', 'Thomas');
-- Ajouter les comptes bancaires
INSERT INTO Compte (id_compte, solde, type_de_compte, taux_dinteret, decouvert,
date_de_creation, id_client, id_employe) VALUES
   (16, 8000.00, 'Epargne', 3.00, 0.00, '2025-02-12', 124, 457),
   (17, 12000.50, 'Courant', 0.00, 500.00, '2025-02-12', 125, 458),
   (18, 1500.75, 'Epargne', 2.75, 0.00, '2025-02-12', 126, 459),
   (19, 10000.00, 'Courant', 0.00, 1000.00, '2025-02-12', 127, 460),
   (20, 25000.30, 'Epargne', 3.50, 0.00, '2025-02-12', 128, 461);
COMMIT;
```

-- Effectuer le dépôt d’un montant dans un compte

-- Insérer l'opération de dépôt dans la table Opération

```
INSERT INTO Operation (ID_Operation, Type_d_operation, Montant, Date_de_Creation,
ID_Compte, ID_Employe)

VALUES (1002, 'Depot', 3000.00, '2025-02-12', 15, 456);

-- Mettre à jour le solde du compte

UPDATE Compte

SET Solde = Solde + 3000.00

WHERE ID_Compte = 15;

-- vérifier que la compte est mis à jour

select * from compte
```

-- EFFECTUER UN RETRAIT DANS UN COMPTE

```
-- 1. Vérifier que le compte a suffisamment de fonds pour le retrait

SELECT Solde, Decouvert FROM Compte WHERE ID_Compte = 15;

-- 2. Insérer l'opération de retrait dans la table Opération

INSERT INTO Operation (ID_Operation, Type_d_operation, Montant, Date_de_Creation,
ID_Compte, ID_Employe)

VALUES (1005, 'Retrait', 9000.00, '2025-02-12', 15, 456);

-- 3. Mettre à jour le solde du compte

UPDATE Compte

SET Solde = Solde - 9000.00

WHERE ID_Compte = 15;

Select * from compte
```

-- Ajouter les intérêts pour les comptes épargnes à la fin du mois

```
START TRANSACTION;

-- Création d'une table temporaire pour stocker le solde moyen des comptes épargne

CREATE TEMPORARY TABLE Temp_Solde_Moyen AS

SELECT

    c.id_compte,
```

```
    CAST(((c.Solde + COALESCE(o.Montant, 0)) / 2) AS DECIMAL(15,2)) AS Solde_Moyen,
    c.Taux_Dinteret
FROM Compte c
LEFT JOIN (
    SELECT id_compte, SUM(Montant) AS Montant
    FROM Operation
    WHERE Date_de_Creation BETWEEN DATE_SUB(LAST_DAY(CURRENT_DATE), INTERVAL 1 MONTH) + INTERVAL 1 DAY
                          AND LAST_DAY(CURRENT_DATE)
    GROUP BY id_compte
) o ON c.id_compte = o.id_compte
WHERE c.Type_de_Compte = 'Epargne';

-- Mise à jour des soldes avec les intérêts calculés
UPDATE Compte c
JOIN Temp_Solde_Moyen t ON c.id_compte = t.id_compte
SET c.Solde = CAST(c.Solde + (t.Solde_Moyen * t.Taux_Dinteret / 12) AS DECIMAL(15,2));

-- Enregistrement des intérêts dans la table Opération
INSERT INTO Operation (id_operation, Type_d_operation, Montant, Date_de_Creation, ID_Compte, ID_Employe)
SELECT
    (SELECT COALESCE(MAX(id_operation), 0) + ROW_NUMBER() OVER() FROM Operation) AS id_operation,
    'depot',
    CAST((Solde_Moyen * Taux_Dinteret / 12) AS DECIMAL(15,2)) AS Montant,
    LAST_DAY(CURRENT_DATE),
    id_compte,
```

```
    NULL
FROM Temp_Solde_Moyen;


-- Suppression de la table temporaire
DROP TEMPORARY TABLE Temp_Solde_Moyen;
COMMIT;
```

-- Effectuer un virement d'un montant d'un compte vers un autre, c'est-à-dire, retirer le montant du compte et le verser vers l'autre compte

```
-- 1. Insérer l'opération de retrait depuis le compte source
INSERT INTO Operation (ID_Operation, Type_d_operation, Montant, Date_de_Creation, ID_Compte, ID_Employe)
VALUES (1006, 'Retrait', 1000.00, '2025-02-12', 15, 456);
UPDATE Compte
SET Solde = Solde - 1000.00
WHERE ID_Compte = 15 AND (Solde - 1000.00 >= -Decouvert);
-- 2. Insérer l'opération de dépôt sur le compte destinataire
INSERT INTO Operation (ID_Operation, Type_d_operation, Montant, Date_de_Creation, ID_Compte, ID_Employe)
VALUES (1006, 'Dépôt', 1000.00, '2025-02-12', 16, 456);
-- 3. Mettre à jour le compte
UPDATE Compte
SET Solde = Solde + 1000.00
WHERE ID_Compte = 16;
-- 4. Vérifier que le compte a été mis à jour
select * from compte
```

-- Créer des vues donnant les informations requises pour les fonctionnalités 6 et 7

-- 6. Pour consulter tous les employés

```
CREATE VIEW Vue_Employes AS

SELECT ID_Employe, Nom_Employe, Prenom_Employe

FROM Employe;

SELECT * FROM Vue_Employes;
```

-- 7. Pour consulter tous les groupes

```
CREATE VIEW Vue_Groupes AS

SELECT ID_Groupe, Nom_Groupe

FROM Groupe;

SELECT * FROM Vue_Groupes;
```

-- Créer une requête SELECT répondant aux fonctionnalités 8 et 10, c'est-à-dire Consulter les clients dont le nom contient la chaine de caractère « ean » et Consulter le total des montants des retraits.

```
SELECT

   C.ID_Client,

   C.Nom_Client,

   (SELECT SUM(O.Montant) FROM Operation O WHERE O.Type_d_Operation = 'Retrait') AS
Total_Retraits

FROM Client C

WHERE C.Nom_Client LIKE '%ean%';
```

- Écrire des requêtes SELECT répondant aux fonctionnalités 11 à 15 en fixant à chaque fois un identifiant de l'entité mentionnée à la fin de la phrase. Ex : pour le point 11, choisir un identifiant de compte et écrire la requête SELECT qui permettrait de trouver la liste des opérations pour le compte.

  -- 11. Consulter la liste des opérations effectuées sur le compte

```
  SELECT

  O.ID_Operation,

  O.Type_d_Operation,
```

```
    O.Montant,
    O.Date_de_Creation,
    O.ID_Compte,
    O.ID_Employe
FROM Operation O
WHERE O.ID_Compte = 15;

-- 12. Consulter le solde du compte
SELECT
    C.ID_Compte,
    C.Solde
FROM Compte C
WHERE C.ID_Compte = 15;
-- 13. Consulter les comptes d'un client
SELECT
    C.ID_Compte,
    C.Date_de_Creation,
    C.Solde,
    C.Type_de_Compte,
    C.Taux_Dinteret,
    C.Decouvert
FROM Compte C
WHERE C.ID_Client = 123;

-- 14. Consulter les comptes créés par un employé
SELECT
    C.ID_Compte,
    C.Date_de_Creation,
```

```
    C.Solde,
    C.Type_de_Compte,
    C.Taux_Dinteret,
    C.Decouvert
FROM Compte C
WHERE C.ID_Employe = 456;

-- 15. Consulter les employés d'un groupe
SELECT
    E.ID_Employe,
    E.Nom_Employe,
    E.Prenom_Employe
FROM Employe E
JOIN Appartenance A ON E.ID_Employe = A.ID_Employe
WHERE A.ID_Groupe = 1;

-- Ecrire des requêtes SELECT implémentant les fonctionnalités 16 et 17 :
-- 16 Consulter la liste des 5 comptes avec le plus grand solde

SELECT
    C.ID_Compte,
    C.Solde
FROM Compte C
ORDER BY C.Solde DESC
LIMIT 5;
-- 17 Consulter la liste des comptes qui ont plus de 10 transactions
SELECT
    C.ID_Compte,
```

```
    COUNT(O.ID_Operation) AS Nombre_Transactions
FROM Compte C
JOIN Operation O ON C.ID_Compte = O.ID_Compte
GROUP BY C.ID_Compte
HAVING COUNT(O.ID_Operation) > 10;
```

-- Ecrire une requête UPDATE implémentant la fonctionnalité 18 :

-- 18. Mettre à jour le nom d'un client

```
UPDATE Client
SET Nom_Client = 'JACQUES'
WHERE ID_Client = 123;
Select * from client
```

Conclusion

En somme, la conception et l'implémentation de cette base de données en langage SQL permet d'optimiser la gestion des comptes bancaires et des transactions en assurant une organisation structurée, sécurisée et efficace des informations. Grâce à un Modèle Conceptuel des Données (MCD) bien défini et à un Modèle Physique des Données (MPD) rigoureusement implémenté, le système garantit l'intégrité et la fiabilité des données.

Les requêtes SQL développées assurent l'exécution des principales fonctionnalités, notamment l'enregistrement des transactions, la mise à jour des soldes et le respect des contraintes de gestion bancaire. **Ce projet constitue ainsi une base solide pour le développement d'un système bancaire performant et évolutif, capable de répondre aux exigences des institutions financières modernes.**