Projet base de données

ALTIERI Aubin CHABANOL Ambroise
CHARLES-MENNIER Matéo CHEN Etienne
ZOUGARI Rania

Novembre 2024

1 Modélisation du problème

1.1 Analyse du problème

Nous avons commencé par extraire les propriétés élémentaires, les dépendances fonctionnelles reliant ces propriétés, ainsi que tous les autres types de contraintes. Elles sont résumées dans ce tableau :

DF	Cont. Valeur	Cont. Multiplicite	Cont. textuelle
Mail → Nom, Prénom, Adresse	Nom et Prénom : non vides		
IdProduit → NomProduit, PrixRevient, Stock	PrixRevient ≥ 0 Stock ≥ 0 NomProduit : non vide		
NomCaractéristiques → Valeur,IdProduit	Valeur : non vide		
NomCatégorie → DescriptionCategorie	Description : non vide		
IdSalle → TypeTemps, TypeSens, EstRevocable, LimiteOffre, NomCategorie	TypeTemps : soit à temps libre soit à durée limitée EstRevocable : bool TypeSens: montante ou descendante LimiteOffre: bool	Une Salle peut avoir plusieurs Ventes, mais une Vente ne peut appartenir qu'à une seule Salle.	
IdVente → IdSalle, IdProduit,Mail		Chaque Vente concerne un seul Produit et inversement.	Une Vente ne peut commencer que si le Stock du Produit associé est non nul.
IdOffre → QuantitéOffre, DateOffre, HeureOffre, PrixOffre, Mail, IdVente	Prix ≥ 0 $0 \leq$ Quantité $\leq$ Stock		Les Offres doivent respecter les règles spécifiques au Type de Vente (montante ou descendante).

Dans ce tableau, toutes les contraintes de valeurs ne sont pas mises notamment celles liés au clé et clés étrangères. Ces deux attributs sont toujours uniques et non null.

1.2 Schéma Entités/Associations

Grâce à ce tableau, nous avons élaboré le schéma suivant :

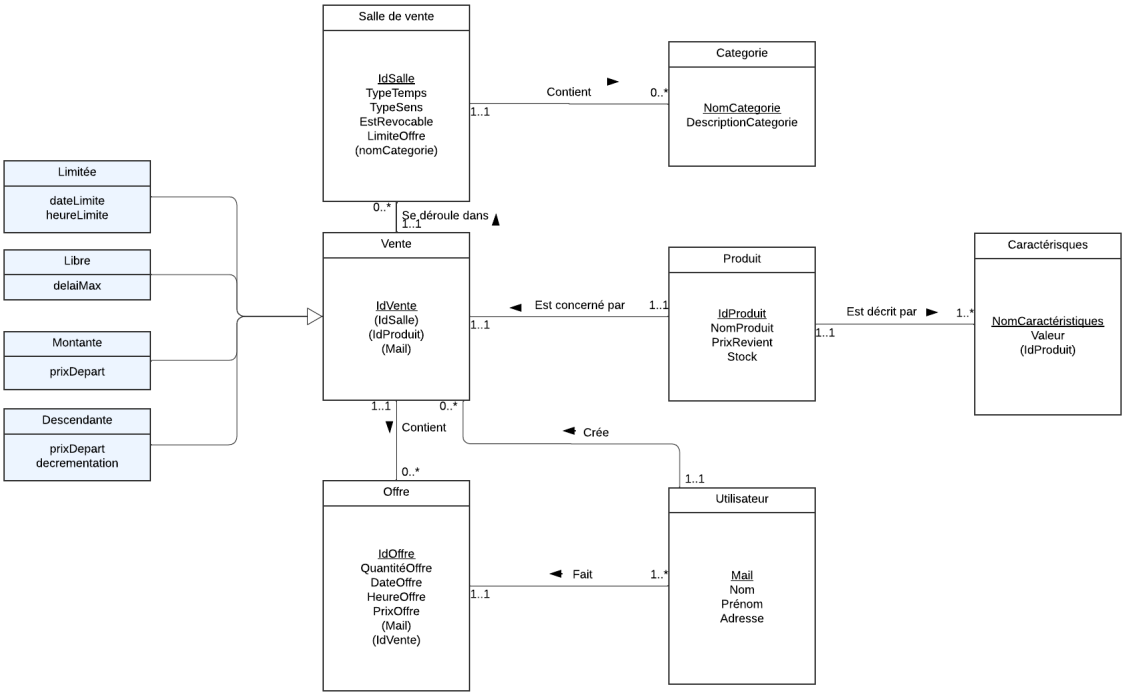

Figure 1: Schéma entité association

1.2.1 Choix de modélisation

Nous avons fait le choix d'associer un produit à une vente. Plus précisément, un produit représente un bien à acheter et non seulement le nom. Donc il peut exister plusieurs produits portant le même nom.

Les particularités d'une vente (Limitée, Montante, ...) sont des sous-types d'entité. Quand l'utilisateur crée sa vente, il doit spécifier la salle de vente dans laquelle elle se déroule, puisque dans celle-ci les caractéristiques y sont décrites. Nous n'avons donc pas besoin de valeurs par défaut pour une vente.

Nous avons créé une table des caractéristiques des produits pour éviter de faire un attribut qui aurait été un tableau dans le produit ce qui n'aurait pas été FN1.

2 Implémentation de la base de données

2.1 Choix de traduction et problèmes rencontrés

Lors de cette étape, nous avons rencontré quelques problèmes. En effet, faire des ALTER TABLE à chaque modification ne nous permettait plus d'avoir une vue d'ensemble finale des tables. Nous avons donc enlevé toute les tables existantes pour les recréer dans leurs états finaux. Ceci explique la présence de l'instruction DROP en début de fichier de *createTableV2.sql*

Bien sûr, l'ordre de création des tables est importante. Il faut les créer à partir de celles qui ont le moins de dépendances vers celles qui en ont le plus.

Nous avons fait le choix d'utiliser trois types de bases :

- **INT** pour toute les valeurs numériques sauf les dates
- **DATE** pour la date et l'heure.
- **VARCHAR** avec une limite dépendant des attributs et de la longueur estimée. En effet, le nom de la catégorie n'a pas la même longueur que sa description.

La majorité des contraintes ont été implémentées dans le fichier de création, tant que celles-ci étaient internes à la table. Ce sont principalement le contraintes de valeur. Par exemple, la contrainte QuantitéOffre $\leq$ Stock sera implémenté par Java et non dans les requêtes SQL.
Le format **DATE** de SQL, nous permet d'enlever l'attribut heure et de le concaténer avec date.

2.2 Formes normales des relations

Prenons comme premier exemple la relation Produit :
R(IdProduit,NomProduit,PrixRevient,Stock)

Produit	IdProduit	NomProduit	PrixRevient	Stock
	1	Pomme	5	20
	2	Banane	10	50

Figure 2: Relation Produit

Les dépendances fonctionnelles identifiées sont :
IdProduit $\rightarrow$ NomProduit
IdProduit $\rightarrow$ PrixRevient
IdProduit $\rightarrow$ Stock

Les attributs de pouvant contenir que des valeurs atomiques et les contraintes de non nullité étant imposée, la relation est 1FN.

Le calcul des clefs donne : C = {IdProduit}

Attribut non clef	Attribut clef
NomProduit	IdProduit
PrixRevient	
Stock	

Les attributs non clefs dépendent pleinement de la clé, d'où la 2FN. Les formes 3FN et 3FNBCK découlent aussi simplement. Evidemment, ce n'était pas le cas au début et il a fallut diviser certaines table en deux, ou encore déplacer certains attributs, pour y parvenir.

3 Implémentation des fonctionalités

Le simulateur est implémenté par niveaux. Chaque niveau a une fonction *handleNomNiveau()* qui attend en entrée les différentes commandes possibles à ce niveau.

Table 1: tableau simulation

Niveau	Nom	Accès	Visualisation	Création
1	Hub	Salles de ventes	Utilisateurs	• Catégorie • Utilisateur • Salle De Vente
2	Salle de Vente	Hub, Vente	• Les ventes dans la salle • Caractéristiques de cette salle	Une vente
3	Vente	Sallede Vente Offre	Les offres et caractéristiques de la vente	Une offre

4 Executer le script

4.1 Creation des tables

Le fichier *creerTable.sql* initialise les tables, en droppant les tables précédemment crées.

4.2 Peupler table

Le fichier *peuplerTable.sql* peuple la table avec des utilisateurs, des caractérisitques, des produits, des ventes, des salles de vente et une offre.

4.3 Java

Au niveau du dossier java, Main.java est à exécuter.Il existe un MakeFile qui permet de compiler et d'exécuter.