Projet Bases de Données : Documentation

Achraf Djarallah, Abderrahmane Gherissi, Baptiste Pignier,
Mariem Triki, Léa Bouabdallah et Youssef El Atia.

2024

Contents

1 Introduction

Ce document constitue la documentation du projet de bases de données pour la société Baie-électronique. Il détaille l'analyse du problème, les contraintes, la conception du modèle de données, et l'implémentation des fonctionnalités nécessaires.

2 Analyse du problème

Le projet vise à informatiser le service des ventes aux enchères de Baie-Électronique en concevant une base de données relationnelle robuste pour gérer les produits, les salles de vente et les enchères. Les principales problématiques incluent la gestion des contraintes fonctionnelles (dépendances, multiplicité, valeurs), l'implémentation de différents types de ventes (montantes, descendantes, révocables ou non) et la prise en charge des enchères concurrentes. Il s'agit également de garantir la cohérence des transactions et d'optimiser les processus décisionnels, tout en intégrant ces fonctionnalités dans une application Java via JDBC pour une utilisation pratique.

3 Conception

3.1 Entitées

Les relations obtenues après la traduction du schéma Entités/Associations sont les suivantes :

Utilisateur

- **Attributs :** `Email`, `nomUtilisateur`, `Prénom`, `Adresse`.

Vente

- **Attributs :** `idVente`, `Révocable`, `prixDépart`, `dateVente`, `Sens`, `OffreUnique`, `Limite`, `idSalle`, `diminutionRate`, `dateHeureFin`, `idProduit`.

Salle

- **Attributs :** `idSalle`, `nomCat`.

Catégorie

- **Attributs :** `nomCat`, `descCat`.

Caractéristique

- **Attributs :** `nomCar`.

Produit

- **Attributs :** `idProduit`, `nomProduit`, `prixRevient`, `stock`, `nomCat`.

Caractéristiques Produit

- **Attributs :** `idProduit`, `nomCar`, `valeurCar`.

3.2 Associations avec attributs

Offre

- **Attributs :** `Email`, `idVente`, `Quantité`, `dateHeureOffre`, `prixOffre`.

ProduitCar (association entre Produit et Caractéristique)

- **Attributs :** `ValeurCar`.

3.3 Contraintes contextuelles

Les contraintes contextuelles identifiées pour le projet sont :

- Si la vente est limitée, la date et l'heure de l'offre doivent être inférieures ou égales à la date de fin de la vente :

$$\text{dateHeureOffre} \leq \text{dateHeureFin}.$$

- Si la vente n'est pas limitée, le délai minimal entre deux offres est de 10 minutes (où dateHeureOffre_n est la date de l'offre n sur une vente donnée) :

$$\text{dateHeureOffre}_{i+1} - \text{dateHeureOffre}_i \geq 10 \text{ minutes}.$$

- La date de l'offre ne peut pas être antérieure à la dernière offre effectuée :

$$\text{dateHeureOffre} > \text{dernière dateHeureOffre}.$$

- La quantité totale d'objets offerts ne peut excéder le stock disponible :

$$\text{total de Quantité} \leq \text{Stock}.$$

- La catégorie des ventes doit correspondre à la catégorie de la salle de vente :

$$\text{Vente.catégorie} = \text{Salle.catégorie}.$$

- Pour accepter une offre, il faut que son prix total soit supérieur au prix de la dérnière offre.
- Chaque salle ne doit contenir qu'un seul type de vente et qu'une seule catégorie de vente.

3.4 Contraintes de valeur

Les contraintes de valeur identifiées sont :

- Le prix de départ doit être inférieur ou égal au prix de l'offre :

$$\text{prixDépart} \leq \text{prixOffre}.$$

- Le stock doit être supérieur ou égal à la quantité offerte :

$$\text{Stock} \geq \text{Quantité}.$$

- Les valeurs suivantes doivent être positives ou nulles :
  - Quantité ≥ 0
  - Stock ≥ 0
  - prixOffre ≥ 0
  - prixDépart ≥ 0
- La date et l'heure de l'offre doivent être antérieures ou égales à la date de fin de vente :

$$\text{dateHeureOffre} \leq \text{dateVente}.$$

- Le prix de revient doit toujours être supérieur à zéro.
- idVenteDescendante $\cup$idVenteUneOffre $\cup$ idVenteLimitee $\subseteq$ idVente

3.5 Dépendances fonctionnelles

Les dépendances fonctionnelles identifiées sont :

$$\text{idSalle} \rightarrow \text{nomCat}$$
$$\text{idVente} \rightarrow \{\text{idProduit, idSalle, Revocable, prixDepart, dateVente}\}$$
$$\text{nomCat} \rightarrow \text{descCat}$$
$$\text{idProduit} \rightarrow \{\text{nomProduit, prixRevient, Stock, nomCat}\}$$
$$\text{idProduit, nomCar} \rightarrow \text{valeurCar}$$
$$\text{idVenteDescendante} \rightarrow \{\text{diminutionRate, quantiteRestante}\}$$
$$\text{Email} \rightarrow \{\text{nomUtilisateur, Prenom, adresse}\}$$
$$\text{idVenteLimitee} \rightarrow \text{dateHeureFin}$$
$$\text{dateHeureOffre, Email, idVente} \rightarrow \{\text{Quantite, prixOffre}\}$$

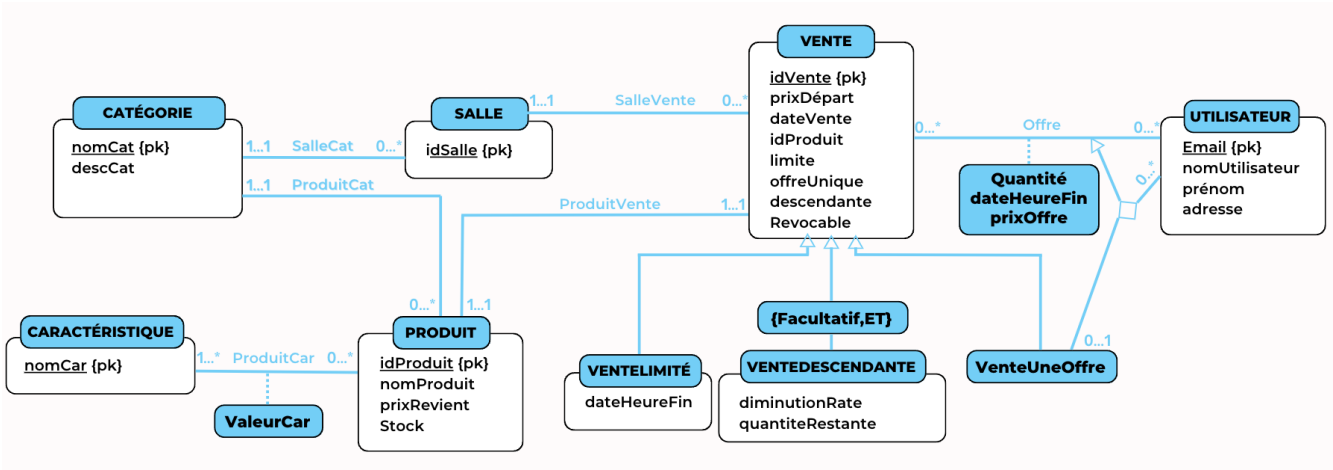
3.6 Dépendances Multivariées

$$\text{Email, idVente} -/-> \{\text{Quantite, prixOffre, dateHeureOffre}\}$$

$$\text{Email, idVenteUneOffre} -/-> \{\text{Quantite, prixOffre, dateHeureOffre}\}$$

$$\text{idProduit} -/-> \{\text{nomCar}\}$$

3.7 Schéma



3.8 Formes normales

Notre schéma de relations est bien normalisé car on a suivi les étapes de conception Entitées/Association avec les règles de traduction définies dans le cours.

3.9 Traduction Relationnelle

À partit du schéma Entitées/Relations

- `Utilisateur(email, nomUtilisateur, prenom, adresse)`
- `Categorie(nomCat, descCat)`
- `Salle(idSalle, nomCat)`
- `Produit(idProduit, nomProduit, prixRevientUnite, stock, nomCat)`
- `Vente(idVente, idProduit, idSalle, revocable, offreUnique, limite, descendante, prixDepart, dateVente)`
- `VenteLimite(idVente, dateHeureFin)`
- `VenteDescendante(idVente, diminutionRate, quantiteRestante)`
- `Caracteristique(nomCar)`
- `CaracteristiqueProduit(idProduit, nomCar, valCar)`
- `Offre(email, idVente, quantite, dateHeureOffre, prixOffre)`

4 Analyse des Fonctionnalités

4.1 Création d'une salle

Pour créer une salle, le programme demande à l'utilisateur de choisir une catégorie de produit. Ensuite, il lui demande le type de ventes qui seront organisées dans la salle. Enfin, une vente est ajoutée à cette salle, comme expliqué ci-dessous.

4.2 Ajout d'une vente à une salle existante

Il faut sélectionner un ou plusieurs produits parmi ceux affichés. Seuls les produits qui ne sont pas encore en vente sont affichés à l'écran.

À ce stade, l'utilisateur renseigne le prix auquel commencera la vente. Si la vente est descendante, le prix sera interprété comme un prix à l'unité. Si la vente est montante, il sera interprété comme un prix pour le lot de produits disponibles. Si la vente est limitée, une date limite est demandée. Il faudra veiller à ce que cette date soit valide et située dans le futur. Si la vente est descendante, un taux de diminution est requis, et celui-ci doit être strictement inférieur au prix de départ.

De cette manière, on garantit qu'une salle ne contienne qu'un seul type de vente et qu'une seule catégorie d'objets mis en vente.

4.3 Création d'une offre

Pour créer une offre, on demande d'abord à l'utilisateur de s'authentifier en sélectionnant son adresse mail dans la liste des utilisateurs affichée. Ensuite, on choisit la vente dans laquelle on souhaite faire une offre. L'interface demande après la quantité de produits vendus que l'utilisateur souhaite acheter. On récupère ensuite le stock du produit. Si la vente est montante, le stock correspond au stock initial de produits en vente. Si la vente est descendante, le stock correspond à la quantité restante du produit et si la quantité souhaitée est inférieure à ce stock, elle est soustraite à la quantité restante qui est alors mise à jour.

Si la vente est limitée, on vérifie si la date de fin n'est pas dépassée. Si elle ne l'est pas, on s'assure que la dernière offre n'ait pas été faite il y a plus de 10 minutes, auquel cas aucune offre n'est acceptée, la vente étant considérée comme terminée.

Viens pour finir, la gestion du prix de l'offre. Dans le cas d'une vente descendante, le prix de l'offre est déterminé en fonction du temps écoulé depuis le début de la vente : Prix unité = prix de départ - taux de diminution * temps écoulé. Si le prix calculé est négatif, l'offre est rejetée, car la vente est considérée comme expirée. Si la vente est révocable et que le prix calculé est inférieur au prix de revient, l'offre est rejetée car considérée comme pas assez rentable pour l'acheteur.

Dans le cas d'une vente montante, on définit le prix d'achat par le prix à l'unité saisie par l'utilisateur multiplié par la quantité voulue précédemment traitée. Si une offre est déjà présente, on vérifie si le prix d'achat est supérieur au prix d'achat de l'offre précédente. Sinon, on vérifie si le prix d'achat est supérieur au prix de départ du lot.

Si toutes les conditions sont satisfaites, on enregistre l'offre.

4.4 Déclaration des gagnants

Pour cette fonctionnalité, on affiche pour chaque vente celui ou ceux qui ont remporté un sous-lot (ou un lot). On effectue alors deux traitements spécifiques en fonction du type de la vente (montante ou descendante).
Pour une vente montante, si on veut afficher le vainqueur à un instant `t`, on cherche dans la table des offres celle qui maximise le bénéfice du vendeur, c'est-à-dire :

$$(\text{prixOffre} - \text{prixRevient}) \times \text{Quantite}$$

Si jamais l'offre est révocable et que ce bénéfice est négatif, on ne prend pas cette offre en compte.

Pour une vente descendante, on affiche toutes les offres qui ont été faites sur celle-ci. On part du principe que notre programme prend en compte uniquement les offres valides. Ainsi, pour une vente descendante, toutes les offres figurant dans la table représentent des ventes bien réalisées.

5 Organisation du projet

Apres avoir lu et analysé le sujet en groupe, on s'est réparti les taches de la facon la plus equitable qu'on jugeait.

- **Achraf :** Analyse, conception et schéma entité association.
- **Youssef :** Traduction du schéma entitée/association en schéma relationnel puis en SQL.
- **Léa :** Contraintes pendant la phase de conception puis traduction en SQL et java.
- **Baptiste :** Implémentation des fonctionnalitées en java.
- **Mariem :** Implémentation des fonctionnalitées en java.
- **Abderrahmane :** Implémentation des fonctionnalitées en java et documentation.