Rémi LANGDORPH - Ryan JABBOUR - Louise KHACHOYAN

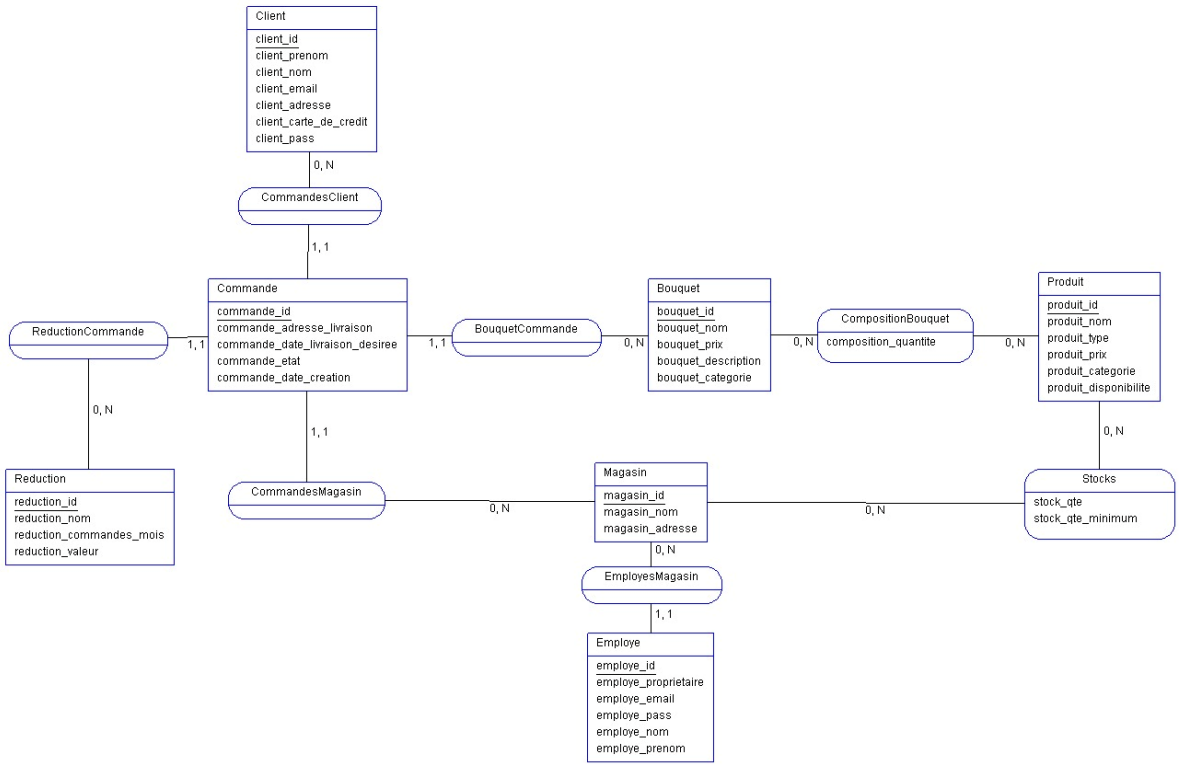
Rapport du projet BelleFleur

Le but du projet était de créer une base de données des magasins du Monsieur BelleFleur, qui pourrait gérer les transactions du magasin entre clients et employés. On s'est acharné pendant plusieurs heures pour le succès du projet final et on vous expliquera par la suite les étapes prises pour arriver à la soutenance finale.

Passage schéma entité/association, schéma relationnel, MySQL

On a commencé par créer le schéma entité/association sur Analyse SI tout en mettant toutes les tables et les associations requises pour le fonctionnement et la sauvegarde de toutes les informations du magasin dans la base de données.

Pour la table Client, on a mis toutes les informations nécessaires à l'achat d'un produit du magasin : le nom, le prénom, les informations bancaires... Après on a ajouté la table Commande qui prend en compte les Réductions éventuellement appliquées ainsi que le Bouquet choisi ou personnalisé par le Client précis. Ensuite, chaque Bouquet prend en compte le Produit désiré : fleur, accessoire..., tout en choisissant le Magasin le plus proche à l'adresse du Client. On a rajouté une table Employé qui est associée à un Magasin donné pour pouvoir prendre des commandes en tant qu'utilisateur et pas seulement en tant que Client.



Puis, suite au schéma entité/association, on a élaboré le schéma relationnel détaillé du schéma ci-dessus.

- client (client_id int, client_prenom VARCHAR(50), client_nom VARCHAR(50), client_email varchar(50), client_adresse VARCHAR(255), client_carte_de_credit VARCHAR(50), client_pass VARCHAR(200));
- reduction (reduction_id int, reduction_nom VARCHAR(50), reduction_commandes_mois int, reduction_valeur decimal)
- magasin (magasin_id int, magasin_nom VARCHAR(50),magasin_adresse VARCHAR(255));
- bouquet (bouquet_id int, bouquet_nom VARCHAR(50), bouquet_prix decimal, bouquet_description VARCHAR(255), bouquet_categorie VARCHAR(50))
- commande (commande_id int, commande_adresse_livraison VARCHAR(255), commande_date_livraison_desiree datetime, commande_etat VARCHAR(50), commande_date_creation datetime, #client_id int, #bouquet_id int, #magasin_id int, #reduction_id)
- produit (produit_id: int, produit_nom: VARCHAR(50), produit_type: VARCHAR(50), produit_prix: decimal, produit_categorie: VARCHAR(50), produit_disponibilite_mois: VARCHAR(50))
- employe (employe_id: int, employe_proprietaire: bool, employe_email: VARCHAR(50), employe_pass: VARCHAR(200), employe_prenom: VARCHAR(50), employe_nom: VARCHAR(50), #magasin_id: int)
- compositionbouquet (#bouquet_id: int, #produit_id: int, composition_quantite: int)
- stocks (#magasin_id: int, #produit_id int, stock_qte int, stock_qte_minimum int)

On est ensuite passé à la création de notre base de données sur MySQL Workbench, en ajoutant toutes les tables requises pour la sauvegarde des informations de l'entreprise, bien sûr en prenant en relief le schéma relationnel qu'on a établi et qui a facilité la tâche sur la rédaction dans MySQL.

Trigger

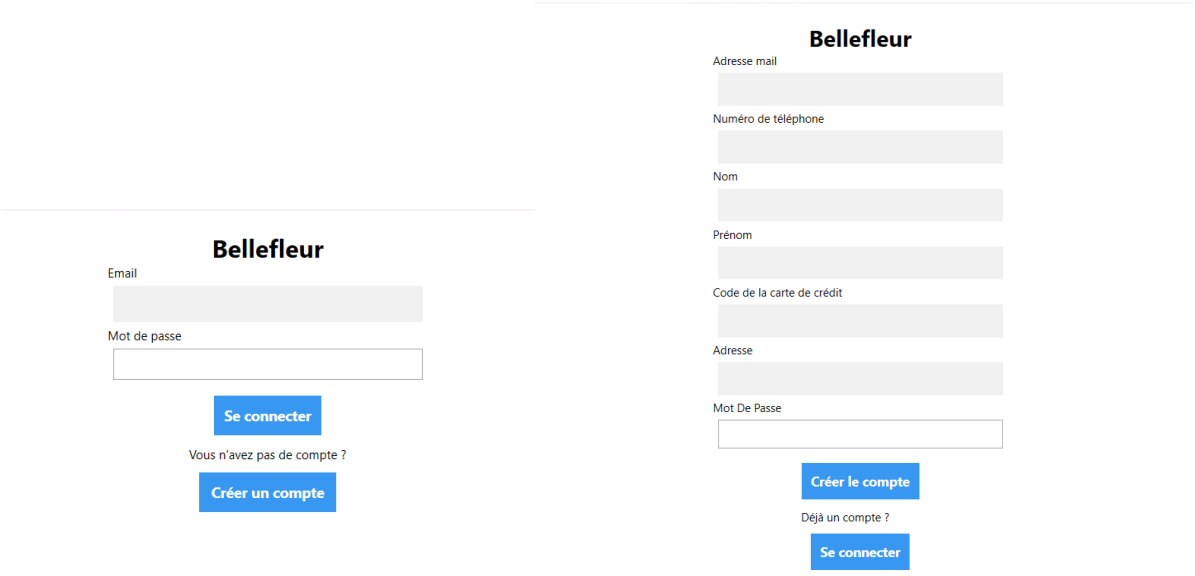
```
mysql> select * from stocks where magasin_id = 4;
Empty set (0.00 sec)

mysql> insert into magasin(magasin_nom, magasin_adresse)VALUES('Test Trigger', 'adresse1');
Query OK, 1 row affected (0.01 sec)

mysql> select * from stocks where magasin_id = 4;
+------------+------------+-----------+-------------------+
| magasin_id | produit_id | stock_qte | stock_qte_minimum |
+------------+------------+-----------+-------------------+
|          4 |          1 |         0 |                 0 |
|          4 |          2 |         0 |                 0 |
|          4 |          3 |         0 |                 0 |
|          4 |          4 |         0 |                 0 |
|          4 |          5 |         0 |                 0 |
|          4 |          6 |         0 |                 0 |
|          4 |          7 |         0 |                 0 |
|          4 |          8 |         0 |                 0 |
|          4 |          9 |         0 |                 0 |
+------------+------------+-----------+-------------------+
9 rows in set (0.00 sec)

mysql>
```

Nous avons codé 3 triggers : 2 pour mettre les stocks à 0 lors de la création d'un produit ou d'un magasin et 1 pour mettre à jour les stocks une fois la commande validée.

Double projet

Nous avons séparé le projet en 2 parties: une librairie “BelleFleurLib” responsable exclusivement de l’intéraction avec MySQL et la logique “métier” du projet et une application Console “FleurConsole” qui utilise la librairie et fait office d’interface uniquement (pas de logique dedans).

WPF



Avant d’implémenter l’application Console, nous avions commencé une application WPF (basée sur le même principe de séparation avec la librairie), nous avons cependant préféré passer à une application Console à cause du manque de temps afin de livrer un projet fonctionnel et stable dans les temps.

Vérifications d’entrée

```
/// <summary>
/// Classe pour gérer les entrées utilisateur et les valider automatiquement
/// </summary>
36 references
public class InputsHelper {
    23 references
    public static int Int(string Nom, int min, int max) {…

    2 references
    public static decimal Decimal(string Nom, decimal min, decimal max) {…

    13 references
    public static string Text(string Nom, int min, int max, bool password=false) {…

    2 references
    public static DateTime Date(string Nom, DateTime min, DateTime max) {…

    1 reference
    public static string Email(string Nom, int min, int max) {…
}
```

Nous avons mis en place une classe pour gérer les différentes entrées utilisateurs et les valider automatiquement afin d’éviter les répétitions et de permettre une meilleure expérience utilisateur.