UFR Informatique – Université Paris Diderot
Master 1

Rapport de projet

BANQUES

Binôme: **CLAVEAU Andéol**
NGUYEN Thi Quynh Nga

Profeseur: **WIESLAW Zielonka**

1/2014

Les choix effectués

L'utilisation de l'héritage

Nous avons beaucoup utilisé l'héritage dans ce projet, en utilisant une clé primaire dans la table fille identique de celle utilisée dans la table mère (vérifiée par une contrainte foreign key). Je pense qu'il est possible d'améliorer la façon dont nous traitons ceci dans les fonctions (le problème de faire une requête select max(id) est que tous les insert/update sont paralysés sur cette table pendant ce traitement peu utile).

Ne pas utiliser de check sur le solde

Nous aurions pu utiliser un check pour nous assurer que le solde ne descendait pas en dessous de la limite imposée pour l'autorisation de dépassement de découvert. Cependant, ceci créait un problème : il interdisait à notre banque de prélever les frais mensuels dans le cas où un compte serait à la limite basse. Afin de ne pas offrir ces frais aux clients déjà douteux, nous avons donc décidé de traiter ce problème dans les fonctions elles-mêmes.

Le relevé de compte

Nous avons choisis de créer une table t_releve_compte, avec une vue associée v_releve_compte. La table contient en clé primaire un couple iban/bic permettant d'identifier le compte. La vue, elle, ne contient pas ce couple : elle est créée de façon à ce qu'une personne la consultant ne voit que les comptes dont il est propriétaire ou mandataire. Elle est remplit tous les mois par le trigger creer_releve_de_compte().

La génération d'IBAN/BIC

Nous n'avons pas pu implémenter cette fonctionnalité, mais nous aurions aimé créer une fonction capable de générer automatiquement l'IBAN et le BIC d'un compte.

Les fonctions

_ajouter un moyen de paiement à un compte :

- ajout_moyen_paiement(int, varchar(20), int)
- les 2 premiers arguments correspondent au couple IBAN/BIC. Le troisième est le code correspondant au moyen de paiement à ajouter.
- La fonction renvoi une erreur si les paramètres sont erronés ou si ce compte possède déjà ce moyen de paiement, ou si le compte est fermé..

_retirer à un compte un moyen de paiement :

- supprimer_moyen_paiement(int, varchar(20), int)
- les 2 premiers arguments correspondent au couple IBAN/BIC. Le troisième est le code correspondant au moyen de paiement à ajouter.
- La fonction renvoi une erreur si les paramètres sont erronés, ou si le compte est fermé. ou si le compte ne possède pas ce moyen de paiement.

_réaliser un virement unique entre 2 comptes d'une même banque :

- virement_unique_intra(int, int, varchar(20), varchar(20), float, int, varchar(40))
- le 1er et le 3e paramètres identifient le débiteur
  le 2e et le 4e paramètre identifient le créditeur
  le 5e paramètre est la somme à virer. Le 6e paramètre indique le moyen utilisé, et le 7e contient un court texte permettant d'identifier le virement (par exemple : "achat billet RATP")
- la fonction renvoi une erreur si les paramètres sont erronés, ou si le solde du compte débiteur n'est pas suffisant, ou si le compte est fermé..
- En cas de solde insuffisant dans le cas d'un paiement par chèque, le client débiteur devient interdit bancaire.
- Chaque virement est enregistré dans la table t_operation.

_mettre en place un virement permanent :

- creer_virement_permanent(int, int, varchar(20), varchar(20), float, char, date)
- le 1er et le 3e paramètres identifient le débiteur
  le 2e et le 4e paramètre identifient le créditeur
  le 5e paramètre est la somme à virer. Le 6ème paramètre corresponds à la périodicité à laquelle le virement est effectué : mensuelle, trimestrielle, semestrielle ou annuelle.
  Le 7ème paramètre indique la date à laquelle commencera le virement.
- La fonction renvoi une erreur si les paramètres sont erronés (valeurs possibles du 6e paramètre : 'M', 'T','S' ou 'A'), ou si le compte est fermé.

_mettre fin à un virement permanent :

- supprimer_virement_permanent(int)
- le paramètre corresponds à l'identifiant du virement.
- La fonction peut renvoyer une erreur si le virement n'existe pas.

_retirer de l'argent :

- retirer(int, varchar(20), int)
- les 2 premiers arguments correspondent au couple IBAN/BIC. Le 3ème indique le montant à retirer.
- La fonction renvoi une erreur si les paramètres sont erronés ou si le solde est insuffisant, ou

si le compte est fermé..

_déposer de l'argent :

- depot(int, varchar(20), int)
- les 2 premiers arguments correspondent au couple IBAN/BIC. Le 3ème indique le montant à retirer.
- La fonction renvoi une erreur si les paramètres sont erronés, ou si le compte est fermé.

_fermer un compte :

- fermeture_compte(int,varchar(20))
- les 2 arguments correspondent au couple IBAN/BIC
- a fonction renvoi une erreur si les paramètres sont erronés, si le solde n'est pas égal à 0, ou si le compte est déja fermé.
- le compte n'est pas supprimé : il est juste désactivé (via la colonne "actif" de la table t_compte)

_ Ouverture compte vue

Ouverture une compte vue pour le nouveau client

- Les entrées:
+ type de client (client)
+ informations de client (nom_client, adresse_client, tel_client, prenom_client, sexe_client) + informations de compte (iban_compe, bic_compte)
- Les sorties: Null
- Les exceptions:

_Ouverture compte

Ouverture une compte vue pour le client qui a existé dans base de donnée:

- Les entrées:
+ identification de client (id_client)

+ informations de compte (iban_compe, bic_compte)

- Les sorties: Null
- Les exceptions: client inexistant

_Créer mandataire

Créer un mandataire de compte:

- Les entrées:
+ informations de mandataire (nom_mandataire, adresse_mandataire, tel_mandataire, prenom_mandataire, sexe_mandataire)
+ informations de compte (iban_compe, bic_compte)
- Les sorties: Null
- Les exceptions: compte inexistant

_Créer mandataire

Créer un mandataire de compte (le futur mandataire existe déjà dans la base de données):

- Les entrées:
+ identification de mandataire (id_mandataire)
+ informations de compte (iban_compe, bic_compte)
- Les sorties: Null

- Les exceptions: compte inexistant, personne inexistant

_Interdit bancaire

Devenir interdit bancaire pour un client

- Les entrées:
+ identification de client (_client)
+ motif d'interdit (_motif)
- Les sorties: Null
- Les exceptions: client inexistant

_Ne plus être interdit bancaire

Ne plus être interdit bancaire pour un client

- Les entrées:
+ identification de client (_client)
- Les sorties: Null
- Les exceptions: client inexistant, client non interdit bancaire

_Autorise découvert

Autorise découvert pour un compte

- Les entrées:
+ informations de compte (iban_compe, bic_compte)
+ découvert autorise (autorisation)
+ dépassement de découvert autorise (depassement)
- Les sorties: Null
- Les exceptions: compte inexistant

_Interdit découvert

Interdit découvert pour un compte

- Les entrées:
+ informations de compte (iban_compe, bic_compte)
- Les sorties: Null

– Les exceptions: compte inexistant

Les Triggers

_virements réguliers engendrés par les virements permanents:

- virement_mensuel()
- A chaque changement de date, ce trigger vérifie pour tous les virements permanents si la date corresponds. Si oui, un virement unique est créé.
- En cas de problème (solde insuffisant, compte inactif...), un raise notice est effectué (plutôt qu'un raise exception dans les fonctions) pour ne pas perturber le reste du trigger.

_Payer les frais liés aux moyens de paiements :

- payer_frais_moyens_paiement()
- A chaque changement de mois, des frais sont prélevés sur le compte pour chacun des moyens de paiements dont celui-ci dispose. (colonne prix de la table t_moyen_paiement)

_Créer relevé de compte

Créer relevé de compte une fois par mois

- Les entrées:
- Les sorties:
- Quand: after update on t_date

_Rémunérer

Rémunérer les compte avec un solde > 1000 une fois par jour

- Les entrées:
- Les sorties:
- Trigger: after update on t_date

_Payer frais découvert

Payer frais découvert pour les comptes avec un solde < 0 une fois par jour

- Les entrées:
- Les sorties:
- Trigger: after update on t_date