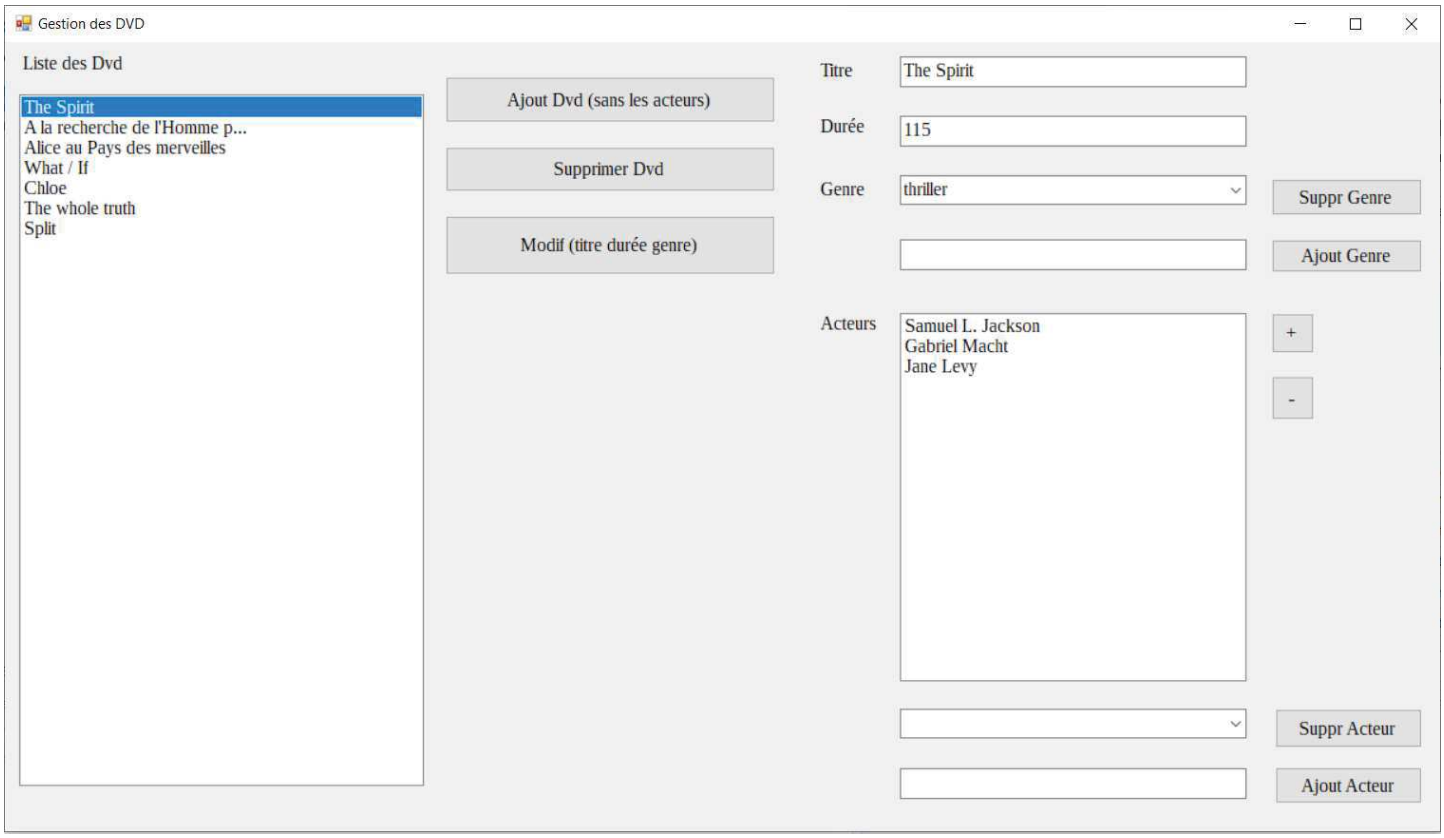
Fiche de Synthèse : Gestion DVD *Mars 2020*

Développement :
Gestion DVD est une petite application destinée, comme son nom l'indique, à gérer les DVD, basée sur une petite base de donnée MySql, que j'ai mise sur WampServer.



Cadre :
Il s'agit d'un TP proposé dans le cadre du cours "Exploitation d'un schéma de données ", par José Gil et Elisabeth Martins Da Silva.

Support :
Développé avec Windesign pour la partie MCD/MLD, et Visual Studio pour le développement en C#.

Contraintes / Organisation :
L'ordre de développement nous était fourni, ainsi que les noms des tables/champs pour la base de données, et les fonctionalités attendues pour chaque bouton de l'application.

Difficultés rencontrées :
Je me suis très très vite rendue compte que la principale difficulté était la rédaction des requêtes : elles doivent arriver en format string à la bdd, mais il faut pourtant bien utiliser des variables.
Je me suis imaginée, le doigt sur mon écran, à compter les guillemets avant et après les caractères d'échappement, pour voir si c'était OK.
J'ai donc, dès le départ ou presque, cherché une solution pour m'éviter ce piège-pour-apprentie-développeuse-fatiguée, et j'ai fini par trouver : j'ai créé une constante, en début de programme, que j'ai appelé quote, et à laquelle j'ai affecté le caractère d'échappement.

```
const string quote = "\"";
```

A partir de là...il n'y avait plus de réelles difficultés.

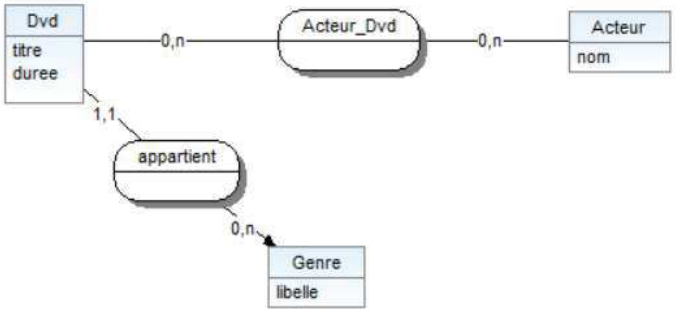
Description détaillée du développement

1. Généralités:

J'ai donc suivi les étapes suivantes, et dans cet odre :
- création du MCD (WinDesign)
- génération du MLD (WinDesign)
- génération du script SQL (WinDesign)
- création de la bdd sous MySQL à partir du script (WinDesign encore – puis phpMyAdmin)
- configuration de l'accès à la bdd à partir d'une application
- création de l'appli qui exploite cette bdd (VisualStudio)

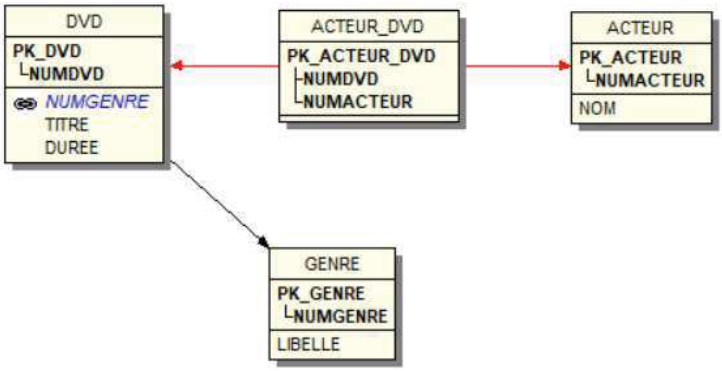
2. Création du MCD :

Le voici, avec les cardinalités appropriées :



3. Génération du MLD

WinDesign fait ça très bien, à partir du MCD, il suffit de lui demander :



4. Génération du script SQL :

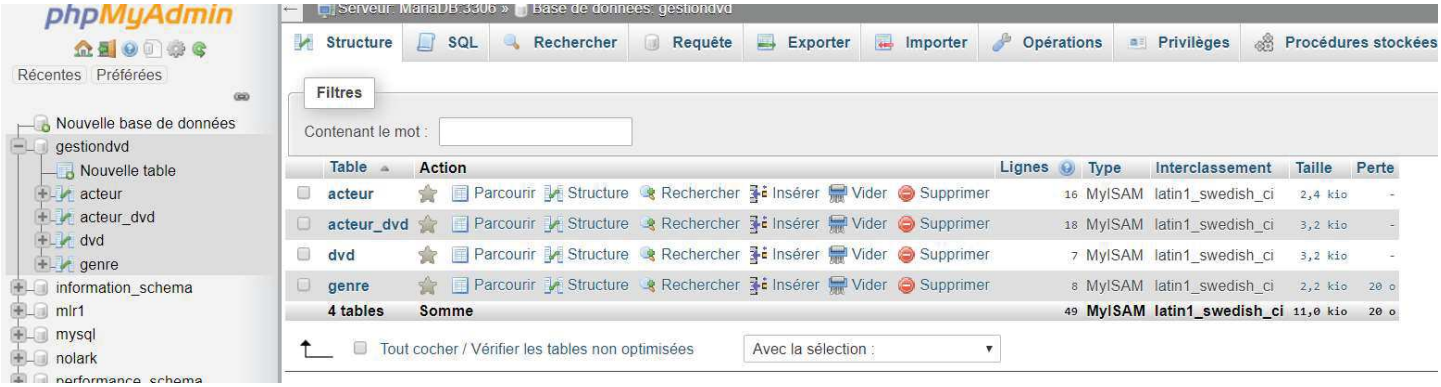
Là encore, Windesign est excellent, il génère le script MySQL correspondant.

Un extrait :

```
# ------------------------------------------------
#       TABLE : DVD
# ------------------------------------------------

CREATE TABLE IF NOT EXISTS DVD
 (
   NUMDVD INTEGER NOT NULL AUTO_INCREMENT ,
   NUMGENRE INTEGER NOT NULL  ,
   TITRE VARCHAR(30) NULL  ,
   DUREE INTEGER NULL
   , PRIMARY KEY (NUMDVD)
 )
 comment = "";

# ------------------------------------------------
#       TABLE : GENRE
# ------------------------------------------------

CREATE TABLE IF NOT EXISTS GENRE
 (
   NUMGENRE INTEGER NOT NULL AUTO_INCREMENT ,
   LIBELLE VARCHAR(30) NULL
```

5. Création de la base de données à partir du script

Cela se fait très facilement, il suffit de copier/coller le script généré par WinDesign, de le coller dans la bdd préalablement créée, et d'exécuter la requête : les tables (vides à ce moment là) sont alors créées.



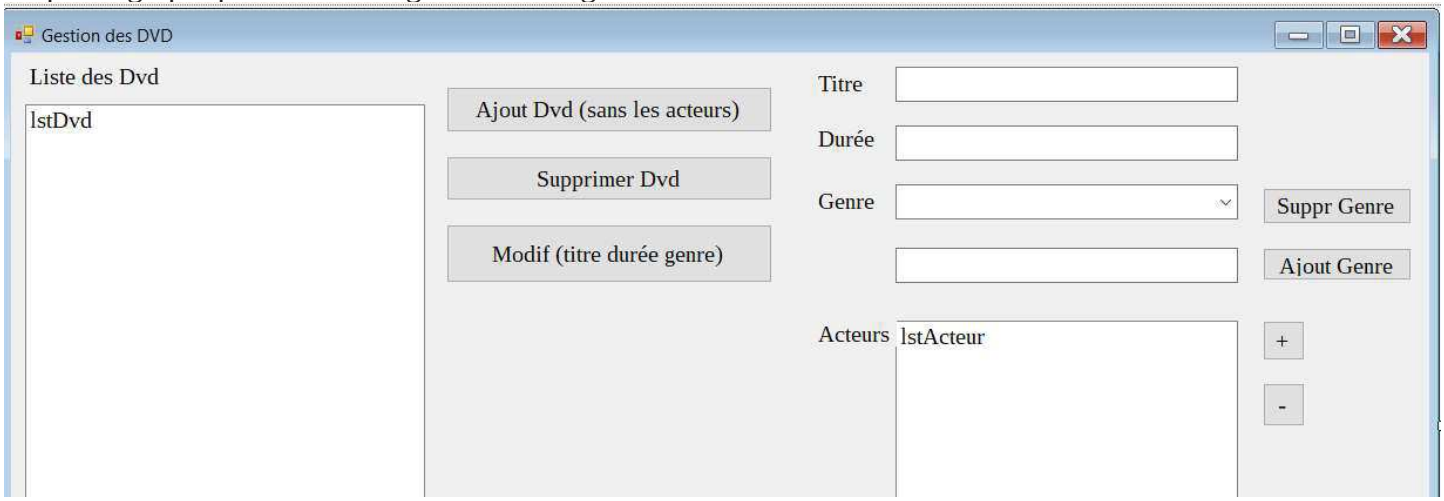
6. Configuration de l'accès à la base de données

Pour accéder à la Base de données, la classe ConnexionSql nous a été fournie : le constructeur construit un curseur qui accède à la base de données, ensuite nous avons les méthodes reqSelect() (envoi d'une requête de sélection), reqUpdate() (envoi d'une requête de mise à jour de la bdd), champ() (récupération d'un champ de la bdd), suivant() (passe à la ligne suivante du curseur), fin() (test de la fin du curseur), et close() (pour la fermeture de la connection).

```
// constructeur
20 références
public ConnexionSql(string chaineConnection)
{
    this.connection = new MySqlConnection(chaineConnection);
    this.connection.Open();
}

// execution d'une requete select
12 références
public void reqSelect(string chaineRequete)
{
    this.command = new MySqlCommand(chaineRequete, this.connection);
    this.reader = this.command.ExecuteReader();
    this.finCurseur = false;
    this.suivant();
}
```

7. Developpement de l'application qui exploite cette base de données

La partie graphique a été créée grâce au Design de Visual Studio :



Le code correspondant est alors généré automatiquement, et il est aisé d'ajouter des évènements.

Au chargement de la fenêtre : la liste des dvd se remplit à partir de la base de données, les combobox Genre et Acteur se chargent, le 1er dvd de la liste est automatiquement sélectionné, les acteurs de ce film apparaissent dans la liste Acteurs, et le titre et la durée s'affichent eux aussi.

```
private void Form1_Load(object sender, EventArgs e)
{
    // Remplissage de la liste des dvd
    chargeDvd();

    if (lstDvd.Items.Count > 0)
    {
        lstDvd.SelectedIndex = 0;

        // Remplissage des 2 comboBox
        chargeCombo(cboGenre, crs, "select libelle from genre", "libelle");
        chargeCombo(cboActeur, crs, "select nom from acteur", "nom");

        // Selection du genre du dvd
        selectionGenre(correspNumDvd_NumGenre(numeroDvd()));
    }
    lstDvd.Focus();
}
```

Ensuite on peut ajouter ou supprimer un dvd, un acteur, un genre, etc.…tout ce qu'on espère d'une application de ce type.

J'ai commencé par créer des fonctions qui me retournaient le n° de l'élement sélectionné. Par exemple, pour le numéro du dvd qui est sélectionné dans la liste ça donne :

```
/**
 * Fonction qui retourne le n° du DVD sélectionné dans la liste lstDvd
 * @return {integer}
 */
9 références
private int numeroDvd()
{
    crs = new ConnexionSql(chaineConnexion);
    crs.reqSelect("select numdvd from dvd where titre = " + quote + lstDvd.SelectedItem.ToString() + quote);
    int num = (int)crs.champ("numdvd");
    crs.close();
    return num;
}
```

Vous noterez que l'utilisation de ma constante " quote " alourdit légèrement la syntaxe de la requête au premier abord, mais elle en simplifie largement la lecture et la vérification.
J'ai des fonctions identiques pour le numéro de Genre, le numéro de l'acteur dans la combobox, dans la liste, etc.

Pour pouvoir supprimer de la bdd un acteur ou un genre, il faut d'abord s'assurer que sa suppression est possible : en effet, imaginez si je supprimais le genre " thriller " : cela créerai une erreur au niveau de la bdd, puisque j'ai au moins 1 film qui est de ce genre.
J'ai donc une fonction booléenne supprimable qui me retourne vrai si, en effet, je peux librement supprimer ce genre/cet acteur :

```
/**
 * Fonction booléenne qui contrôle si on peut supprimer une ligne de la Bdd
 */
2 références
private bool supprimable(ComboBox combo, ConnexionSql curs, string requete, string champBd,int numASupp)
{
    bool supprimable = true;
    curs = new ConnexionSql(chaineConnexion);
    curs.reqSelect(requete);
    while (!curs.fin())
    {
        if ((int)curs.champ(champBd) == numASupp)
        {
            supprimable = false;
            MessageBox.Show("Impossible de supprimer ce champ : il est lié à au moins 1 enregistrement");
        }
        curs.suivant();
    }
    curs.close();
    return supprimable;
}
```

Une fois ce contrôle validé, je peux effectuer la suppression :

```
/*
 * Procédure qui supprime une ligne de la Bdd et du Combo associé :
 */
2 références
private void supprComboEtBdd(ComboBox combo, ConnexionSql curs, string requete, int numASupp)
{
    // suppression dans la bdd
    curs = new ConnexionSql(chaineConnexion);
    curs.reqUpdate(requete + numASupp);
    curs.close();
    // suppression dans le combo
    combo.Items.Remove(combo.SelectedItem);
}
```

Pour faire un ajout dans une combobox et dans la base de données (ou dans une liste et la bdd), j'ai développé des procédures de ce type :

```
/*
 * Procédure qui ajoute un champ d'une textbox dans la bdd et dans la combobox reliée
 */
2 références
private void ajoutComboEtBdd(ComboBox combo, TextBox textbox, ConnexionSql curs, string requete)
{
    combo.Items.Add(textbox.Text);
    curs = new ConnexionSql(chaineConnexion);
    curs.reqUpdate(requete + quote + textbox.Text + quote + ")");
    curs.close();
    textbox.Text = "";
}
```

Une fois toutes les fonctions et procédures nécessaires écrites, il suffit de les ajouter sur les évènements correspondants, le plus souvent un clic sur l'un des boutons :

```
1 référence
private void btnAjoutGenre_Click(object sender, EventArgs e)
{
    if (txtAjoutGenre.Text != "")
    {
        // controle si le genre saisi n'existe pas dans la combo
        if (different(cboGenre,txtAjoutGenre))
        {
            // et ensuite ajout dans la combobox + la bdd
            ajoutComboEtBdd(cboGenre, txtAjoutGenre, crs, "insert into genre (numgenre,libelle) values(null,");
        }
    }
    lstDvd.Focus();
}
```

Le développement de cette application m'a permi d'expérimenter l'accès et la mise à jour à une base de données à partir d'un petit programme. Il y a des bases de données partout, et pour tout, ce que j'ai appris à l'occasion de ce TP me servira probablement jusqu'à la fin de ma future vie professionnelle.