



Hiver 2021

Introduction à ADO.NET



Saliha Yacoub
CLG

Table des matières

Chapitre1, introduction	3
Chapitre 2, prés-requis :.....	6
A. Méthode 1 , plus simple recommandée.....	6
B. Méthode 2, si la méthode 1 ne marche pas.....	10
Chapitre3, l'objet OracleConnection.....	13
Description de l'objet OracleConnection	13
Propriétés importantes de l'objet OracleConnection	13
Propriété ConnectionString.....	13
Propriété State.	14
Méthodes importantes :.....	14
La méthode Open().....	14
La méthode Close().....	14
Exemple	15
Chapitre 4, l'objet OracleCommand.....	17
Quelques propriétés de l'objet OracleCommand	17
Méthodes importantes de l'objet OracleCommad	18
La méthode ExecuteNonQuery().....	18
La méthode ExecuteReader()	19
La méthode ExecuteScalar()	19
Chapitre 5, l'objet OracleDataReader	20
Quelques propriétés importantes de l'OracleDataReader.....	20
Quelques méthodes importantes de OracleDataReader	20
Chapitre 6, exemple avec Windows Form.....	23
Présentation	23
Construction de l'interface.....	24
Le DataGridView	25
Le code C#	27
Explications, la fonction affichertous().....	30
Explications de la fonction listerEquipe()	31
Explications de la fonction listejoueursEquipe()	33

Insertion des données dans la base de données.....	34
Exemple de fonction, qui compte les joueurs dans chaque équipe.....	35
Sources	37

Chapitre1, introduction

ADO.NET est un ensemble de classes qui exposent les services d'accès aux données pour les programmeurs .NET Framework. ADO.NET propose un large ensemble de composants pour la création d'applications distribuées avec partage de données. Partie intégrante du .NET Framework, il permet d'accéder à des données relationnelles, XML et d'application. ADO.NET répond à divers besoins en matière de développement, en permettant notamment de créer des clients de bases de données frontaux et des objets métier de couche intermédiaire utilisés par des applications, outils, langages ou navigateurs Internet.

ADO.NET intègre deux modèles d'accès aux données qui sont :

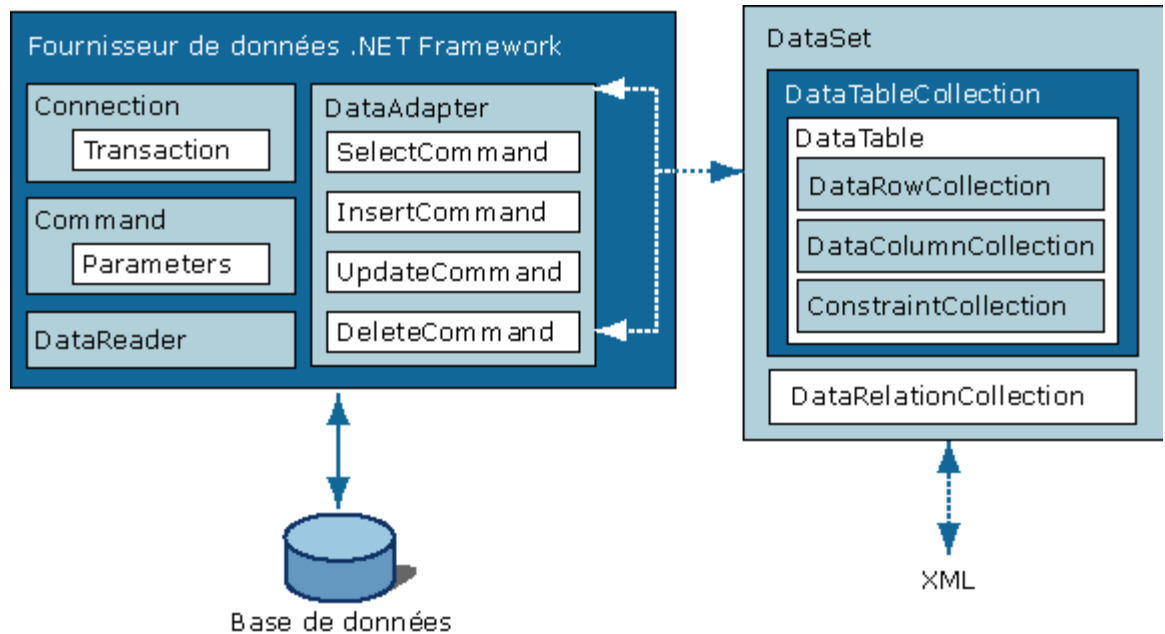
- Le modèle « **connecté** » qui est bien adapté aux applications à deux couches traditionnelles; c'est ce mode que nous allons aborder
- Le modèle « **déconnecté** » qui est destinés aux applications multicouches en utilisant un DataSet. Ne sera pas abordé ici.

Les sources de données peuvent être :

- des SGBD relationnels tels **Microsoft SQL Server** et **Oracle**
- des sources de données exposées via **OLE DB**.
- des sources de données exposées via **XML**.

Les composants de ADO.NET ont été conçus de façon à distinguer l'accès aux données de la manipulation de données. Cette distinction est rendue possible par deux composants centraux de ADO.NET : le **DataSet** et le **fournisseur de données**.

Le schéma suivant représente les composants de l'architecture ADO.NET.



Le fournisseur de données

Un fournisseur de données est utilisé pour :

- La connexion à une base de données ;
- L'exécution de commandes;
- L'extraction de résultats.

En ADO.NET, les principaux fournisseurs de données sont les suivants :

- Fournisseur de données « .NET Framework » pour SQL Server ;
- **Fournisseur de données « Oracle Data Provider pour le NET (ODP.NET) » pour Oracle**
- Fournisseur de données « .NET Framework » pour OLE DB;
- Fournisseur de données « .NET Framework » pour ODBC ;

Avec ADO.NET, le fournisseur de données est conçu pour être léger et créer une couche minimale entre la source de données et votre code, afin d'augmenter les performances sans réduire la fonctionnalité. Il se comprend un ensemble de composants comprenant les objets **Connection**, **Command**, **DataReader** et **DataAdapter**.

Ces composants sont explicitement conçus pour la manipulation des données et un accès aux données rapide. L'objet **Connection** assure la connectivité avec une source de données. L'objet **Command** permet l'accès aux commandes de base de données pour retourner des données, modifier des données, exécuter des procédures stockées et envoyer ou extraire des informations sur les paramètres. Le **DataReader** fournit un flux très performant de données en provenance de la source de données. Enfin, le **DataAdapter** établit une passerelle entre l'objet **DataSet** et la

source de données. Le **DataAdapter** utilise les objets **Command** pour exécuter des commandes SQL au niveau de la source de données afin d'une part d'approvisionner le **DataSet** en données, et d'autre part de répercuter dans la source de données les modifications apportées aux données contenues dans le **DataSet**.

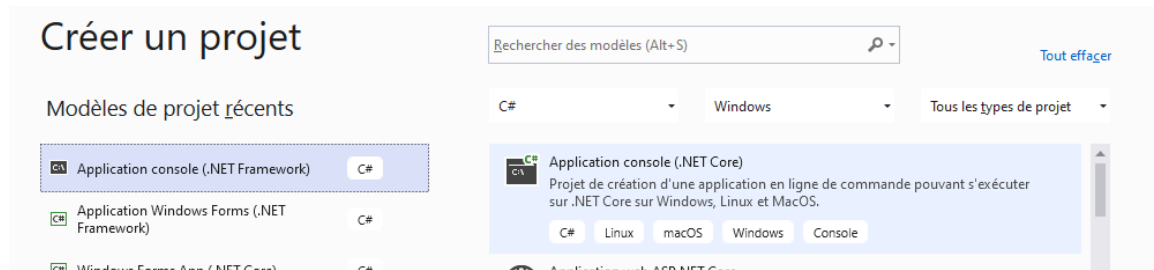
Dans le document qui suit, et puisque notre base de données est ORACLE, nous utiliserons le provider ODP.NET, qui est le fournisseur d'accès aux données oracle

Chapitre 2, prés-requis :

Pour pouvoir utiliser les classes d'Oracle Developer Tools for Visual Studio.

A. Méthode 1¹, plus simple recommandée

Démarrer une application C# Console



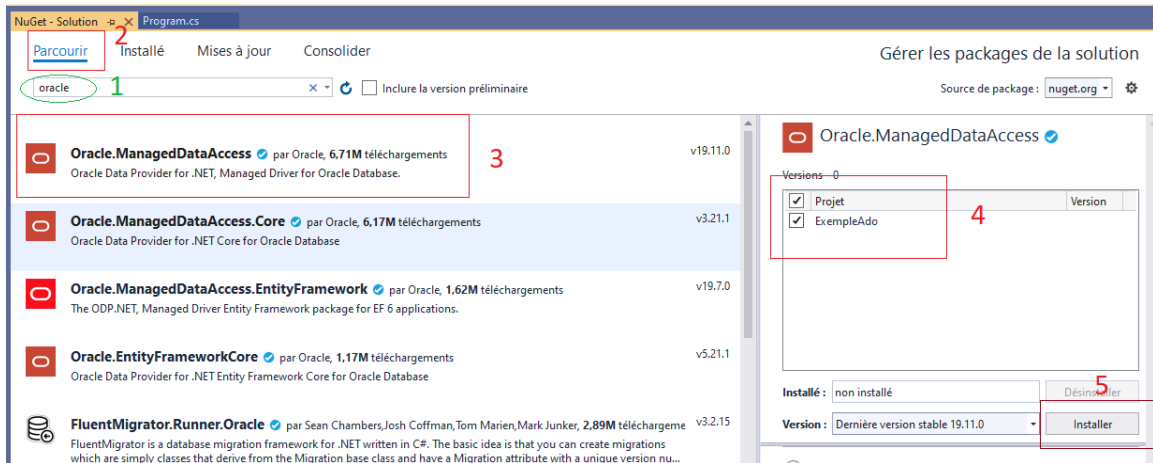
Dans l'onglet Outils, aller à Gérer les packages NuGet pour la solution



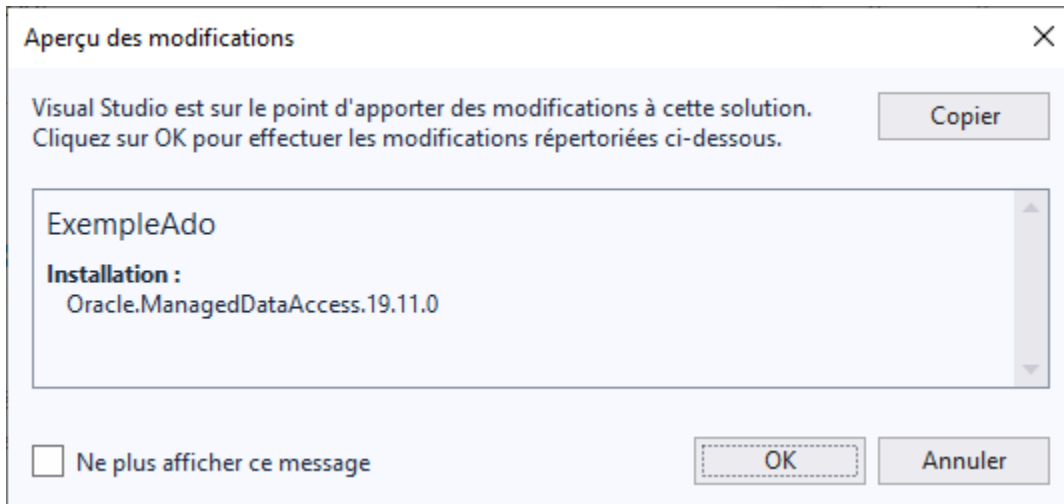
Puis la fenêtre suivante s'ouvre :

- 1- Tapez oracle dans la barre de recherche
- 2- Cliquer sur parcourir
- 3- Choisir le package Oracle.ManagedDataAccess
- 4- Cocher les deux cases (projet et solution)
- 5- Installer

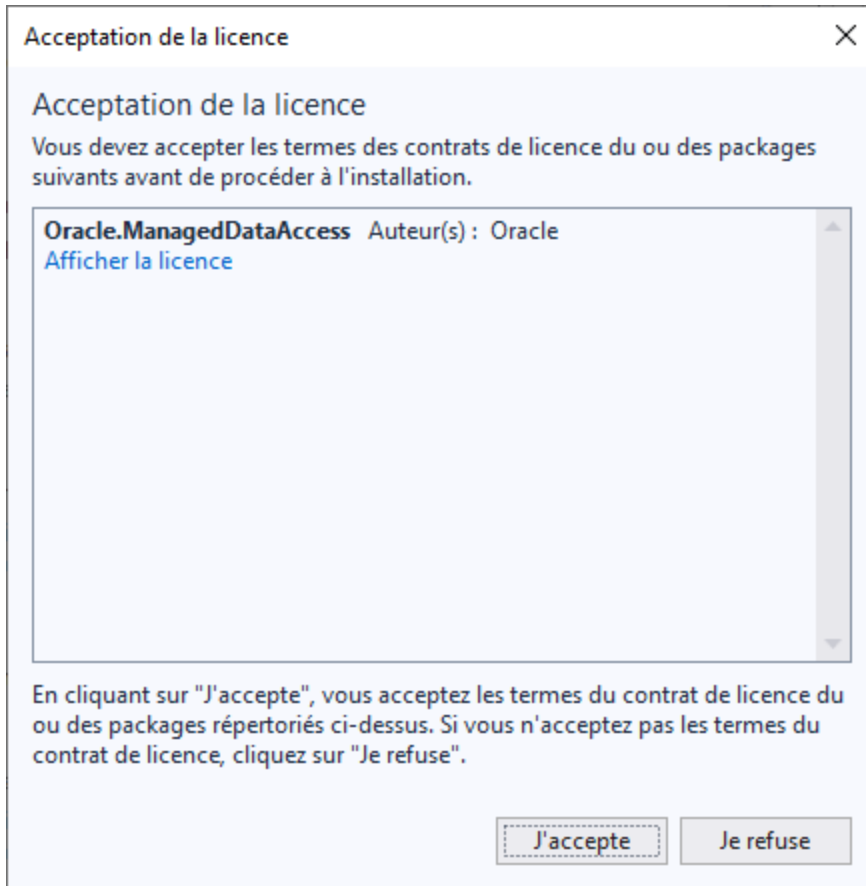
¹ De Marc Beaulne



Un message vous indiquant que Visual Studio va apporter des modifications à votre solution.
Cliquer OK

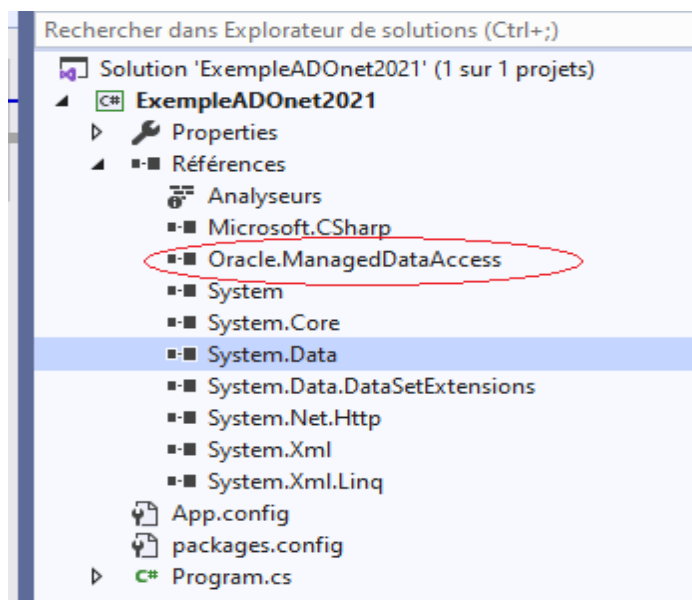


La figure suivante apparaît. Accepter la licence.



Cliquer sur Accepter la licence

Lorsque cette étape est complétée alors, vous allez voir dans les propriétés du projet qu'une référence a été ajoutée au projet (voir la figure suivante) :



Dans votre projet, ajouter la ligne suivante

```
using Oracle.ManagedDataAccess.Client;
```

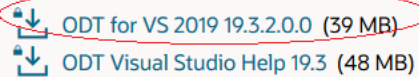
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using Oracle.ManagedDataAccess.Client;
7
8  namespace ExempleAdo
9  {
10     0 références
11     class Program
12     {
13         0 références
14         static void Main(string[] args)
15         {
16     }
```

B. Méthode 2, si la méthode 1 ne marche pas.

1. Installer ODT (Oracle Development Tools) for Visual Studio 2019 allez à l'adresse.
https://www.oracle.com/database/technologies/dotnet-odtvsix-vs2019-downloads.html?elq_mid=152472&sh=022624120625082615181926150607150413&cmid=WWMK180508P00032C0020

Oracle Developer Tools for Visual Studio 2019 Downloads

Oracle Developer Tools for Visual Studio 2019



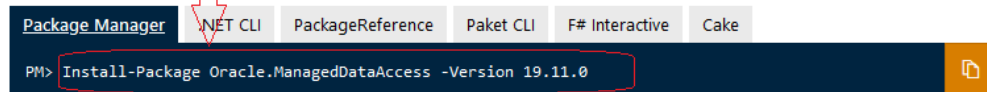
2. Dézipper le fichier, puis installer.
3. Démarrer un projet Visual Studio (application console)
4. Aller sur le site :
<https://www.nuget.org/packages/Oracle.ManagedDataAccess/>
5. Copier l'instruction suivante : (voir figure plus bas)

```
Install-Package Oracle.ManagedDataAccess -Version 19.11.0
```

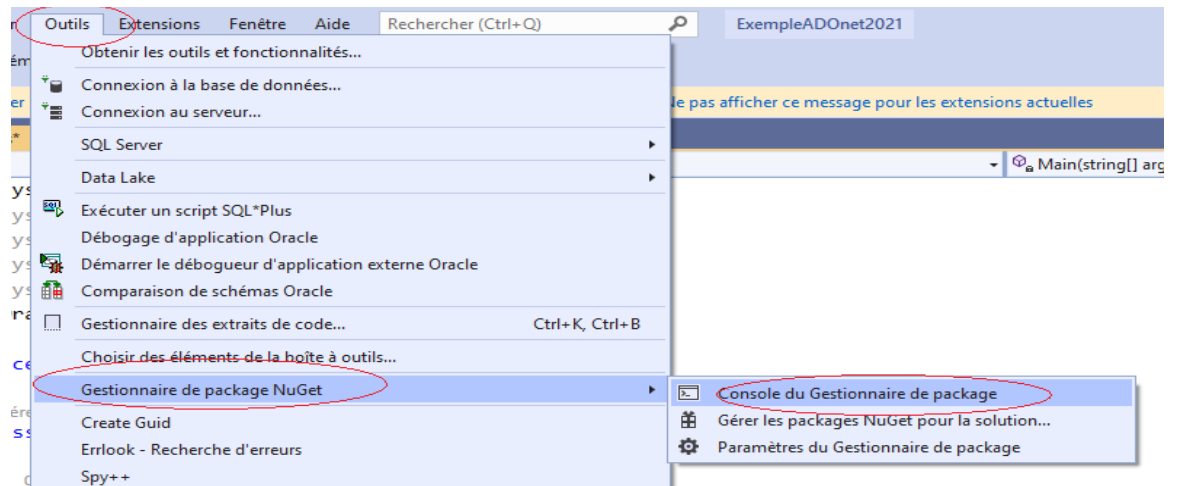


Oracle.ManagedDataAccess 19.11.0 ✓

ODP.NET, Managed Driver is a 100% native code .NET Framework driver for Oracle Database. No additional Oracle Client software is required to be installed to connect to Oracle Database.



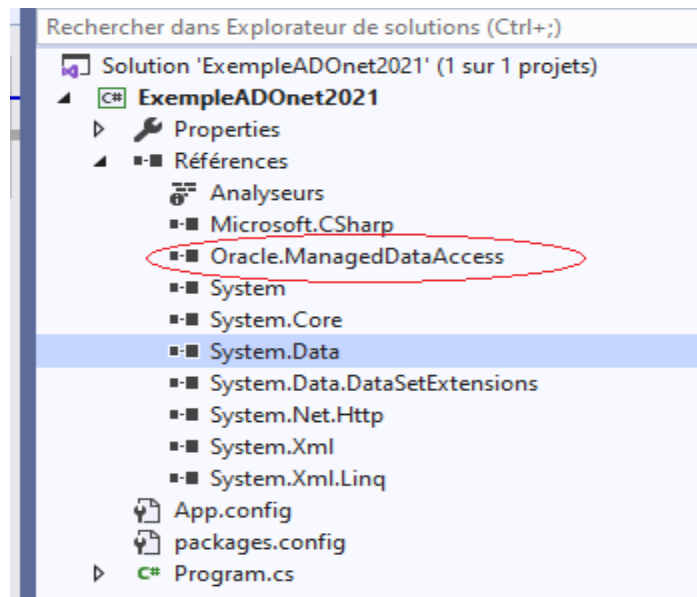
6. Dans votre projet Visual studio faites les opérations suivantes :
 - a. Allez a Outils
 - b. Puis Gestionnaire de Package Nuguet
 - c. Puis Console du gestionnaire de Packges (voir figure)



- d. à la console faites CTRL-V pour copier l'instruction **Install-Package** voir la figure suivante

```
PM> Install-Package Oracle.ManagedDataAccess -Version 19.11.0
Le package 'Oracle.ManagedDataAccess.19.11.0' existe déjà dans le projet 'ExempleADONet2021'
Temps écoulé : 00:00:00.0052905
PM> Install-Package Oracle.ManagedDataAccess -Version 19.11.0
```

7. Lorsque cette étape est complétée alors, vous allez voir dans les propriétés du projet qu'une référence a été ajoutée au projet (voir la figure suivante)



8. Pour votre projet, ajouter l'espace de nom :

Oracle.ManagedDataAccess.Client;

`using Oracle.ManagedDataAccess.Client;`

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using Oracle.ManagedDataAccess.Client;
7
8  namespace ExempleAdo
9  {
10     0 références
11     class Program
12     {
13         0 références
14         static void Main(string[] args)
15         {
16     }
```

Chapitre3, l'objet OracleConnection

Description de l'objet OracleConnection

Un objet OracleConnection représente une connexion à la base de données Oracle. Il a donc pour rôle d'établir une connexion à la base de données.

Les connexions sont utilisées pour « parler » aux bases de données et sont présentées par la classe OracleConnection.

On crée un objet OracleConnection avec la méthode **new**

```
OracleConnection conn = new OracleConnection();
```

Propriétés importantes de l'objet OracleConnection

Propriété ConnectionString

Il s'agit de la chaîne de connexion qui comprend des paramètres requis pour établir la connexion initiale. La valeur par défaut de cette propriété est une chaîne vide ("").

Les principaux paramètres pouvant être inclus dans la chaîne sont les suivants :

- Data source : le nom de serveur ou de la source de données ;
- User Id : Compte de connexion Oracle ;
- Password : Mot de passe de la session du compte Oracle

Le **Data Source** correspond à la description de la base de donnée, cette description est contenue dans le fichier tnames.ora (situé dans le C:\app\product\11.2.0\client_1\Network\Admin).

En général, la description du Data Source est de la forme suivante :

```
OraDb=
(DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)(HOST=ORASRV)(PORT1521))
  )
  (CONNECT_DATA=
    (SERVER=DEDICATED)
    (SERVICE_NAME=ORCL)
  )
)
```

Attention : , Votre Data Source est de la forme :

```
DSource =(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)
(HOST = 205.237.244.252)(PORT = 1521)))
(CONNECT_DATA =(SERVICE_NAME = orc1)))
```

Une chaîne de connexion est toujours un string et de la forme :

```
string chaîne = "Data Source = DSoucre; User Id = user; password = passe";
```

En d'autres mots, il faudra fournir au système l'ensemble des paramètres que vous fournissez lorsque vous utilisez SQL Developer

Exemple:

La chaîne de connexion doit inclure le Data Source, le nom du User son mot de passe.

```
string maBase = "(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)" +
"(HOST = 205.237.244.252)" + "(PORT = 1521)))" +
"(CONNECT_DATA =(SERVICE_NAME = orc1)))";

string ChaîneConnexion = "Data Source =" + maBase
+ ";User ID =user ;Password = pass";
```

Pour passer la chaîne de connexion, on utilise la propriété **ConnectionString** de l'objet OracleConnection.

```
conn.ConnectionString = ChaîneConnexion; (conn étant défini plus haut)
```

Propriété State.

Indique l'état actuel de la connexion. ODP a deux valeurs pour cette propriété. Closed ou Open. Par défaut la valeur est Closed

Méthodes importantes :

La méthode Open()

Permet d'ouvrir une connexion à la base de données

La méthode Close()

Permet de fermer une connexion ouverte à la base de données.

Exemple

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Oracle.ManagedDataAccess.Client;

namespace ExempleAdo
{
    class Program
    {
        static void Main(string[] args)
        {
            //on instancie un objet OracleConnection.
            OracleConnection conn = new OracleConnection();

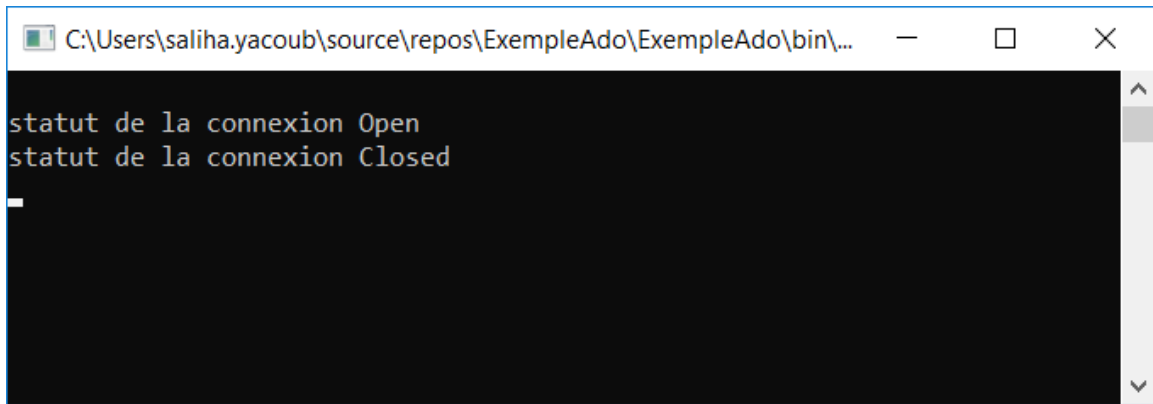
            //on declare le Data Source. Son nom est maBase
            string maBase = "(DESCRIPTION = (ADDRESS_LIST = " +
                "(ADDRESS = (PROTOCOL = TCP)" +
                "(HOST = 205.237.244.252)" + "(PORT =1521)))" +
                "(CONNECT_DATA =(SERVICE_NAME = orcl)))";

            //on declare la chaine de Connexion
            string ChaineConnexion = "Data Source =" + maBase
                + ";User ID =user ;Password = pass";

            try {
                //On fournit la chaine de connexion
                conn.ConnectionString = ChaineConnexion;
                //on ouvre une connexion
                conn.Open();
                Console.Read();
                //on affiche l'eta de la connexion
                Console.WriteLine("statut de la connexion" + " " + conn.State);
                Console.Read();
            }

            catch (Exception ex) {Console.WriteLine(ex.Message); };
            //on ferme la connexion
            conn.Close();
            //on affiche l'etat de la connexion
            Console.WriteLine("statut de la connexion" + " " + conn.State);
            Console.Read();
        }
    }
}
```


Ce qui donne ceci :




```
C:\Users\saliha.yacoub\source\repos\ExempleAdo\ExempleAdo\bin\...  
statut de la connexion Open  
statut de la connexion Closed  
-
```

Chapitre 4, l'objet OracleCommand

L'objet OracleCommand contient les commandes envoyées aux SGBD. Ces commandes sont envoyées soit en utilisant des requêtes simples, soit en utilisant des procédures stockées. Lorsque la requête SQL ou procédure retourne un résultat, il est retourné dans un OracleDataReader ou autre (voir **plus loin**).

L'objet OracleCommand se crée avec la méthode **new()**

Attention : 

Pour envoyer une requête à la base de donnée, on utilise un objet OracleCommand, en fournissant une connexion et une requête SQL

OracleCommand possède plusieurs constructeurs, celui qu'on va utiliser le constructeur suivant :

OracleCommand(String,OracleConnection)

Exemple : (conn étant le nom de l'objet OracleConnection)

```
string sql1 = "update employes set salaire = 10 where EMPNO = 11";  
OracleCommand oraCmd = new OracleCommand(sql1, conn);
```

Quelques propriétés de l'objet OracleCommand

CommandText	Obtient ou définit l'instruction SQL ou la procédure stockée à exécuter sur la base de données
CommandType	Obtient ou définit une valeur indiquant la manière dont la propriété CommandText doit être interprétée (instruction SQL ou procédure)
Connection	Obtient ou définit l'objet OracleConnection utilisé par cette instance de OracleCommand .
Parameters	Spécifie les paramètres de la requête SQL ou de la procédure stockée

Méthodes importantes de l'objet OracleCommand

ExecuteNonQuery()	Exécute une instruction SQL sur Connection et retourne le nombre de lignes affectées.
ExecuteReader()	Surchargé. Envoie CommandText à Connection et génère OracleDataReader
ExecuteScalar()	Exécute la requête et retourne la première colonne de la première ligne.

La méthode ExecuteNonQuery()

Cette méthode permet d'exécuter une requête DML (INSERT, UPDATE et DELETE). Cette méthode retourne un **int**, indiquant le nombre de lignes affectées par la requête.

Exemple

```
namespace ExempleAdo
{
    class Program
    {
        static void Main(string[] args)
        {
            //on instancie un objet OracleConnection.
            OracleConnection conn = new OracleConnection();
            //on declare le Data Source
            string maBase = "(DESCRIPTION = (ADDRESS_LIST = " +
                "(ADDRESS = (PROTOCOL = TCP)" +
                "(HOST = 205.237.244.252)" + "(PORT = 1521)))" +
                "(CONNECT_DATA =(SERVICE_NAME = orcl)))";
            //on declare la chaine de Connexion
            string ChaineConnexion = "Data Source =" + maBase
                + ";User ID =user ;Password = pass";

            try {
                //On fournit la chaine de connexion
                conn.ConnectionString = ChaineConnexion;
                //on ouvre une connexion
                conn.Open();
                Console.Read();
                //on affiche l'état de la connexion
                Console.WriteLine("statut de la connexion" + " " + conn.State);
                Console.Read();
                //on déclare une requete sql. C'est un string
                string sql1 = "update employes set salaire = 100 where EMPNO =10";

                //on crée l'objet OracleCommand avec la command SQL et la connection
                OracleCommand oraCmd = new OracleCommand(sql1, conn);

                //On utilise ExecuteNonQuery() pour executer la requête.
                int n= oraCmd.ExecuteNonQuery();

                Console.WriteLine("nombre de MAJ" + " " + n);
            }
        }
    }
}
```

```
    }  
    catch (Exception ex) {Console.WriteLine(ex.Message); }  
    //on ferme la connexion  
    conn.Close();  
    //on affiche l'état de la connexion  
    Console.WriteLine("statut de la connexion" + " " + conn.State);  
    Console.Read();  
  }  
}  
}
```

La méthode **ExecuteReader()**

Cette méthode permet d'exécuter une requête SELECT et retourne un **OracleDataReader**, contenant le résultat de la requête.


La méthode **ExecuteScalar()**

Exécute la requête et retourne la première colonne de la première ligne. Elle est utilisée pour des requêtes comme SELECT COUNT, SELECT MIN etc...

Chapitre 5, l'objet OracleDataReader

Les objets **DataReader** servent à extraire d'une base de données un flux de données en lecture seule et dont le défilement se fera par en avant uniquement (read-only, forward-only,). Les résultats sont retournés pendant que la requête s'exécute et stockés dans la mémoire tampon de réseau sur le client jusqu'à ce que vous les demandiez au moyen de la méthode Read de **DataReader**.

L'objet OracleDataReader se crée par la méthode **ExecuteReader** de l'objet **OracleCommand**

Attention : 

L'objet OracleDataReader NE SE Crée pas avec la méthode new

Quelques propriétés importantes de l'OracleDataReader

FieldCount	Obtient le nombre de colonnes figurant dans la ligne en cours.
HasRows	Obtient une valeur indiquant si OracleDataReader contient une ou plusieurs lignes.
IsClosed	Indique si OracleDataReader est fermé.

Quelques méthodes importantes de OracleDataReader

Close	Ferme l'objet OracleDataReader
Dispose	Libère toutes les ressources occupées par l'objet OracleDataReader
Read	Permet d'avancer l'objet OracleDataReader jusqu'à l'enregistrement suivant.
GetDateTime	Obtient la valeur de la colonne spécifiée sous la forme d'un objet DateTime.
GetDecimal	Obtient la valeur de la colonne spécifiée sous la forme d'un objet Decimal.
GetDouble	Obtient la valeur de la colonne spécifiée sous la forme d'un nombre de type Double.
GetFloat	Obtient la valeur de la colonne spécifiée sous la forme d'un nombre de type Float.
GetInt16	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 16 bits.
GetInt32	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 32 bits.

GetInt64	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 64 bits.
GetLifetimeService	Extrait l'objet de service de durée de vie en cours qui contrôle la stratégie de durée de vie de cette instance.
GetName	Obtient le nom de la colonne spécifiée.
GetString	Obtient la valeur de la colonne spécifiée sous la forme d'une chaîne.

Consulter http://docs.oracle.com/html/B28089_01/OracleDataReaderClass.htm#i1004157 pour le reste des méthodes.

Par défaut, un **DataReader** charge une ligne entière en mémoire à chaque appel de la méthode **Read()**. Il est possible d'accéder aux valeurs de colonnes soit par leurs noms soit par leurs références ordinales. Une solution plus performante est proposée permettant d'accéder aux valeurs dans leurs types de données natifs (**GetInt64**, **GetDouble**, **GetString**). Par exemple si la première colonne de la ligne indiquée par 0 est de type **int**, alors il est possible de la récupérer à l'aide de la méthode **GetInt64** de l'objet **DataReader**.

Exemple (on est déjà connecté)

```
string sql2 = "select nom, prenom,salaire from employes";
OracleCommand oraSql2 = new OracleCommand(sql2, conn);
OracleDataReader oraRead = oraSql2.ExecuteReader();
while (oraRead.Read())
{
    Console.WriteLine("{0} - {1} - {2}",
        oraRead.GetString(0), oraRead.GetString(1), oraRead.GetDecimal(2));
}
oraRead.Close();
```

```

namespace ExempleAdo
{
    class Program
    {
        static void Main(string[] args)
        {
            //on instancie un objet OracleConnection.
            OracleConnection conn = new OracleConnection();
            //on declare le Data Source
            string maBase = "(DESCRIPTION = (ADDRESS_LIST = " +
                "(ADDRESS = (PROTOCOL = TCP)" +
                "(HOST = 205.237.244.252)" + "(PORT = 1521)))" +
                "(CONNECT_DATA =(SERVICE_NAME = orcl))";
            //on declare la chaine de Connexion
            string ChaineConnexion = "Data Source =" + maBase
                + ";User ID =user1 ;Password = user1";

            try {
                //On fournit la chaine de connexion
                conn.ConnectionString = ChaineConnexion;
                //on ouvre une connexion
                conn.Open();
                Console.Read();
                //on affiche l'eta de la connexion
                Console.WriteLine("statut de la connexion" + " " + conn.State);

                //on déclare une requête SELECT
                string sql2 = "select nom, prenom,salaire from employes";

                //on cree l'objet OracleCommand avec la command SQL et la connection
                OracleCommand oraSql2 = new OracleCommand(sql2, conn);

                //On execute la requete sql2. On obtient un OracleDataReader
                OracleDataReader oraRead = oraSql2.ExecuteReader();

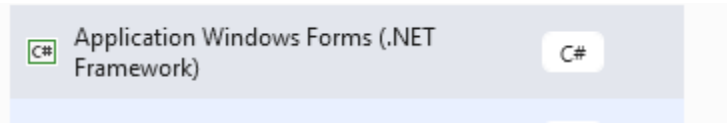
                //On lit le OracleDataReader.
                while (oraRead.Read())
                {
                    Console.WriteLine("{0} - {1} - {2}",
                        oraRead.GetString(0), oraRead.GetString(1), oraRead.GetDecimal(2));
                }
                oraRead.Close();

            }
            catch (Exception ex) {Console.WriteLine(ex.Message); };
            //on ferme la connexion
            conn.Close();
            //on affiche l'état de la connexion
            Console.WriteLine("statut de la connexion" + " " + conn.State);
            Console.Read();
        }
    }
}

```

Chapitre 6, exemple avec Windows Form

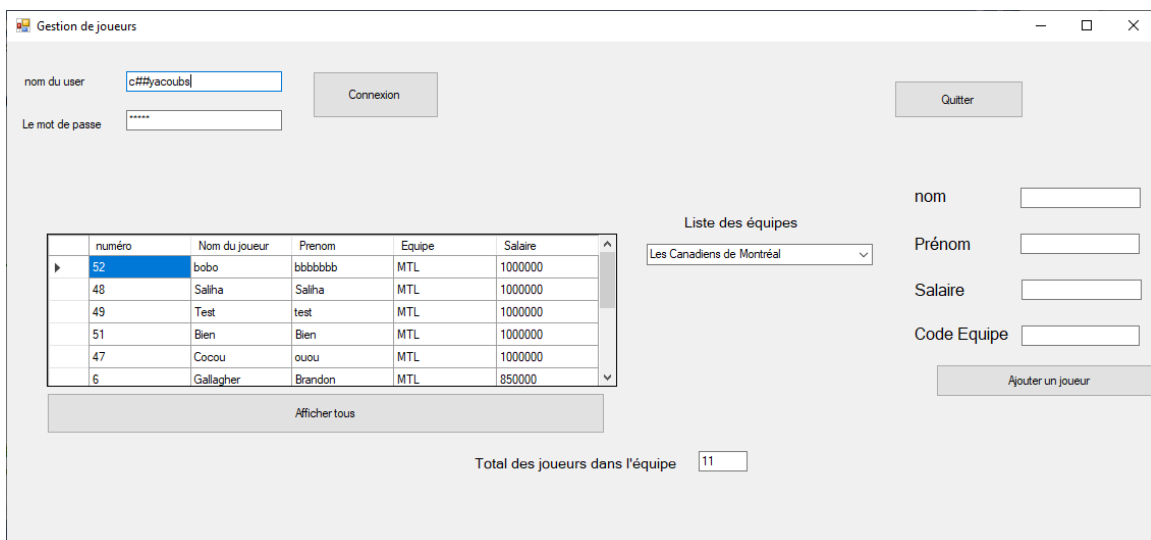
1. Avant tout, vous démarrez un projet : Application Windows Forms(.Net Framework)



2. Puis vous ajoutez les packages nécessaires (comme pour une application console—page 6).

Présentation

Voici la forme que nous souhaitons avoir.



numéro	Nom du joueur	Prenom	Equipe	Salaire
52	bobo	bbbbbbb	MTL	1000000
48	Salha	Salha	MTL	1000000
49	Test	test	MTL	1000000
51	Bien	Bien	MTL	1000000
47	Cocou	ouou	MTL	1000000
6	Gallagher	Brandon	MTL	850000

Sur la forme, il ya les objets

- Un bouton pour se connecter
- Un bouton pour se déconnecter
- Un bouton pour afficher tous les joueurs
- Un bouton pour ajouter un joueur.

Chaque bouton appelle la fonction qui lui correspond.

Il ya aussi

- deux zones de texte pour saisir le nom du user et le mot de passe pour à la base de données et
- 4 zones de texte pour rentrer les données du joueur.

Également

- Un Combobox pour contenir la liste des noms des équipes (le nom de l'équipe). Les comboBox peuvent contenir des ITEMS, les **Items** vont correspondre à une colonne de la BD
- Un DataGridView pour contenir la liste de joueurs. Un DataGridView est un objet qui possède des lignes **Rows**. et des colonnes. Même structure qu'une table.

Tous ces objets sont dans une Form Gestion des joueurs.

Construction de l'interface. (Attention !! cette méthode n'est pas la meilleure façon de faire)

Une fois que votre projet est démarré, vous trouverez les contrôles dans **la boîte à outils**. Voici un aperçu. Il suffit de glisser le contrôle et de le mettre sur la Form

ImageKey	[] (aucun)
ImageList	(aucun)
RightToLeft	No
Text	button1
TextAlign	MiddleCenter
TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackColor	True
UseWaitCursor	False
Comportement	
Design	
(Name)	button1
GenerateMember	True

Chaque contrôle a ses propriétés. Quand le contrôle est sélectionné, vous verrez ses propriétés à droite. La propriété la plus importante et le NOM (name), car cette propriété va représenter un nom de variable. Exemple pour le bouton, La propriété Text va représenter ce qui va être afficher sur le bouton. Le name va être le nom bouton dans le code C#

Même principe pour les zones de texte. Le nom est très important

	NumericUpDown
	PictureBox
	ProgressBar
	RadioButton
	RichTextBox
	TextBox
	ToolTip
	TreeView

Comportement	
Design	
(Name)	textNom
GenerateMember	True
Locked	False
Modifiers	Private
Disposition	

Le DataGridView

Une fois que votre DataGridView (DGV) est placé sur la form

- 1- NE CHOISIR aucune source de données.
- 2- Donner lui un nom significatif
- 3- Puis cliquer sur ajouter une colonne.

Le nom est le nom de la variable elle-même, texte de l'entête est ce qui va s'afficher sur le DGV

Ajouter une colonne ? X

Colonne liée aux données

Colonnes du DataSource

Colonne indépendante

Nom :

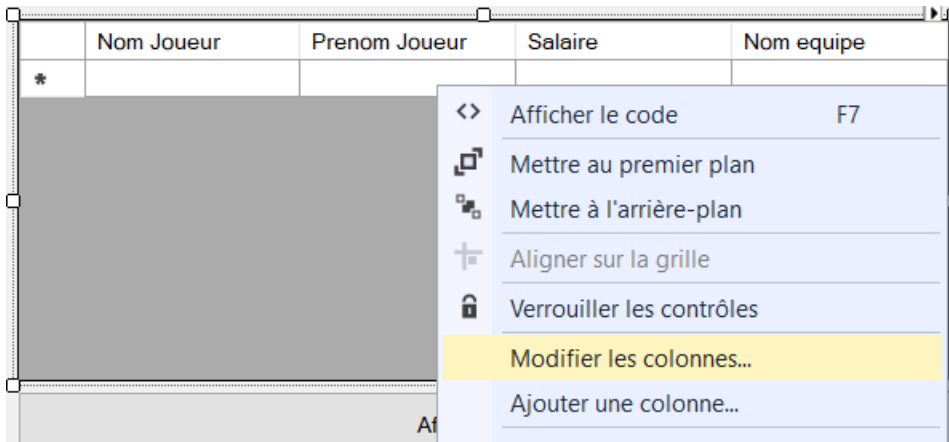
Type :

Texte de l'en-tête :

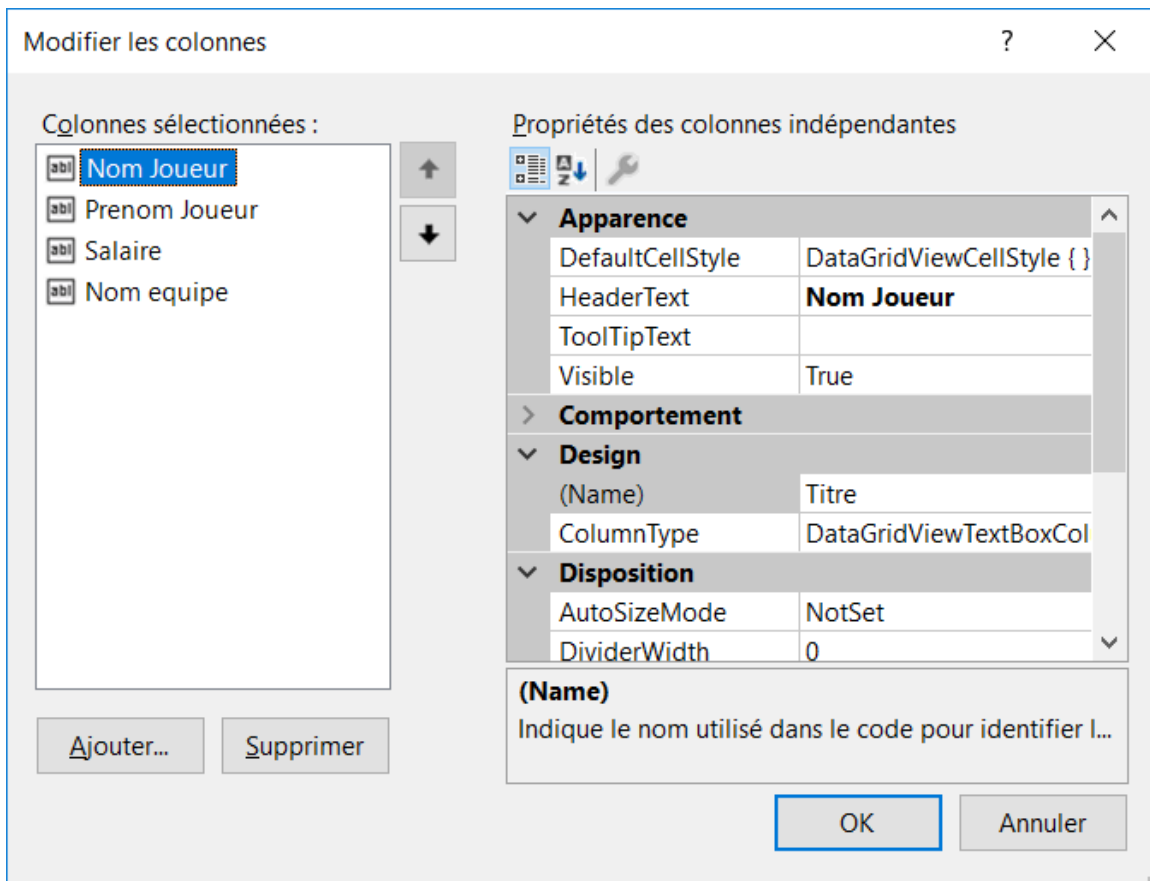
Visible Lecture seule Figé

Ajouter toutes les colonnes qui vont correspondre au résultat de votre requête

Note : Quand le DGV est sélectionné, par le bouton droit de la SOURIS, vous pouvez le gérer comme vous voulez.




Si vous avez choisi : Modifier une colonne, vous allez avoir ceci et donc modifier vos colonnes.



Le code C#

Comme nous l'avons mentionné plus chaque bouton va appeler la fonction qui lui correspond.

Pour accéder au code du bouton, il suffit de double cliquer dessus.

Attention : 

Votre objet OracleConnection doit être global, puisque TOUS les objets OracleCommand vont l'utiliser.

```
namespace Kb6Tpno1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```
OracleConnection oraconn = new OracleConnection();
```

```
string maBase = "(DESCRIPTION = (ADDRESS_LIST = " +
    "(ADDRESS = (PROTOCOL = TCP)" +
    "(HOST = 205.237.244.252)" + "(PORT = 1521)))" +
    "(CONNECT_DATA =(SERVICE_NAME = orcl)))";
```

```
-----Suite du code
}
}
```

Le bouton btnConnecter

```
private void btnConnecter_Click(object sender, EventArgs e)
{
    connecter();

    listerEquipe();
}
```

Voir les explications de listerEquipe() plus bas.

Démarche pour la fonction **connecter()**

- 1- `string user = textUser.Text`, on récupère le nom du User par le TextBox de nom TextUser. Et on fait la même chose pour le mot de passe.
- 2- On construit la chaîne de connexion
- 3- On fixe la propriété `ConnectionString` de l'objet `oraconn` déclaré plus haut
- 4- On ouvre une connexion avec `Open()`;
- 5- Le code est sur la page suivante.

La fonction **connecter()** a le code suivant :

```

private void connecter()
{
    string user = textUser.Text;
    string pass = textPasseword.Text;
    string ChaineConnexion = "Data Source =" + maBase + ";user id="
        + user + ";Password =" + pass;

    try
    {
        oraconn.ConnectionString = ChaineConnexion;
        oraconn.Open();
        MessageBox.Show(oraconn.State.ToString());
    }
    catch (Exception exconn)
    {
        MessageBox.Show(exconn.Message.ToString());
    }
}

```

Toute connexion ouverte doit être fermée. (bouton **Deconnecter**)

```

private void btnDeconnectin_Click(object sender, EventArgs e)
{
    deconncter();
}

```

La fonction deconnecter()

```

private void deconncter()
{
    oraconn.Close();
    MessageBox.Show(oraconn.State.ToString());
    Application.Exit();
}

```

Après la connexion, on pourra alors consulter liste des joueurs en utilisant le bouton

Afficher Tous

Voici le code du bouton **Afficher Tous**

```

private void btnAfficher_Click(object sender, EventArgs e)
{
    affichertous();
}

```

La fonction **affichertous()**

```
private void affichertous()
{
    dgvListedisques.Rows.Clear();
    try {
string sql1 = "select nom, prenom,salaire, nomequipe  from joueurs inner join
equipes on joueurs.CODEEQUIPE = equipes.CODEEQUIPE";

        OracleCommand oracmd1 = new OracleCommand(sql1, oraconn);
        OracleDataReader oraRead = oracmd1.ExecuteReader();

        while (oraRead.Read())
        {
            dgvListeJoueurs.Rows.Add(
                oraRead.GetString(0),
                oraRead.GetString(1),
                oraRead.GetDouble(2),
                oraRead.GetString(3)
            );
        }
        oraRead.Close();
    }
    catch (Exception exsql1)
    {
        MessageBox.Show(exsql1.Message.ToString());
    }
}
```

Explications, la fonction **affichertous()**

dgvListedisques.Rows.Clear(); permet de vider le DGV, sinon il va se remplir encore et encore à chaque clique du bouton

string sql1 = "select nom, " on construit la requêtes SQL. On construit la requête dans le try pour pouvoir vérifier sa validité.

OracleCommand oracmd1 = new OracleCommand(sql1, oraconn);On crée un Obejt OracleCommand en lui passant la requete sql1 et la connexion oraconn.

OracleDataReader oraRead = oracmd1.ExecuteReader(); Comme c'Est une requête SELECT, on utilise la méthode ExecuteReader() de l'objet OracleCommnad. Ce qui permet d'obtenir un objet OracleDataReader.

while (oraRead.Read()); On lit l'objet OracleDataReader ligne par ligne en utilisant la méthode Read();

oraRead.GetString(0), On utilise la méthode getTypedeDonnee(Indicedecolonne) pour lire le contenu d'une colonne.

```

while (oraRead.Read())
{
    dgvListedisques.Rows.Add(oraRead.GetString(0),
    oraRead.GetString(1),oraRead.GetDouble(2),oraRead.GetString(3));
}

```

La boucle while qui permet de lire un DataReader ligne par ligne, va permettre de remplir le DGV ligne par ligne aussi. On utilise la propriété **Rows** et la méthode **Add()** pour ajouter une ligne à chaque fois.

oraRead.Close(); Il faut fermer l'objet OracleDataReader avec sa méthode Close();

La fonction listeEquipe(), pour afficher la liste des équipes dans un ComboBox ou dans une liste (pas un DGV)

La fonction qui correspond à afficher la liste des équipes dans un ComboBox est la suivante :

```

private void listerEquipe()
{
    string sqlequipe = "Select nomequipe from equipes";
    OracleCommand oracmd2 = new OracleCommand(sqlequipe, oraconn);
    OracleDataReader oraRedEquipe = oracmd2.ExecuteReader();
    while (oraRedEquipe.Read())
    {
        comEquipes.Items.Add(oraRedEquipe.GetString(0));
    }
    comEquipes.SelectedIndex = 0;
    oraRedEquipe.Close();
}

```

Explications de la fonction listerEquipe()

On liste les équipes et on les affiche dans un ComboBox. Le nom du ComboBox est comEquipes


Vous l'avez compris. Tout le code avant le while s'explique de la même façon que pour la fonction affichertous());

Vous avez compris aussi que la requête SELECT dans ce cas renvoi une seule colonne. Dans ce cas on va utiliser un ComboBox pour contenir le résultat de la requête.

Pour le ComboBox on va utiliser la propriété **Items** et la méthode **Add()**

comEquipes.Items.Add(oraRedEquipe.GetString(0)); Va permettre d'ajouter un Items(une colonne) à notre ComboBox. Évidemment cette instruction est dans un while puisqu'il faut lire tout le OracleDataReader oraReadEquipe.

`comEquipes.SelectedIndex = 0;` va positionner l'index de la ComboBox à zéro, donc va pouvoir afficher le premier Items (ou la première ligne de la requête).

Attention : 

La fonction `listerEquipe()` doit être appelée. Ici nous avons décidé de l'appeler juste après la connexion.

Et si on veut afficher la liste des joueurs d'une équipe donnée ?

Le nom de l'équipe est déjà dans le comboBox, il suffit de faire une requête avec un WHERE sur le contenu du ComboBox.

Voici la fonction qui liste les joueurs selon leur équipe.

Fonction `listeJoueursEquipe()`

```
private void listeJoueursEquipe()
{
    dgvListeJoueurs.Rows.Clear();
    string nomEquip = comEquipes.Text;
    try
    {
        string sql2 = "select nom, prenom, salaire, nomequipe from
joueurs inner join equipes on joueurs.CODEEQUIPE = equipes.CODEEQUIPE where
nomEquip ='" + nomEquip + "'";

        OracleCommand oracmd2 = new OracleCommand(sql2, oraconn);
        OracleDataReader oraReadjoueur = oracmd2.ExecuteReader();

        while (oraReadjoueur.Read())
        {
            dgvListeJoueurs.Rows.Add(
                oraReadjoueur.GetString(0),
                oraReadjoueur.GetString(1),
                oraReadjoueur.GetDouble(2),
                oraReadjoueur.GetString(3));
        }
        oraReadjoueur.Close();
    }
    catch (Exception exsql2)
    {
        MessageBox.Show(exsql2.Message.ToString());
    }
}
```

Explications de la fonction `listejoueursEquipe()`

On utilise une requête avec jointure, puisque la clause WHERE porte sur le nom de l'équipe.

- La requête est `SELECT nom, prenom, salaire` Cette requête renvoi plusieurs lignes et plusieurs colonnes. Donc pour afficher le résultat on utilise un `Un DataGridView`. (voir plus haut).
- La requête présente une clause WHERE sur le nom de l'équipe. Or le nom de l'équipe est dans le `comboBox` de nom `comEquipes`. Il suffit de le récupérer comme suit :
`string nomEquip = comEquipes.Text;`

La requête est donc construite ainsi:

```
string sql2 = "select nom, prenom, salaire, nomequipe from joueurs inner join  
equipes on joueurs.CODEEQUIPE = equipes.CODEEQUIPE where nomEquip ='" + nomEquip  
+ "'";
```

Une fois la requête construite, il suffit d'utiliser un `OracleCommand` pour l'envoyer au SGBD et appliquer la méthode `ExecuteReader()` pour obtenir le résultat dans un `OracleDataReader`.

On lit le `OracleDataReader` avec la méthode `Read()` et une boucle `while`. Il suffit d'envoyer le résultat de la lecture dans le `DataGridView`. On ajoute une ligne à la fois avec `Rows.Add(..)`


```
while (oraReadjoueur.Read())  
{  
    dgvListeJoueurs.Rows.Add(oraReadjoueur.GetString(0),  
    oraReadjoueur.GetString(1), oraReadjoueur.GetDouble(2),  
    oraReadjoueur.GetString(3));  
}
```

Cette fonction sera appelée lorsqu'on choisit (on change) le nom de l'équipe dans le `comboBox`. Le `comboBox` a un évènement `SelectedIndexChanged` (lorsque l'index est changé)

Code du `comboBox` :

```
private void comEquipes_SelectedIndexChanged(object sender, EventArgs e)  
{  
  
    listejoueursEquipe();  
    compter();  
  
}
```

`dgvListeJoueurs.Rows.Clear();` permet de vider le `DataGridView`. Sinon il va se remplir tout le temps.


Attention : 

Il faut effacer le contenu de votre DataGridView si vous ne voulez pas qu'il se remplisse à l'infini

Insertion des données dans la base de données.

L'insertion des données va se faire par les zones de texte texNom, textPrenom, textSalaire et textCodeEquipe. Donc la première chose à faire est de récupérer leur contenu, et les mettre dans des variables :

```
string nomj = texNom.Text;
string prnj = textPrenom.Text;
string salj = textSalaire.Text;
string codeEquip = textCodeEquipe.Text;
```

Attention : 

- 1- Le numéro du joueur est inséré automatiquement par une séquence. Ça évite la duplication de la PK(primary key)
- 2- Remarquez que les simples guillemets lorsque la colonne concernée par la saisie est de type VARCHAR2 (donc un string)

Il suffit de construire la requête INSERT INTO ..Ici, le numéro du joueur est IDENTITY BY DEFAULT

```
public void AjouterJoueur()
{
    string nom = textNom.Text;
    string prenom = textPrenom.Text;
    string salaire = textSalaire.Text;
    string codeEquipe = textCodeEquipe.Text;
    try
    {
        string sqlTxt = $"insert into joueurs" +
            $" (nom, prenom, salaire, codeEquipe)" +
            $" values ('{nom}', " +
            $" '{prenom}', " +
            $" {salaire}, " +
            $" '{codeEquipe}');"

        OracleCommand oraCmd = new OracleCommand(sqlTxt, conn);
        int n = oraCmd.ExecuteNonQuery();

        MessageBox.Show($"{n} Joueur ajouté");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString());
    }
}
```

Une fois que la requête est construite, on utilise un OracleCommand avec sa méthode ExecuteNonQuery() pour envoyer et exécuter la requête. Rien de plus simple.

Comme, les zones de textes seront utilisées pour effectuer plusieurs insertions, il est conseillé de les vider pour la prochaine insertion.

```
private void vider()
{
    texNom.Clear();
    textPrenom.Clear();
    textCodeEquipe.Clear();
    textSalaire.Clear();
}
```

La fonction est appelée par le bouton btnInsérer comme suit.

```
private void btnInsérer_Click(object sender, EventArgs e)
{
    AjouterJoueur()
}
```

Exemple de fonction, qui compte les joueurs dans chaque équipe.

Le résultat sera affiché dans une zone de texte appelée : textTotal.

Cette fonction est appelée juste après listeJoueursEquipe();

On utilise la méthode ExecuteScalar() de l'objet OracleCommand.

Dans la variable equipe, on récupère le nom de l'équipe qui est dans le comboBox. Cette variable sera utilisée dans le WHERE.

```
private void compter()
{
    try
    {
        string equipe = comboEquipes.Text;
        string sql5 = "select count(*) from equipes inner join joueurs on " +
            "equipes.codeEquipe = joueurs.codeequipe " +
            "where nomequipe = '" + equipe + "'";
        OracleCommand oraCmd5 = new OracleCommand(sql5, conn);
        textTotal.Text = oraCmd5.ExecuteScalar().ToString();
    }
    catch (Exception sql5)
    {
        MessageBox.Show(sql5.Message.ToString());
    }
}
```

Sources

<https://docs.microsoft.com/fr-fr/dotnet/framework/data/adonet/>

<https://docs.oracle.com/en/database/oracle/oracle-data-access-components/19.3/>