BRIEF PROJET - Créer une base de données Projet pour la préparation à la journée de sélection pour le parcours de formation Data Engineer.

Gregory EL BAJOURY, 31/01/2025

Note Synthétique : Conception de la Base de Données

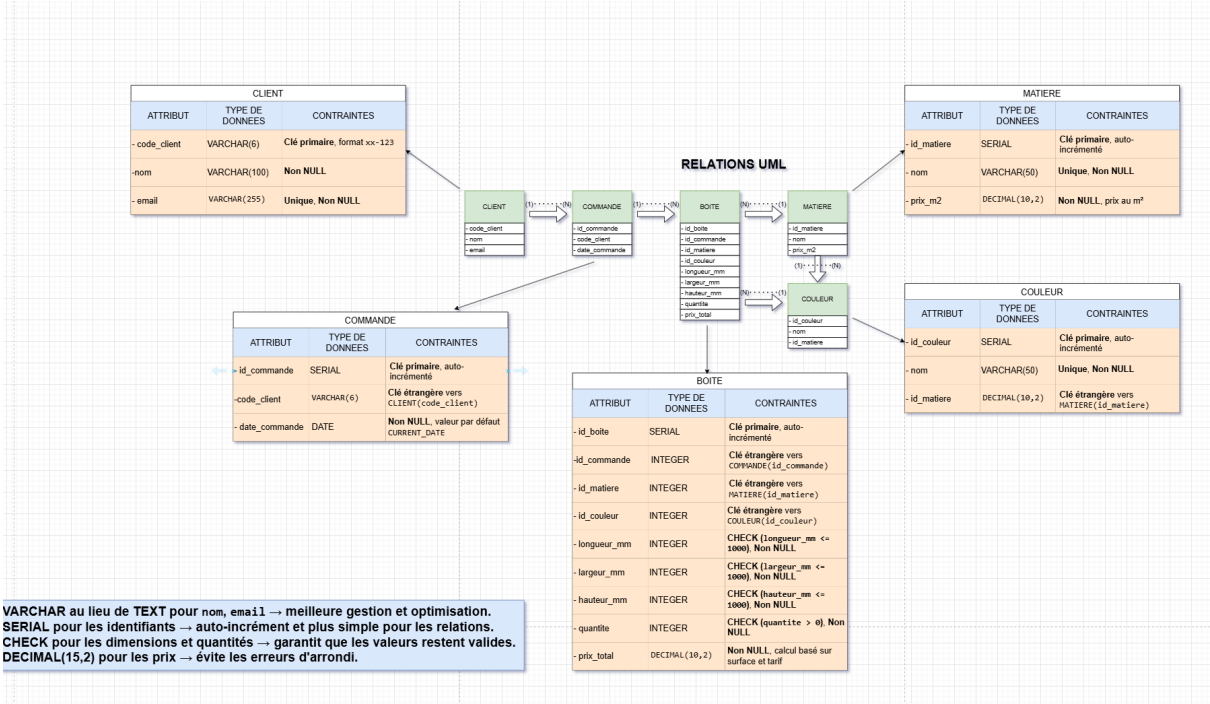
1.Contexte du Projet

L'objectif de cette base de données est de permettre la gestion des commandes de **boîtes de rangement personnalisables**.

Elle doit répondre aux contraintes métier suivantes :

- **Les clients peuvent personnaliser leurs boîtes** (matière, couleur, dimensions).
- **Les commandes peuvent contenir plusieurs boîtes** avec des prix calculés dynamiquement.
- **Un tarif dégressif s'applique en fonction des quantités commandées**.
- **Les relations entre les entités doivent être bien définies pour garantir l'intégrité des données**.

Modèle Logique des données



2.Modélisation des Données

Conception d'un **Modèle Logique des Données (MLD)** avec les entités suivantes :

1. **Client** (`code_client`, `nom`, `email`)
2. **Commande** (`id_commande`, `code_client`, `date_commande`)
3. **Boîte** (`id_boite`, `id_commande`, `id_matiere`, `id_couleur`, `longueur_mm`, `largeur_mm`, `hauteur_mm`, `quantite`, `prix_total`)
4. **Matière** (`id_matiere`, `nom`, `prix_m2`)
5. **Couleur** (`id_couleur`, `nom`, `id_matiere`)

Relations clés :

- **Un client peut passer plusieurs commandes (`1:N`)**
- **Une commande peut contenir plusieurs boîtes (`1:N`)**
- **Une boîte est liée à une matière et une couleur (`N:1`)**
- **Certaines couleurs sont spécifiques à certaines matières (`1:N`)**

Implémentation SQL

Utilisation de **PostgreSQL** pour créer cette base de données relationnelle avec :

- **Les contraintes de clés primaires (`PRIMARY KEY`) et étrangères (`FOREIGN KEY`)** pour assurer l'intégrité des données.
- **Des contrôles (`CHECK`) sur les dimensions des boîtes** pour garantir le respect des règles métier.
- **Une fonction SQL + Trigger** pour **calculer automatiquement le prix total d'une boîte** en fonction de sa surface et de sa quantité.
- **Une vue SQL `rapport_commandes`** pour obtenir **le total des commandes par client**.

Calculs Automatisés

Le prix d'une boîte est calculé dynamiquement en fonction de :

- **Sa surface extérieure en mm²**
- **Le prix du matériau utilisé**
- **Un tarif dégressif appliqué en fonction des quantités commandées✔️ Un Trigger PostgreSQL** permet de **recalculer le prix total à chaque mise à jour de la quantité**.

Résultats et Vérifications

Test la base en ajoutant des données réelles :

- **Insertion de clients, commandes, matières et couleurs**
- **Ajout de plusieurs boîtes et vérification du calcul des prix**
- **Mise à jour des quantités et vérification de l'application des réductions**
- **Exécution de requêtes analytiques pour voir le total des commandes par client**

Scripts

creation_bdd_boites.sql :

```
-- TABLE CLIENT
CREATE TABLE Client (
    code_client VARCHAR(6) PRIMARY KEY CHECK (code_client ~ '
    nom VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL
);

-- TABLE COMMANDE
CREATE TABLE Commande (
    id_commande SERIAL PRIMARY KEY,
    code_client VARCHAR(6) NOT NULL,
    date_commande DATE NOT NULL DEFAULT CURRENT_DATE,
    FOREIGN KEY (code_client) REFERENCES Client(code_client) (
);

-- TABLE MATIERE
CREATE TABLE Matiere (
    id_matiere SERIAL PRIMARY KEY,
    nom VARCHAR(50) UNIQUE NOT NULL,
    prix_m2 DECIMAL(15,2) NOT NULL CHECK (prix_m2 > 0)
);

-- TABLE COULEUR
CREATE TABLE Couleur (
    id_couleur SERIAL PRIMARY KEY,
    nom VARCHAR(50) UNIQUE NOT NULL,
    id_matiere INTEGER NOT NULL,
    FOREIGN KEY (id_matiere) REFERENCES Matiere(id_matiere) O
);

-- TABLE BOITE
CREATE TABLE Boite (
    id_boite SERIAL PRIMARY KEY,
    id_commande INTEGER NOT NULL,
    id_matiere INTEGER NOT NULL,
    id_couleur INTEGER NOT NULL,
    longueur_mm INTEGER NOT NULL CHECK (longueur_mm BETWEEN 1
    largeur_mm INTEGER NOT NULL CHECK (largeur_mm BETWEEN 1 A
```

```
    hauteur_mm INTEGER NOT NULL CHECK (hauteur_mm BETWEEN 1 A
    quantite INTEGER NOT NULL CHECK (quantite > 0),
    prix_total DECIMAL(15,2) NOT NULL DEFAULT 0,
    FOREIGN KEY (id_commande) REFERENCES Commande(id_commande
    FOREIGN KEY (id_matiere) REFERENCES Matiere(id_matiere) O
    FOREIGN KEY (id_couleur) REFERENCES Couleur(id_couleur) O
);

-- Supprimer l'ancienne fonction si elle existe
DROP FUNCTION IF EXISTS calculer_prix_total;

-- Créer la nouvelle fonction
CREATE OR REPLACE FUNCTION calculer_prix_total()
RETURNS TRIGGER AS $$
DECLARE
    prix_unitaire DECIMAL(15,2);
    surface_m2 DECIMAL(15,6);
    reduction DECIMAL(5,2) := 0;
BEGIN
    -- Convertir la surface de mm² en m²
    surface_m2 := (2 * (NEW.longueur_mm * NEW.largeur_mm +
                        NEW.longueur_mm * NEW.hauteur_mm +
                        NEW.largeur_mm * NEW.hauteur_mm)) / 1

    -- Récupérer le prix au m² de la matière
    SELECT prix_m2 INTO prix_unitaire
    FROM Matiere WHERE id_matiere = NEW.id_matiere;

    -- Appliquer la réduction en fonction des quantités
    IF NEW.quantite BETWEEN 6 AND 10 THEN
        reduction := 0.05;  -- 5% de réduction
    ELSIF NEW.quantite BETWEEN 11 AND 20 THEN
        reduction := 0.10;  -- 10% de réduction
    ELSIF NEW.quantite > 20 THEN
        reduction := 0.15;  -- 15% de réduction
    END IF;

    -- Calcul du prix total
```

```
    NEW.prix_total := surface_m2 * prix_unitaire * NEW.quanti

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Supprimer l'ancien trigger s'il existe
DROP TRIGGER IF EXISTS trigger_calcul_prix ON Boite;

-- Créer le trigger pour appliquer la réduction automatiqueme
CREATE TRIGGER trigger_calcul_prix
BEFORE INSERT OR UPDATE ON Boite
FOR EACH ROW EXECUTE FUNCTION calculer_prix_total();
```

data_test_boites.sql:

```
-- INSÉRER DES CLIENTS
INSERT INTO Client (code_client, nom, email) VALUES
('ab-001', 'Alice Dupont', 'alice.dupont@email.com'),
('cd-002', 'Bob Martin', 'bob.martin@email.com');

-- INSÉRER DES COMMANDES
INSERT INTO Commande (code_client, date_commande) VALUES
('ab-001', '2024-02-01'),
('cd-002', '2024-02-02');

-- INSÉRER DES MATIÈRES
INSERT INTO Matiere (nom, prix_m2) VALUES
('Plastique', 20.50),
('Bois', 35.75),
('Métal', 50.00);

-- INSÉRER DES COULEURS
INSERT INTO Couleur (nom, id_matiere) VALUES
('Rouge', 1),  -- Plastique
('Bleu', 1),   -- Plastique
('Marron', 2), -- Bois
```

```
('Gris', 3);   -- Métal

-- INSÉRER DES BOÎTES
INSERT INTO Boite (id_commande, id_matiere, id_couleur, longu
VALUES
(1, 1, 1, 500, 300, 200, 10),  -- Plastique Rouge
(1, 2, 3, 800, 500, 400, 5),   -- Bois Marron
(2, 3, 4, 1000, 700, 600, 2);  -- Métal Gris

SELECT * FROM Client;
SELECT * FROM Commande;
SELECT * FROM Matiere;
SELECT * FROM Couleur;
SELECT * FROM Boite;  -- Vérifier que prix_total est bien cal

SELECT c.code_client, cmd.id_commande, b.id_boite, b.prix_tot
FROM Client c
JOIN Commande cmd ON c.code_client = cmd.code_client
JOIN Boite b ON cmd.id_commande = b.id_commande;


UPDATE Boite
SET quantite = 15
WHERE id_boite = 1;

SELECT * FROM Boite WHERE id_boite = 1;


SELECT id_boite, longueur_mm, largeur_mm, hauteur_mm, quantit
FROM Boite;

CREATE OR REPLACE VIEW rapport_commandes AS
SELECT
    c.code_client,
    c.nom AS client_nom,
    cmd.id_commande,
    cmd.date_commande,
    SUM(b.prix_total) AS total_commande
```

```
FROM Client c
JOIN Commande cmd ON c.code_client = cmd.code_client
JOIN Boite b ON cmd.id_commande = b.id_commande
GROUP BY c.code_client, c.nom, cmd.id_commande, cmd.date_comm
ORDER BY cmd.date_commande DESC;

SELECT table_name
FROM information_schema.views
WHERE table_name = 'rapport_commandes';

SELECT id_boite, prix_total FROM Boite;

SELECT c.code_client, cmd.id_commande, b.id_boite, b.prix_tot
FROM Client c
LEFT JOIN Commande cmd ON c.code_client = cmd.code_client
LEFT JOIN Boite b ON cmd.id_commande = b.id_commande;

SELECT * FROM rapport_commandes;
```