Projet conception d'une base de données

Ryan El Aroud, Noé Lavallée, Malo Nicolas-Coqueron, Sylvain Desbiolles,

Mohammed-Reda Belfaida, Taha Aftiss

MMIS G3 EQUIPE 5

Sommaire :

1. Analyse du problème

Nous avons premièrement listé les entités présentes dans le texte.

L'analyse du problème distingue quatre types de propriétés définissant la manière d'implanter le schéma entités / associations.

Premièrement, nous avons identifié des contraintes de valeurs listées ci-dessous :

- Prix d'une offre > mise à prix > 0
- Prix de revient, Prix de départ, Prix d'achat > 0
- Stock >= 0
- Quantité de produits dans la vente > Quantité de produits dans une enchère > 0

Ensuite, nous avons identifié les dépendances fonctionnelles sur les différents attributs :

- Id_produit => nom, catégorie, stock, prix de revient
- Catégorie, type de vente => Salle de vente
- Id_vente => id_produit, Salle
- Id_offre => id_vente, prix d'achat
- Date_offre => validité
- Email => nom, prénom, adresse

Nous avons ensuite repéré les contraintes de multiplicité :

- Une salle contient plusieurs ventes :

  Salle 1..1 -> 0..* id_vente

- Une salle ce contient qu'un type d'offre et de catégorie :

  Salle 0..* -> 1..1 type de vente, catégorie

- Une vente ne concerne qu'un produit :

  id_vente 0..* -> 1..1 id_produit

- Un utilisateur peut faire plusieurs offre pour une même vente (cas où le nombre d'offre n'est pas limité) :

  offre 0..* -> 1..1 utilisateur

- Un utilisateur est limité à 1 offre pour chaque vente (cas où le nombre d'offre est limité) :

  offre 0..1 -> 1..1 utilisateur

- Plusieurs adresse emails peuvent correspondre au même nom resp. prénom resp. adresse postale :

  adresse postale 1..1 => 0..* adresse mail

  prénom 1..1 => 0..* adresse mail

  nom 1..1 => 0..* adresse mail

Enfin, nous avons analysé les <u>contraintes contextuelles</u> :

- Si la vente est à durée limitée :

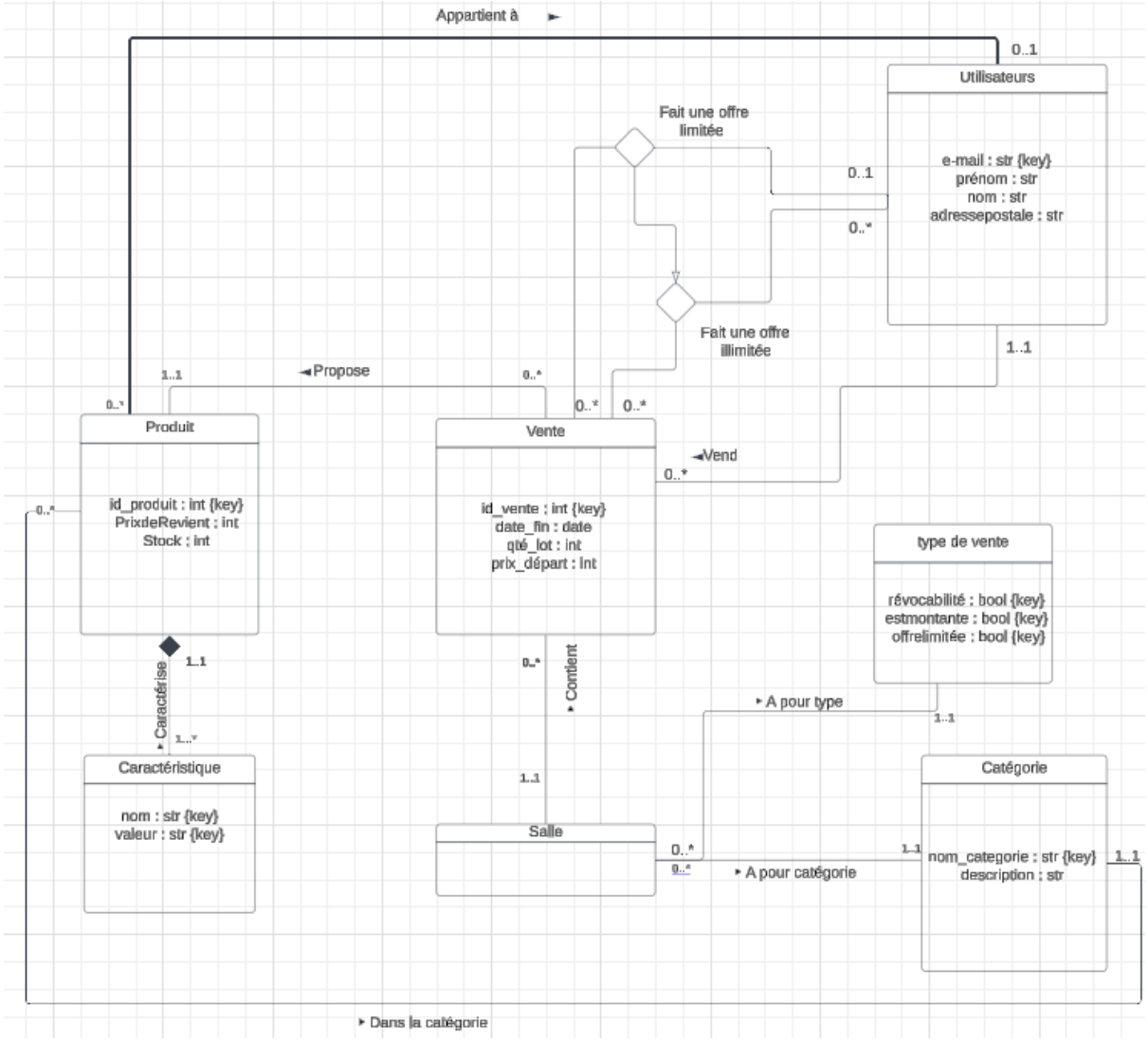
  Date offre d'achat > Date création de la vente

- Si la vente est à durée non limitée :

  Date offre d'achat < 10min après la dernière offre réalisée

2. Schéma Entités/Associations

Nous avons ainsi été apte à modéliser le problème sous forme d'un schéma Entités/Associations au format UML:



La principale difficulté rencontrée à cette étape était de modéliser l'existence de ventes limitant le nombre d'offres. Nous avons finalement fait le choix de modéliser une offre par le sous-type d'association reliant les tables utilisateurs et ventes.

3. Traduction en schéma relationnel

En s'appuyant sur ce schéma Entités/Associations, nous avons pu concevoir le schéma relationnel suivant les étapes de constructions. Les clés primaires sont soulignées, les éléments en gras lors d'une étape sont ceux modifiés suite à cette étape de traitement :

1. Etat initial :

   Utilisateurs (email, prenom, nom, adresse postale)

   Produit (id_produit, prix de revient, stock)

   Vente (id_vente, date_fin, qté_lot, prix_départ)

   Caractéristiques (nom, valeur)

   Salle ()

   Catégorie (Nom_categorie, Description)

   Type_de_vente (estMontante, Offrelimitée, Révocabilité)

2. Traitement des entités faibles :

   Utilisateurs(email, prenom, nom, adresse postale)

   Produit(id_produit, prix de revient, stock)

   Vente(id_vente, date_fin, qté_lot, prix_départ)

   Caractéristiques(nom**, id_produit**, valeur)

   Salle()

   Catégorie(Nom_categorie, Description)

   Type_de_vente (estMontante, Offrelimitée, Révocabilité)

3. Traitement des associations avec cardinalité 1..1

   Utilisateurs(email, prenom, nom, adresse postale)

   Produit(id_produit, prix de revient, stock, **nom_categorie**)

   Vente(id_vente, **id_produit**, date_fin, qté_lot, prix_départ, email, **catégorie, estMontante, Offrelimitée, Révocabilité**)

   Caractéristiques(nom**,** id_produit, valeur)

   Salle(**catégorie, Révocabilité, EstMontante, OffreLimitée**)

   Catégorie(Nom_categorie, Description)

4. Traitement des associations avec cardinalité 0..1

   Utilisateurs(email, prenom, nom, adresse postale)

   Produit(id_produit, prix de revient, stock, nom_categorie)

   Vente(id_vente, id_produit, date_fin, qté_lot, prix_départ, email, catégorie, estMontante, Offrelimitée, Révocabilité)

   Caractéristiques(nom**,** id_produit, valeur)

   Salle(catégorie, Révocabilité, EstMontante, OffreLimitée)

   Catégorie(Nom_categorie, Description)

   Appartient_à(**email**, **id_produit**)

5. Traitement des associations avec cardinalité ?..*

   Utilisateurs(email, prenom, nom, adresse postale)

   Produit(id_produit, prix de revient, stock, nom_categorie)

   Vente(id_vente, id_produit, date_fin, qté_lot, prix_départ, email, catégorie, estMontante, Offrelimitée, Révocabilité)

   Caractéristiques(nom**,** id_produit, valeur)

Salle(catégorie, Révocabilité, EstMontante, OffreLimitée)

Catégorie(Nom_categorie, Description)

Appartient_à(email, id_produit)

Offre(date, id_offre, prixAchat,qtéAcheté,email, id_vente)

Une fois que nous avons eu notre schéma relationnel, il a fallu réaliser le traitement des redondances et l'ajout des clés étrangères :

La seule redondance que nous avons identifiée dans notre schéma relationnel est la présence de l'attribut catégorie dans la table vente, qui pouvait être déduit par la présence de id_produit => catégorie.

Schéma relationnel sans redondance :

- Utilisateurs(email, prenom, nom, adresse postale)
- Produit(id_produit, prix de revient, stock, nom_categorie)
- Vente(id_vente, id_produit, date_fin, qté_lot, prix_départ, email, estMontante, Offrelimitée, Révocabilité)
- Caractéristiques(nom, id_produit, valeur)
- Salle(catégorie, Révocabilité, EstMontante, OffreLimitée)
- Catégorie(Nom_categorie, Description)
- Appartient_à(email, id_produit)
- Offre(date, id_offre, prixAchat,qtéAcheté,email, id_vente)

Ensuite, nous avons précisé les clés étrangères :

- Utilisateurs(email, prenom, nom, adresse postale)
- Produit(id_produit, prix de revient, stock, nom_categorie)

  nom_catégorie référence catégorie.nom_catégorie
- Vente(id_vente, id_produit, date_fin, qté_lot, prix_départ, email, estMontante, Offrelimitée, Révocabilité)

  Id_produit référence produit.id_produit

  email référence utilisateur.email

  estMontante référence salle.EstMontante

  Offrelimitée référence salle.OffreLimitée

  Révocabilité référence salle.OffreLimitée
- Caractéristiques(nom**,** id_produit, valeur)

  Id_produit référence produit.id_produit
- Salle(catégorie, Révocabilité, EstMontante, OffreLimitée)

  Catégorie référence catégorie.nom_description
- Catégorie(nom_categorie, Description)
- Appartient_à(email, id_produit)

  email référence utilisateur.email

  id_produit référence produit.id_produit
- Offre(date, id_offre, prixAchat,qtéAcheté,email, id_vente)

  email référence utilisateur.email

  id_vente référence vente.id_vente

Ensuite, nous nous sommes intéressés aux formes normales :

Pour la table Caractéristiques, l'attribut valeur dépend pleinement des deux clés primaires nom et id_produit, la troisième forme normale est donc vérifiée.

Toutes les autres tables sont dans l'un des deux cas suivants :

- La table possède une unique clef
- La table ne possède pas d'attribut non clef

Ainsi, cela nous permet de conclure que chacune de nos tables vérifie la troisième forme normale.

La 3FNBCK est également vérifiée dans notre schéma relationnel étant donné que les membres de gauche de nos dépendances fonctionnelles sont uniquement des clés des tables en question.

4. Analyse des fonctionnalités et implantation en SQL2

4.1. Création d'une salle de vente et sélection de produis déjà disponibles à la vente et permettant le choix du type d'enchère.

Pour cette partie, il a fallu tout d'abord regarder quelles catégories d'articles sont disponibles.

[SELECT nom FROM Categorie]

On demande ensuite à l'utilisateur de sélectionner la catégorie pour laquelle il veut créer une salle via des « input » lancés sur le terminal par le programme Java. On interroge la base de données pour considèrer un résultat différent si une salle ayant les mêmes caractéristiques que celles demandées par l'utilisateur existe déjà :

[SELECT * FROM Salle WHERE categorie = ? AND estmontante = ? AND revocabilite = ? AND offre_limite = ?]

Ainsi, si le résultat est nul – Une telle salle n'existe pas encore -, on crée une telle salle :

[INSERT INTO Salle VALUES ('?', ?, ?, ?)]

Sinon, on n'en crée pas.

Enfin, on affiche les ventes reliées à cette salle :

[SELECT id FROM Vente JOIN Produit ON (Vente.idproduit = Produit.id_produit) WHERE Produit.nom_categorie ='" + categorie +"' AND revocabilite = ? AND estMontante = ? AND offrelimite = ?]

4.2. Création d'une offre à une vente existante

Le processus d'offre se divise en deux parties principales. La première consiste en la vérification de l'offre pour s'assurer qu'elle est valide et conforme aux critères définis par le système. La seconde partie est la gestion de l'ajout de l'offre à la base de données si elle est acceptée.

A. Vérification des informations

Avant d'ajouter une offre, plusieurs vérifications sont effectuées : Vérification de la vente : On commence par vérifier que l'ID de la vente fournie par l'utilisateur existe dans la base de données. Cette vérification se fait par la requête suivante :

SELECT * FROM Vente WHERE id = <id_vente>;

Vérification de l'utilisateur : On s'assure également que l'utilisateur qui propose l'offre est inscrit dans la base de données en utilisant son adresse e-mail. La requête utilisée est : SELECT * FROM Utilisateur WHERE Email = <email>.

B. Vérification de la conformité de l'offre

Une fois que la vente et l'utilisateur sont validés, l'offre est ensuite comparée aux critères définis par le type de la vente. Vérification de la quantité disponible : Peu

importe le type de vente, la quantité demandée par l'utilisateur doit être inférieure ou égale à la quantité restante pour la vente. Si la quantité demandée est supérieure à la quantité disponible, l'offre est rejetée. Vérification du prix (pour les ventes montantes) : Si la vente est de type "montante" (les prix augmentent au fur et à mesure des enchères), l'offre de l'utilisateur doit être supérieure à la dernière offre enregistrée pour cette vente. Pour cela, la dernière offre est récupérée avec la requête suivante :

SELECT * FROM Offre WHERE idvente = <id_vente> ORDER BY dateoffre DESC LIMIT 1;

Vérification des offres déjà faites (pour les ventes limitées) : Si la vente est de type « limitée » (l'utilisateur peut faire une seule offre), on vérifie si l'utilisateur a déjà proposé une offre sur cette vente en utilisant la requête suivante :

SELECT * FROM Offre WHERE idvente = <id_vente> AND email = '<email>';

C. Ajout de l'offre

Si toutes les vérifications passent avec succès, l'offre peut être ajoutée à la base de données. Les informations de l'offre (quantité, prix, utilisateur, etc.) sont insérées dans la table Offre avec une requête d'insertion appropriée.

Remarques :

Vente montante : Pour ce type de vente, le prix de l'offre doit toujours être supérieur à celui de la dernière offre. Cela assure que l'enchère progresse et évite qu'un utilisateur propose un prix inférieur à celui déjà proposé.

Vente limitée : Pour ce type de vente, un utilisateur ne peut proposer qu'une seule offre. Toute tentative d'offrir plusieurs fois sur la même vente sera rejetée.

4.3. Lister les enchères terminées et déterminer les gagnants d'une enchère

Pour cette fonctionnalité, nous avons développé un processus permettant d'identifier les enchères terminées et de désigner leurs gagnants en fonction des offres soumises. Ce processus repose sur des requêtes SQL exécutées via Java, et est composé de plusieurs étapes :

A. Lister les enchères terminées

Les enchères terminées sont identifiées selon deux critères principaux :

Pour les enchères avec une limite d'offre, la date de fin de la vente doit être passée.

Pour les enchères sans limite d'offre, il faut vérifier que plus de 10 minutes se sont écoulées depuis la dernière offre enregistrée.

La requête SQL utilisée est la suivante :

```
SELECT * FROM VENTE
JOIN PRODUIT ON VENTE.IDPRODUIT = PRODUIT.ID_PRODUIT
WHERE ( (VENTE.OFFRELIMITÉ <> 0 AND VENTE.DATEFIN < SYSDATE)
    OR (VENTE.OFFRELIMITÉ = 0 AND (SYSDATE + INTERVAL '10' MINUTE) >
        (SELECT MAX(DATEOFFRE) FROM OFFRE WHERE OFFRE.IDVENTE = VENTE.ID) ) );
```

Les résultats obtenus sont ensuite parcourus pour extraire les identifiants des ventes terminées.

B. Déterminer les gagnants d'une enchère

Une fois les enchères terminées identifiées, les étapes suivantes sont réalisées pour chaque enchère afin de désigner les gagnants :

Récupérer la quantité totale disponible

Une requête permet de récupérer la quantité totale d'articles disponibles pour l'enchère :

SELECT QTÉLOT FROM VENTE WHERE ID = ?;

Récupérer et trier les offres

Les offres soumises pour cette enchère sont récupérées et triées par prix unitaire décroissant (prix d'achat divisé par la quantité achetée) :

SELECT IDOFFRE, PRIXACHAT, QTEACHETEE FROM OFFRE

WHERE IDVENTE = ?;

Distribuer les lots

Les lots disponibles sont attribués en fonction des offres, en donnant priorité aux offres avec le prix unitaire le plus élevé. Si une offre demande une quantité supérieure à celle restante, elle est ignorée.

Identifier les gagnants

Les identifiants des offres gagnantes sont utilisés pour récupérer les emails des gagnants :

SELECT EMAIL FROM OFFRE WHERE IDOFFRE IN (?);

5. Bilan du projet

Une certaine fierté se dégagera de ce projet où nous avons réussi à concilier six individus autour d'un travail commun en une durée courte. L'organisation a été compliquée car nous avons fait le choix de ne pas commencer l'implantation des fonctionnalités sur Oracle tant que notre schéma relationnel n'était pas terminé. L'accent a été mis sur cet aspect du projet qui nous a demandé beaucoup d'investissement de manière à créer un schéma relationnel sur lequel tout le monde était d'accord. Le problème était donc que la conception du schéma relationnel a été assez chronophage au vu des divergences de points de vue des membres du groupe. Une fois celui-ci terminé, nous avons pu paralléliser les tâches de manière à avancer plus vite tout en continuant à faire des allers-retours avec le schéma relationnel pour répondre aux problématiques rencontrées lors de l'implantation des fonctionnalités Nous aurions tout de même souhaité plus de temps pour implanter des fonctionnalités optionnelles afin de prendre plus de plaisir.

Ci-dessous une répartition temporelle grossière des différentes étapes :

Semaine 1 : Réalisation du Schéma Entités Associations

Semaine 2 : Réalisation du Schéma Relationnel

Semaine 3 : Implantation de la base de données sur le serveur et implantation des fonctionnalités.

6. Mode d'emploi du démonstrateur

Pour lancer le démonstrateur, se placer dans le répertoire src/Java :

```
javac Demonstrateur.java
java Demonstrateur
```

Ensuite, suivre les indications du menu.

Noter que pour le choix de la catégorie lors de la création de la salle, il faut taper dans le terminal le nom exact de la salle.

FIN