RABOLIN William, RAKAI Mohamed, TD Q

Rapport – Projet BDD

1) Organisation de la base de données

Diagramme E/A de la base:

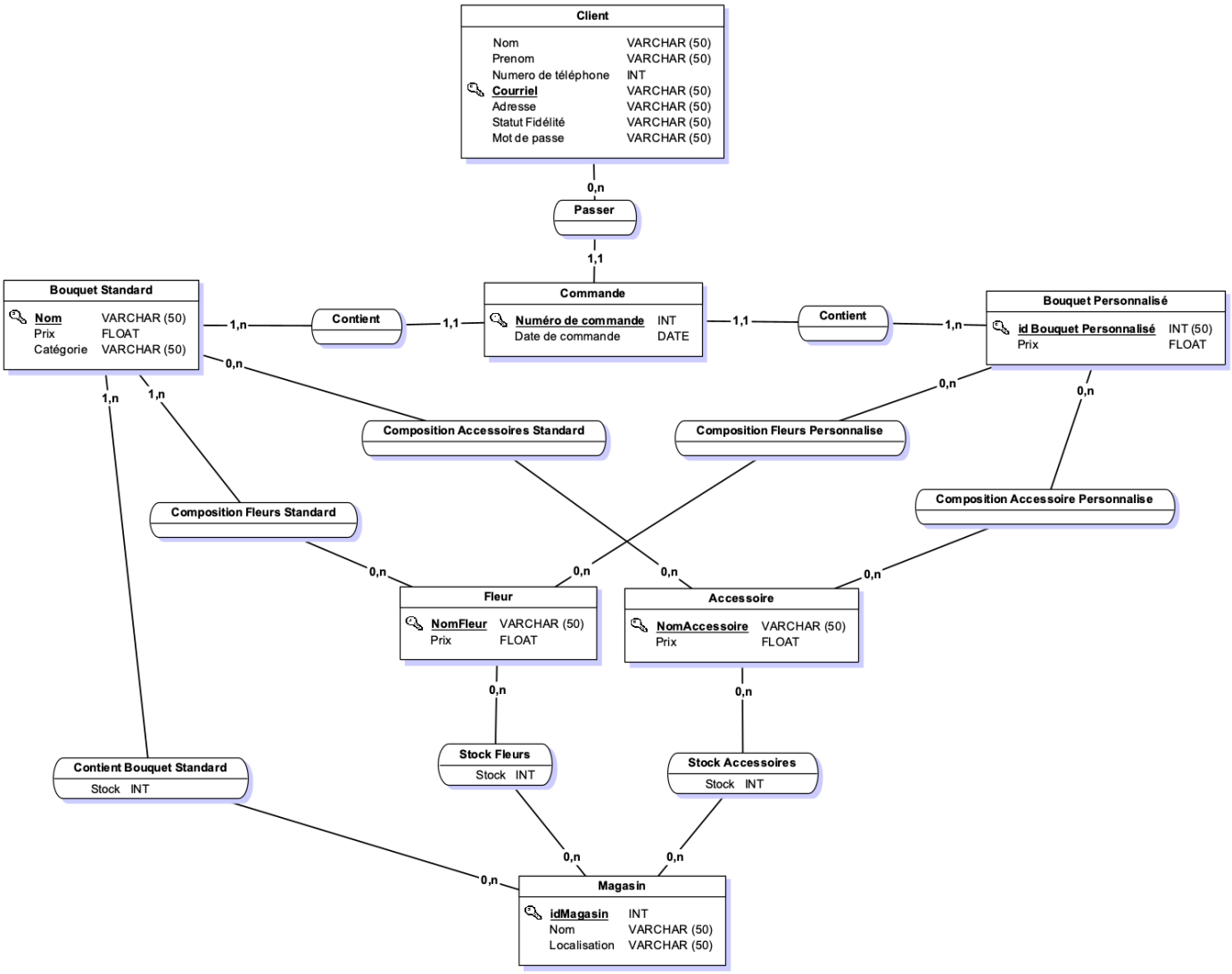
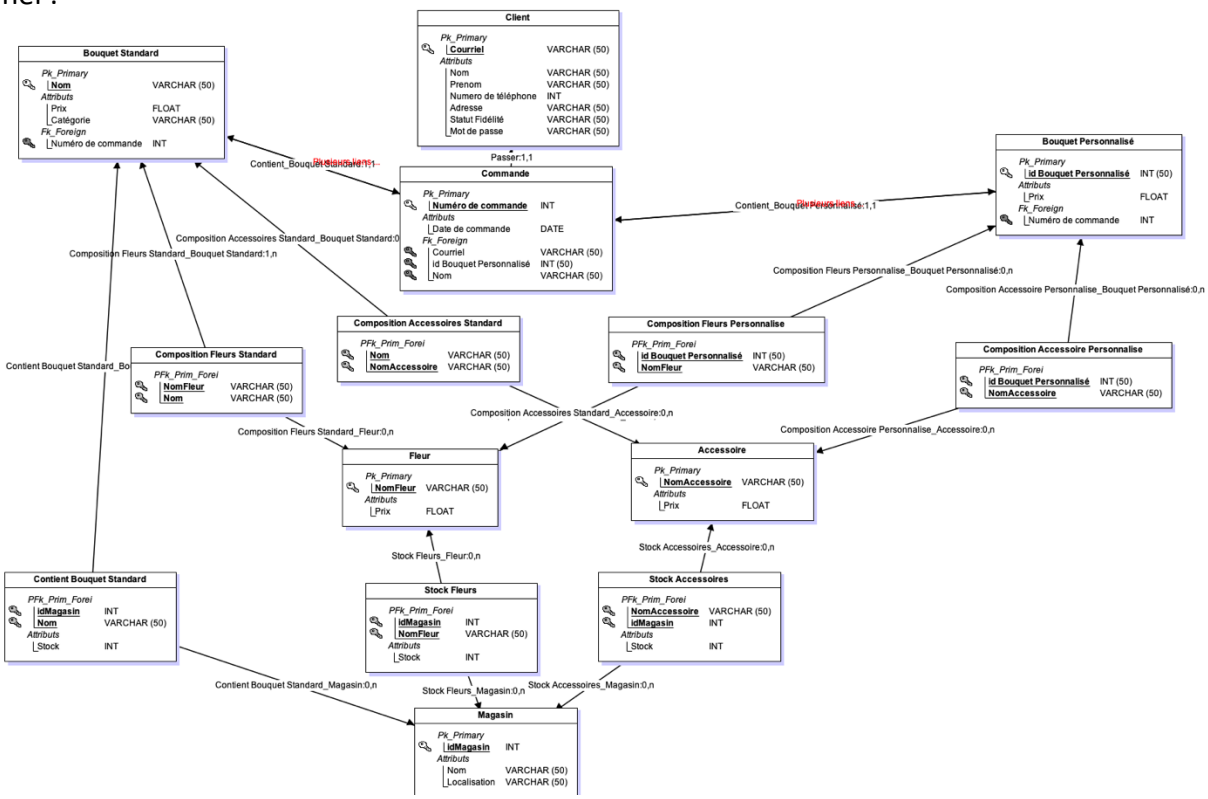


Schéma relationnel :



La base de données a ensuite été créée sur MYSQL Workbench. On l'a remplie avec quelques entités pour chaque table à titre d'exemples. En effet, le code C# permet de rajouter des entités client, fleur, accessoire, commande, …

2) Structure du code

Le code est organisé de la manière suivante : une fonction main appelle une fonction connexion qui renvoie un string qui est le courriel de la personne connectée (avec sécurité pour le mot de passe, …). En fonction de si la connexion est un client ou un administrateur, la console affiche différentes pages. La page client permet au client de passer des commandes, afficher la composition de bouquets standards ou encore de regarder son historique de commandes en appelant des fonctions annexes (définies plus loin dans le code). La page administrateur permet de gérer beaucoup plus de choses :

- Nouvelle fleur, nouvel accessoire
- Faire des requêtes SQL directement depuis la console
- Afficher toutes les commandes en préparation
- Préparer commande
- Statistiques de l'entreprise
- Nouveaux arrivages dans les magasins
- Afficher des fiches client
- Afficher les stocks d'un magasin
- Export en XML des clients à plusieurs commandes dans le dernier mois
- Changement du mot de passe
- Suppression de fleurs ou accessoires non assignés à un bouquet

```
namespace Projet
{
    0 références
    internal class Program
    {
        0 références
        static void Main(string[] args)...  //differencier interface client et admin

        //Fonctions utiles

        1 référence
        static void requete(MySqlConnection connection)...

        0 références
        static void afficherTable(string table,MySqlConnection connection)...

        9 références
        static List<string> getPrimaryKeys(string table,string pk,MySqlConnection connection) ...

        13 références
        static List<string> getPrimaryKeysLower(string table, string pk, MySqlConnection connection)...


        //Fonctions du projet

        1 référence
        static string connexion(MySqlConnection connection) ...

        2 références
        static void ajouterClient(MySqlConnection connection)...

        1 référence
        static void ficheClient(MySqlConnection connection)...

        1 référence
        static void nouvelleFleur(MySqlConnection connection)...
```

Le main ouvre une connexion MySQL et les fonctions annexes ont toutes pour argument d'entrée cette connexion. La connexion est fermée uniquement à la fin du main.

3) Quelques consignes d'utilisation

- Un employé de magasin doit se connecter en écrivant « admin » à moment où la console demande un courriel pour se connecter
- Lorsque la console demande d'entrer un magasin, il faut entrer le code de ce dernier.
- Lorsqu'un client ou un employé est connecté, il peut taper le nom d'une action à faire, il peut taper « exit » pour terminer l'exécution ou alors taper « actions » pour avoir la liste de toutes les actions qu'il peut effectuer
- A part cela, il faut juste entrer ce que la console indique de rentrer afin de pouvoir gérer au mieux les magasins de fleurs et les commandes !

4) Problèmes rencontrés

Nous n'avons pas réussi à implémenter le fait que le client dise la somme qu'il veut dépenser afin de créer un bouquet standard dont le prix sera inférieur à cette somme.

Un autre problème est la préparation des commandes. Lorsqu'un client commande une quantité qui n'est pas disponible, on lui indique simplement de commander moins mais on ne met pas sa commande en attente en attendant un arrivage dans le magasin.

5) Quelques extraits du code

Le main pour la partie « administrateur » :

```
if (co == "admin") //admin
{

    Console.Clear();
    Console.Write("Que désirez vous faire aujourd'hui ? Pour afficher la liste des actions possibles, tapez actions : ");

    List<string> actions = new List<string>();
    actions.AddRange(new string[] { "requete", "statistiques", "xml", "actions","supprimer item", "afficher commandes", "preparer commande",

    string actionDesiree = Console.ReadLine();
    while (actions.Contains(actionDesiree.ToLower()) == false)
    {
        Console.Write("Action non reconnue, veuillez réessayer : ");
        actionDesiree = Console.ReadLine();
    }


    while (actionDesiree.ToLower() != "exit")
    {
        Console.WriteLine();
        if (actionDesiree.ToLower() == "arrivage" || actionDesiree == "arrivages") { arrivages(connection); }
        else if (actionDesiree.ToLower() == "afficher commandes") { afficherCommandes(connection); }
        else if (actionDesiree.ToLower() == "statistiques") { statistiques(connection); }
        else if (actionDesiree.ToLower() == "supprimer item") { supprimerItem(connection); }
        else if (actionDesiree.ToLower() == "preparer commande") { preparerCommande(connection); }
        else if (actionDesiree.ToLower() == "nouveau bouquet standard") { nouveauBouquetStandard(connection); }
        else if (actionDesiree.ToLower() == "nouveau client") { ajouterClient(connection); }
        else if (actionDesiree.ToLower() == "nouvelle fleur") { nouvelleFleur(connection); }
        else if (actionDesiree.ToLower() == "nouvel accessoire") { nouvelAccessoire(connection); }
        else if (actionDesiree.ToLower() == "xml") { exportXMLClientsFideles(connection); }
        else if (actionDesiree.ToLower() == "infos client") { ficheClient(connection); }
        else if (actionDesiree.ToLower() == "stock") { affichageStockMagasin(connection); }
        else if (actionDesiree.ToLower() == "requete") { requete(connection); }
        else if (actionDesiree.ToLower() == "former bouquet standard") { formerBouquetStandard(connection); }
        else if (actionDesiree.ToLower() == "afficher composition bouquet standard") { afficherFormationBouquetStandard(connection); }
        else if (actionDesiree.ToLower() == "actions")
        {
            Console.Write("Voici la liste de toutes les actions disponibles : ");
            foreach (string i in actions) { Console.Write(i + ", "); }
            Console.WriteLine();
        }
        Console.WriteLine();
        Console.Write("Action terminée, veuillez en entrer une nouvelle : ");
        actionDesiree = Console.ReadLine();
        while (actions.Contains(actionDesiree.ToLower()) == false)
        {
            Console.Write("Action non reconnue, veuillez réessayer : ");
            actionDesiree = Console.ReadLine();
        }
    }
}
```

Une requête :

```
foreach (string i in nomFleurs)
{
    quantiteFleurActuelle = quantiteFleurs[cpt];
    command = connection.CreateCommand();
    command.CommandText = $"SELECT stock FROM stock_fleurs NATURAL JOIN composition_fleurs_standard WHERE idMagasin = {magasin} AND NomFleur='{i}' AND Nom_Bouquet_Standard='{bouquetForme}';";
    reader = command.ExecuteReader();
    reader.Read();
    stockFleurActuelle = int.Parse(reader.GetValue(0).ToString());
    if (quantiteFleurActuelle > stockFleurActuelle) { erreur = 1; }
    cpt++;
    reader.Close();
}
```