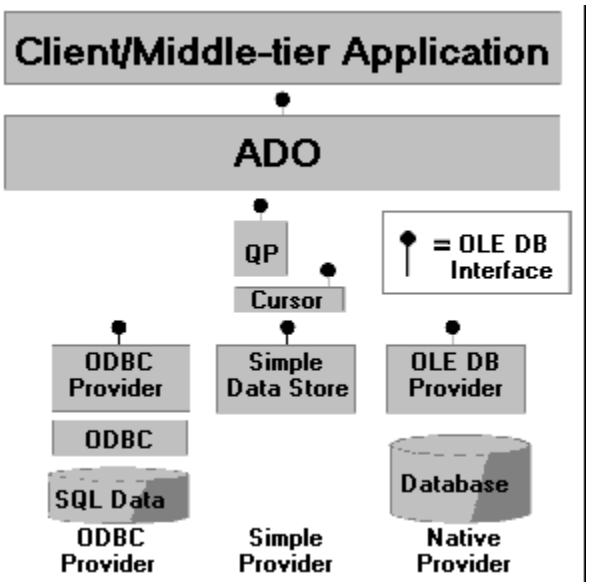
1. Un tour rapide d'ADO:

ADO (ActiveX Data Object) est un composant ActiveX permettant d'accéder aux bases de données de façon beaucoup plus facile sans se soucier de tout ce qui est allocation des environnements de travail. ADO fournit des objets qui permettent de se connecter à une base et de réaliser des requêtes SQL sur cette base.

1.1 - Active Data Objects (ADO).



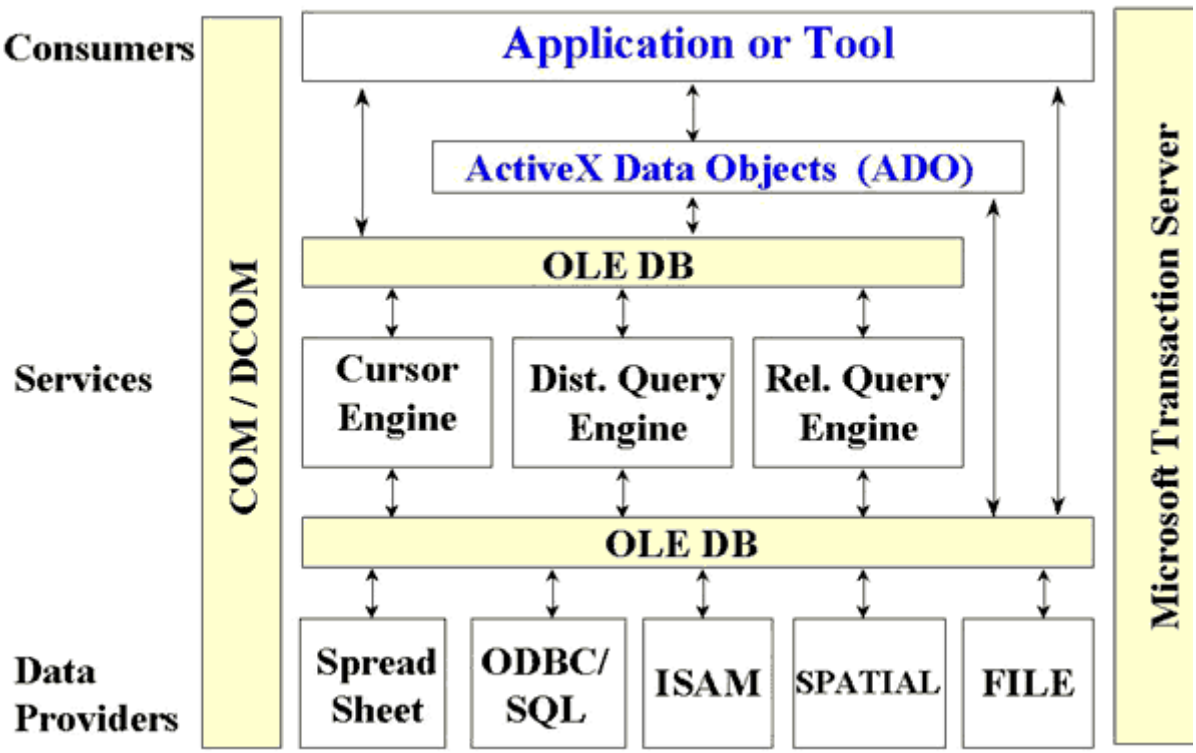
L'interface ADO implémente le modèle OLE DB

ADO implémente le modèle OLE DB qui lui permet par l'intermédiaire de "PROVIDER" d'accéder a toutes les sources de données. OLE DB ne remplace donc pas ODBC, mais il fournit un "PROVIDER" qui permet d'accéder les sources de données ODBC.

```
Dim oConn As Object
Dim oRS As Object

Set oConn = CreateObject("ADO.Connection")
oConn.Connect "pubs", "sa", ""
Set oRS = oConn.Execute("select * from authors")
Do While Not oRS.EOF
   Debug.Print oRS.Fields("au_lname").Value
   oRS.MoveNext
Loop
```

Exemple d'utilisation de ADO avec la base Pubs



OLE DB implémente trois types de composants : les providers, les services, et les consumers.

Les "data providers" sont des composants qui représente les sources de données comme les bases de données SQL, les fichiers séquentiel indexés, et les données issues d'un tableur.

Les "providers" expose les informations à OLE DB de manière uniforme en utilisant une abstraction commune appelée rowset.

Les Services sont des composants qui exploite et produisent les données OLE DB. Par exemple :

- Un "cursor engine " est un composant "service" qui peut exploiter les données d'un fichier séquentiel, pour produire une source de données "scrollable".
- Un "relational query engine " est un exemple de service qui surcharge les données OLE DB et qui produit des rowsets qui satisfont à une condition particulière.

Les Consumers sont des composants qui exploite les données OLE DB, par exemples des services tels que les "query processor".

Pour pouvoir utiliser ADO dans un projet Visual Basic ou Access vous avez deux solutions pour y parvenir :

- La première est de créer un **Projet de données**.

- La seconde est de rajouter dans le menu Projets - Références, **Microsoft AtiveX Data Objects 2.x Library**.

ADO propose les objets suivants :

- **Command** : permet d'exécuter des requêtes
- **Connection** : connexion à une source de données (aussi bien un fichier texte, qu'un fichier Excel, ou une base de données)
- **Error** : ensemble des erreurs retournées par le SGBD
- **Parameter** : permet de définir un paramètre d'une requête
- **Recordset** : jeu d'enregistrements retournés lors de l'exécution d'un SELECT

2. Etablir une connexion avec ADO et l'objet Connection:

Pour établir une connexion à une base de données avec ADO, on utilise l'objet Connection. Vous trouverez dans ce chapitre un exemple de connexion à une base de données Access, Oracle et SQL Server.

Une connexion à une base de données se définit par :

- l'hôte sur lequel se trouve la base de données
- le nom de la base de données
- le nom de l'utilisateur
- le mot de passe

L'ensemble de ces champs est appelé *chaîne de connexion*. Les champs "hôte" et "nom de la base de données" peuvent soit être définis dans le programme soit dans un DSN (Data Source Name). Un DSN se configure dans le panneau de configuration avec l'outil *Source de données (ODBC)*.

Comment réaliser une connexion ?

Tout d'abord il faut déclarer la variable associée à la connexion.

```
Dim cnx As New ADODB.Connection
```

ou alors

```
Dim cnx As ADODB.Connection
Set cnx = New ADODB.Connection
```

L'étape suivante consiste à écrire la chaîne de connexion. Comme il a été vu précédemment, il existe deux possibilités soit en utilisant un DSN soit en mettant les informations nécessaires dans la chaîne de connexion.

2.1 Connexion à une base de données Access sans DSN:

```
'Déclaration de la variable de connexion
Dim cnx As ADODB.Connection
Set cnx = New ADODB.Connection

'Définition du pilote de connexion
cnx.Provider = "Microsoft.Jet.Oledb.3.51"
'Définition de la chaîne de connexion
```

```
cnx.ConnectionString = "C:\maBase.mdb"
'Ouverture de la base de données
cnx.Open
```

Le pilote 3.51 permet d'accéder à Access 95 et 97. Pour Access 2000, il faut utiliser la version 4.0.

2.2 Connexion à une base de données SQL Server sans DSN:

```
'Déclaration de la variable de connexion
Dim cnx As ADODB.Connection
Set cnx = New ADODB.Connection
...

'Définition de la chaîne de connexion
cnx.ConnectionString = "UID=" & NomUtilisateur &";PWD=" & MotDePasse & ";" &
"DRIVER={SQL Server};Server=" & NomServeur & ";Database=" & NomBaseDeDonnées
& ";"

'Ouverture de la base de données
cnx.Open
```

Comme vous avez pû le constater, il n'y a pas la ligne cnx.Provider. En effet dans cet exemple le pilote à utiliser est décrit dans la chaîne de connexion : *DRIVER={SQL Server}*.

2.3 Connexion à une base de données Oracle sans DSN:

```
'Déclaration de la variable de connexion
Dim cnx As ADODB.Connection
Set cnx = New ADODB.Connection
...

'Définition de la chaîne de connexion
cnx.ConnectionString = "UID=" + NomUtilisateur & ";PWD=" & MotDePasse & ";" &
"DRIVER=msdaora;Server=" & NomServeur & ";Database=" & NomBaseDeDonnées & ";"

'Ouverture de la base de données
cnx.Open
```

Vous avez pû voir des chaînes de connexion avec comme DRIVER {Microsoft ODBC for Oracle}. Ceci marche mais n'est pas très recommandé.

2.4 Connexion à une base de données avec un DSN:

```
'Déclaration de la variable de connexion
Dim cnx As ADODB.Connection
Set cnx = New ADODB.Connection
...

'Définition de la chaîne de connexion
cnx.ConnectionString = "DSN=" & NomDuDSN & ";UID=" & NomUtilisateur & ";PWD=" &
MotDePasse & ";"

'Ouverture de la base de données
```

```
cnx.Open
```

Dans le cas où il y a le DSN, vous n'avez plus besoin de spécifier l'hôte sur lequel se trouve la base de données ainsi que le nom de la base de données.

Pour finir cette partie, voici une petite fonction de connexion avec DSN :

```
'=========================================================
==
' FUNCTION : InitConnection(...)
' DESCRIPTION : Initiliase la connexion à la base de données
' PARAMS : * DSN : Nom du DSN associé à la connexion
' * UserName : Nom de l'utilisateur
' * Password : Mot de passe de l'utilisateur
' VERSION : 1.1
'=========================================================
==
Public Function InitConnection(DSN As String, UserName As String, PassWord As String)
As Boolean
  Dim query As String
  Dim cnxString As String
  Dim RequeteOk As Boolean
  Dim mRst As New ADODB.Recordset

  InitConnection = False
  'Initialisation de la chaine de connexion
  ADOCnx.ConnectionString = "DSN=" & DSN & ";"

  'Vérifie que la connexion est bien fermée
  If ADOCnx.State = adStateOpen Then
    ADOCnx.Close
  End If
  On Error GoTo BadConnection
  'Connexion à la base de données
  ADOCnx.Open cnxString, UserName, PassWord, adAsyncConnect
  'Attente que la connexion soit établie
  While (ADOCnx.State = adStateConnecting)
    DoEvents
   Wend
  'Vérification des erreurs dans le cas d'une mauvaise connexion
  If ADOCnx.Errors.Count > 0 Then
    'Affichage des erreurs
    MsgBox ADOCnx.Errors.Item(0)
    InitConnection = False
    Exit Function
  Else
    InitConnection = True
   End If
   Exit Function

BadConnection:

If ADOCnx.Errors.Count > 0 Then
    'Affichage des erreurs
    MsgBox ADOCnx.Errors.Item(0)
    InitConnection = False
```

```
   Exit Function
Else
   MsgBox err.Description
End If
End Function
```

2.5 Quelques informations supplémentaires sur les pilotes:

Liste des pilotes ODBC sans DSN

dBase
Driver={Microsoft dBASE Driver (*.dbf)};DriverID=277;Dbq=chemin\nombd.dbf;

MS Access
Driver={Microsoft Access Driver (*.mdb)};Dbq=chemin\nombd.mdb;Uid=NomUtilisateur;Pwd=MotDePasse;

MS SQL Server
Driver={SQL Server};Server=NomDuServeur;Database=nombd;Uid=NomUtilisateur;Pwd=MotDePasse;

MS Text Drive
Driver={Microsoft Text Driver (*.txt; *.csv)};Dbq=chemin\;Extensions=asc,csv,tab,txt;Persist Security Info=False;

MySQL
Driver={mysql};database=nombd;server=NomDuServeur;uid=NomUtilisateur;pwd=MotDePasse;option=16386;

Oracle
Driver={Microsoft ODBC for Oracle};Server=ServeurOracle.schema;Uid=NomUtilisateur;Pwd=MotDePasse;

Visual Foxpro
Driver={Microsoft Visual FoxPro Driver};SourceType=DBC;SourceDB=chemin\nombd.dbc;Exclusive=No;

Liste des pilotes OLEDB

MS Access
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=chemin\nombd.mdb;User Id=NomUtilisateur;Password=MotDePasse;

MS SQL Server
Provider=SQLOLEDB;Data Source=NomServeur;Initial Catalog=nombd;User ID=NomUtilisateur;Password=MotDePasse;

MS SQL Server avec une adresse IP
Provider=SQLOLEDB; Data Source=xx.xx.xx.xx,1433; Network Library=DBMSSOCN; Initial Catalog=dbname;User ID=NomUtilisateur;Password=MotDePasse;

MS Text Driver
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=chemin;Extended Properties='text;FMT=Delimited'"

Oracle
Provider=OraOLEDB.Oracle;Data Source=nombd;User Id=NomUtilisateur;Password=MotDePasse;

3. Réaliser des requêtes avec l'objet Recordset

Comme pour l'objet Connection vous devez commencer par déclarer une variable de type Recordset.

```
Dim rst As New ADODB.Recordset
```

ou alors

```
Dim rst As ADODB.Recordset
Set rst = New ADODB.Recordset
```

Une fois ces lignes de code tapées, vous pouvez exécuter votre requête. Pour cela vous devez utiliser la méthode **Open** de l'objet Recordset. Cette méthode prend en paramètre :

- la requête
- la connexion sur laquelle vous souhaitez exécuter la requête
- le type du curseur (je vous renvoie à des cours de SGBD)
- le type de blocage
- le type de requête

L'ensemble de ces champs sont facultatifs, mais je vous conseille d'au moins passer les deux premiers paramètres à la méthode. Celà rend le code plus clair. Dans les exemples qui vont suivre je n'utiliserai que les deux premiers paramètres. Si vous souhaitez plus d'informations, je vous renvoie sur MSDN qui est très bien documenté.

Voici un petit exemple :

```
'Déclaration des variables
Dim cnx As ADODB.Connection
Dim rst As ADODB.Recordset

'Instanciation des variables
Set cnx = New ADODB.Connection
Set rst = New ADODB.Recordset

'Connexion à la base de données
cnx.ConnectionString = "Provider=" & PiloteDaccesAlaBaseDeDonnées & ";DSN=" &
NomDuDSN & ";UID=" & NomUtilisateur & ";PWD=" & MotDePasse & ";"
cnx.Open

'Exécution de la requête
rst.Open "SELECT nom, prenom, adresse FROM Client", cnx
```

Une fois le Open exécuté, l'ensemble des enregistrements retournés par le SELECT se trouvent dans l'objet Recordset, ici rst.

Pour accéder à ces enregistrements vous devez utiliser le champ Field. Chaque champ Field contient une colonne. Dans notre cas nous en avons 3 numérotées de 0 à 2. Vous avez donc deux solutions pour accéder à un champ. Prenons l'exemple du champ prenom. Vous faites rst.Fields(1) ou bien rst.Fields("prenom"). Vous avez pû voir certainement du code avec rst(1) ou bien rst("prenom"), c'est la même chose. Et oui les développeurs sont un peu faignants dès qu'il s'agit de tapper du code ;o)
Accèder à un champ c'est bien, mais pouvoir naviguer dans l'ensemble des enregistrements c'est mieux. Pour celà, il existent des méthodes permettant de le faire :

- MoveFirst : sélectionne le premier enregistrement
- MoveLast : sélectionne le dernier enregistrement
- MoveNext : sélectionne l'enregistrement suivant
- MovePrevious : sélectionne l'enregistrement précédent

Il y a deux propriétés de l'objet Recordset à connaître pour la navigation qui sont:

- BOF (Begin Of File) : est à vrai si l'objet Recordset pointe sur le début d'enregistrement
- EOF (End Of File) : est à vrai si l'objet Recorset pointe sur la fin de l'enregistrement

Voici un exemple de boucle permettant de parcourir un jeu d'enregistrement et d'afficher le résultat :

```
While Not(rst.EOF)
  MsgBox rst("nom") & " " & rst("prenom") & " habite au " & rst("adresse") & "."
  rst.MoveNext
Wend
```

Il arrive que certaines fois, il y est un problème avec l'objet Recordset et il ne pointe pas au début de l'enregistrement. Donc pour remédier à ce problème vous pouvez tappez les lignes suivantes après avoir effectué le Open.

```
rst.MoveLast
rst.MoveFirst
```

Une fois que vous n'utilisez plus le Recordset, pensez à le fermer avec la méthode Close.

```
rst.Close
```

Il existe une autre propriété qui peut être intéressante qui est RecordCount. Elle vous permet de savoir le nombre d'enregistrements stockés dans l'objet Recordset.

Voici un petit exemple de fonction pouvant exécuter tout type de requêtes via un Recordset :

```
'====================================================='
' FUNCTION : ExecSQL(...)
' DESCRIPTION : Execute une requête SQL
' PARAMS : * query : Requête à exécuter
'          * rst : Variable permettant de stocker les enregistrements
'====================================================='
Public Function ExecSQL(query As String, ByRef rst As ADODB.Recordset, ByRef cnx As
ADODB.Connection) As Boolean
```

```
  'Initialisation du RecordSet
  If rst.State <> adStateClosed Then rst.Close

  'Ouvre une transaction pour ne pas à avoir à réaliser de commit en fin de traitement
  ADOCnx.BeginTrans

  'Positionne le curseur côté client
  rst.CursorLocation = adUseClient
  'Vérifie que la connexion passée est bonne
  Set rst.ActiveConnection = cnx On Error GoTo ErrHandle
  'Exécute la requête
  rst.Open query, ADOCnx
  'Valide la transaction
  ADOCnx.CommitTrans
  ExecSQL = True
  Exit Function

ErrHandle:
  ExecSQL = False
  MsgBox "ADOManager.ExecSQL:ErrHandle" & vbCr & vbCr & err.Description, vbCritical
End Function
```

4. Réaliser des requêtes avec l'objet Command

Comme pour les autres objets, vous devez commencer par déclarer une variable de type Command.

```
Dim rst As New ADODB.Command
```

ou alors

```
Dim rst As ADODB.Command
Set rst = New ADODB.Command
```

L'objet Command est un peu plus complexe que le Recordset quoique. L'avantage de l'objet Command par rapport à l'objet Recordset est de pouvoir facilement paramétrer les requêtes mêmes les SELECT.

Pour pouvoir utiliser des requêtes paramétrables il faut utiliser le symbôle ? dans la requête SQL puis rajouter un objet Parameter à l'objet Command.

Pour être plus clair voici un exemple :

```
'Déclaration des variables
Dim cnx As ADODB.Connection
Dim cmd As ADODB.Command
Dim prm1 As ADODB.Parameter
Dim rst As ADODB.Recordset

'Instanciation des variables
```

```
Set cnx = New ADODB.Connection
Set cmd = New ADODB.Command
Set prm1 = New ADODB.Parameter
Set rst = New ADODB.Recordset

'Connexion à la base de données
cnx.ConnectionString = "Provider=" & PiloteDaccesAlaBaseDeDonnées & ";DSN=" &
NomDuDSN & ";UID=" & NomUtilisateur & ";PWD=" & MotDePasse & ";"
cnx.Open

'Préparation de l'objet Command
cmd.CommandText = "SELECT * FROM Client WHERE nom = ?"

'Préparation du paramètre
prm1.Name = "nom" 'Nom du champ correspondant
prm.Type = adVarChar 'Type du champ
prm.Direction = adInput 'Type de paramètre : Entrée, Sortie, Entrée/Sortie
prm.Size = 40 'Taille maximale du champ
prm.Value = "Dupond" 'Valeur du paramètre

'Exécution de la requête
Set rst = cmd.Execute
```

Comme vous avez pû le remarquer, j'ai utilisé un Recordset dans cet exemple. Voici une façon de récupérer les enregistrements retournés par un SELECT. L'objet Recordset n'est nécessaire que dans le cas d'un SELECT. Dans les autres cas vous pouvez taper juste *cmd.Execute*. La méthode Execute de l'objet Command peut prendre trois paramètres qui sont falcutatifs:

- le premier est le nombre d'enregistrements affectés par la requête. Il est de type Long
- le deuxième est un tableau de Variant contenant les paramètres de la requête SQL
- le troisème indique le type de valeur que le fournisseur doit attribuer à la propriété CommandText

La propriété CommandText contient la requête à exécuter. Ca peut être une requête standard comme une requête de modification de l'architecture d'une table ou de la base de données.