Projet de bases de données

Table Des Matières

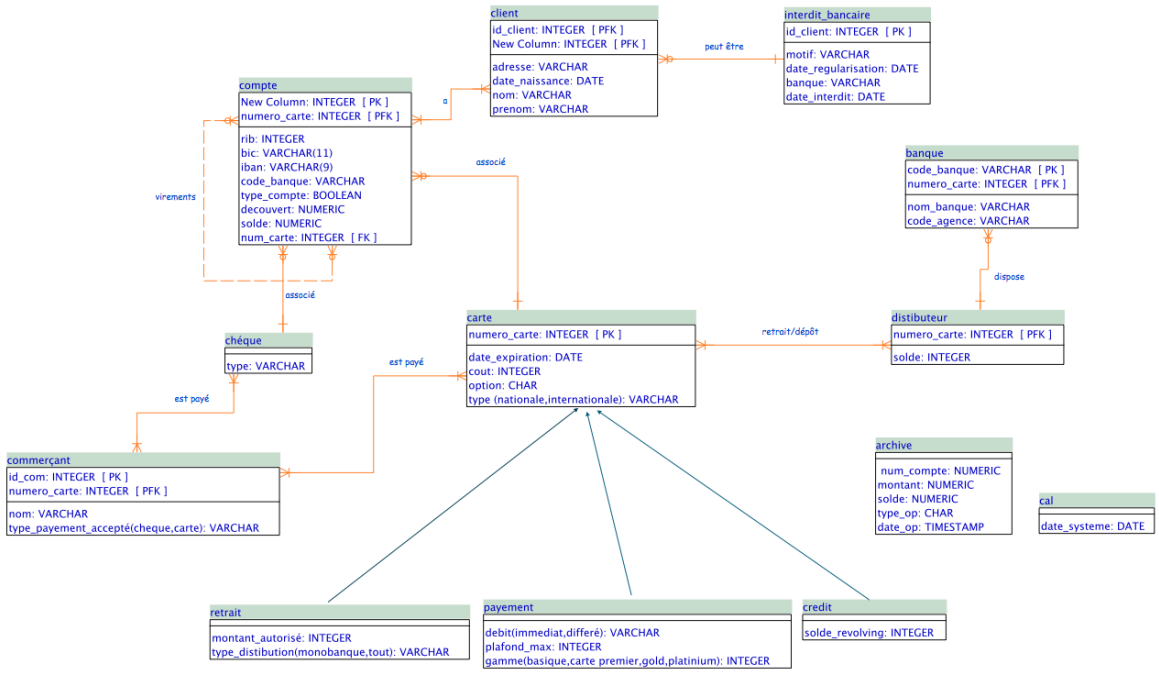
Présentation et lancement de l'application

Le but du projet est d'implémenter une application capable de gérer différentes fonctionnalités d'une banque (ouverture et fermeture d'un compte, virement unique et permanents..etc) que nous aborderons à la suite de notre rapport.

La création, le remplissage des tables ainsi que le lancement du logiciel se fera avec la commande :

```
$ ./compile.sh
```

Diagramme Entité / Association



-Modèle conceptuel de données-

Description des choix effectués

Dans un premier temps, nous avons étudié le sujet à deux, afin de cerner tous les aspects du projet. Nous avions plusieurs points de divergence, mais nous avons fini par coordonner la plupart de nos idées. Certains points restaient encore flous, mais nous avons pu établir une première modélisation, qui nous a semblé satisfaisante après quelques essais de requêtes.

Notation utilisé

Pour avoir une meilleure lecture du rapport, voici la notation qu'on a utilisé dans le projet :

- table : *t_nom_table*

- relation entre deux tables : *r_nom_relation*

- *vue : v_nom_vue*

- fonctions déclenchées par les triggers : *f_function()*

- triggers : *tg_nom_trigger*

- autres fonctions paramétrés ou auxiliaires.

Pour le MCD on a utilisé la notation Crow's foot avec le logiciel SQL Power Architect, ce dernier ne disposant pas de toutes les fonctionnalités pour la notation E/A notamment le nom des relations, héritages...etc, on a du le finir avec les moyens du bord.

SCHÉMA RELATIONNEL

T_BANQUE (code_banque,nom_banque,code_agence)

T_COMPTE (num_compte,rib,iban,bic,code_banque*,solde,decouvert,type_compte)

T_CLIENT (id_client,nom,prenom,adresse,date_naissance)

R_CLIENT_COMPTE(id_client*,num_compte*,mandataire,date_creation,date_fermeture)

R_VIREMENT(id_virement*,num_compte_creditaire*,iban_benifiaire,bic_benificiaire,type_virement, periode,date_virement)

T_INTERDIT_BANCAIRE(banque,id_client*,motif,date_interdit,date_regularisation)

T_CAL (date_cal)

T_ARCHIVE(num_compte,type_op,montant,solde,date_operation)

Note : *La dernière table (**t_archive**) nous sert à archiver une opération de retrait ou de dépôt pour garder une trace du solde, date..etc lors d'un **UPDATE** sur t_compte.*

CONTRAINTES D'INTÉGRITÉS

Check sur les tables

- **Table t_Client :** sur la date de naissance puisque l'age doit être > 18 ans pour ouvrir un compte

- **Relation r_virement** : -type virement doit être soit unique ou permanents.

-périodicité : soit mensuel, trimestriel...etc.

Triggers et fonctions

1. Ouverture d'un compte : La fonction *f_ouverture_compte()* retourne un ***TRIGGER*** sur la table ***t_client*** qui se déclenche après insertion d'un nouveau client, la fonction vérifie d'abord si le client n'est pas interdit bancaire, elle lui attribut ensuite un nouveau numéro de compte (calculé à partir de fonctions auxiliaires voir *fun_aux.sql*) avec un RIB,IBAN,..et toutes les modalités d'un compte bancaire.

2. Fermeture d'un compte : La fonction *f_fermeture_compte()* retourne un ***TRIGGER*** sur la table ***t_client*** qui se déclenche avant suppression du client, la fonction vérifie si le client est bien le mandataire du compte et s'il n'est pas interdit bancaire, dans ce cas il devrait d'abord régulariser sa situation.

3. Consultation d'un solde : La fonction *consult_solde(numeric)* retourne le solde du numéro de compte passé en paramètre.

4. Retrait/dépôt d'espèces : L'opération *op(numeric,numeric,char)* fait un ***UPDATE*** sur la table ***t_compte*** du numéro de compte ***$1*** pour retirer ou déposer ('R' ou 'D' ***$3***) de l'argent d'un montant ***$2***. La fonction vérifie d'abord si le client n'est pas interdit bancaire. Dans le cas d'un retrait 'R' (paramètre ***$3*** dans la fonction) la fonction vérifie si le client n'est pas débiteur, dans ce cas il ne peut pas retirer de l'argent.

On dispose aussi d'une fonction qui agit sur ce ***UPDATE*** sur la table ***t_compte***, qui a pour rôle de lancer une fonction ***f_maj_releve ()*** qui actualise une requête qui calcule le relevé mensuel de chaque compte.

5. Virement entre compte d'une banque :

> **Virement unique:** La fonction *f_virement()* retourne un ***TRIGGER*** sur la table ***r_virement*** qui est une relation entre comptes d'une banque qui se déclenche après insertion d'un nouveau virement unique (paramètre 'U' dans ***INSERT***). La fonction retourne une ***EXCEPTION*** si le virement est inférieur à 10€ ou, si le client n'est pas le mandataire du compte, ou si le client est interdit bancaire.

La deuxième étape de la fonction est de faire la transaction : créditer le compte bénéficiaire et débiter le client créditeur, on notera que le bloc de la transaction est entre ***BEGIN...EXCEPTION*** pour l'annuler en cas d'un incident technique.

En fin la fonction vérifie si le créditeur n'est pas le même que le mandataire et facture la transaction au créditeur selon le critère du virement.

Virement périodique : La fonction *f_virement_perm()* retourne un ***TRIGGER*** sur la table ***t_cal*** qui a pour attribut une date fictive qu'on peut manipuler à la place de la date du système, ainsi quand on fait un ***UPDATE*** sur cette table sa déclanche le ***TRIGGER*** qui exécute la procédure *f_virement_perm()*. Le virement périodique n'est autre chose qu'un virement unique qui se lance mensuellement, semestriellement,trimestriellement ou annuellement.

6. Interdiction bancaire :

- *is_interdit(numeric) :* La procédure vérifie si le numéro de compte donné en paramètre est oui ou non interdit bancaire,pour quel motif, pour combien de temps.

- *ajout_interdit (int,varchar):* Quand un client dépasse le découvert autorisé par sa banque il devient un interdit bancaire, cette procédure permet d'ajouter un client passé en ***$1*** dans les interdits bancaire pour le motif ***$2***.

- *f_del_interdit() :* La procédure retourne un ***TRIGGER*** sur la table ***t_interdit_bancaire*** qui se déclenche avant suppression d'un interdit bancaire, la procédure vérifie d'abord si la date de régularisation a été dépassé.

Contraintes d'intégrités pressenties (non implémentés)

- Pour le compte joint, le mandataire = titulaire du compte par défaut.

-Pour les retraits/dépôt d'une carte via un distributeur automatique, faire un ***TRIGGER*** sur la table distributeur, qui vérifie s'il dispose encore de billets ou pas.

- Dans la relation *est payé* entre Commerçant et Carte, spécifier si c'est une carte payement ou à crédit revolving.

- Dans la relation entre Compte et Carte ''*associé*'', penser à mettre une contrainte pour qu'aucune carte soit associé à un compte sans payement.

- Même chose pour la relation Chèque et Compte.

- L'attribut option dans la table Carte est par défaut ***NULL*** si elle n'est pas une carte à autorisation systématique.

- Pour la carte Retrait (qui hérite de Carte), penser à mettre un ***TRIGGER*** pour le débit différé, une fois par mois par exemple.

''Ce projet nous a beaucoup apporté, il nous a permis de nous familiariser avec le langage procédural PL/pgSQL. C'était ainsi l'occasion pour nous de concrétiser nos connaissances acquises tout au long du semestre''.