

Documentation Projet BD

KAMAL Mohyieddine (kamalm), LAHMOUZ Amine (lahmouza), LE BIGOT Evan (lebigote),
LEMOINE Olivier (lemoineo), MARAH Mohamed Taha (marahm)

November 30, 2025

1 Analyse du problème

Dépendances fonctionnelles

On fait le choix d'inclure le numéro de téléphone et l'adresse email dans les coordonnées de contact du producteur.

- idProducteur → nom, adresse, latitude, longitude, mail, telephone

Identification du produit. On considère les contenants comme des produits, on a donc des producteurs de contenants.

- idProduit → nom, categorie, description, idProducteur

On fait le choix de distinguer les produits en tant que tels des items qui peuvent être achetés en magasin. Un item peut être soit un article, soit un contenant (cf typeItem) les idArticle et idContenant valent NULL selon la valeur de typeItem.

- idItem → typeItem, idArticle, idContenant

On identifie un article :

- idArticle → idProduit, modeConditionnement, poids, prixAchatProducteur, prixVenteClient, delaiDisponibilite

On identifie un arrivage de lot d'articles. La quantité disponible comment à la quantité initiale et va évoluer selon les commandes clients. On indique un pourcentage de réduction du prix quand le produit arrive à sa date de péremption, qu'on pourra appliquer dans la partie applicative :

- idLot → idArticle, dateReception, quantiteInitiale, quantiteDisponible, datePeremption, typePeremption, prixLot, pourcentageReduction

On associe les contenants en ajoutant une valeur unite qui indique si la capacité est en kg ou en litre :

- idContenant → typeContenant, capacite, unite, reutilisable, stock, prixVente

On peut perdre soit une partie d'un lot, soit des contenants. On a alors ajouté deux id, idLot et idContenant qui valent NULL ou un id selon le type d'item perdu :

- $\text{idPerte} \rightarrow \text{idLot}, \text{idContenant}, \text{datePerte}, \text{naturePerte}, \text{quantitePerdue}$

Un client a pour données personnelles son nom, prénom, mail et numéro de téléphone

- $\text{idClient} \rightarrow \text{nom}, \text{prenom}, \text{mail}, \text{telephone}$

On choisit d'identifier les adresses avec un identifiant à part :

- $\text{idAdresse} \rightarrow \text{idClient}, \text{adressePostale}, \text{pays}, \text{latitude}, \text{longitude}$

Le prix de la commande est le prix incluant les frais de livraison. On indique le mode de paiement et le mode de récupération et on a l'identifiant de l'adresse de la livraison si le mode de récupération est la livraison :

- $\text{idCommande} \rightarrow \text{idClient}, \text{idAdresseLivraison}, \text{dateCommande}, \text{statut}, \text{modePaiement}, \text{modeRe-cuperation}, \text{fraisLivraison}, \text{montantTotal}$

Contraintes de multiplicité

Un producteur peut avoir plusieurs activités :

- $\text{idProducteur} - >> \text{nomActivite}$

On modélise la saisonnalité des intervalles de temps par une date de début et de fin de disponibilité :

- $\text{idProduit} - >> \text{dateDebut}, \text{dateFin}$

Un produit peut avoir plusieurs caractéristiques (allergènes, bio...)

- $\text{idProduit} - >> \text{caracteristiques}$

Un client peut avoir plusieurs adresses de livraison :

- $\text{idClient} - >> \text{idAdresse}$

Pour chaque ligne de commande, on décrit l'article concerné, la quantité et le prix unitaire appliqué pour cet article (selon la réduction, le prix peut varier selon l'item).

- $\text{idCommande} - >> \text{idLigne}, \text{idItem}, \text{quantite}, \text{prixUnitaireApplique}$

Contraintes de valeur

$\text{poids} > 0$

$\text{prixAchatProducteur} > 0$

$\text{prixVenteClient} > 0$

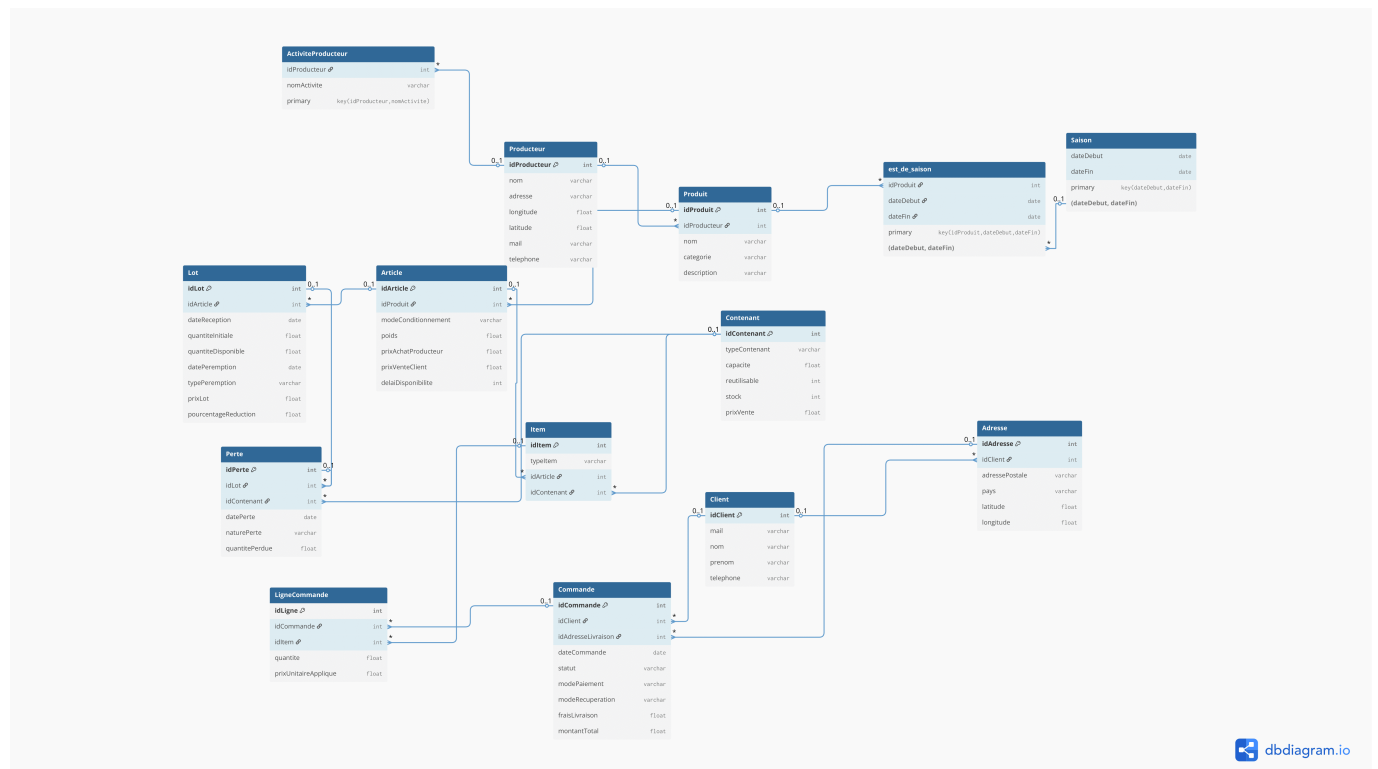
$\text{quantiteInitiale} > 0$

$\text{quantiteDisponible} > 0$

prixLot > 0
 0 < pourcentageReduction < 100
 prixVente > 0
 stock > 0
 quantitePerdue > 0
 fraisLivraison > 0
 montantTotal > 0
 prixUnitaireApplique > 0
 typeItem ∈ {"ARTICLE", "CONTENANT"}
 typeDatePeremption ∈ {"DLC", "DLUO"}
 modePaiement ∈ {"Carte", "Espèces"}
 modeRecuperation ∈ {"Livraison", "Boutique"}
 typeItem ∈ {"ARTICLE", "CONTENANT"}
 statut ∈ {"En preparation", "Prete", "En livraison", "Livree", "Annulee"}
 reutilisable ∈ {0, 1}
 unite ∈ {"kg", "L"}

2 Passage en modèle entité/association

Le schéma est également disponible dans le projet en grand format



3 Schéma relationnel

On décrit ici le schéma relationnel associé au modèle entité/association qui précède, avec leur forme normale.

- Producteur(idProducteur, nom, adresse, longitude, latitude, mail, telephone), 3FNBCK

- ActiviteProducteur(idProducteur, nomActivite), 3FNBCK
 - idProducteur référence Producteur
 - Produit(idProduit, idProducteur, nom, categorie, description), 3FNBCK
 - idProducteur référence Producteur
 - Contenant(idContenant, typeContenant, capacite, unite, reutilisable, stock, prixVente), 3FNBCK
 - Article(idArticle, idProduit, modeConditionnement, poids, prixAchatProducteur, prixVenteClient, delaiDisponibilite), 3FNBCK
 - idProduit référence Produit
 - Item(idItem, typeItem, idArticle, idContenant), 3FNBCK
 - idArticle référence Article
 - idContenant référence Contenant
 - Lot(idLot, idArticle, dateReception, quantiteInitiale, quantiteDisponible, datePeremption, typePeremption, pourcentageReduction), 2FN car quantiteDisponible dépend de quantiteInitiale
 - idArticle référence Article
 - Perte(idPerte, idLot, idContenant, datePerte, naturePerte, quantitePerdue), 3FNBCK
 - idLot référence Lot
 - idContenant référence Contenant
 - Client(idClient, mail, nom, prenom, telephone), 3FNBCK
 - Adresse(idAdresse, idClient, adressePortale, pays, latitude, longitude), 3FNBCK
 - idClient référence Client
 - Commande(idCommande, idClient, idAdresseLivraison, dateCommande, status, modePaiement, modeRecuperation, fraisLivraison, montantTotal), 3FNBCK
 - idClient référence Client
 - idAdresseLivraison référence Adresse
 - LigneCommande(idLigne, idCommande, idItem, quantite, prixUnitaireApplique), 3FNBCK
 - iditem référence Item
 - idCommande référence Commande
- Contrainte non représentée : Une commande ne peut avoir qu'une seule ligne par article
- Saison(dateDebut, dateFin), 3FNBCK
 - est_de_saison(idProduit, dateDebut, dateFin), 3FNBCK

4 Analyse des fonctionnalités

4.1 Fonctionnalité 1

La commande va d'abord détecter de quel type d'item il s'agit :

```
SELECT typeItem, idArticle, idContenant FROM Item WHERE idItem = [id]
```

Si l'item est un article, on vérifie si l'article est hors saison :

```
SELECT 1 FROM est_de_saison e JOIN Article a ON e.idProduit = a.idProduit  
WHERE a.idArticle = [id] AND SYSDATE NOT BETWEEN e.dateDebut AND e.dateFin
```

On vérifie si l'article est en stock :

```
SELECT COALESCE(SUM(quantiteDisponible), 0) FROM Lot WHERE idArticle=[id]
```

Sinon, on vérifie si l'article est sur commande :

```
SELECT delaiDisponibilite FROM Article WHERE idArticle = [id]
```

Pour les contenant, on vérifie simplement que le stock est suffisant :

```
SELECT stock FROM Contenant WHERE idContenant = [id]
```

Récupération du prix :

```
SELECT typeItem, idArticle, idContenant FROM Item WHERE idItem = [id]
```

Si il y a livraison, on calcule les frais de livraison, d'abord selon le poids (uniquement pour les articles), puis selon la distance.

Pour le poids, on récupère l'adresse et le poids de l'article :

```
SELECT a.poids, a.modeConditionnement, ad.pays FROM Article a, Adresse ad  
WHERE a.idArticle=[idArticle] AND ad.idClient=[idClient]
```

Les frais de livraison pour le poids valent poids*tarif, avec le tarif calculé selon le pays :

Si le pays de livraison est la france, tarif = 0.5

Si on est en DOM-TOM, tarif = 5

Sinon, tarif = 8

Pour la distance, on récupère la latitude et la longitude de l'adresse :

```
SELECT pays, latitude, longitude FROM Adresse  
WHERE idClient = [idClient]
```

Les frais de livraison pour la distance valent distance*prixKm, avec prixKm calculé selon le pays:

Si le pays de livraison est la france, prixKm = 0.05

Si on est en DOM-TOM, prixKm = 0.15

Sinon, prixKm = 0.2

On crée la commande :

```
INSERT INTO Commande (idClient, dateCommande, statut, modeRecuperation, modePaiement,  
montantTotal, fraisLivraison)  
VALUES ([idClient], SYSDATE, 'En préparation', [modeRecuperation], [modePaiement], [mon-  
tantTotal], [fraisLivraison])
```

Et pour chaque ligne :

```
INSERT INTO LigneCommande (idCommande, idItem, quantite, prixUnitaireApplique)  
VALUES ([idCommande], [idItem], [quantite], [prixUnitaireApplique])
```

4.2 Fonctionnalité 2

Pour la liste d'alerte des gérants :

```
SELECT p.nom, l.datePeremption FROM Lot l JOIN Article a ON l.idArticle=a.idArticle  
JOIN Produit p ON a.idProduit=p.idProduit  
WHERE l.datePeremption < SYSDATE+7 AND l.quantiteDisponible>0
```

Pour les articles soumis à une réduction, elle est automatiquement appliquée, la commande prend l'article qui arrivera le plus vite à péremption :

```
SELECT pourcentageReduction FROM Lot  
WHERE idArticle=[idArticle] AND quantiteDisponible>0  
ORDER BY datePeremption ASC
```

4.3 Fonctionnalité 3

Pour la gestion d'une commande, on commence par verrouiller la commande :

```
SELECT idCommande, statut FROM Commande  
WHERE idCommande = [idCommande] FOR UPDATE
```

Ensuite, on verrouille le lot pour les commandes d'articles :

```
SELECT idLot, quantiteDisponible FROM Lot  
WHERE idArticle = [idArticle] AND quantiteDisponible > 0  
ORDER BY datePeremption ASC FOR UPDATE
```

Enfin, on décrémente le stock :

```
UPDATE Lot SET quantiteDisponible = quantiteDisponible - [quantite] WHERE idLot=[idLot]
```

Même chose pour les contenants :

```
UPDATE Contenant SET stock = stock - [quantite] WHERE idContenant = [idContenant]
```

La commande devient alors prête :

```
UPDATE Commande SET statut = 'Prête' WHERE idCommande = [idCommande]
```

Pour expédier une commande, on met à jour le statut :

```
SELECT statut, modeRecuperation FROM Commande WHERE idCommande = [idCommande]  
FOR UPDATE  
UPDATE Commande SET statut = 'En livraison' WHERE idCommande = [idCommande]
```

Pour indiquer qu'une commande a été livrée, même chose :

```
SELECT statut, modeRecuperation FROM Commande WHERE idCommande = [idCommande]  
FOR UPDATE  
UPDATE Commande SET statut = 'Livrée' WHERE idCommande = [idCommande]
```

De même pour une commande récupérée en boutique :

```
SELECT statut, modeRecuperation FROM Commande WHERE idCommande = [idCommande]  
FOR UPDATE  
UPDATE Commande SET statut = 'Récupérée' WHERE idCommande = [idCommande]
```

Et enfin, pour annuler une commande :

```
SELECT statut, modeRecuperation FROM Commande WHERE idCommande = [idCommande]  
FOR UPDATE  
UPDATE Commande SET statut = 'Annulée' WHERE idCommande = [idCommande]
```