Introduction aux bases de donnés

ADO.NET, introduction

Introduction a ADO.NET

- Introduction
- Pres-requis : installation des composants
- L'objet OracleConenction
  - Propriétés
  - méthodes
- L'objet OracleCommand
  - Propriétés
  - méthodes
- L'objet OracleDataReader
  - Propriétés
  - méthodes

Introduction

- Il est rare que l'exploitation des bases de données se fasse directement par l'interface client des SGBDs (Sqldeveloper, SSMS, MySQL Workbench etc…).
- Les développeurs conçoivent des applications afin de rendre l'accès au données facile et convivial.
- Il existe au moins deux solutions pour d'accéder aux données d'une bases de données :

1 - Une solution offerte par les SGBDs:

les objets permettant l'accès aux données est du côté du SGBD (Exemple: Mysqli, Oracle Call Interface). Dans ce cas les programme C# qui va accéder à un BD SQL Server et à une BD Oracle sont très différents. Les solutions ne sont pas portables d'un SGBD à l'autre. Par contre les accès aux données sont très rapides.

Introduction

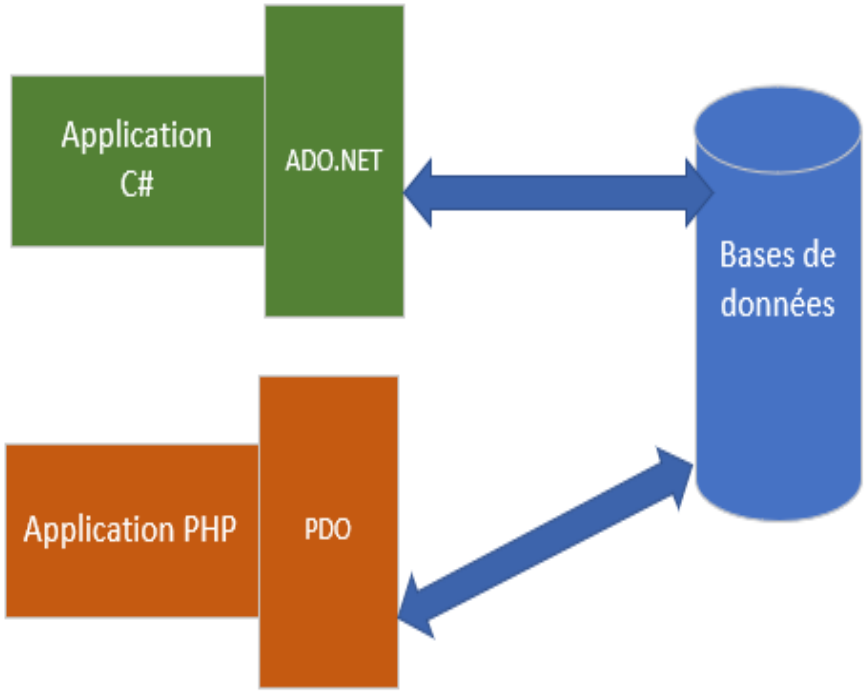
2- Une solution offerte par les langages de programmation. (ADO.NET,PDO,JDBC etc).

Dans ce cas , la solution est portable d'un SGBD à un autre. L'accès par C# à une BD Oracle ou une BD SQL Server est presque le même.

La figure ci-après montre les accès côté application à une BD.

Une application C#, par ADO.NET

Une application PHP par PDO.



ADO.NET, définition

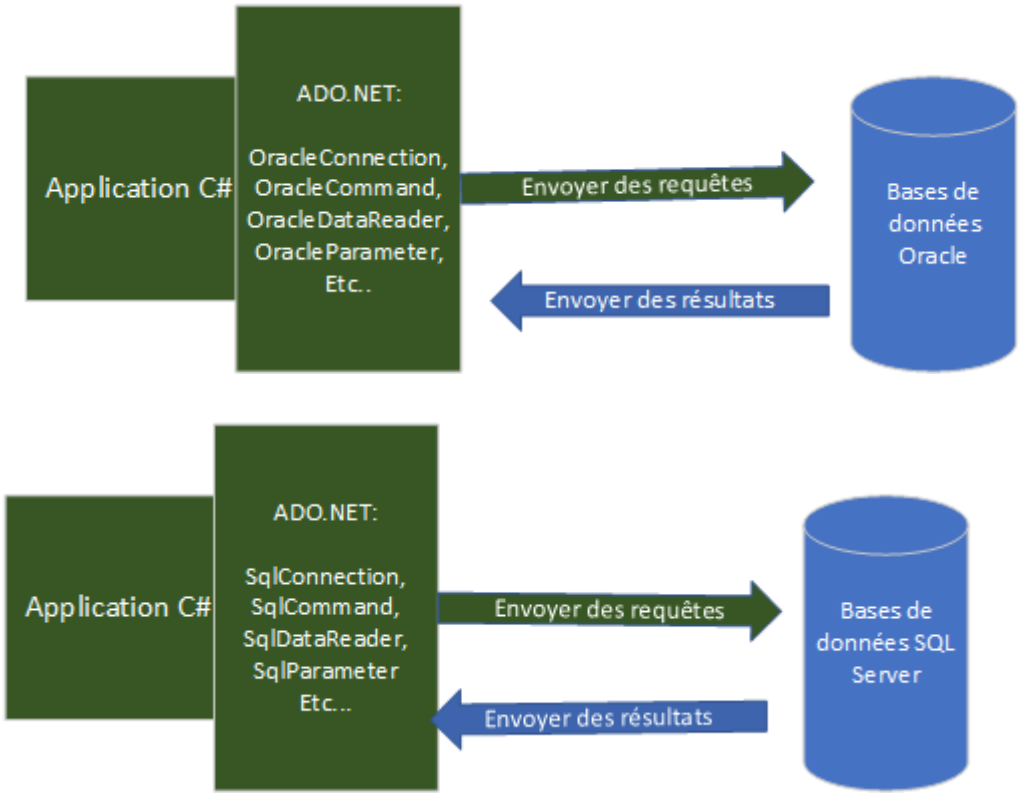
ADO.NET est un ensemble de classes qui exposent des services standardisés d'accès aux données.

Ces classes permettent donc aux programmeurs de concevoir des applications permettant de se connecter à des sources de données variées et, d'extraire, de manipuler et de mettre à jour ces données.

Dans une application ADO.NET, pour accéder à Oracle ,ou SQL server ou MySQL seul le nom des classes change. Les propriétés et les méthodes restent les même

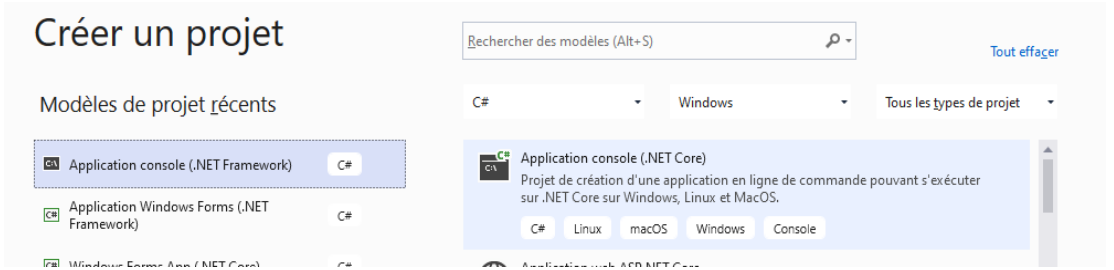
Pour utiliser Oracle avec C#, vous avez besoin du composant ODT (Oracle Developer Tools) for Visual Studio

ODT contient les classes qui permettent d'accéder et d'exploiter une base de données Oracle par Visual Studio.



Installation des composants

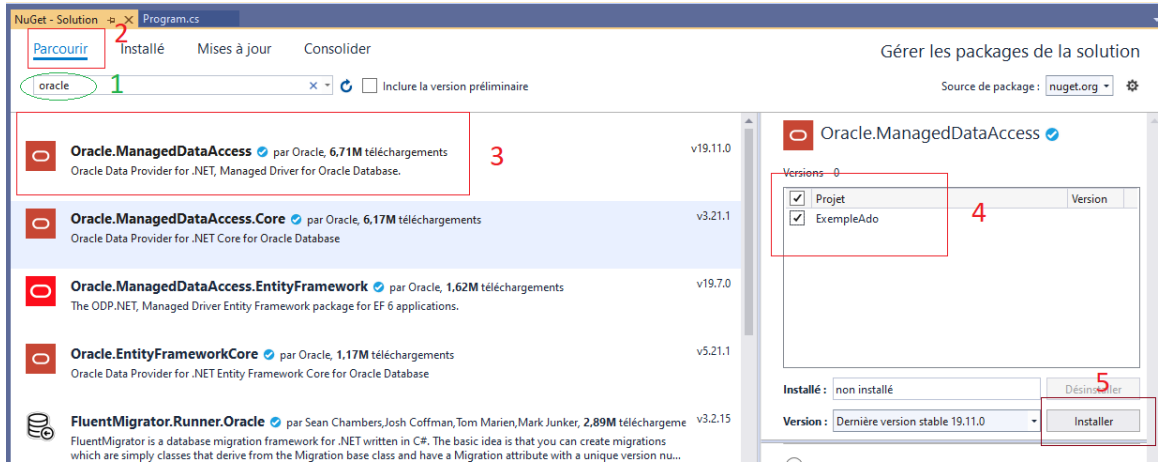
On commence par créer une application en mode console



Dans le menu Outils aller à Gestionnaire de package NuGet, puis Gérer les packages NuGet pour la solution

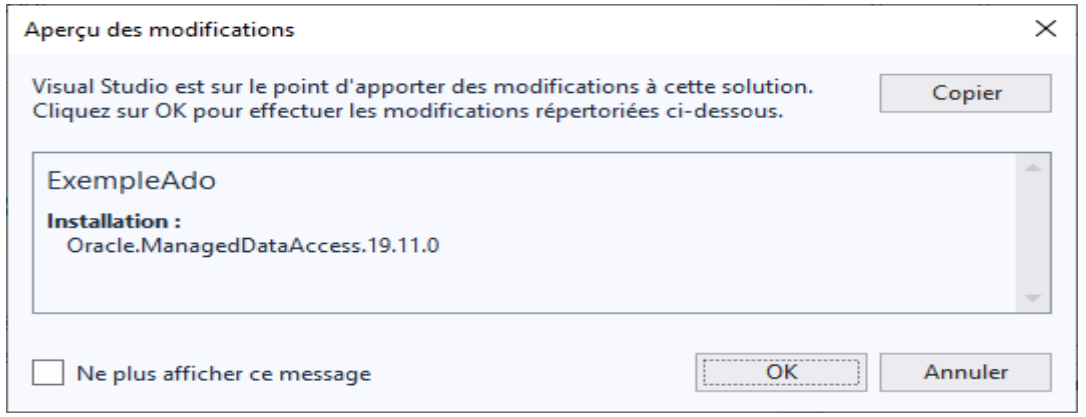
Puis la fenêtre ci-après s'ouvre :

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

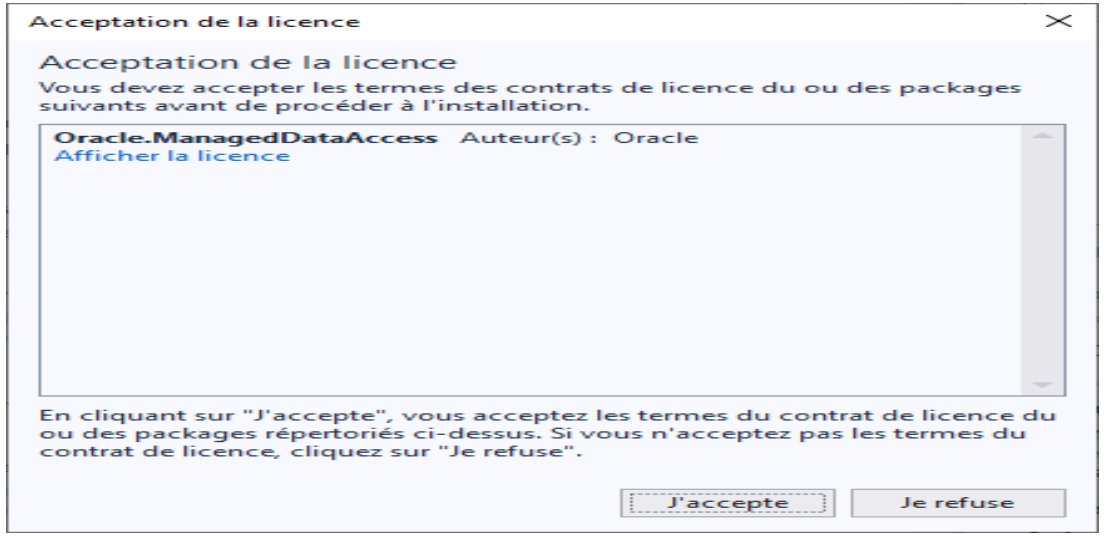


Installation des composants

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

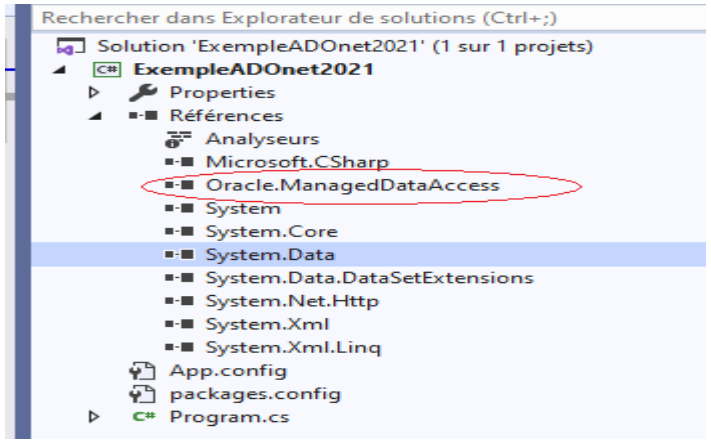


La figure ci-après apparait: Accepter la licence.



Installation des composants

- Lorsque l'installation est terminée vous allez revenir à votre projet, et remarquez que vous avec une référence: **Oracle.ManagedDataAccess**.



Dans votre projet, ajoutez 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   {
        0 références
10      class Program
11      {
            0 références
12          static void Main(string[] args)
13          {
14          }
15      }
16  }
```

Vous êtes prêts pour commencer à coder

La classe OracleConnection

La classe OracleConnection: Définition

- 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()**

```
private OracleConnection oraConn  = new OracleConnection();
```

Pour se connecter à la base de données, nous avons besoin:

- De fournir les information de connexion (nom du serveur, nom du user, le mot de passe). Ces informations de connexion **formeront une chaine de connexion**.
- Ouvrir une connexion.

La classe OracleConnection

Propriétés importantes: ConnectionString et State.

ConnectionString

Il s'agit de la **chaîne de connexion (string)**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 : (séparés par un point virgule)

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

La chaine de connexion est de la forme:

```
string ChaineConnexion = "Data Source = DSource; User Id = user; password = passe";
```

La classe OracleConnection

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

- Pour Oracle, le Data Source doit contenir les informations qui vous permettent d'accéder à votre base de données Oracle. En d'autres mots doit contenir les informations que vous saisissez dans votre SQL Developer:
  - Le nom du serveur ou son adresse IP
  - Le numéro du port
  - Le SID (System ID :nom global de la base de données). En général le SID et identique au Service_Name)

  Exemple:

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

La classe OracleConnection

- Exemple

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

string ChaineConnexion = $"Data Source = {DSource}; " +  $"User ID ={userName} ;Password = {password}";
```

La classe OracleConnection

À ce stade nous avons:

- un objet OracleConnection: oraConn
- une chaine de connexion: ChaineConnexion.

La propriété ConnectionString de l'objet OracleConnection va permettre définir la chaine de connexion qui sera utilisée pour ouvrir une connexion

```
oraConn.ConnectionString = ChaineConnexion;
```

La propriété State:

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

```
oraConn.State ;
```

La classe OracleConnection

Méthodes importantes: Open() et Close()

- La méthode Open() permet d'ouvrir une connection

```
oraConn.Open()
```

- La méthode Close() permet de fermer une connexion

```
oraConn.Close()
```

Toute connexion ouverte doit être fermée. C'est une obligation

La classe OracleConnection

```
OracleConnection oraConn = new OracleConnection();
        string userName = "c##user1";
        string password = "user1";
        string DSource = "(DESCRIPTION = (ADDRESS_LIST = " +
           "(ADDRESS = (PROTOCOL = TCP)" +
           "(HOST = 205.237.244.252)" + "(PORT = 1521)))" +
           "(CONNECT_DATA =(SERVICE_NAME = orcl)))";

        string ChaineConnexion = $"Data Source = {DSource}; " +  $"User ID ={userName} ;Password = {password}";
           try
                {
                 oraConn.ConnectionString = ChaineConnexion;
                oraConn.Open();
                Console.WriteLine("statut de la connexion" + " " + oraConn.State);
                 }
       catch (Exception ex) { Console.WriteLine(ex.Message); };
       oraConn.Close();
       Console.WriteLine("statut de la connexion" + " " + oraConn.State);
```

La classe OracleCommand

La classe OracleCommand, définition

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 (un Data Set**).**

OracleCommand possède plusieurs constructeurs, celui que nous allons utiliser est le constructeur suivant :

```
OracleCommand(string SQL , OracleConnection oraConn)
```

Exemple:

```
OracleCommand oraCmd = new OracleCommand(sql, oraConn);
```

Un objet OracleCommand se crée avec la méthode **new();**

sql est une requête SQL

oraConn est un objet OracleConnection

La classe OracleCommand

La classe OracleCommand, Propriétés importantes

Propriétés	Signification
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 stockée).
Parameters	Spécifie les paramètres de la requête SQL ou de la procédure stockée

Si CommandText et CommadType ne sont pas précisés alors ils sont par défaut. Donc nous avons à faire à une instruction SQL dans le code et non une procédure stockée (ce qui est notre cas).

La classe OracleCommand

La classe OracleCommand, méthodes importantes

Méthodes	Significations
ExecuteNonQuery()	Exécute une instruction SQL sur Connection et retourne le nombre de lignes affectées. Pour toutes les requêtes DML
ExecuteReader()	Surchargé. Envoie CommandText à Connection et génère OracleDataReader. Pour les requêtes SELECT
ExecuteScalar()	Exécute la requête et retourne la première colonne de la première ligne. (en général pour les SELECT COUNT, SELECT MIN …

- **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.
- **La méthode ExecuteReader():** Cette méthode permet d'exécuter une requête SELECT et retourne un **SqlDataReadr** 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…

La classe OracleCommand

La classe OracleCommand, Exemple

Nous somme déjà connectés

Exemple1:

```
string sql = "insert into divisions values ('S','Sud')";
OracleCommand oraCmd = new OracleCommand(sql, oraConn);
int nbInsertion = oraCmd.ExecuteNonQuery();
Console.WriteLine("Nombre d'insertion" + nbInsertion);
```

Exemple2

```
string sqlcount = "select count(*) from joueurs";
 OracleCommand oraCmd2 = new OracleCommand(sqlcount, conn);
 decimal totaljoueurs =(decimal)oraCmd2.ExecuteScalar();
 Console.WriteLine("Nombre de joueurs" + totaljoueurs);
```

La classe OracleDataReader

La classe OracleDataReader, définition

un objet **DataReader** sert 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,).** Le contenu d'un OracleDataReader correspond à une requête SELECT (plusieurs lignes, plusieurs colonnes. En ce sens c'est comme une table).

L'objet OracleDataReader **NE se crée PAS** avec la méthone new()

L'objet OracleDataReader se crée avec la méthode ExecuteReader() de l'objet OracleCommand

```
OracleCommand oraCmd = new OracleCommand(sql, oraConn);
OracleDataReader oraRead = oraCmd. ExecuteReader()
```

La classe OracleDataReader

La classe OracleDataReader, Méthodes importantes

Méthodes	Signification
Colse()	Ferme l'objet SqlDataReader
Dispose ()	Libère toutes les ressources occupées par l'objet SqlDataReader
Read ()	Permet d'avancer l'objet SqlDataReader jusqu'à l'enregistrement suivant.
GetDecimal(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un objet Decimal.
GetDouble(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un nombre de type Double.
GetFloat(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un nombre de type Float.
GetInt16(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 16 bits.
GetInt32(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 32 bits.
GetInt64(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'un entier signé 64 bits.
GetName(indicolonne)	Obtient le nom de la colonne spécifiée.
GetString(indicolonne)	Obtient la valeur de la colonne spécifiée sous la forme d'une chaîne.

La classe OracleDataReader

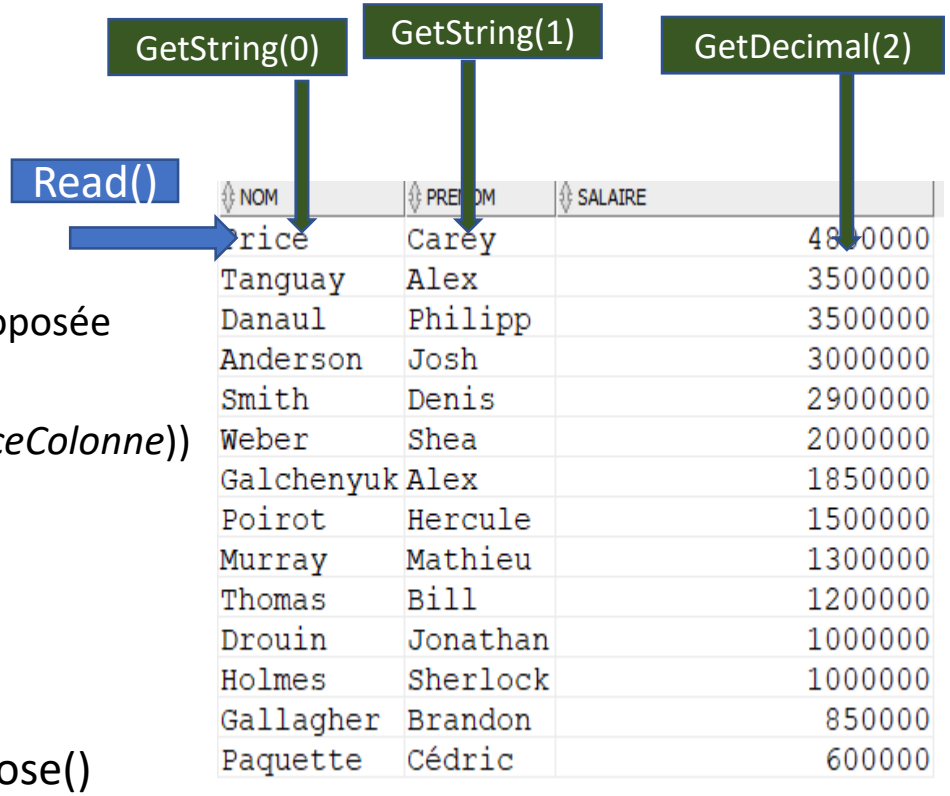
La classe OracleDataReader, lecture

Un objet OracleDataReader peut être vu comme un tableau de lignes et de colonnes.

Les colonnes sont les colonnes de la requêtes SELECT.

Les lignes sont les résultats de la requête SELECT

Pour lire **une ligne** on utilise la méthode **Read().** Par défaut, un DataReader charge une ligne entière en mémoire à chaque appel de la méthode Read().



Lorsque la ligne est chargée, il faudra lire les colonnes. La solution performante proposée permet d'accéder aux valeurs dans leurs types de données natifs (Type du SGBD)

Les valeurs de colonnes sont obtenues par la méthode Get() (*GetTypedonnées(indiceColonne)*)

- **Selon le type de colonne dans le SGBD Oracle**
- **Selon l'indice de colonne**

L'indice de la première colonne est toujours 0

On utilise une boucle while pour lire l'ensemble des ligne du DataReader

L'objet OracleDataReader doit être fermé à la fin. On utilise la méthode Close()

La classe OracleDataReader

La classe OracleDataReader, Exemple

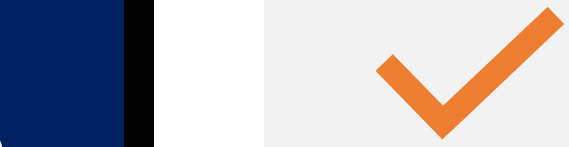
```
try
{
    string sql2 = "select nom, prenom, salaire from joueurs";
    OracleCommand oraSql2 = new OracleCommand(sql2, conn);
    OracleDataReader oraRead = oraSql2.ExecuteReader())

      while (oraRead.Read())
            {
              Console.WriteLine($"{oraRead.GetString(0),-10} - " +
                                 $"{oraRead.GetString(1),-10} - " +
                                 $" {oraRead.GetDecimal(2),-5}");
            }
    oraRead.Close()
    }

catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

Application

Écrire le code ADO.NET,C# qui permet de:

1. Se connecter à la base de données Oracle
2. D’ajouter une division
3. D’ajouter au joueur numéro 2 , 1% de son salaire
4. D’afficher le nombre total de joueurs de l’équipe MTL
5. D’afficher la liste des joueurs de MTL.
6. De fermer la connexion.

Introduction à ADO.NET



Conclusion

Questions