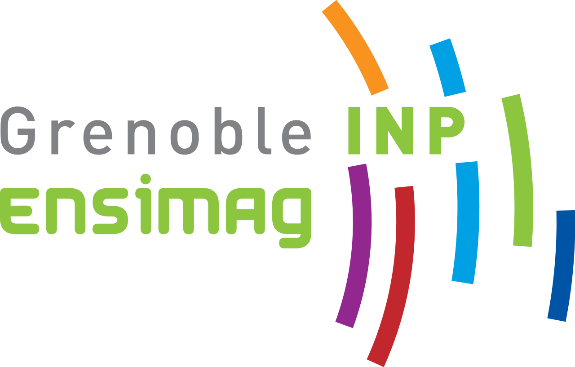
**Rendu intermédiaire : Projet BDCO**



Hassan OBEID

Vincent LIM

Fabien VERMEULEN

Jules SARRAN

DF:

IdProduit  intitulé, prixCourant, descP, URL

Email  mdp, nom, prenom, dateNaissance, adressePostale

IdOffre  dateOffre, prixPropose

IdOffre --> email, IdProduit

email, IdProduit  IdOffre

IdProduit → nomCat

Contraintes de valeurs:

prixCourant > = 0

nomCatMere inclu dans (nomCat)

prixCourant = prixPropose (de l’Offre avec la dateOffre la plus grande)

Contraintes de multiplicité :

email -|->> idOffre

idProduit -|->> idOffre

nomCatMere –|->> nomCat

nomCat -|-> nomCatMere

IdProduit -|- >> IdProduit, nomCara, valeur

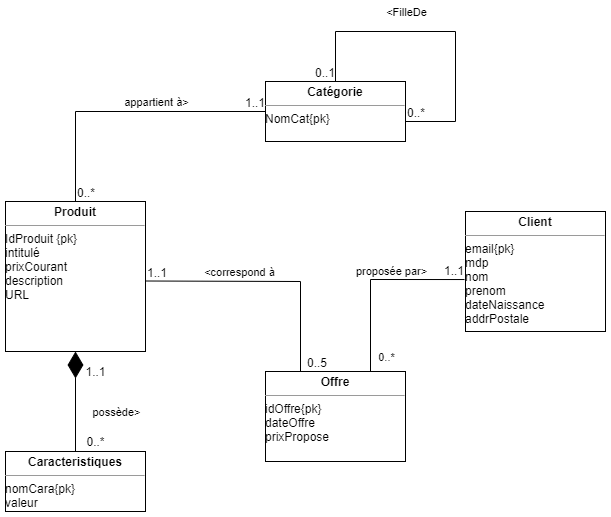
Contraintes contextuelles :

Il ne peut y avoir que 5 offres au maxium différentes pour un même produit.

Prix courant d’un produit est le prix maximum associé aux offres.

La cinquième offre représente l’achat

Le nouveau prix proposé doit être supérieur au prix courant actuel et ce prix courant doit être modifié pour valoir le nouveau prix proposé



Produit( IdProduit, intitulé, prixCourant, descP, URL, NomCat)

* NomCat référence Categorie(NomCat dans Categorie)
* prixCourant = prixProposé de la dernière offre si elle existe

1 FN (Valeurs atomiques + aucune valeur pouvant valoir nulle) → 2 FN (Tout dépend de IdProduit, voir tâche A) → 3 FN → 3FNBCK

F = { (IdProduit → intitulé, prixCourant, descP, URL, NomCat)}

Offre(IdOffre, dateOffre, prixPropose, IdProduit, email)

* email référence Client(email dans Client)
* IdProduit référence Produit(IdProduit dans Produit)
* Vérifier qu’il y ait 5 offres maximum pour un même IdProduit
* Une offre a un prixCourant supérieur au prixCourant d’une autre offre, si elle existe, ayant le même IdProduit et une dateOffre inférieure

1 FN (Valeurs atomiques + aucune valeur pouvant valoir nulle) → 2 FN (tout dépend de IdOffre, voit tâche A) → 3FN (Oui) → 3FNBCK

F = { (IdOffre → dateOffre, prixPropose, IdProduit, email)}

Client(email, mdp, nom, prenom, dateNaissance, addrPostale)

1 FN (Valeurs atomiques + aucune valeur pouvant valoir nulle) → 2 FN (tout dépend de email, voit tâche A) → 3FN (Oui) → 3FNBCK

F={ (email → mdp, nom, prenom, dateNaissance, addrPostale )}

Caracteristiques(IdProduit, nomCara, valeur)

* IdProduit référence Produit(IdProduit dans Produit)

1 FN (Valeurs atomiques + aucune valeur pouvant valoir nulle) → 2 FN (tout dépend de (IdProduit, nomCara), voit tâche A) → 3FN (Oui)

F={ (IdProduit, nomCara → valeur )}

Categorie(NomCat)

1 FN (Valeurs atomiques + aucune valeur pouvant valoir nulle) → 2 FN (il n’y a que le clé primaire) → 3FN (Oui) → 3FNBCK

FilleDe(NomCat, NomCatMere)

* NomCat référence Categorie(NomCat dans Categorie)
* NomCatMere référence Categorie(NomCat dans Categorie)

1 FN (Valeurs atomiques + aucune valeur pouvant valoir nulle) → 2 FN (NomCatMere dépend de NomCat) → 3 FN (oui) → 3FNBCK

Requêtes SQL non verifiées :

create table Categorie

(

nomCat VARCHAR(20) constraint PK\_nomCatCategorie primary key

);

create table Produit

(

IdProduit INTEGER constraint PK\_IdP primary key,

intitule VARCHAR(20),

prixCourant MONEY check (prixCourant > 0),

descP VARCHAR(30),

URL VARCHAR(20),

nomCat VARCHAR(20) constraint FK\_nomCat references Categorie(nomCat)

);

create table Client

(

email VARCHAR(20) constraint PK\_email primary key,

mdp VARCHAR(20),

nom VARCHAR(20),

prenom VARCHAR(20),

dateNaissance date,

addrPostale VARCHAR(20),

);

create table Offre

(

IdOffre INTEGER constraint PK\_IdOffre primary key,

dateOffre DATE,

prixPropose MONEY,

IdProduit INTEGER constraint FK\_IdProduitOffre references Produit(IdProduit),

email VARCHAR(20) constraint FK\_email references Client(email)

);

create table Caracteristiques

(

IdProduit INTEGER constraint FK\_IdProduitCaracteristiques references Produit(IdProduit),

nomCara VARCHAR(20),

valeur FLOAT,

constraint PK\_IdP\_nomCara primary key (IdProduit, nomCara)

);

create table FilleDe

(

nomCat VARCHAR(20) constraint FK\_nomCatt references Categorie (nomCat),

NomCatMere VARCHAR(20) constraint FK\_nomCatMere references Categorie (nomCat),

constraint PK\_nomCatFilleDe primary key (nomCat)

);