5/15/2021

Conception d'un système d'information pour des organismes Bancaires

Oumayma Mtat

ECOLE NATIONALE D'INGENIEURS DE CARTHAGE

Introduction

On va concevoir un système d'information pour les organismes bancaires.

Ce système permet à un client inscrit dans une des agences d'une banque de créer un compte dont le type répond à son besoin, ainsi d'effectuer les opérations (services) et les transactions qu'il souhaite.

De plus ce système permet aux clients de demander des prêts, des chèques, de renouveler leurs cartes.

Et aussi, le client aura la possibilité d'effectuer du parrainage à d'autres clients et le système lui calcule les primes de fidélité résultants.

MODELE CONCEPTUEL

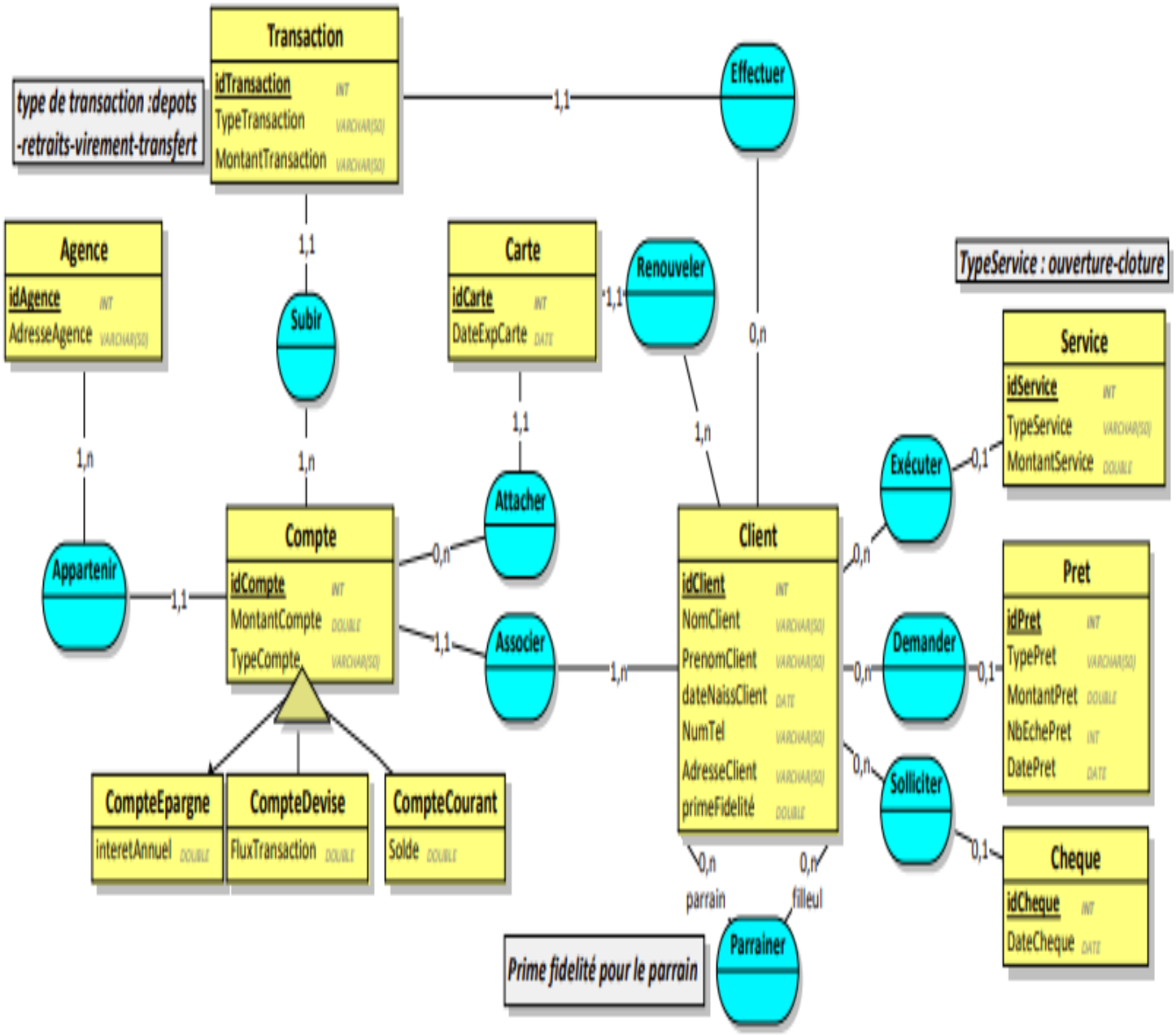

Figure 1: Diagramme conceptuel d'un système d'information des organismes bancaires

MODELE LOGIQUE

Client(idClient, NomClient, PrenomClient, dateNaissClient, NumTel, AdresseClient, primeFidelité)

Parrainage(#idClientParrain,#idClientFilleul)

Transaction(idTransaction, TypeTransaction, MontantTransaction,#idCompte,#idClient)

Compte(idCompte, MontantCompte, TypeCompte,#idAgence,#idClient)

CompteEpargne(#idCompteEpargne,interetAnnuel)

CompteDevise(#idCompteDevise,FluxTransaction)

CompteCourant(#idCompteCourant,Solde)

Pret(idPret, TypePret, MontantPret, NbEchePret, DatePret,#idClient)

Service(idService, TypeService, MontantService,#idClient)

Carte(idCarte, DateExpCarte,#idClient,#idCompte)

Cheque(idCheque, DateCheque,#idClient)

Agence(idAgence, AdresseAgence)

MODELE PHYSIQUE

Création des tables en précisant les contraintes d'intégrité :

```
CREATE TABLE Client (
  idClient DECIMAL(5) NOT NULL,
  NomClient varchar(255),
  PrenomClient varchar(255),
  dateNaissClient date,
  NumTel varchar(255),
  AdresseClient varchar(255),
  primeFidelite float,
  PRIMARY KEY (idClient)
);

CREATE TABLE Parrainage (
  idClientParrain int NOT NULL,
  idClientFilleul int NOT NULL,
  interetAnnuel float,
  PRIMARY KEY (idClientParrain,idClientFilleul),
  CONSTRAINT FK_CClientParrain FOREIGN KEY (idClientParrain) REFERENCES Client(idClient),
  CONSTRAINT FK_CClientFilleul FOREIGN KEY (idClientFilleul) REFERENCES Client(idClient)
);
CREATE TABLE Agence (
  idAgence int NOT NULL,
  AdresseAgence varchar(255),
  PRIMARY KEY (idAgence)
);
```

```
CREATE TABLE Compte (
  idCompte int NOT NULL,
  MontantCompte float,
  TypeCompte varchar(255),
  idAgence int,
  idClient int,
  PRIMARY KEY (idCompte),
  CONSTRAINT FK_CompteAgence FOREIGN KEY (idAgence) REFERENCES Agence (idAgence),
  CONSTRAINT FK_CompteClient FOREIGN KEY (idClient) REFERENCES Client (idClient),
  CONSTRAINT chk_TypeCompte CHECK (TypeCompte IN ('epargne', 'devise', 'courant'))
);

CREATE TABLE CompteEpargne (
  idCompteEpargne int NOT NULL,
  interetAnnuel float,
  PRIMARY KEY (idCompteEpargne),
  CONSTRAINT FK_CCompteEpargne FOREIGN KEY (idCompteEpargne) REFERENCES Compte(idCompte)
);

CREATE TABLE CompteDevise (
  idCompteDevise int NOT NULL,
  FluxTransaction float,
  PRIMARY KEY (idCompteDevise),
  CONSTRAINT FK_CCompteDevise FOREIGN KEY (idCompteDevise) REFERENCES Compte(idCompte)
);
CREATE TABLE CompteCourant (
  idCompteCourant int NOT NULL,
  Solde float,
  PRIMARY KEY (idCompteCourant),
  CONSTRAINT FK_CCompteCourant FOREIGN KEY (idCompteCourant) REFERENCES Compte(idCompte)
);
```

```
CREATE TABLE Pret (
  idPret int NOT NULL,
  TypePret varchar(255),
  MontantPret float,
  NbEchePret int,
  DatePret date,
  idClient int,
  PRIMARY KEY (idPret),
  CONSTRAINT FK_PretClient FOREIGN KEY (idClient) REFERENCES Client(idClient)
);

CREATE TABLE Transaction (
  idTransaction int NOT NULL,
  TypeTransaction varchar(255),
  MontantTransaction varchar(255),
  idClient int,
  idCompte int,
  PRIMARY KEY (idTransaction),
  CONSTRAINT FK_TransactionClient FOREIGN KEY (idClient) REFERENCES Client(idClient),
  CONSTRAINT FK_TransactionCompte FOREIGN KEY (idCompte) REFERENCES Compte(idCompte),
 CONSTRAINT chk_TypeTransaction CHECK (TypeTransaction IN ('depot', 'retrait', 'virement', 'transfert'))
);

CREATE TABLE Cheque (
  idCheque int NOT NULL,
  DateCheque date,
  idClient int,
  PRIMARY KEY (idCheque),
  CONSTRAINT FK_ClientCheque FOREIGN KEY (idClient) REFERENCES Client(idClient)
);
```

```
CREATE TABLE Carte (
  idCarte int NOT NULL,
  DateExpCarte date,
  idClient int,
  idCompte int,
  PRIMARY KEY (idCarte),
  CONSTRAINT FK_CarteClient FOREIGN KEY (idClient) REFERENCES Client(idClient),
  CONSTRAINT FK_CarteCompte FOREIGN KEY (idCompte) REFERENCES Compte(idCompte)
);

CREATE TABLE Service (
  idService int NOT NULL,
  TypeService varchar(255),
  MontantService float,
  idClient int,
  PRIMARY KEY (idService),
  CONSTRAINT FK_ServiceClient FOREIGN KEY (idClient) REFERENCES Client(idClient),
  CONSTRAINT chk_TypeService CHECK (TypeService IN ('ouverture', 'cloture'))
);
```

INSERTION DE JEU DE DONNEES

On a généré automatiquement les données via le logiciel : **dbForge Data Generator for Oracle**

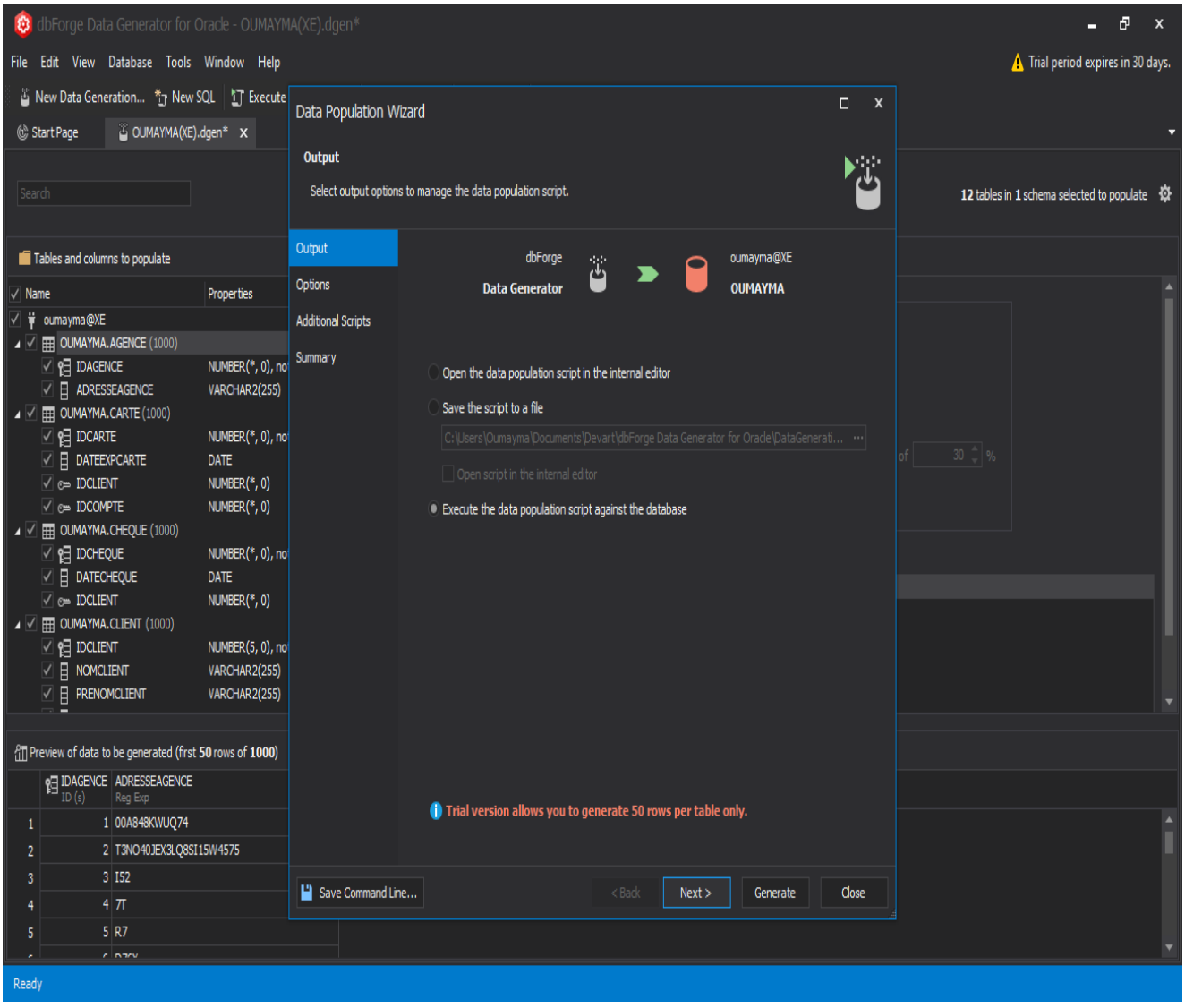

Figure 2: Insertion de jeu de données

REQUETER LA BASE DE DONNEES

1-Donner les noms et prénoms des clients, triés, nées après 1985

```
select nomclient,prenomclient  from client
where datenaissclient >= TO_DATE('1985-01-01', 'YYYY-MM-DD')
 order by nomclient;
```

2-afficher la liste des clients ayant au moins un compte devise

```
select nomclient,prenomclient from client cl,compte co where cl.idclient=co.idclient
And co.typecompte = Any (select TYpecompte from compte where typecompte='devise');
```

3- donner le numero de compte ayant subi l'opération de virement la plus élévée

```
select idcompte from transaction
where montanttransaction > All(select montanttransaction from transaction where
typetransaction = 'virement');
```

4- donner le nombre de prets,s'il y en a, effectues pour le client n°42

```
select sum(nbechepret) from pret
where exists(select * from pret where idclient=42);
```

5- donner la liste des clients qui sont à la fois filleul et parrain

```
select nomclient, prenomclient from client cl,parrainage p where p.Idclientparrain = cl.idclient
INTERSECT
select nomclient,prenom from client cl,parrainage p where p.idclientfilleul = cl.idclient;
```

6- donner la moyenne des montants dont on a cloturé les comptes

select avg(montantservice) as moyennecloturecomptes from service where typeservice='cloture';

7- donner les agences du banque outre que celle qui contient le compte coutant n°24

select * from agence Minus select * from agence where idagence = (select idagence from compte

where idcompte = (select idcomptecourant from comptecourant where idcomptecourant = 24));

8- Donner le nombre de cartes pour chaque clients

select cl.nomclient||cl.prenomclient as client,count(c.Idcarte) as nbcarte

from carte c,client cl group by(cl.nomclient||cl.prenomclient);

9- Donner la liste des clients ayant plus que 100 cheques

select cl.nomclient||cl.prenomclient as client,count(ch.datecheque)

from cheque ch,client cl group by(cl.nomclient||cl.prenomclient)

having(count(ch.datecheque) >100);

10- Supprimer toutes les transactions dont le montant est null

Delete from transaction where montanttransaction is null;

Conclusion

Dans ce projet, on a conçu un système d'information qui peut être utilisé pour les organismes bancaires.

On a commencé par l'élaboration du diagramme d'entité-association, qui le modèle conceptuel.

Ensuite, on a appliqué les règles de passage au relationnel ce qui nous a permis d'élaborer le modèle logique.

L'étape suivante, consiste à mettre en œuvre ce modèle logique sur un SGBD (oracle dans ce cas), et donc on a écrit les scripts SQL relatives à la création des tables de la tables de base de données avec l'identification des contraintes d'intégrités.

Maintenant, que notre base de données est prête, c'est le moment de la remplir et de l'interroger.

Pour le remplissage, on a choisi de la remplir automatiquement avec des données aléatoires pour effectuer les requêtes désirées.