université PARIS DIDEROT PARIS 7

Master 1 Informatique
2013-2014

Projet Base de Données Avancé
BANQUES

CASTELLANOS Alejandro
HUAM Erica

Sommaire

I Le diagramme de classe, son passage aux tables

1 Les Comptes et Clients

La modélisation de notre base de données a été désignée à partir d'une table centrale nommée **Compte**, qui a comme attribut principal un identificateur *PK NbCompte*. Chaque compte peut avoir un ou plusieurs titulaires identifiés par le champ *Unique PK Id_Perso* dans la table **Personne**. Dans cette table, l'information pertinente aux clients des banques est enregistrée.

Les comptes peuvent aussi être de différents types, définis dans la table **Type_compte**. Si le compte choisi est de type « joint » et comprend deux titulaires, il faut faire la relation entre ce compte et la table C**omptes_joints** où sont stockés les informations supplémentaires pour ce type particulier. Le type de compte permet de connaître le genre et les frais liés au type choisi dans la banque donnée.

Un compte est aussi lié à une agence qui est elle-même temps liée à une banque spécifique avec la relation *FKAgence FOREIGN KEY (BanqueId_banque)*.

2 Les Cartes Bancaires, Chèques et Transactions

Les comptes donnent la possibilité aux titulaires d'avoir des cartes bancaires et chèques liés à eux-même grâce aux références vers *PK Nbcompte* dans les tables **Cheque** et **Carte_bancaire** respectivement. Il est également important de mentionner que les cartes bancaires peuvent avoir différents types qui sont prédéfinis dans la table **Type_carte**.

La fonctionnalité principale des comptes bancaires sont les transactions. Ces dernières couvrent tout type de « flux », c'est-à-dire de transfert d'argent. C'est la table **nature_trans** qui permettra de savoir quel genre de transfert nous effectuons. Un simple retrait par exemple, peut utiliser la table **Debits** pour l'opération. Un prélèvement pour sa part, mettra en relation la table des **Tiers**, reconnus dans le système bancaire pour leurs RIBs. Ce qui permet de connaître toutes les données bancaires nécessaires pour effectuer la transaction. Les virements peuvent être unitaires ou automatisés à des périodes définies grâce aux paramètres de *TIMESTAMP NOT NULL date_trans*, *date_effect* à la table **periodicite.**

Il est important de connaître la nature des opérations bancaires afin de bien effectuer les débits stockés dans la table **Debits** et référencés par les *FKdebits_compte FOREIGN KEY (Nbcompte) REFERENCES Compte (NbCompte)*. De manière opposée les fonds seront crédités et additionnés au solde du compte.

Cet ensemble de tables permettent les opérations compte à compte. Le choix du modèle est principalement dû à notre envie de pouvoir modéliser les flux d'argent qui ne sont pas toujours visibles par les clients. Par exemple les frais liés aux comptes, aux opérations payantes sont prises en charge. Notamment grâce au paramètre *int4 NOT NULL Actifs* de la table **Banque** qui permet des échanges entre les comptes clients et l'entité banque qui est presque considéré comme un compte particulier.

II Règles de gestion implémentées

1 Le compte à vue et ouverture d'un compte

Au moment de l'entrée en relation avec une banque, la première opération que nous effectuons est celle de l'ouverture d'un compte avec la fonction `creer_compte()`.

C'est le compte ordinaire que l'on utilise pour déposer notre argent gérer notre budget. La convention de compte dépend du type de compte. Ce dernier indique si notre compte fonctionne avec ou sans moyen de paiement (chéquier et/ou cartes bancaire). Par ailleurs, il donne lieu à un relevé de compte au moins une fois par mois.

Si les comptes sont rémunérés, ils auront des intérêts positifs annuellement. À titre d'indication, les taux utilisés sont les taux indiqué dans le sujet. Ils sont appliqué selon les revenus déclarés : *revenues_anuelles* stockés dans la table personne.

2 Fermeture d'un compte

La fermeture est faite à partir d'un dernier virement vers un compte tiers ou un retrait des espèces restants dans le compte. La fermeture utilise la fonction de `creer_transaction()` puis planifie la fermeture du compte en utilisant le paramètre *bool NOT NULL Valide* et le modifie à FALSE afin de le faire « disparaitre » des tables lors des opérations. La meilleure optimisation serait d'utiliser un trigger qui serait lancé à la fin de la transaction demandée.

3 Consultation du solde

La consultation du solde d'un compte est fait comme un simple SELECTdans la fonction `get_solde()` ; il vérifie le numéro du compte donné en paramètre ainsi que l'identification de l'agence. Ces informations seront entrés par l'utilisation d'un carte bancaire ou d'une personne en guichet d'agence.

4 Retrait d'espèces, virement entre comptes d'une banque, mise en place d'un virement périodique

La fonction `creer_transaction()` permet de créer tout type de transaction. Selon les paramètres la transaction permet un « flux » d'argent. Ce flux est identifié par un char(1)='O' (pour « out ») lorsqu'il est dirigé vers « l'extérieur », tels que les retraits, prélèvements, virements, etc... et par un char(1)='I' (i.e. « in ») pour des dépôts par chèque, espèce ou virements également.

5 Interdiction bancaire

L'interdit bancaire est liée à la base de donné FCC dans laquelle on trouve les comptes bancaires en découvert et ayant une date de régularisation dépassée.

Un compte peut avoir un dépassement autorisé ou non, c'est à dire qu'il est possible d'avoir un solde négatif. À partir de l'opération changeant le solde en négatif, si le dépassement n'est pas autorisé alors l'opération n'a pas lieue. S'il est autorisé alors, selon un paramètre de la banque et du type de compte, une période de régularisation est mise en place à partir du jour du solde négatif. Au terme de cette période de temps, le client doit régulariser sa situation en remboursant sa dette et en payant les frais dûs à son dépassement. À la fin de ce temps, si le client n'a toujours pas régularisé son compte alors il devient interdit bancaire et ses informations bancaires sont envoyées à la FCC, ses opérations de type flux out (débit) sont bloqués.

Cette fonctionalité n'a pas été implémentée ! La table **Interdiction_bancaire** a été modélisée pour permettre de stocker la date limite de régularisation et celle de début d'interdiction.

III Améliorations futures

1 Interdiction Bancaire

L'implémentation optimale à laquelle nous avons pensé est de créer une fonction permettant de vérifier, chaque jour, si une date de régularisation est actuelle et si c'est le cas et que le client n'est pas en règle alors il deviendra interdit bancaire. Le problème rencontré est de trouver une fonction qui boucle sur 24h sans utiliser de trigger.

IV Mode d'emploi

1 Création de la base, des tables, fonctions et insertions

Nous utilisons les fichiers suivants directement dans le terminal à l'aide de \i

- crear.sql
- insert.sql
- creer_functions.sql

2 Requêtes pour créer une liste de comptes

```
select creer_compte(1,1,86786,76567);
select creer_compte(2,2,40000,12345);
select creer_compte(2,4,67890,12345);
select creer_compte_joint(4,1,2,50000,76567,true,false);
select creer_compte_joint(3,2,4,40000,12345,false,true);
```

3 Requêtes pour mettre à jour les soldes des comptes créés

```
update compte set solde=3400 where id_compte=1;
update compte set solde=4700 where id_compte=2;
update compte set solde=5000 where id_compte=3;
update compte set solde=900 where id_compte=4;
update compte set solde=2100 where id_compte=5;
```

4 Requêtes permettant la consultation de soldes

```
select get_solde('20147770001',86786);
select get_solde('20147770002',40000);
select get_solde('20147770004',50000);
```

5 Requêtes de transactions

- **Transaction de crédit par chèque**

```
select creer_transaction('I',2,100,2014-01-06,1,'12345678901234567890123', '2014011',
'0987654321098765') ;
```

- **Transaction de crédit par virement unitaire**

select creer_transaction ('I',4, 100, 2014-01-06 ,1, '123456789012345678901 23', null, '0987654321098765') ;

- **Transaction de crédit par virement périodique**

select creer_transaction ('I',5, 100, 2014-01-06 ,5, '123456789012345678901 23', null, '0987654321098765') ;

- **Transaction de crédit par depôt d'espèces avec carte bancaire**

select creer_transaction ('O',6, 100, 2014-01-06 ,1, null, null,'0987654321098765') ;

- **Transaction de débit par prélèvement mensuel**

select creer_transaction ('O',1, 100, 2014-01-06 ,5, '123456789012345678901 23', '2014011', '0987654321098765') ;

- **Transaction de débit par retrait d'espèces par carte bancaire**

select creer_transaction('O',3,100,2014-01-06 ,1, null, '2014011', '0987654321098765') ;

- **Transaction de débit par virement unitaire**

select creer_transaction ('O',4, 100, 2014-01-06 ,1, '123456789012345678901 23', null, '0987654321098765') ;

- **Transaction de débit par virement mensuel entre comptes d'un même client**

select creer_transaction ('O',5, 100, 2014-01-06 ,5, '123456789012345678901 23', null, '0987654321098765') ;