10/12/2018

Victor Bros

Sébastien Herbreteau

Sophia Vilayvong

Projet Base de données

Documentation

Ensimag – 2ème année

Table des matières

[1. Analyse du problème et réflexion sur la modélisation de la base de données 2](#_Toc532073772)

[1.1 Identification des propriétés élémentaires 2](#_Toc532073773)

[1.2 Identification des contraintes 3](#_Toc532073774)

[1.2.1 Identification des dépendances fonctionnelles 3](#_Toc532073775)

[1.2.2 Contraintes de valeurs 5](#_Toc532073776)

[1.2.3 Contraintes de multiplicité 5](#_Toc532073777)

[1.2.4 Contraintes autres 6](#_Toc532073778)

[1.2.5 Résultat de l’analyse statique 6](#_Toc532073779)

[2. Modèle Entité/Association et passage au relationnel 0](#_Toc532073780)

[2.1 Modèle Entité/Association 0](#_Toc532073781)

[2.2 Passage au relationnel 0](#_Toc532073782)

[2.3 Formes normales des relations obtenues : 3FNBCK 1](#_Toc532073783)

[2.4 Scripts SQL 1](#_Toc532073784)

[2.5 Commentaires sur la création de la base de données et des tables 2](#_Toc532073785)

[3. Analyse des fonctionnalités 2](#_Toc532073786)

[3.1 « Mise en place d’une salle de vente et sélection de produits déjà disponibles à la vente et permettant le choix du type d’enchères et du prix de départ » 2](#_Toc532073787)

[3.2 Enchère faite par un utilisateur sur un produit mise en vente dans une salle de vente 3](#_Toc532073788)

[3.3 Processus de fin de vente 3](#_Toc532073789)

# Analyse du problème et réflexion sur la modélisation de la base de données

On commence par lire et analyser le cahier des charges du client pour en dégager les propriétés élémentaires, les dépendances fonctionnelles, les contraintes de valeurs et de multiplicités et éventuellement d’autres contraintes qui ne pourraient pas s’écrire sous la forme de contraintes de multiplicités.

Commençons par identifier les propriétés élémentaires.

## Identification des propriétés élémentaires

Les utilisateurs de *Baie-électronique* peuvent mettre en vente des produits ou enchérir sur des produits mis en vente par d’autres. Pour organiser un tant soit peu les ventes aux enchères, *Baie électronique* se base sur le concept des Salles de Ventes. Une Salle de Vente propose la vente aux enchères d’une sélection de produits d’une même catégorie dans un même type d’enchères. Chaque vente, identifiée de façon unique, ne concerne qu’un seul produit et n’a lieu que dans une seule salle. Une catégorie de produit est identifiée par son nom et possède une description. Un produit est identifié de façon unique et est décrit par son nom, son prix de revient (prix à partir duquel le vendeur fait des bénéfices) et le stock proposé à la vente (un lot fait l’objet que d’une seule vente, éventuellement divisée en sous-lots). Un ensemble de caractéristiques (couples nom et valeur) permet de décrire spécifiquement le produit.

Les types de vente aux enchères sont variés mais ils possèdent tous un prix de départ à partir duquel démarreront les enchères. Une vente peut être montante (les offres sont toujours croissantes) ou descendante (le prix des produits diminue toutes les minutes, le premier utilisateur à faire une offre remporte la vente). Une vente, qu’elle soit montante ou descendante, peut autoriser un même utilisateur à enchérir plusieurs fois sur le même produit (en respectant le sens, bien sûr), ou seulement une seule fois. Une vente peut également être à durée limitée (la date et l’heure de fin sont alors précisées) ou à durée libre (le délai maximal entre deux offres est fixé à 10 minutes). Enfin, une vente peut être révocable, la vente est annulée si le prix de revient du produit n’est pas atteint, ou non révocable : le vendeur peut être amené à vendre à perte. Par défaut, les ventes sont montantes, non révocables, sans limite de temps et permettent à un même utilisateur d’enchérir plusieurs fois.

Une enchère, effectuée par un seul utilisateur, porte sur une vente en particulier et propose un prix d’achat. La date et l’heure de l’enchère sont également des éléments importants pour décider de sa validité et pour déterminer qui remporte la vente. De plus, une enchère peut ne concerner qu’un sous-lot du produit mis en vente, c’est pourquoi la quantité de produit concerné est toujours précisée. Il est possible en pratique qu’une enchère se termine par la vente d’un (ou de quelques) sous-lots du produit concerné. L’objectif est alors que la vente rapporte le plus possible au vendeur, même si tous les produits ne sont pas vendus et si les sous-lots sont finalement vendus à des prix différents (en fonction des enchères qui ont été faites).

Les utilisateurs de l’application sont identifiés par leur email et décrits par leur nom, leur prénom et leur adresse postale (pour la livraison). Nous ne nous occupons pas ici des informations de paiement, cela étant géré par une autre application déjà en place.

Voici la liste des propriétés :

{ idSalle, typeVente, estRevocable, estLibre, enchereMultiple

nomCategorie, descriptionCategorie,

idVente, prixDepart, dateFin, heureFin,

idProduit, nomProduit, prixRevient, stock,

nomCaracteristique, valeurCaracteristique

idEnchere, prixPropose, dateEnchere, heureEnchere, quantiteEnchere,

email, nomUtilisateur, prenomUtilisateur, adresseUtilisateur }

Choix de conception pour la base de données :

*typeVente, estRevocable, estLibre et enchereMultiple* prennent leur valeur dans l’ensemble {0,1}. 1 correspond aux valeurs par défaut définies dans l’énoncé et 0 à leur antonyme.

Ainsi :

* Si typeVente = 1 alors la vente est montante, sinon elle est descendante ;
* Si estRevocable = 0 alors la vente est non révocable, sinon elle l’est ;
* Si estLibre = 1 alors la vente est à durée libre, sinon elle est à durée limité ;
* Si enchereMultiple = 1 alors un même utilisateur peut enchérir plusieurs fois sur la même vente, sinon il ne peut qu’enchérir qu’une seule fois.

Remarque : la propriété *idEnchere* correspond davantage à un choix de conception plutôt qu’à une spécificité du cahier des charges. Nous nous justifierons dans la partie liée aux dépendances fonctionnelles.

Nous allons à présent analyser les contraintes.

## Identification des contraintes

### Identification des dépendances fonctionnelles

« Une Salle de Vente propose la vente aux enchères d’une sélection de produits d’une même catégorie » :

idSalle --> nomCategorie

« Chaque vente, identifiée de façon unique, ne concerne qu’un seul produit et n’a lieu que dans une seule salle » :

idVente --> idProduit, idSalle

« Une catégorie de produit est identifiée par son nom et possède une description » :

Categorie --> nomCategorie, descriptionCategorie

« Un produit est identifié de façon unique et est décrit par son nom, son prix de revient (prix à partir duquel le vendeur fait des bénéfices) et le stock proposé à la vente » :

idProduit --> nomProduit, prixRevient, stock

« Un ensemble de caractéristiques (couples nom et valeur) permet de décrire spécifiquement le produit » :

idProduit, idCaracteristique --> nomCaracteristique, valeurCaracteristique

« ils [les ventes] possèdent tous un prix de départ à partir duquel démarreront les enchères. […] Une vente peut également être à durée limitée (la date et l’heure de fin sont alors précisées) ou à durée libre (le délai maximal entre deux offres est fixé à 10 minutes) » :

idVente --> prixDepart, dateFin, heureFin

Choix de conception pour la base de données : nous choisissons de mettre les attributs dateFin et heureFin quel que soit le type de vente. Une vente à durée libre aura ces deux champs initialisés *null*.

« Une enchère, effectuée par un seul utilisateur, porte sur une vente en particulier et propose un prix d’achat. La date et l’heure de l’enchère sont également des éléments importants pour décider de sa validité et pour déterminer qui remporte la vente » :

idEnchere --> email, idVente, prixPropose, dateEnchere, heureEnchere

« De plus, une enchère peut ne concerner qu’un sous-lot du produit mis en vente, c’est pourquoi la quantité de produit concerné est toujours Précisée » :

idEnchere --> quantiteEnchere

Remarque sur les dépendances fonctionnelle induites par *idEnchere* : nous pouvons nous passer de cette propriété. En effet, nous avons les dépendances suivantes :

email, idVente --> prixPropose, dateEnchere, heureEnchere

email, idVente --> quantiteEnchere

Nous le gardons pour des choix de conceptions qui sont notamment à visée préventive.

« Les utilisateurs de l’application sont identifiés par leur email et décrits par leur nom, leur prénom et leur adresse postale (pour la livraison) » :

email --> nomUtilisateur, prenomUtilisateur, adresseUtilisateur

Remarque :

Les dépendances suivantes vont être modifiées dans la suite du travail pour des choix liés à la non redondance des données et pour respecter le cahier des charges. Nous présenterons nos choix de conceptions dans la partie liée aux « contraintes autres ».

« Une vente peut être montante (les offres sont toujours croissantes) ou descendante (le prix des produits diminue toutes les minutes, le premier utilisateur à faire une offre remporte la vente) » :

idVente --> typeVente

« Une vente, qu’elle soit montante ou descendante, peut autoriser un même utilisateur à enchérir plusieurs fois sur le même produit (en respectant le sens, bien sûr), ou seulement une seule fois » :

idVente --> enchereMultiple

« Une vente peut également être à durée limitée (la date et l’heure de fin sont alors précisées) ou à durée libre (le délai maximal entre deux offres est fixé à 10 minutes) » :

idVente --> estLibre

« Enfin, une vente peut être révocable, la vente est annulée si le prix de revient du produit n’est pas atteint, ou non révocable » :

idVente --> Revocable

### Contraintes de valeurs

prixDepart >=0, double

dateFin au format date

heureFin au format heure

prixPropose >= 0, double

dateEnchere au format date

heureEnchere au format heure

quantiteEnchere >0, int

prixRevient >=0, double

stock >=0, int

« Une enchère peut ne concerner qu’un sous-lot du produit mis en vente » : quantiteEnchere <= stock

### Contraintes de multiplicité

Une salle **peut** concerner plusieurs ventes :

idSalle -|->> idVente

Une vente **peut** comporter plusieurs enchères :

idVente -|->> idEnchere

Un produit **peut** comporter plusieurs caractéristiques :

idProduit -|->> nomCaracteristique

Une catégorie **peut** rassembler plusieurs produits :

idCategorie -|->> idProduit

« Une Salle de Vente propose la vente aux enchères d’**une** **sélection** de produits » :

idSalle -|->> idProduit

« Les utilisateurs de Baie-électronique peuvent mettre en vente des produits ou enchérir sur des produits mis en vente par d’autres » :

* Un utilisateur peut vendre plusieurs produits :

Email -|->> idProduit

* Plusieurs produits d’un utilisateur peuvent être ouvert à la vente :

Email -|->> idVente

* Un utilisateur peut enchérir sur des produit mis en vente par d’autres :

Email -|->> idEnchere

### Contraintes autres

« Une Salle de Vente propose la vente aux enchères d’une sélection de produits d’une **même catégorie** dans un **même type d’enchères** » :

Cela signifie dans toutes les ventes d’une même salle de vente doivent être de même type. Ainsi, nous faisons la modification suivante :

* idVente --> typeVente devient idSalle --> typeVente
* idVente --> estLibre devient idSalle --> estLibre
* idVente --> estRevocable devient idSalle --> estRevocable
* idVente --> enchereMultiple devient idSalle --> enchereMultiple

« Par défaut, les ventes sont montantes, non révocables, sans limite de temps et permettent à un même utilisateur d’enchérir plusieurs fois ».

### Résultat de l’analyse statique

* **Propriétés :**

**{ idSalle, typeVente, estRevocable, estLibre, enchereMultiple**

**nomCategorie, descriptionCategorie,**

**idVente, prixDepart, dateFin, heureFin,**

**idProduit, nomProduit, prixRevient, stock,**

**nomCaracteristique, valeurCaracteristique**

**idEnchere, prixPropose, dateEnchere, heureEnchere, quantiteEnchere,**

**email, nomUtilisateur, prenomUtilisateur, adresseUtilisateur }**

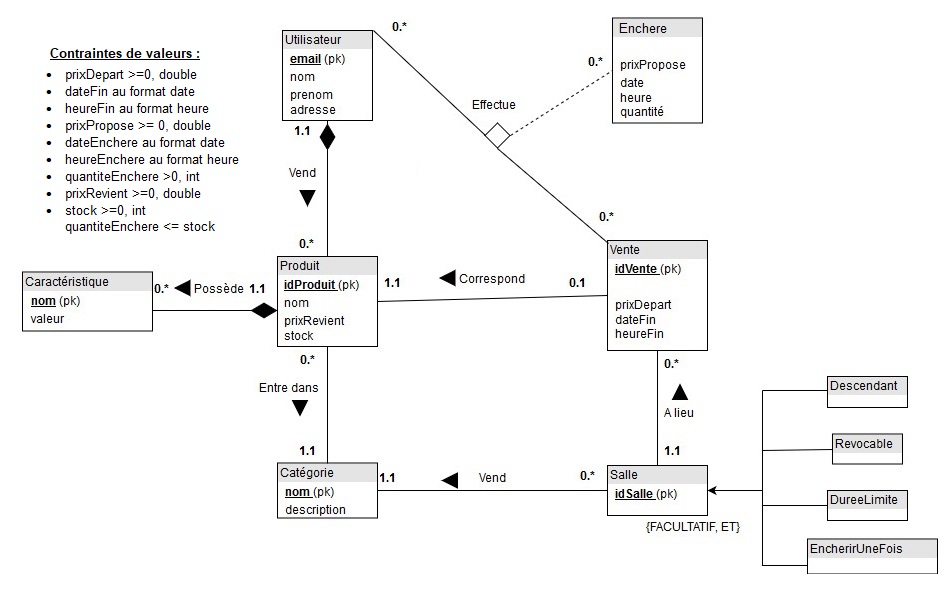
Les contraintes sont affichées dans le tableau ci-dessus :

* **Contraintes :**

|  |  |  |  |
| --- | --- | --- | --- |
| Dépendances Fonctionnelle | Contraintes de valeurs | Contraintes de multiplicité | Contraintes autres |
| * idSalle --> nomCategorie, typeVente, estLibre, estRevocacle, enchereMultiple * idVente --> idProduit, idSalle, prixDepart, dateFin, heureFin * Categorie --> nomCategorie, descriptionCategorie * idProduit --> nomProduit, prixRevient, stock * idProduit, idCaracteristique --> nomCaracteristique, valeurCaracteristique * idEnchere --> email, idVente, prixPropose, dateEnchere, heureEnchere, quantite * email --> nomUtilisateur, prenomUtilisateur, adresseUtilisateur | * prixDepart >=0, double * dateFin au format date * heureFin au format heure * prixPropose >= 0, double * dateEnchere au format date * heureEnchere au format heure * quantiteEnchere >0, int * prixRevient >=0, double * stock >=0, int * quantiteEnchere <= stock | * idSalle -|->> idVente * idVente -|->> idEnchere * idProduit -|->> nomCaracteristique * idCategorie -|->> idProduit * idSalle -|->> idProduit * Email -|->> idProduit * Email -|->> idVente * Email -|->> idEnchere |  |

# Modèle Entité/Association et passage au relationnel

## Modèle Entité/Association

Nous avons construit le schéma Entité/Association résultant de notre analyse du problème. Nous l’avons représenté sous la forme d’un diagramme UML :

## Passage au relationnel

Nous avons construit les tables suivantes à partir du diagramme UML :

* Utilisateur(email, nom, prenom, adresse)
  + email non null
* Categorie(nom, description)
  + nom non null
* Produit(idProduit, nom, nom\_categorie, email, prixRevient, stock)
  + idProduit non null
  + nom\_categorie référence Categorie(nom)
  + email référence Utilisateur(email)
* Caracteristique(nom, idProduit, valeur)
  + nom non null
  + idProduit référence Produit(idProduit)
* Salle(idSalle, nom\_categorie, type\_vente, est\_libre, est\_revocable, enchere\_multiple)
  + nom\_categorie référence Categorie(nom)
* Vente(id\_vente, id\_produit, id\_salle, prix\_depart, temps)
  + Id\_produit référence Produit(id\_produit)
  + Id\_salle référence Salle(id\_salle)
* Enchere(id\_enchere, email, id\_vente, prix\_propose, temps, quantite)
  + email référence Utilisateur(email)
  + id\_vente référence Vente(idVente)

## Formes normales des relations obtenues : 3FNBCK

Nous rappellons les relations et leurs dépendances fonctionnelles :

* Salle(idSalle, nom\_categorie, type\_vente, est\_libre, est\_revocable, enchere\_multiple)
  + idSalle --> nomCategorie, typeVente, estLibre, estRevocacle, enchereMultiple
* Vente(id\_vente, id\_produit, id\_salle, prix\_depart, temps)
  + idVente --> idProduit, idSalle, prixDepart, dateFin, heureFin
* Categorie(nom, description)
  + Categorie --> nomCategorie, descriptionCategorie
* Produit(idProduit, nom, nom\_categorie, email, prixRevient, stock)
  + idProduit --> nomProduit, prixRevient, stock
* Caracteristique(nom, idProduit, valeur)
  + idProduit, idCaracteristique --> nomCaracteristique, valeurCaracteristique
* Enchere(id\_enchere, email, id\_vente, prix\_propose, temps, quantite)
  + idEnchere --> email, idVente, prixPropose, dateEnchere, heureEnchere, quantite
* Utilisateur(email, nom, prenom, adresse)
  + email --> nomUtilisateur, prenomUtilisateur, adresseUtilisateur

Toute nos relations sont en 1FN car tous les attributs de chaque relation ont des valeurs atomiques.

Elles sont en 2FN car tous les attributs non clés sont pleinement dépendants des clés.

Elles sont aussi clairement en 3FN car tous les attributs non clés sont directement et pleinement dépendants des clés.

Elles sont même en 3FNBCK (les relations ne possèdent qu’une seule clé).

## Scripts SQL

Vous trouverez dans le dossier sql les scripts suivants :

* A.sql : il réinitialise la base de données en supprimant les tables qui portent les mêmes noms que les tables que nous allons créer puis les créer en respectant toutes les contraintes ;
* B.sql : il permet de peupler la base de données ;
* t.sql : il regroupe l’ensemble des transactions que nous avons implantés et des requêtes sql que nous avons utilisées pour réaliser l’application.

## Commentaires sur la création de la base de données et des tables

Pour générer les identifiants (id\_produit, id\_enchere, id\_vente, id\_salle), nous avons utilisé la commande « INT GENERATED ALWAYS AS IDENTITY » qui permet de générer automatiquement un entier par incrémentation. Pour chaque nouvel ajout dans la table, l’identifiant du nouvel élément sera le numéro du dernier identifiant (i.e. : l’identifiant de valeur maximale de la table) + 1. Ainsi, en supposant qu’une table contient deux objets d’id 1 et 2, alors si j’ajoute un nouvel objet, celui portera l’id 3 et le suivant l’id 4, etc.

Ainsi à la fin de chaque insertion dans une table (produit, enchere, vente, salle), dans la même transaction, il faudra récupérer l’id à l’aide d’une requête.

# Analyse des fonctionnalités

## « Mise en place d’une salle de vente et sélection de produits déjà disponibles à la vente et permettant le choix du type d’enchères et du prix de départ »

Etapes :

1. Choisir la catégorie des produits disponibles à la vente, qui seront vendu dans ladite salle
2. Création de la salle de vente en précisant le type de vente
3. Choisir les produits qui seront ouvert à la vente aux enchères en précisant le prix de départ
4. Création des ventes associées

Implantation en SQL : exemple pour une vente montante, à durée libre, non révocable et qui autorise un utilisateur à enchérir plusieurs fois sur une même vente. Pour cette transaction, on se place en mode « SERIALIZABLE ». Les autres

Begin ;

--choix de la catégorie

SELECT DISTINCT nom\_categorie FROM Produit1 ;

--ajout de la salle de vente

INSERT INTO Salle1(nom\_categorie, type\_vente, est\_libre, est\_revocable, enchere\_multiple) values ('Informatique', 0, 1, 1, 1);

--on récupère l’id de la salle créée

SELECT max(id\_salle)

FROM Salle1

WHERE nom\_categorie=Vetement

AND type\_vente=0 AND est\_libre=1 AND est\_revocable=1 AND enchere\_multiple=1 ;

--ajouter les produits souhaites parmis ceux autorisés

insert into Vente1(id\_produit, id\_salle, prix\_depart, temps) values (3, 1, 80, TO\_DATE('2018-12-06 10:02:08', 'YYYY-MM-DD HH24:MI:SS'));

commit;

insert into Vente1(id\_produit, id\_salle, prix\_depart, temps) values (4, 1, 50, TO\_DATE('2018-12-06 10:02:08', 'YYYY-MM-DD HH24:MI:SS'));

insert into Vente1(id\_produit, id\_salle, prix\_depart, temps) values (5, 1, 30, TO\_DATE('2018-12-06 10:02:08', 'YYYY-MM-DD HH24:MI:SS'));

commit;

Pour récupérer les produits autorisés, on utilise la requête suivante :

--pour récupérer les produits disponibles la vente

--dans l'éventualité de les ajouter dans une salle de vente

SELECT Produit1.id\_produit, Produit1.nom, Produit1.prix\_revient

From Produit1

Where Produit1.nom\_categorie = 'Vetement'

MINUS

Select Produit1.id\_produit, Produit1.nom, Produit1.prix\_revient

From Salle1, Vente1, Produit1

Where Salle1.nom\_categorie = 'Vetement'

AND Produit1.nom\_categorie = 'Vetement'

AND Vente1.id\_salle = Salle1.id\_salle

AND Vente1.id\_produit = Produit1.id\_produit

Remarque : des produits pourront être rajoutés ultérieurement. La seule condition est d’avoir l’identifiant de la salle.

## Enchère faite par un utilisateur sur un produit mise en vente dans une salle de vente

Etape :

1. L’utilisateur doit choisir la catégorie du produit qu’il souhaite acheter ;
2. Il doit ensuite sélectionner le produit parmi les produits mise à la vente dans la catégorie qu’il a choisi ;
3. L’application récupérer le type de la vente ;
4. L’utilisateur propose une offre ;
5. L’application vérifie les conditions d’enchères suivant le type de vente ;
6. L’application refuse ou accepte l’offre.

Implantation en SQL :

--choix de la categorie

SELECT DISTINCT nom\_categorie FROM Salle1 ;

## Processus de fin de vente