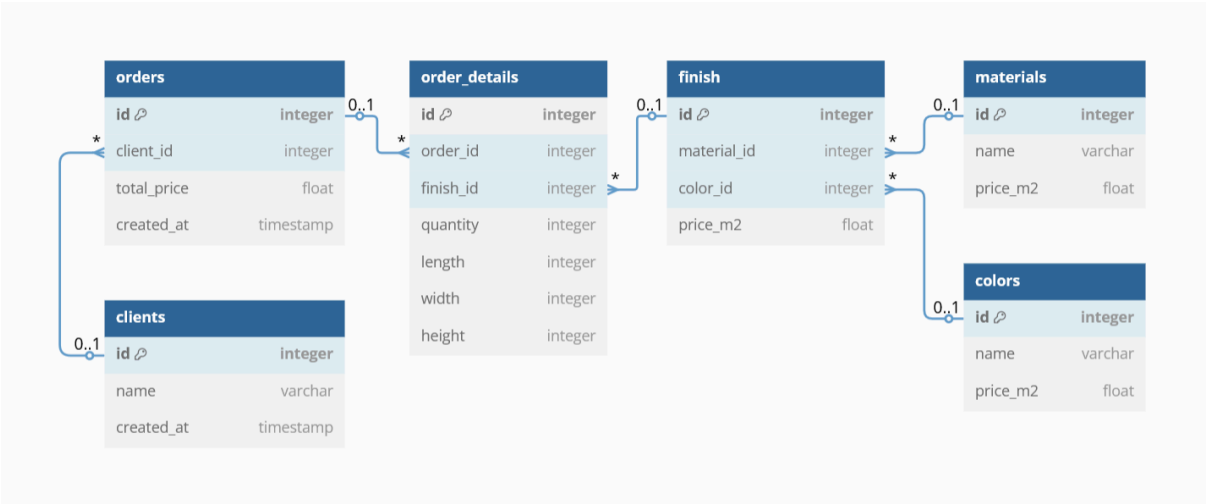
Exercice de positionnement

Data Engineer | ***Anatole Bahurel***



Modèle logique des données

Justification des choix de tables

- clients

Une table pour enregistrer tous les clients qui souhaitent passer commande. Avant de placer une commande, on vérifiera si le client existe bien dans la table.

- materials

On liste ici les différents matériaux disponibles pour la fabrication d'une boîte. On y précise, en plus de l'identifiant unique, un intitulé et un prix au m^2.

- colors

Comme pour la table des matériaux, on enregistre les différentes couleurs disponibles pour la fabrication d'une boîte. On y précise, en plus de l'identifiant unique, un intitulé et un prix au m^2.

- finish

C'est ici que l'on définit les différentes combinaisons matériaux/couleurs possibles. Le client ne pourra personnaliser ses boîtes uniquement avec les finitions listées dans cette table. On construit des paires à partir des identifiants des éléments présents dans les tables "materials" et "colors".
Le prix de la combinaison est calculé en ajoutant celui du matériau avec celui de la couleur choisie. C'est un choix arbitraire pour l'exercice, on aurait pu directement choisir un prix pour chaque paire.

- orders

Dans cette table, on enregistre toutes les commandes passées avec un identifiant unique, le client qui a passé la commande et le prix total calculé ici. Le détail se trouve dans une autre table.

- order_details

C'est ici que le détail des commandes est enregistré. On y précise à quelle commande cela correspond, la quantité et les options des boîtes commandées. Une contrainte de valeur maximale est appliquée sur les champs de dimensions (inférieur ou égal à 1000 mm).

Fonctions déclencheurs

- set_created_at()

Permet de renseigner la valeur des champs "*created_at*" avec le timestamp actuel à chaque insertion dans les tables "clients" et "orders". Un ajout qui peut être utile pour garder un trace chronologique de chaque enregistrement.

- update_finish_price()

Pour chaque création ou modification de combinaison matériau/couleur dans la table "*finish*", cette fonction ajoute le prix du matériau à celui de la couleur associée, puis enregistre le résultat dans le champ "*price_m2*". On peut modifier la fonction si la méthode de calcul du prix venait à changer.

materials.price_m2
colors.price_m2
>>> finish.price_m2

- update_order_total_price()

De la même manière que pour le prix des combinaisons matériau/couleur, le prix total de chaque commande est calculé en fonction du détail de cette commande. La fonction calcule dans un premier temps la quantité totale de boîtes commandées afin de déterminer le montant de la réduction à appliquer au prix total (tarif dégressif). Ensuite, le prix des boîtes est calculé en fonction de la superficie de matière utilisée, de la finition choisie et de la quantité demandée. On applique alors le rabais déterminé plus tôt.

order_details.quantity
order_details.length
order_details.width
order_details.height
finish.price_m2
>>> orders.total_price

Choix des outils

Par habitude et car les outils/technologies étaient déjà installés et configurés sur mon ordinateur, j'ai choisi d'utiliser **Postgres** pour ma base de données, que j'explorais et modifiais avec **DBeaver**.