# Exercice TD3 : Java & PostgreSQL (JDBC) - Normalisation de données et gestion de relation ManyToMany

**Date d'émission : Jeudi 15 janvier 2026 | Correction : Mardi 20 janvier 2026**

Nous pouvons remarquer qu'une entité "ingrédient" appartient à une entité "plat" ce qui veut dire qu'un ingrédient avec un même nom peut être inséré plusieurs fois dans la base de données mais avec différents identifiants et associés à différents plats.

Par exemple, reconsidérons l'exemple du plat "Salade fraîche" composé des ingrédients : "Laitue" et "Tomate". Voici comment sont enregistrés les données dans la base de données pour correspondre à ce besoin :

Dans la table Dish :

| id | name | dish_type |
|---|---|---|
| 1 | Salade fraîche | START |

Dans la table Ingredient :

| id | name | price | category | id_dish |
|---|---|---|---|---|
| 1 | Laitue | 800.00 | VEGETABLE | 1 |
| 2 | Tomate | 600.00 | VEGETABLE | 1 |

Imaginons maintenant que nous voulons ajouter un nouveau plat "Salade de tomates", dont l'ingrédient est également "Tomate". Nous ne pouvons dissocier l'ingrédient (id=2, name="Tomate", id_dish=1) pour l'associer au nouveau plat mais nous allons devoir devoir créer une nouvel ingrédient pour correspondre au "Tomate", donc avec les même propriétés, mais seule valeur de id_dish change en celle correspondant à l'identifiant du nouveau plat "Salade de tomates".

Autrement dit, voici comment sont enregistrés les données dans la base de données :

Dans la table Dish :

| id | name | dish_type |
|---|---|---|
| 6 | Salade de tomates | STARTER |

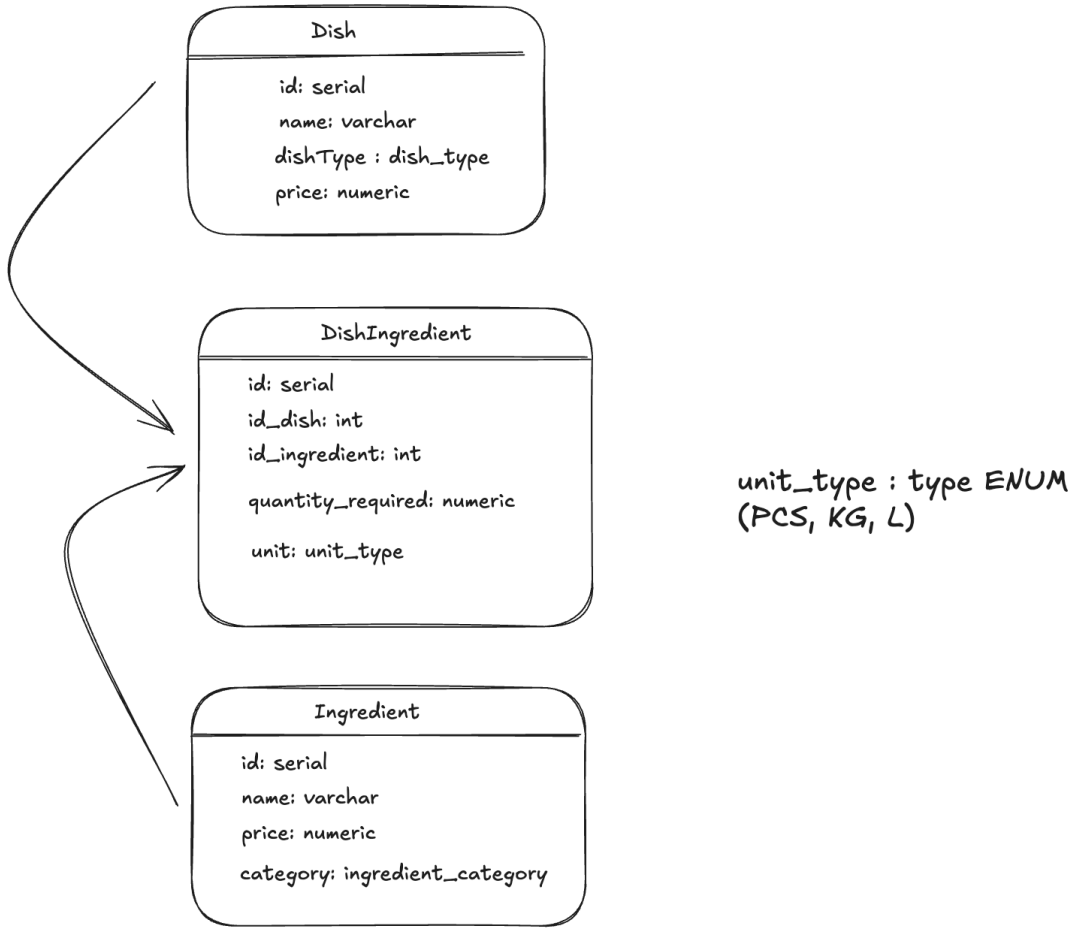
Dans la table Ingredient :

| id | name | price | category | id_dish |
|---|---|---|---|---|
| 6 | Tomate | 600.0 | VEGETABLE | 6 |

Hormis les remarques déjà effectuées plus tôt, un problème réside dans la mise à jour des propriétés de cet ingrédient, notamment si on veut mettre à jour le prix des tomates alors il faut mettre à jour toutes les lignes d'ingrédients avec un nom "Tomate", où il suffit qu'on ait écrit "tomate" au lieu de "Tomate" ou encore "Tomates" (au pluriel) pour que ça ne marche pas (en plus des problème de performance).

Pour gérer ceci plus efficacement, il faut donc que nous ajustons les relations des tables au sein de notre base de données où nous allons devoir gérer une relation ManyToMany et appliquer ce que l'on appelle la **NORMALISATION DE DONNÉES (**https://www.datacamp.com/fr/tutorial/normalization-in-sql)

Pour cela, il faut que vous ajouter une nouvelle table de jointure entre Ingredient et Dish que nous allons appeler "DishIngredient" et qui va avoir la structure suivante :




Dorénavant, un ingrédient est enregistré comme un seul élément que vous pouvez directement associer à plusieurs plats en même temps à travers la table de jointure DishIngredient.

Remarquez que dans cette nouvelle table, nous spécifions également la quantité requise d'un ingrédient pour composer un plat, ainsi que l'unité utilisée pour la quantité. Voici quelques exemples pour comprendre cela :

- Une salade fraîche est composée de laitue et de tomate, mais il faut être plus précis sur cette composition. On peut par exemple dire qu'il faut une pièce de laitue et 0,25 KG de tomates, en particulier car de façon générale, lorsqu'on achète ces ingrédients, on achète la laitue par pièce et les tomates par unité de kilogramme.

- Un autre exemple, pour le plat poulet grillé, enrichissons un peu les ingrédients : il faut 0,5 KG de poulet et 0,15 L d'huile, encore une fois car lorsque nous achetons du poulet, on achète par unité en KG et de l'huile en unité en litre (L).

1. Effectuer les modifications nécessaires pour normaliser les données auprès de la base de données Postgresql et écrire les scripts correspondant à ces modifications dans un nouveau fichier "**new_schema.sql**". En particulier, il faut que vous ajoutez, si ce n'est pas fait un prix de vente à un plat, qui est optionnel (peut être null).

2. Insérez les données suivantes
   a. Dans la table DishIngredient :

| id | id_dish | id_ingredient | quantity_required | unit |
|---|---|---|---|---|
| 1 | 1 | 1 | 0.20 | KG |
| 2 | 1 | 2 | 0.15 | KG |
| 3 | 2 | 3 | 1.00 | KG |
| 4 | 4 | 4 | 0.30 | KG |
| 5 | 4 | 5 | 0.20 | KG |

   b. Dans la table Dish (pour la mise à jour des prix) :

| id | name | dish_type | selling_price |
|---|---|---|---|
| 1 | Salade fraîche | START | 3500.00 |
| 2 | Poulet grillé | MAIN | 12000.00 |
| 3 | Riz aux légumes | MAIN | NULL |
| 4 | Gâteau au chocolat | DESSERT | 8000.00 |
| 5 | Salade de fruits | DESSERT | NULL |

3. Effectuez les modifications nécessaires dans DataRetriever pour maintenir les méthodes existantes (en y ajoutant de nouvelles méthodes car il y a une nouvelle entité à sauvegarder) notamment la récupération de données (tout ce qui est méthode qui commence par **find** et la sauvegarde de données tout ce qui est méthode qui commence par **save**).

4. Mettre à jour la méthode **Double getDishCost()** en y intégrant les quantités requises dans le calcul. Pour le moment, nous n'allons pas considérer la conversion des unités.

5. Ajouter la méthode permettant de calculer la marge brute de chaque plat en la méthode précédente et en ajoutant, si ce n'est pas encore le cas, un prix de vente aux plats. Il faut savoir que c'est un attribut pouvant être nul, auquel cas une exception pourrait être levée durant le calcul si c'est ce qui se produit.

Voici les valeurs attendues pour les données de tests actuels :

- Pour la méthode getDishCost() :

| Plat | Coût attendu |
|---|---|
| Salade fraîche | **250.00** |
| Poulet grillé | **4500.00** |
| Riz aux légumes | **0.00** |
| Gâteau au chocolat | **1400.00** |
| Salade de fruits | **0.00** |

- Pour la méthode getGrossMargin() :

| Plat | Marge attendue |
|---|---|
| Salade fraîche | **3250.00** |
| Poulet grillé | **7500.00** |
| Riz aux légumes | ❌ **Exception (prix NULL)** |
| Gâteau au chocolat | **6600.00** |
| Salade de fruits | ❌ **Exception (prix NULL)** |