C# et ADO.net

Date	Commentaire	
Avril 2018	Création	
06/04/2018	Corrections diverses et ajout schéma front-end / back-end	
01/04/2018	Corrections MySqlConnection + liste objet (schéma) + SQL	
08/04/2018	Explications chaine en C# (usage $) + SQL insert into...	

TABLE DES MATIÈRES

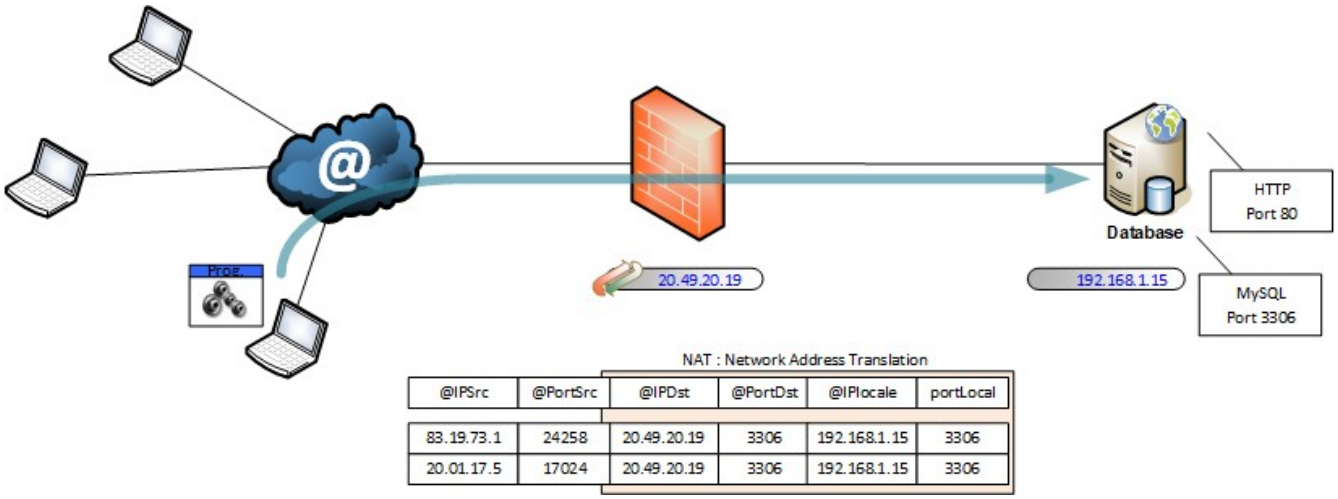
1 Visual Studio et MySQL

1.1 Les fondamentaux

1.1.1 Fonctionnement d'une base de données en réseau

Une base de données peut être accédée à distance, via le réseau. Il s'agit d'un service au même titre qu'une messagerie (SMTP port 25), qu'un DNS (port 53) ou un serveur web (port 80 ou 443).

Pour le programmeur, peu importe ce qui se passe entre les équipements réseaux : VPN, NAT, changement de protocole (IPv4 / IPv6), etc.



NAT : Network Address Translation

@IPSrc	@PortSrc	@IPDst	@PortDst	@IPlocale	portLocal
83.19.73.1	24258	20.49.20.19	3306	192.168.1.15	3306
20.01.17.5	17024	20.49.20.19	3306	192.168.1.15	3306

Cela signifie qu'un programme utilisant une base de données peut le faire depuis Internet. Il faudra cependant prendre garde à ce que :

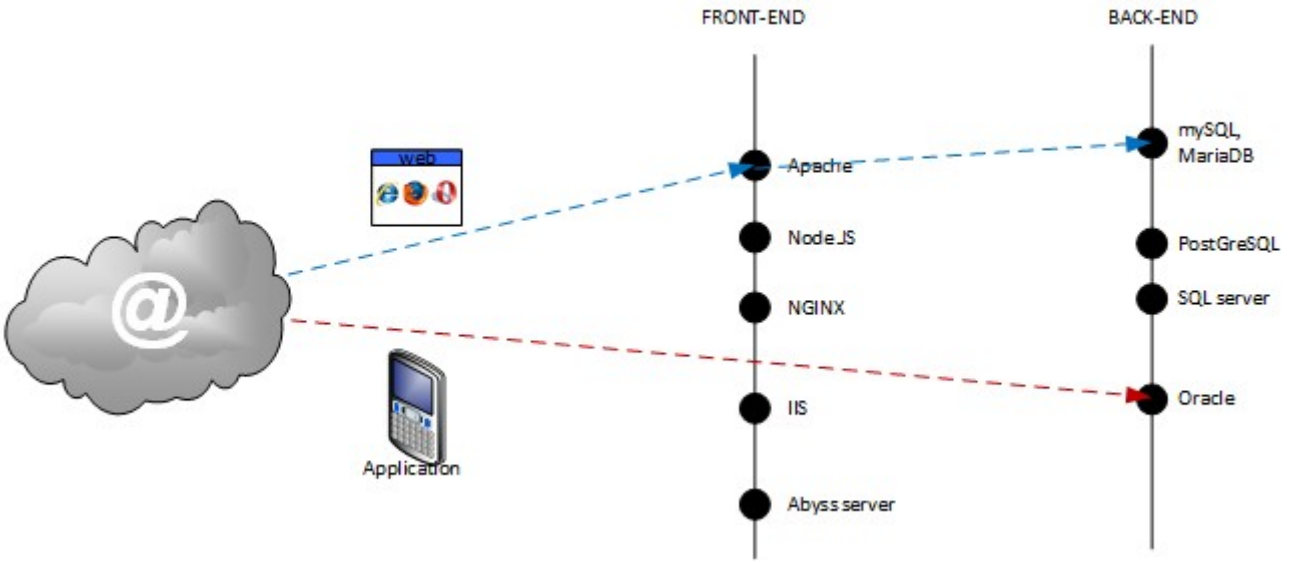
- Le moteur de base de données soit actif (et compatible avec des accès réseaux. C'est le cas de MySQL, PostgreSQL. Ce n'est pas le cas de SQLite).
- Le pare-feu accepte les connexions vers la base (règles de filtrages)
- Le pare-feu fasse de la translation d'adresse, si le réseau local utilise des adresses IP privées (RFC1918).
- Le programme utilise un compte autorisé depuis un accès réseau et bien sûr, que la configuration de la base de données autorise les accès réseaux (en général, PHP accède à la base en local et c'est une sécurité d'accès pratique pour éviter les attaques externes.

Une fois cela évalué, il faut créer la connexion, sous la forme d'un index ou *handle*, comme pour n'importe quel accès à un fichier ou flux réseau.

1.1.2 Fonctionnement d'entreprise

Il est évident que le modèle réel ne place pas la base de données sur le serveur web. Cela induirait des risques de sécurité importants (si le serveur web est compromis, la base de données le serait aussi).

On trouve l'appellation "back-end" et "front-end" qui désigne la spécialité des codeurs… c'est aussi une notion de topologie.

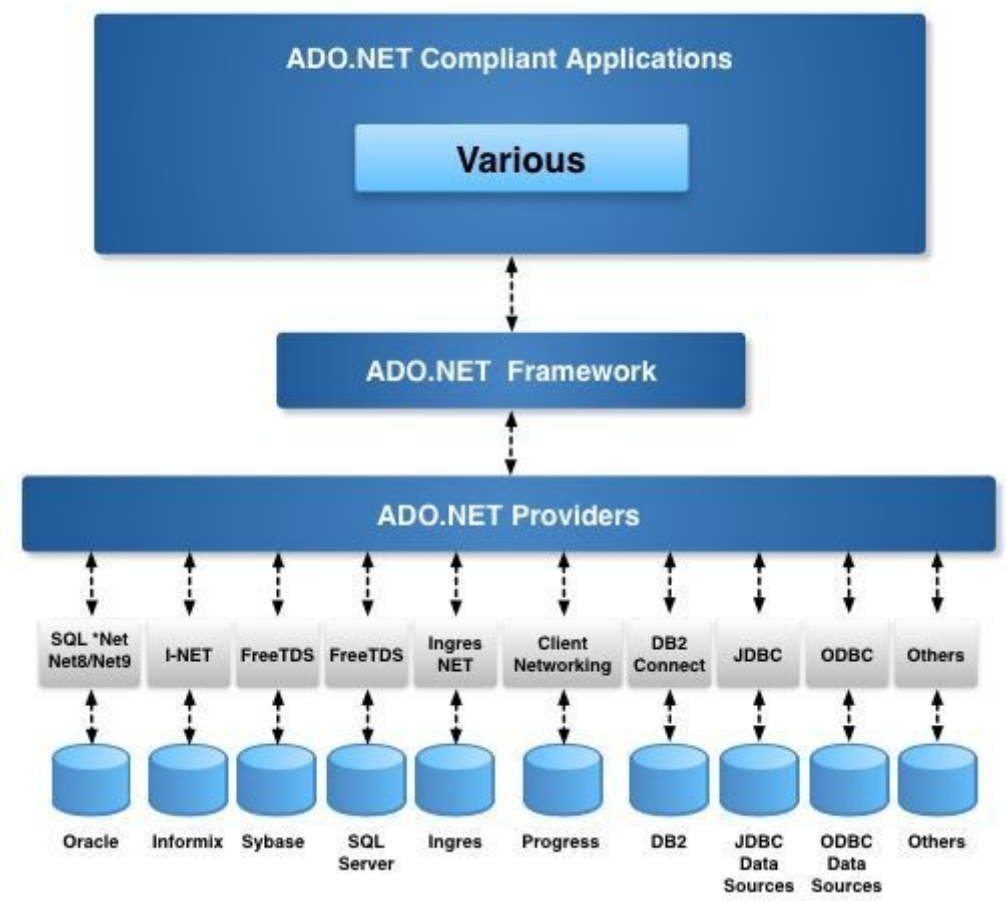


Ainsi, s'il est fréquent de rencontrer le cas en bleu, par les sites web, il n'est pas rare qu'une application Android ou iOS accède directement à la base de données (cas en rouge).

L'idée est donc de programmer en C# un client lourd (une application) qui n'utiliserait pas un serveur front-end mais accéderait aux mêmes données.

1.1.3 ADO.Net

Pour accéder aux bases de données, Microsoft a choisi de proposer un support permettant le développement simple d'interfaces. Ce modèle, présent dans le système DotNet permet donc l'utilisation de "pilotes" adaptés pour les accès aux bases de données, comme le montre le schéma suivant :

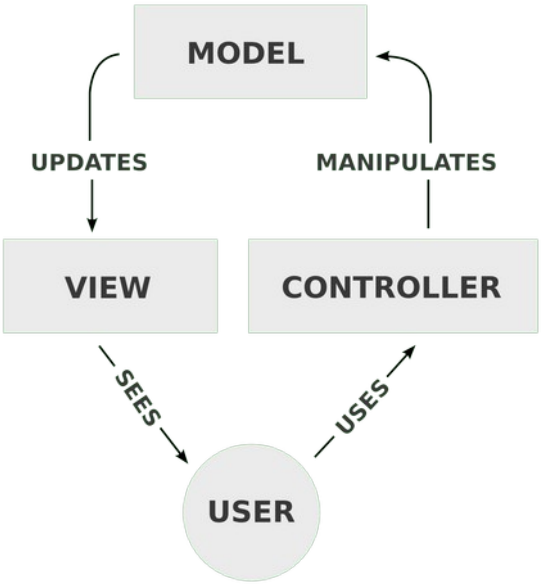


Concrètement, il s'agit d'un ensemble de classes, permettant d'interagir avec les différents modules de bases de données. Certains attribuent le nom historique à "ActiveX Datas Objects" mais les évolutions sont si importantes qu'il ne faut pas confondre les deux produits.

1.1.4 Modèle MVC et développement

Ce TP reprend le code source disponible à la fin du document (sur CodeProject).

Il faut donc intégrer ce "design pattern" : il s'agit d'un modèle utilisé dans le développement. C'est principalement une manière de penser, il n'y a pas de normes définies !



Ainsi, les trois éléments constituant le modèle MVC sont :

- Le modèle, qui représente majoritairement les données et leurs chaînes de traitements.
- La vue, qui affiche le résultat des données à l'utilisateur.
- Le contrôleur, qui permet les interactions avec le modèle.

Prenons un exemple de la vie de tous les jours : un conducteur ne regarde pas comment fonctionne son moteur (quantité d'air, pression du turbo, déclenchement de la vanne EGR…). Le constructeur gère le modèle et en tire des informations qu'il affiche au tableau de bord (la vue). En fonction de cette vue, le conducteur va appuyer sur les pédales ou activer les éléments utiles (phares, clignotants, essuie-glaces…) via le "contrôleur".

L'intérêt pour le développement informatique est de moduler son code en 3 parties. Ainsi, le changement de modèles (changement de moteur de base de données) ne nécessitera de ne retoucher le code que de cette partie (la vue et le contrôleur ne devrait pas être affectés).

Inversement, un nouveau système d'interface impliquera la mise à jour de la vue et du contrôleur, mais le code lié au modèle restera identique (et donc fiable).

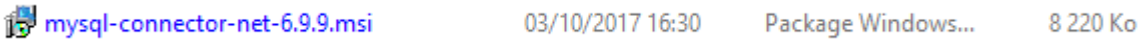
1.2 Mise en œuvre des concepts

1.2.1 Installation connecteur MySQL (ADO)

Téléchargez puis installez le connecteur ADO pour MySQL (Oracle).

https://dev.mysql.com/downloads/connector/net/

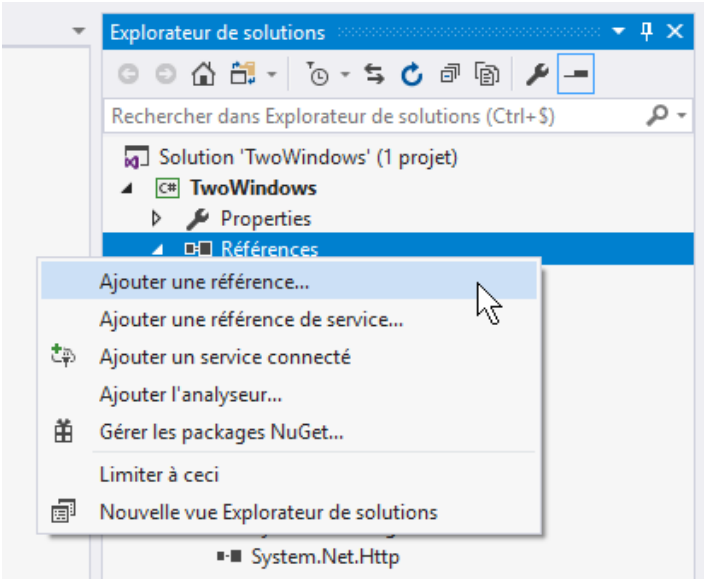
Installez le connecteur...



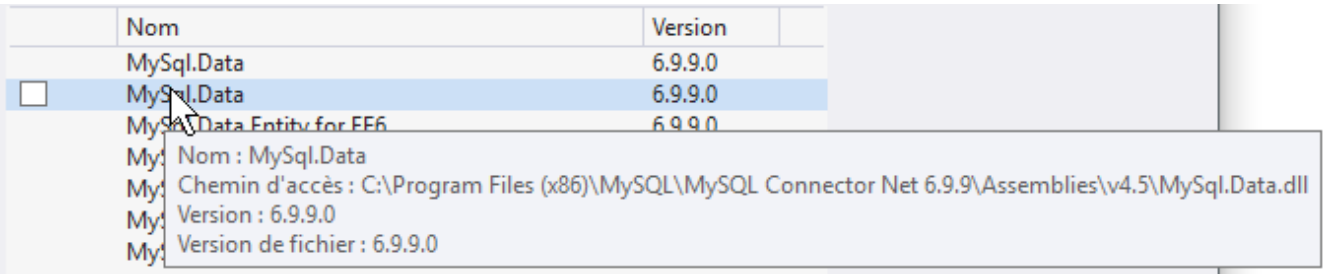
1.2.2 Création d'un projet C#

Créer un nouveau projet Windows Form

Dans l'Explorateur de solutions, ajouter une nouvelle référence, chercher MySql.Data (une dll).

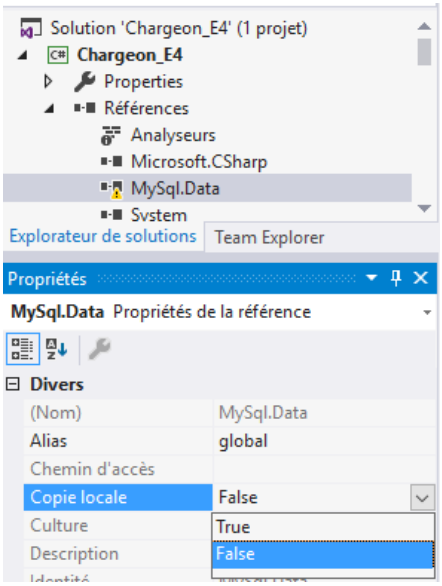


Choisir la version la plus intéressante (4.0 ou 4.5 visibles en passant la souris au-dessus du nom) :



Par défaut, la propriété sur MySql.Data doit avoir "Copie Locale" à **True**.

Dans la zone propriétés, dépliez l'attribut "Références" puis cliquez sur MySqlData et vérifier la valeur.



Initialisation de connexion

Il est nécessaire d'appeler la bibliothèque de fonction MySqlConnection :

```
using MySql.Data.MySqlClient;
```

Si les mots clés restent soulignés, n'oubliez pas d'ajouter la référence au projet MySqlData.

Il faut initialiser l'attribut connexion qui est de type MySqlConnection :

```
MySqlConnection cnxBase;
```

Comme pour un accès en PHP, l'utilisation d'ADO permet d'utiliser une chaîne de caractère pour la connexion :

```
"SERVER=localhost ; DATABASE='nomBase' ; UID='root' ; PASSWORD='' ; "
```

ou bien par l'usage de variables :

```
String server   = "localhost";
String database = "connectcsharptomysql";
String uid      = "username";
String password = "password";
string chaineConnexion;
chaineConnexion = "SERVER=" + server + ";" + "DATABASE=" +
            database + ";" + "UID=" + uid + ";" + "PASSWORD=" + password + ";";
```

une fois la chaîne de connexion préparée, il suffit de l'utiliser avec

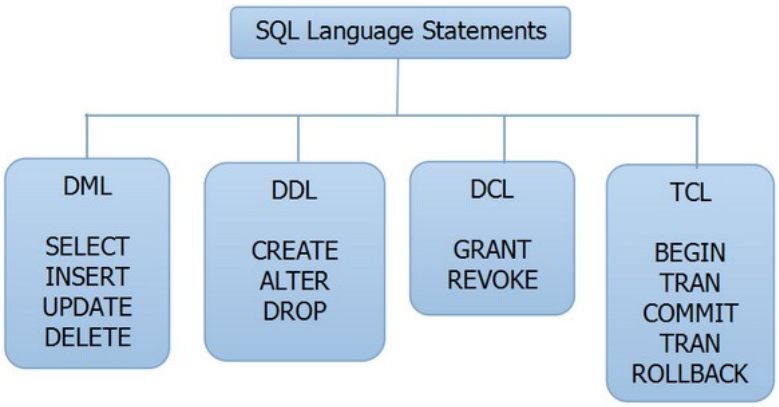
```
cnxBase = new MySqlConnection(chaineConnexion) ;
```

1.2.3 Ouverture et fermeture de la connexion

Une fois la chaîne de connexion formatée et préparée, il faut ouvrir la connexion avec la base :

```
private void btnConnecter_Click(object sender, EventArgs e) {
      string chaineConnexion;
      chaineConnexion = "SERVER=" + txtHost.Text + ";" + "DATABASE=" + txtDB.Text +
                  ";" + "UID=" + txtID.Text + ";" + "PASSWORD=" + txtMDP.Text + ";";
      if (btnConnecter.Text == "Connecter") {
            cnxBase = new MySqlConnection(chaineConnexion);
            try {
                  cnxBase.Open();
                  btnConnecter.Text = "Déconnecter";
            }
            catch (MySqlException ex) {
                  MessageBox.Show("Err ouv." + ex.ToString() + ": " + ex.Message);
            }
      } else {
            try {
                  cnxBase.Close();
                  btnConnecter.Text = "Connecter";
            }
            catch (MySqlException ex) {
                  MessageBox.Show("Err fin " + ex.ToString() + ": " + ex.Message);
            }
      }
}
```

Il est désormais possible de travailler sur la base de données :



La plupart des exemples proposent de préparer la commande, de l'exécuter puis de fermer la connexion. Cependant en termes de performances, n'est pas efficace.

Pour cet exemple qui utilise une fenêtre graphique, nous conservons la connexion active, jusqu'à ce que l'utilisateur estime ne plus en avoir besoin.

1.2.4 Écrire, modifier et lire dans la base

Imaginons une base '**heroesDB**' avec une table personnageHeroes, constituée des colonnes suivantes :

userFirstName	userlastName	userBirthDay	UserID
Texte	Texte	Date	Int (AUTO)
Jones	INDIANA[1]	1899-07-01	1

Il faut la créer (par exemple, utilisez PHPMyAdmin et les commandes dans l'annexe).

Nous pouvons interagir utilisant les commandes SQL et les commandes propres à C# pour les manipulations :

- **ExecuteNonQuery() :** cette commande ne renvoie rien. Elle est utilisée pour les insertions, les mises à jour et les effacements (Insert, update et delete).
- **ExecuteReader() :** cette commande peut retourner de 0 à plusieurs enregistrements, comme dans la commande SQL 'Select'.
- **ExecuteScalar() :** cette commande n'est employée que lorsque le résultat ne contient qu'une valeur de retour (par exemple "Select Count(*)").

1.2.4.1 Écrire

La commande la plus simple est l'écriture, dont voici la syntaxe :

Pas de quotes ou guillemets

```
string query = "INSERT INTO madatabase (userFirstName, userlastName, userBirthDay)
VALUES('Bruce','WAYNE','1972-02-19')";

MySqlCommand cmd = new MySqlCommand(query, cnxBase);
cmd.ExecuteNonQuery();
```

quotes ou guillemets pour les valeurs

Pour créer la variable 'query' en C# à partir de variables, il est possible d'utiliser la syntaxe suivante :

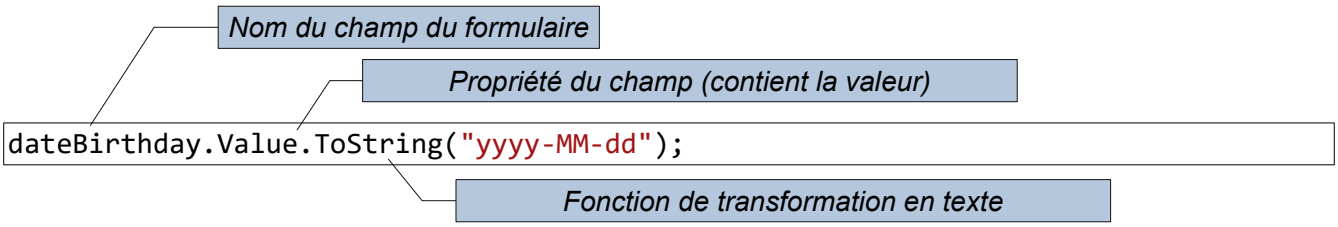
```
string query = $"INSERT INTO personnageheroes (userFirstName, userlastName, userBirthDay)
VALUES ('{txtPrenom.Text}','{txtNom.Text}','{dateBirthday.Value.ToString("yyyy-MM-
dd")}')";
```

sinon, il faut concaténer les variables

```
string query = "INSERT INTO personnageheroes (userFirstName, userlastName, userBirthDay)
VALUES ('"+txtPrenom.Text+"','"+txtNom.Text+"','"+dateBirthday.Value.ToString("yyyy-MM-
dd")+"')";
```

1 http://indianajones.wikia.com/wiki/Indiana_Jones

Le cas particulier d'une écriture d'un champ DateTime de C# vers **Date** de MySQL :



MySQL utilise une date au format année-mois-jour : il est donc nécessaire de convertir le champ du formulaire C# (un nombre entier depuis 1970) en date acceptable par la base, grâce au paramètre "yyyy-MM-dd" dans la fonction ToString.

Attention, yyyy-mm-dd entraîne une erreur car mm = minutes et MM = mois.

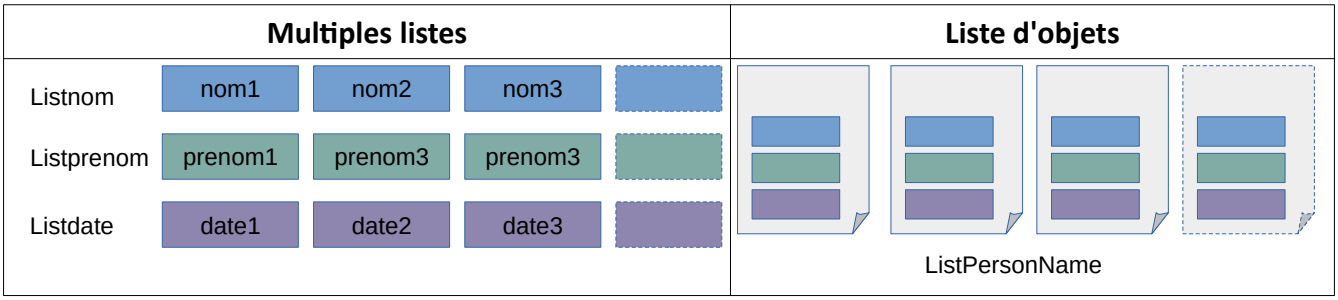
1.2.4.2 Mettre à jour

Le principe est similaire, avec un WHERE pour sélectionner la ligne à modifier :

```
string query = "UPDATE personnageheroes SET userFirstName='John' WHERE
userlastName='WAYNE'";
MySqlCommand cmd = new MySqlCommand(query, cnxBase);
cmd.ExecuteNonQuery();
```

1.2.4.3 Lire

L'opération est un peu plus compliquée, car il faut récupérer les valeurs. Sans connaître leur nombre, il faut utiliser des listes (ou collections). Or, plusieurs listes donnent un risque de désynchronisation, il devient préférable d'utiliser un objet contenant les mêmes attributs que la base de données.



Voici une classe d'exemple :

```
public class PersonName
{
    // attributs de l'objet : un par colonne
    private String prenom;
    private String nom;
    private DateTime dnaissance;

    // méthodes d'accès (get/set)
    public String Prenom { get => prenom; set => prenom = value; }
    public String Nom { get => nom; set => nom = value; }
    public DateTime Dnaissance { get => dnaissance; set => dnaissance = value; }
   // utilisation objet.Prenom = "Toto" ou maVariable = objet.Prenom

    // méthodes de conversion SQL <-> C# (les formats sont différents)
    public String Convert2MySQL(DateTime dt) {
        return dt.ToString("yyyy-MM-dd HH:mm:ss");
     }

    public PersonName(String prenom, String nom, DateTime madate) {
        this.prenom = prenom;
        this.nom = nom;
        this.dnaissance = madate;
     }
}
```

*Notez la méthode de conversion d'une date en C# vers le format texte de la date en mySQL ! ToString("yyyy-MM-dd HH:mm:ss") qui est le format **DateTime** de MySQL (format anglophone).*

Pour faire les choses proprement, les attributs sont privés et nous définissons des getters et setters (accesseurs et mutateurs) . Vous pouvez le faire en utilisant [Alt]+[Entrée] sur les attributs.

Pour utiliser les accesseurs ou mutateurs ainsi créé, rien de plus simple :

```
PersonName Objet = new PersonName();
Private String unPrenom = Objet.Prenom;
Objet.Prenom = "Frédéric";
```

Créer un objet de la classe PersonName
Lire la valeur de l'attribut prenom
Écrire la valeur de l'attribut prenom

Il suffit maintenant de créer une liste d'objet PersonName, ayant pour nom lstPerson :

```
private List<PersonName> lstPerson = new List<PersonName>();
```

Nous aurons besoin de convertir la date de MySQL en objet DateTime de C#…

```
public DateTime Convert2C(String sqlDate) {
    return DateTime.Parse(sqlDate);
}
```

Lecture de plusieurs lignes

Désormais, une simple boucle va nous permettre de remplir notre liste. Comme il s'agit d'une application graphique, le bouton "Voir" n'est actif que lorsqu'une connexion à la base est ouverte.

```
private void btnVoir_Click(object sender, EventArgs e) {
      string query = "SELECT * FROM utilisateurs";
      MySqlCommand cmd = new MySqlCommand(query, cnx);
      MySqlDataReader dataReader = cmd.ExecuteReader();
      txtVoir.Text = "";
      lstPerson.Clear();
      while (dataReader.Read()) {
            lstPerson.Add(new PersonName(dataReader["userFirstName"].ToString(),
            dataReader["userlastName"].ToString(),
            Convert2C(dataReader["userBirthDay"].ToString())));
      }
      dataReader.Close();

      foreach(PersonName tttt in lstPerson) {
            txtVoir.Text = txtVoir.Text + tttt.Prenom + "\t" + tttt.Nom + "\t" +
                  tttt.Dnaissance.ToString("dd-MM-yyyy") + Environment.NewLine;
      }
}
```

Équivalent fetch en PHP

Cela reste la partie la plus lourde à écrire mais facile à comprendre :

- cmd.**ExecuteReader()** va ouvrir un flux avec la base de données
- dataReader.**Read()** lit le flux et renvoie 'true' tant qu'il reste des données à lire
- dans l'objet dataReader, on isole les index des colonnes (userFirstName correspond à la première colonne de notre base).
- Lorsqu'il n'y a plus de données, on clôture le flux avec dataReader.**Close()**.

Lecture d'une valeur simple

Pour lire une seule valeur, l'usage de la commande **ExecuteScalar**() simplifie le codage :

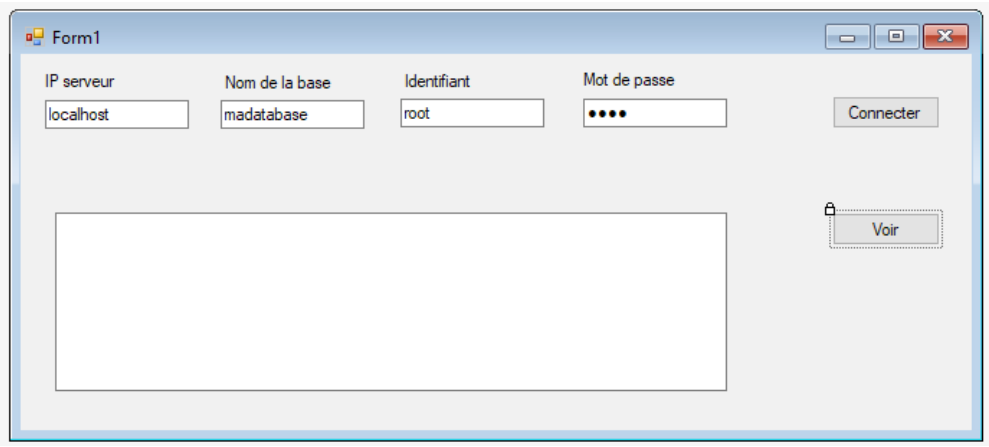
```
string query = "SELECT password FROM utilisateurs where login ='"+login+"'";
MySqlCommand cmd = new MySqlCommand(query, cnx);
String password = (String)cmd.ExecuteScalar();
```

Notez l'usage de la déclaration explicite du type d'attribut.

1.3 SOURCES

https://www.codeproject.com/kb/database/connectcsharptomysql.aspx (Etienne RACHED)

1.4 CODE SOURCE :




```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace MySQLAccess
{
    public partial class Form1 : Form
    {
        MySqlConnection cnx;
        private List<PersonName> lstPerson = new List<PersonName>();

        public Form1() {
            InitializeComponent();
        }

        private void btnConnecter_Click(object sender, EventArgs e) {
            string chaineConnexion;
            chaineConnexion = "SERVER=" + txtHost.Text + ";" + "DATABASE=" + txtDB.Text + ";" +
"UID=" + txtID.Text + ";" + "PASSWORD=" + txtMDP.Text + ";";
            if (btnConnecter.Text == "Connecter") {
                cnx = new MySqlConnection(chaineConnexion);
                try {
                    cnx.Open();
                    btnConnecter.Text = "Déconnecter";
                    btnVoir.Enabled = true;
                }
                catch (MySqlException ex) {
                    MessageBox.Show("Erreur ouverture " + ex.ToString() + ": " + ex.Message);
                }
            } else {
                try {
                    cnx.Close();
```

```
                    btnConnecter.Text = "Connecter";
                    btnVoir.Enabled = false;
                }
                catch (MySqlException ex) {
                    MessageBox.Show("Erreur cloture " + ex.ToString() + ": " + ex.Message);
                }
            }
        }
        public DateTime Convert2C(String sqlDate) {
            return DateTime.Parse(sqlDate);
        }

        private void btnVoir_Click(object sender, EventArgs e) {
            string query = "SELECT * FROM utilisateurs";
            MySqlCommand cmd = new MySqlCommand(query, cnx);
            MySqlDataReader dataReader = cmd.ExecuteReader();
            txtVoir.Text = "";
            lstPerson.Clear();
            while (dataReader.Read()) {
                lstPerson.Add(new PersonName(dataReader["userFirstName"].ToString(),
dataReader["userlastName"].ToString(), Convert2C(dataReader["userBirthDay"].ToString())));
            }
            dataReader.Close();
            foreach(PersonName tttt in lstPerson) {
                txtVoir.Text = txtVoir.Text + tttt.Prenom + "\t" + tttt.Nom + "\t" +
tttt.Dnaissance.ToString("dd-MM-yyyy") + Environment.NewLine;
            }
        }
    }

    public class PersonName
    {
        // attributs de l'objet : un par colonne
        private String prenom;
        private String nom;
        private DateTime dnaissance;

        // méthodes d'accès (get/set)
        public String Prenom { get => prenom; set => prenom = value; }
        public String Nom { get => nom; set => nom = value; }
        public DateTime Dnaissance { get => dnaissance; set => dnaissance = value; }

        // méthodes de conversion SQL <-> C# (les formats sont différents)
        public String Convert2MySQL(DateTime dt) {
            return dt.ToString("yyyy-MM-dd HH:mm:ss");
        }

        public PersonName(String prenom, String nom, DateTime madate) {
            this.prenom = prenom;
            this.nom = nom;
            this.dnaissance = madate;
        }
    }
}
```

1.5 CRÉATION DATABASE 'HEROESDB'

Code SQL pour créer la base de l'exemple.

`Table personnageHeroes`

userFirstName	userlastName	userBirthDay	UserID
Texte	Texte	Date	Int (AUTO)
Jones	INDIANA[2]	1899-07-01	1

```
CREATE DATABASE `heroesDB` /*!40100 COLLATE 'utf8mb4_general_ci' */

CREATE TABLE `personnageHeroes` (
`UserID` INT NULL,
`userFirstName` TEXT NULL DEFAULT NULL,
`userLastName` TEXT NULL DEFAULT NULL,
`userBirthDay` DATE NULL DEFAULT NULL,
UNIQUE INDEX `UserID` (`UserID`)
)
COLLATE='utf8mb4_general_ci'
AUTO_INCREMENT=1
;

INSERT INTO personnageheroes (userFirstName, userLastName, userBirthDay) VALUES
('Chuck','NORRIS','1940-03-10'),
INSERT INTO personnageheroes (userFirstName, userLastName, userBirthDay) VALUES
('Indiana','JONES','1899-07-01');
```

2 http://indianajones.wikia.com/wiki/Indiana_Jones