RAPPORT PROJET : BDD

Mr BelleFleur et la gestion de différentes parties de son business

THOMINE Bérénice & THOMASSIN Pablo

I/ Conception de la base de données et modèle entité-association

- Client : sa clé primaire est son id, on répertorie ici l'ensemble des clients de Mr BelleFleur avec différents attributs pour pouvoir les caractériser au mieux
- Statut_Fidelite : sa clé primaire est id_fidelite, l'objectif de cette table est de gérer le statut de fidélité des clients, on pourra donc le voir et aussi le mettre à jour
- Produit : sa clé primaire est id_ produit, on répertorie ici l'ensemble des items, soit des fleurs disponibles dans le stock total
- Accessoire : sa clé primaire est id_accessoire, on répertorie ici aussi l'ensemble des accessoires pouvant être utilisé dans une commande personnalisée.
- Bouquet_Standard : sa clé primaire est id_bouquet_standard, on répertorie l'ensemble des bouquets existants et pouvant être vendu directement au client, ici on ne s'occupe pas de faire une quantité par bouquet, comme vue dans le cahier des charges on vérifiera les items via la table Produit
- Magasin : sa clé primaire est id_magasin. Dans cette table on répertorie l'ensemble des magasins de Mr BelleFleur et avec les produits, les bouquets standards et les accessoires disponibles dans ce magasin en passant en clé étrangère les id de ces trois tables
- Commande_Standard : sa clé primaire est id_commande_standard, ici on répertorie toutes les commandes standards passé, on utilise la clé de magasin pour définir dans quel magasin la commande a été passé, avec de même pour les bouquets standards et aussi pour les clients pour savoir qui passe la commande.
- Commande_Personnalisee : sa clé primaire est id_commande_personnalisee, ici on fait comme pour une commande standard, mais dans le cadre de la commande personnalisée on a donc les clés étrangères des tables suivantes : Accessoire, Produit, Magasin, Client. Pour traiter correctement la commande en fonction du client et de ce qu'elle contient
- Details_Commande : sa clé primaire est id_details_commande, ici on cherche à répertorier en détails le contenu de chaque commande, qu'elle soit personnalisée ou standard. On permettra donc l'accès aux quantités exactes utilisé dans la commande.

De plus on considère que chaque vente de produit implique une commande unique, par exemple si un client veut 3 bouquets standard, il passera alors 3 commandes standards. Cela permet d'éviter une table commande globale et un choix entre personnalisée et standard. C'est un choix que nous avons fait pour nous simplifier le travail des commandes dans un premier temps. Néanmoins après avoir développé notre application en C# (et avec un peu plus d'expérience donc…) on remarque qu'il

serait en effet pertinent d'avoir cette table commande et de permettre des commandes multiples pour éviter de générer 3 tuples (par exemples) pour 3 commandes devant être dû à un même client.

De plus pour les objets, on a fait le choix de ne pas faire de liste de produit dans les bouquets standards, magasins, commande personnalisée et pareille pour les accessoires. En faite cela est juste une simplification unitaire, si on autorise sous forme de liste on récupère un attribut en C# de type List<> et on traite dessus avec les méthodes usuelles de ce type. Cela serait plus proche de la réalité.

II/ Option de développement C# et produit final

II.1 Un développement par classe

Pour le développement de notre application, on effectue un affichage console, car l'un de nous travaillait sous MacOS et pour ne pas avoir à gérer Windows form sous mac. Cela constitue donc un gros point d'amélioration pour avoir une interface plus « user friendly ». Néanmoins nous proposons une nomenclature simple et compréhensible ou le choix se fait via un nombre.

On commence tout d'abords par une demande de connexion à la base de données (avec l'option bozo/bozo disponible), cela permettra pour bozo notamment d'avoir des retours de message d'erreur dû à son manque d'accès sur certaine fonctionnalité.

Par la suite un menu s'affiche proposant les modules ainsi que les 2 exports souhaité. Puis dans chaque module, de la même manière différentes options sont affichés permettant de réaliser par exemple : l'ajout, la suppression, la modification de table …

Nous avons fait le choix d'un développement orienté objet. En effet chaque table possède sa propre classe (SAUF Détails_Commande étant une classe de stockage on répertorie ses informations dans les classes Commande_Standard ou Commande_Personnalisee respectivement et selon notre besoin). Cela permet de définir des attributs ayant les mêmes types que nos attributs pour chaque table dans notre base de données et donc d'effectuer des opérations en C# sur ces attributs pour vérifier des conditions, permettre une saisie de l'utilisateur, avoir des objets pour nos tables. Mais surtout d'avoir une bonne organisation du code, avec les requêtes sont majoritairement dans les classes et leur utilisation dans le fichier Program.

Un exemple d'affichage :

=== Menu Principal ===

1. Module Client

2. Module Produit

3. Module Commande

4. Module Statistiques

5. Export en XML (requête synchronisée)

6. Export en JSON

7. Quitter

Choix :

III/ Requêtes réglementaires du projet

III.1 : Requête avec auto-jointure

Notre requête avec auto-jointure :

"SELECT DISTINCT c1.nom, c1.prenom, c1.adresse_facturation FROM Client c1 INNER JOIN Client c2 ON c1.adresse_facturation = c2.adresse_facturation AND c1.id_client <> c2.id_client"

Permet de trouver les clients distincts ayant la même adresse de facturation

III.2 : Requête synchronisée

Notre requête synchronisée se situe dans l'export xml, on autorise une lecture seulement en définissant : `private static readonly object syncLock = new object();`

Et en insérant la requête usuel par C# dans : `lock (syncLock){}`

Cela permet de ne pas modifier les données que nous allons exporter au format XML

III.3 : Requête avec union

Notre requête avec union :
```
@"SELECT SUM(montant_total) FROM (
    SELECT montant_total FROM Commande_Standard
    WHERE YEAR(date_livraison) = @annee AND MONTH(date_livraison) = @mois
    UNION ALL
    SELECT montant_total FROM Commande_Personnalisee
    WHERE YEAR(date_livraison) = @annee AND MONTH(date_livraison) = @mois
) AS chiffre_affaires_mois"
```
Notre requête avec union permet de récupérer un chiffre d'affaires mensuel total en récupérant le prix de vente de toutes les commandes standards et personnalisées dans le mois à l'année choisie par l'utilisateur.

ANNEXE REQUÊTE

NB : Dans les classes produits et clients, la méthode supprimer n'est pas écrite sur ce document, néanmoins elle est bien implémentée, pour se faire on désactive temporairement certaines clés étrangères pour pouvoir effectuer une suppression. Cela est dû au fait qu'à la création de la base de données nous n'avons pas défini en cascade ces clés étrangères, cela constitue donc un point d'amélioration mais résoluble par d'autre moyen.

--- Dans le fichier programme :

EXPORT EN XML :
```
$@"SELECT c.*
                FROM Client c
                JOIN (
                    SELECT client_id
```

```
                    FROM Commande_Standard
                    WHERE date_commande >= '{lastMonthDate.ToString("yyyy-MM-
dd")}'
                    GROUP BY client_id
                    HAVING COUNT(id_commande_standard) > 1
                    UNION
                    SELECT client_id
                    FROM Commande_Personnalisee
                    WHERE date_commande >= '{lastMonthDate.ToString("yyyy-MM-
dd")}'
                    GROUP BY client_id
                    HAVING COUNT(id_commande_personnalisee) > 1
                ) AS t ON c.id_client = t.client_id";
```

EXPORT EN JSON :

```
@"SELECT * FROM Client WHERE id_client NOT IN (
                    SELECT DISTINCT client_id FROM Commande_Standard WHERE
date_commande >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH) + INTERVAL 1 DAY
                    UNION
                    SELECT DISTINCT client_id FROM Commande_Personnalisee WHERE
date_commande >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH) + INTERVAL 1 DAY
                )"
```

AFFICHAGE DU STOCK DES MAGASINS :

```
@"
            SELECT m.nom AS nom_magasin,
                   GROUP_CONCAT(DISTINCT bs.nom SEPARATOR ', ') AS
bouquet_standard_present,
                   GROUP_CONCAT(DISTINCT CONCAT(p.nom, ' (',
p.quantitéDisponible, ')') SEPARATOR ', ') AS produits_present,
                   GROUP_CONCAT(DISTINCT CONCAT(a.nom, ' (', a.quantité, ')')
SEPARATOR ', ') AS accessoires_present
            FROM Magasin m
            LEFT JOIN Bouquet_Standard bs ON m.bouquet_standard_id =
bs.id_bouquet_standard
            LEFT JOIN Produit p ON m.produit_id = p.id_produit
            LEFT JOIN Magasin m2 ON m.nom = m2.nom
                                    AND (bs.id_bouquet_standard IS NULL OR
bs.id_bouquet_standard = m2.bouquet_standard_id)
                                    AND (p.id_produit IS NULL OR p.id_produit =
m2.produit_id)
            LEFT JOIN Accessoire a ON m.accessoire_id = a.id_accessoire
            WHERE (p.quantitéDisponible > 0 OR a.quantité > 0)
                  AND m2.id_magasin IS NOT NULL
            GROUP BY m.id_magasin, m.nom";
```

REQUETE POUR TROUVER UN NOUVEL ID A UN AJOUT : cette requête se retrouve plusieurs fois dans le code selon qu'on ajoute un client, une commande, un accessoire ….

```
"SELECT MAX(id_bouquet_standard) FROM Bouquet_Standard"
```

----- Dans la classe commande_standard :

AJOUTER UNE COMMANDE :

```
"INSERT INTO Commande_Standard (id_commande_standard, bouquet_standard_id,
date_commande, date_livraison, adresse_livraison, message_accompagnement,
montant_total, etat, client_id, magasin_id) VALUES (@id, @bouquetStandardId,
@dateCommande, @dateLivraison, @adresseLivraison, @messageAccompagnement,
@montantTotal, @etat, @clientId, @magasinId)"
```

SUPPRIMER UNE COMMANDE :

```
"DELETE FROM Commande_Standard WHERE id_commande_standard = @id"
```

OBTENIR UNE COMMANDE STANDARD :

```
"SELECT * FROM Commande_Standard WHERE id_commande_standard = @commandeId" :
```

OBTENIR LES COMMANDEs STANDARDS LIVREES POUR UN ID CLIENT :

```
"SELECT * FROM Commande_Standard WHERE etat = 'CL' AND client_id = {clientId}"
```

OBTENIR TOUTES LES COMMANDES STANDARDS :

```
"SELECT * FROM Commande_Standard"
```

MODIFIER L'ETAT D'UNE COMMANDES STANDARDS :

---- Dans la classe Commande_Personnalisee :

AJOUT :

```
"INSERT INTO commande_personnalisee(id_commande_personnalisee, client_id,
date_commande, date_livraison, adresse_livraison, message_accompagnement,
montant_total, etat, magasin_id, produit_id, accessoire_id)
VALUES(@id_commande_personnalisee, @clientId, @dateCommande, @dateLivraison,
@adresseLivraison, @messageAccompagnement, @montantTotal, @etat, @magasinId,
@produitID, @accessoireID)"
```

SUPPRESSION :

```
"DELETE FROM Commande_Personnalisee WHERE id_commande_personnalisee = @id"
```

OBTENIR UNE COMMANDE PERSONNALISEE SELON ID D'UN CLIENT :

```
"SELECT * FROM Commande_Personnalisee WHERE id_commande_personnalisee =
@commandeId "
```

OBTENIR UNE COMMANDE PERSONNALISEE SELON ID D'UN CLIENT ET SON ETAT :

```
"SELECT * FROM Commande_Personnalisee WHERE etat = 'CL' AND client_id =
{clientId}"
```

OBTENIR TOUTES LES COMMANDES PERSONNALISEE :

```
"SELECT * FROM Commande_Personnalisee"
```

MODIFIER L'ETAT D'UNE COMMANDE PERSONNALISEE :

```
"UPDATE Commande_Personnalisee SET etat='{etat}' WHERE
id_commande_personnalisee={commandeId}"
```

---- Dans la classe Magasin :

OBTENIR LA LISTE DES MAGASINS :

```
"SELECT * FROM Magasin"
```

---- Dans la classe Statut_Fidelite :

CHANGER LE STATUT SELON LE CAHIER DES CHARGES :

```
@"
                UPDATE Statut_Fidelite
                SET statut = CASE
                    WHEN (
                        (SELECT COUNT(*) FROM Commande_Standard WHERE client_id =
@clientId AND date_commande BETWEEN DATE_SUB(NOW(), INTERVAL 12 MONTH) AND NOW())
+
                        (SELECT COUNT(*) FROM Commande_Personnalisee WHERE
client_id = @clientId AND date_commande BETWEEN DATE_SUB(NOW(), INTERVAL 12
MONTH) AND NOW())
                    ) >= 5 THEN 'OR'
                    WHEN (
                        (SELECT COUNT(*) FROM Commande_Standard WHERE client_id =
@clientId AND date_commande BETWEEN DATE_SUB(NOW(), INTERVAL 12 MONTH) AND NOW())
+
                        (SELECT COUNT(*) FROM Commande_Personnalisee WHERE
client_id = @clientId AND date_commande BETWEEN DATE_SUB(NOW(), INTERVAL 12
MONTH) AND NOW())
                    ) >= 1 THEN 'Bronze'
                    ELSE NULL
                END
                WHERE client_id = @clientId"
```

OBTENIR LE STATUT DE FIDELITE

```
"SELECT statut FROM Statut_Fidelite WHERE client_id = @clientId"
```

---- Dans la classe Bouquet_Standard :

OBTENIR TOUS LES BOUQUETS STANDARDS :

```
"SELECT * FROM Bouquet_Standard"
```

AJOUTER :

```
"INSERT INTO Bouquet_Standard (id_bouquet_standard, nom, description, prix,
catégorie, produit_id) VALUES (@id_bouquet_standard, @nom, @description, @prix,
@catégorie, @produit_id)
```

SUPPRIMER :

```
"DELETE FROM Bouquet_standard WHERE id_bouquet_standard = {id}"
```

METTRE A JOUR :

```
"UPDATE Bouquet_Standard SET nom = @nom, description = @description, prix =
@prix, catégorie = @categorie, produit_id = @produit_id WHERE id_bouquet_standard
= @id"
```

---- Dans la classe Accessoire :

OBTENIR TOUS LES ACCESSOIRE :

```
"SELECT * FROM Accessoire"
```

AJOUTER UN ACCESSOIRE :

```
"INSERT INTO Accessoire (id_accessoire, nom, description_accessoire, quantité,
prix) VALUES (@id_accessoire, @nom, @description, @quantite, @prix)"
```

SUPPRIMER :

```
"DELETE FROM Accessoire WHERE id_accessoire = {id}"
```

METTRE A JOUR :

```
"UPDATE Accessoire SET nom = @nom, description_accessoire = @description, prix =
@prix, quantité = @qttdisponible WHERE id_accessoire = @id"
```

--- Dans la classe Statistique :

MONTANT TOTAL DES COMMANDES POUR APRES CALCULER LE PRIX MOYEN :

```
"SELECT SUM(montant_total) AS total, COUNT(*) AS count FROM (SELECT montant_total
FROM commande_standard UNION ALL SELECT montant_total FROM
commande_personnalisee) as t"
```

MEILLEUR CLIENT DU MOIS:

```
@"SELECT client_id
                            FROM (
                                SELECT client_id, SUM(montant_total) AS
total_chiffre_affaires
                                FROM (
```

```
                                    SELECT id_commande_standard, client_id,
montant_total
                                    FROM Commande_Standard
                                    WHERE YEAR(date_livraison) = @annee AND
MONTH(date_livraison) = @mois
                                    UNION ALL
                                    SELECT id_commande_personnalisee, client_id,
montant_total
                                    FROM Commande_Personnalisee
                                    WHERE YEAR(date_livraison) = @annee AND
MONTH(date_livraison) = @mois
                                ) AS commandes
                                GROUP BY client_id
                                ORDER BY total_chiffre_affaires DESC
                            ) AS meilleurs_clients
                            LIMIT 1"
```

NOMBRE DE COMMANDE POUR UN MOIS ET UNE ANNEE :

```
@"SELECT COUNT(*) FROM (
                                SELECT id_commande_standard AS id_commande, 0 AS
type_commande FROM Commande_Standard
                                WHERE YEAR(date_livraison) = @annee AND
MONTH(date_livraison) = @mois
                                UNION
                                SELECT id_commande_personnalisee AS id_commande,
1 AS type_commande FROM Commande_Personnalisee
                                WHERE YEAR(date_livraison) = @annee AND
MONTH(date_livraison) = @mois
                            ) AS commandes_mois"
```

NOMBRE DE COMMANDE POUR UNE ANNEE :

```
@"SELECT COUNT(*) FROM (
                            SELECT id_commande_standard AS id_commande, 0 AS
type_commande FROM Commande_Standard
                            WHERE YEAR(date_livraison) = @annee
                            UNION
                            SELECT id_commande_personnalisee AS id_commande, 1 AS
type_commande FROM Commande_Personnalisee
                            WHERE YEAR(date_livraison) = @annee
                        ) AS commandes_annee"
```

MONTANT TOTAL DES COMMANDES POUR UN MOIS ET UNE ANNEE :

```
@"SELECT SUM(montant_total) FROM (
                                SELECT montant_total FROM Commande_Standard
                                WHERE YEAR(date_livraison) = @annee AND
MONTH(date_livraison) = @mois
                                UNION ALL
                                SELECT montant_total FROM Commande_Personnalisee
                                WHERE YEAR(date_livraison) = @annee AND
MONTH(date_livraison) = @mois
                            ) AS chiffre_affaires_mois";
```

MONTANT TOTAL DES COMMANDES POUR UNE ANNEE :

```
@"SELECT SUM(montant_total) FROM (
                                SELECT montant_total FROM Commande_Standard
                                WHERE YEAR(date_livraison) = @annee
                                UNION ALL
                                SELECT montant_total FROM Commande_Personnalisee
                                WHERE YEAR(date_livraison) = @annee
                            ) AS chiffre_affaires_mois"
```

MEILLEUR PRODUIT DU MOIS :

```
"SELECT bs.nom AS NomProduit, COUNT(*) AS NombreDeCommandes " +
                                "FROM Commande_Personnalisee cs " +
                                "JOIN Produit bs ON cs.produit_id = bs.id_produit
" +
                                "WHERE MONTH(cs.date_commande) = @mois AND
YEAR(cs.date_commande) = @annee " +
                                "GROUP BY cs.produit_id " +
                                "ORDER BY NombreDeCommandes DESC " +
                                "LIMIT 1;"
```

MEILLEUR BOUQUET STANDARD DU MOIS :

```
"SELECT bs.nom AS NomBouquetStandard, COUNT(*) AS NombreDeCommandes " +
                                "FROM Commande_Standard cs " +
                                "JOIN Bouquet_Standard bs ON
cs.bouquet_standard_id = bs.id_bouquet_standard " +
                                "WHERE MONTH(cs.date_commande) = @mois AND
YEAR(cs.date_commande) = @annee " +
                                "GROUP BY cs.bouquet_standard_id " +
                                "ORDER BY NombreDeCommandes DESC " +
                                "LIMIT 1;"
```

--- Dans la classe Produit:

OBTENIR TOUS LES PRODUITS :

```
"SELECT * FROM Produit"
```

OBTENIR UN PRODUIT AVEC SON ID :

```
"SELECT * FROM Produit WHERE id_produit = @id_produit"
```

METTRE A JOUR UN PRODUIT :

```
"UPDATE Produit SET nom = @nom, description = @description, prix = @prix,
quantitéDisponible = @qttdisponible, disponibilité = @disponibilite WHERE
id_produit = @id"
```

AJOUTER UN PRODUIT :

```
"INSERT INTO Produit (id_produit, nom, description, prix, quantitéDisponible,
disponibilité) VALUES (@id_produit, @nom, @description, @prix, @qttdisponible,
@disponibilite)"
```

TROUVER LE SEUIL POUR LE STOCK DES PRODUITS AVEC ETAT DES COMMANDE VINV OU CPAV :

```
@"SELECT SUM(dc.quantite) AS quantite_totale
                                    FROM Details_Commande dc
                                    INNER JOIN Commande_Standard cs ON
dc.commande_type = 'STANDARD' AND dc.commande_id = cs.id_commande_standard
                                    WHERE cs.etat IN ('VINV', 'CPAV')
                                    UNION ALL
                                    SELECT SUM(dc.quantite) AS quantite_totale
                                    FROM Details_Commande dc
                                    INNER JOIN Commande_Personnalisee cp ON
dc.commande_type = 'PERSONNALISÉE' AND dc.commande_id =
cp.id_commande_personnalisee
                                    WHERE cp.etat IN ('VINV', 'CPAV')"
```

--- Dans la classe Client :

AJOUTER :

```
"INSERT INTO Client (id_client, nom, prenom, telephone, courriel, mot_de_passe,
adresse_facturation, carte_credit) VALUES (@id, @nom, @prenom, @telephone,
@courriel, @motDePasse, @adresseFacturation, @carteCredit)"
```

METTRE A JOUR :

```
"UPDATE Client SET nom = @nom, prenom = @prenom, telephone = @telephone, courriel
= @courriel, mot_de_passe = @motDePasse, adresse_facturation =
@adresseFacturation, carte_credit = @carteCredit WHERE id_client = @id"
```

OBTENIR TOUS LES CLIENTS :

```
"SELECT * FROM Client"
```

OBTENIR UN CLIENT SELON SON ID :

```
"SELECT * FROM Client WHERE id_client = @id"
```