Projet Bases de Données

Théo Leducq, Jérémie O'Brien, Fabrice Pont, Cécile Robichon

Décembre 2018

1 Analyse

1.1 Propriétés

{ idSalle, idVente, idProduit, caracteristique, idTypeEnchere, idEnchere, idUtilisateur, typeEnchereLimitee, typeEnchereCroissant, typeEnchereRevocable, typeEnchereMultiple, dateMiseEnActivite, venteFinie, catProduit, description, nomProduit, prixRevient, stock, valCaracteristique, prixDepart, duree, dateHeure, prixEnchere, quantite, email, nomUtilisateur, prenomUtilisateur, adresseUtilisateur }

1.2 Analyse statique

- Dépendances fonctionelles

idSalle → catProduit, typeEnchere
idVente → idProduit, idSalle, dateMiseEnActivite
venteFinie → idVente
catProduit → description
idProduit → nomProduit, prixRevient, stock, idUtilisateur, catProduit
idProduit, caracteristique → valCaracteristique
idTypeEnchere → prixDepart
typeEnchereLimitee → idTypeEnchere, duree
typeEnchereCroissant → idTypeEnchere
typeEnchereRevocable → idTypeEnchere
typeEnchereMultiple → idTypeEnchere
idEnchere → idUtilisateur, idVente, dateHeure, prixEnchere, quantite
idUtilisateur → email, nomUtilisateur, prenomUtilisateur, adresseUtilisateur

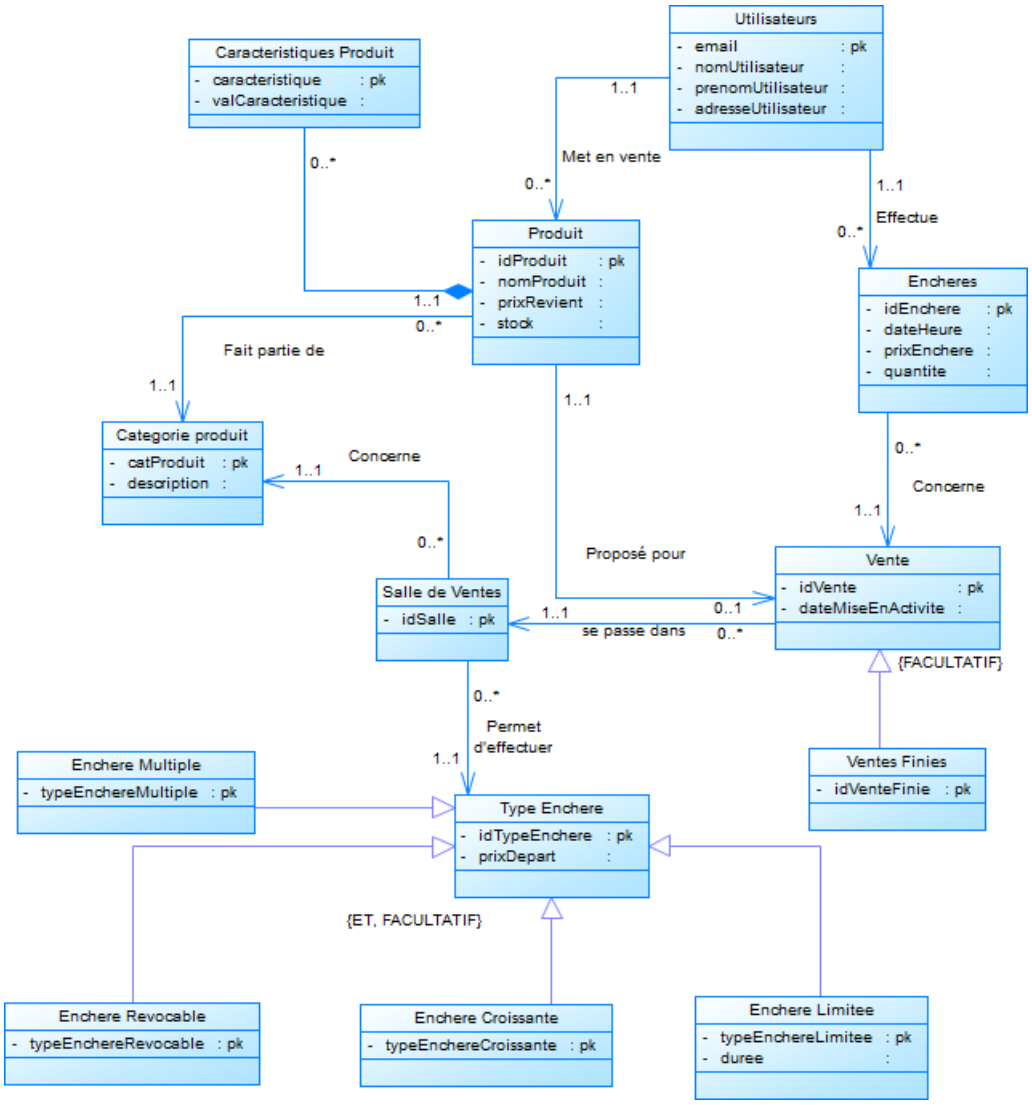
- Contraintes de valeur

prixRevient > 0, stock ≥ 0,
prixDepart ≥ 0
duree > 0
prix ≥ 0, quantite > 0
Ext(typeEnchereLimitee) $\subseteq$ Ext(idTypeEnchere)
Ext(typeEnchereCroissant) $\subseteq$ Ext(idTypeEnchere)
Ext(typeEnchereRevocable) $\subseteq$ Ext(idTypeEnchere)
Ext(typeEnchereMultiple) $\subseteq$ Ext(idTypeEnchere)
Ext(venteFinie) $\subseteq$ Ext(idVente)

- Contraintes de multiplicité

- Un utilisateur peut mettre en vente plusieurs produits: idUtilisateur $-\|-$ >> idProduit
- Un utilisateur peut effectuer plusieurs enchères: idUtilisateur $-\|-$ >> idEnchere
- Un produit peut avoir plusieurs caractéristiques : idProduit $-\|-$ >> idProduit, caracteristique
- Une catégorie de produit peut être la catégorie de plusieurs salles de vente catProduit $-\|-$ >> idSalle
- Une catégorie de produit peut être la catégorie de plusieurs produits catProduit $-\|-$ >> idProduit
- Une vente peut être l'objet de plusieurs enchères idVente $-\|-$ >> idEnchere

1.3 Schéma entités/associations



1.4 Passage au relationnel

- Schéma Relationnel

SalleDeVente (idSalle, typeSalle, nomCategorie)
typeSalle référence TypeEnchere
nomCategorie référence CatégorieProduit

TypeEnchere (idTypeEnchere, prixDepart, montante (booléen), encheresMultiples (booléen), revocable (booléen), dureeLimitee (booléen), duree)

CategorieProduit (nomCat, descriptionCat)

Vente (idVente, dateMiseActivite, finie (booléen), salle, produitPropose)
salle référence SalleDeVente
produitPropose référence Produit

Produit (idProduit, nomProduit, prixRevient, stock, nomCatProduit, emailVendeur)
nomCatProduit référence CatégorieProduit
emailVendeur référence Utilisateur

Caracterisitique (idProduitCarac, nomCarac, valeur)
idProduitCarac référence Produit

Enchere (idEnchere, idVenteEnchere, prixEnchere, dateEnchere, quantite, emetteur)
idVenteEnchere référence Vente
emetteur référence Utilisateur

Utilisateur (email, nomUtilisateur, prenomUtilisateur, adresse)

- Formes Normales

$1^{ère}$ Forme normale : Pour toutes les relations, les attributs ont des valeurs atomiques elles sont donc bien 1FN.

2^{nde} Forme normale : Pour toutes les relations, les attributs non clés sont complètement dépendants de la clé primaire de la relation, elles sont donc bien 2FN.

$3^{ème}$ Forme normale : Pour toutes les relations, les attributs non clés sont dépendants non transitivement de la clé primaire de la relation, elles sont donc bien 3FN.

$3^{ème}$ Forme normale BCK : Pour toutes les relations, les attributs non clés ne sont pas sources de dépendance fonctionnelle vers toute ou partie de la clé primaire de la relation, elles sont donc bien 3FNBCK.

2 Transactions

2.1 Transactions qui concernent les utilisateurs

- Création d'un nouvel utilisateur
- Ajout d'un produit
- Ajout de caractéristiques produit pour un produit ajouté par l'utilisateur
- Ajout d'une proposition d'enchère pour une vente

2.2 Transactions qui concernent l'administrateur

- Ajout d'un nouveau produit
- Suppression d'un produit
- Création d'une nouvelle catégorie de produit
- Association d'un produit à une catégorie de produit
- Création d'un nouveau type d'enchère
- Création d'une nouvelle salle de vente
- Ajout d'une vente (mise en vente d'un produit)
- Affichage des enchères d'une vente qui est terminée
- Suppression d'une vente qui est finie et pour laquelle l'administrateur a déterminé le gagnant de la vente.

3 Choix de conceptions et fonctionnement de l'application

3.1 Fonctionnement global de l'application

Notre application fonctionne comme un lien entre un administrateur et les utilisateurs de l'application.

Les utilisateurs peuvent s'inscrire puis ils peuvent proposer à la vente des produits. Des requêtes `sql` permettent de créer des enchères mais malheureusement nous n'avons pas eu le temps de permet à l'utilisateur de le faire via l'interface graphique.

L'administrateur, quant à lui, peut créer des salles de vente en requête ou via l'interface graphique. Il peut proposer dans ces salles un produit de la base de données pour une enchère. Les produits peuvent être ajoutés à la base de données à la fois par l'administrateur et par les utilisateurs.

Nous ne sommes pas parvenus à implémenter la gestion automatique des "gagnants" d'une enchère donc pour l'instant, c'est l'administrateur qui peut obtenir la liste des enchères effectuées pour une vente et qui peut désigner celui qui gagne l'enchère via des requêtes `sql`.

Seul l'administrateur peut créer une vente. C'est dans celle-ci qu'il déterminera le produit, la salle et le type d'enchère de cette vente.

3.2 Nos choix de conceptions

Nous avons fait le choix d'avoir une application qui fait office de lien entre l'administrateur et les utilisateurs, mais qui ne gère pas d'elle même la mise en vente des produits proposés par les utilisateurs. Par ailleurs, nous avons décidé de limiter le pouvoir des utilisateurs, ils ne peuvent pas d'eux-mêmes choisir la salle de vente et donc le type d'enchère auquel sera soumis leur produit, c'est le rôle de l'administrateur lorsqu'il crée une vente.

Nous avons décidé d'introduire un booléen dans la table Vente qui permet d'indiquer si une vente est en cours ou si elle est terminée, ainsi il suffit de parcourir le tableau des ventes pour vérifier lesquelles sont terminées. Ensuite, on peut, pour toutes les ventes terminées, déterminer le(s) "gagnant(s)" puis supprimer cette vente du tableau Vente.

Pour la bonne réalisation de l'application nous avons utilisé plusieurs technologies que nous allons détailler. Pour assurer la connection avec la base de données de EnsiOracle nous avons créé le module SConnect. Nous avons ensuite utilisé des driver pour s'assurer de créer une unique connection. Dans le code `java`, nous avons utilisé des DAO pour les requêtes `sql`. La classe abstraite DAO donne une classe extends pour chacunes des tables. Dans c'est classe, il y a une méthode `create`, `delete` et `find` qui permettent respectivement d'ajouter des éléments dans la table, d'en supprimer et d'un trouver un avec sa clé primaire. Pour tester le fonctionnement des DAO, voir le fichier `testAjout.java`. Pour

la création de l'interface graphique et son utilisation nous avons utilisé Java Swing. Dans un premier temps, nous avons téléchargé JFdesigner qui nous a permis d'utiliser Java Swing graphiquement sur IntelliJ.