Excel et les bases de données

Travailler avec des bases Access

Avec Excel, il est également possible de travailler avec de véritables bases de données, par exemple de type Access.

Le grand avantage des bases de données sur les simples fichiers de données est qu'elles disposent de la puissance du langage SQL et qu'elles permettent de mieux structurer l'information en travaillant sur plusieurs tables de données.

Nous n'étudierons ici que les bases de données Access, ce logiciel étant intégré, comme Excel, à la suite Microsoft Office.

Nous nous concentrerons sur les relations entre Excel et Access et nous supposerons que la base de données ainsi que les tables qui la composent ont été créées directement dans Access.

Les connexions ADO

Pour travailler sur une base Access à partir de Visual Basic pour Excel, il faut d'abord pouvoir s'y connecter. Nous utiliserons ici un type de connexion disponible sur Excel, les connexions ADO.

Dans le programme Visual Basic nous devons créer une connexion définie par une chaîne de connexion, c'est-à-dire un texte expliquant au programme comment se connecter. Mais, avant tout, nous devons configurer Visual Basic. Dans le menu *Outils* de Visual Basic nous devons sélectionner *Références* puis cocher :

- Microsoft Ado Ext. 6.0 for DDL and Security
- Microsoft ActiveX Data Objects 6.1 Library

Par exemple, si nous donnons le nom *Connect* à notre connexion, nous devons d'abord la déclarer puis la créer :

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

On peut alors ouvrir la connexion en fournissant au programme le type de connexion et le chemin d'accès à la base de données :

```
With Connect
    .ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\Essais\Commerce.accdb"
    .Open
End With
```

Nous nous connectons ainsi à la base Access *Commerce.accdb* située dans le dossier *C:\Essais*. Cette base contient une table *Ventes* définie par trois champs :

- *Produit* défini comme un texte court de 20 caractères ;
- *Vendeur* défini comme un texte court de 20 caractères ;
- *Valeur* défini comme un numérique réel double.

Les commandes

Nous nous proposons de charger les données suivantes contenues dans la feuille *Données* du classeur.

	A	B	C
1	Robes	Martin	1234,12
2	Manteaux	Martin	675,32
3	Pantalons	Martin	3455,64
4	Chemises	Martin	2145,12
5	Robes	Dupond	3421,12
6	Manteaux	Dupond	2541,34
7	Pantalons	Dupond	2365,15
8	Chemises	Dupond	3215,37
9			

Nous allons commencer par vider la table *Ventes* pour s'assurer qu'elle ne contient aucun enregistrement. Pour cela nous allons utiliser la commande SQL : *DELETE FROM*.

Pour lancer une requête SQL depuis Visual Basic nous devons définir une commande que nous avons appelé ici *Commande* de la manière suivante :

```
Dim Commande As ADODB.Command
Set Commande = New ADODB.Command
Set Commande.ActiveConnection = Connect
```

Nous indiquons ici au programme que nous voulons utiliser la connexion *Connect* que nous avons définie précédemment.

Nous pouvons maintenant définir la commande et la lancer :

```
With Commande
    .CommandText = "DELETE FROM Ventes "
    .CommandType = adCmdText
    .Execute
End With
```

Nous pouvons vérifier dans Access que les enregistrements présents ont bien été effacés.

Nous pouvons écrire nos données dans la table *Ventes* par le programme suivant :

```
Set f = ThisWorkbook.Sheets("Données")
For i = 1 To 8
    produit = f.Cells(i, 1)
    vendeur = f.Cells(i, 2)
    va = f.Cells(i, 3)
    valeur = Replace(va, ",", ".")
    With Commande
        .CommandText = "INSERT INTO Ventes VALUES(" & "'" & produit
& "','" & vendeur & "'," & valeur & ")"
        .CommandType = adCmdText
        .Execute
    End With
Next i
```

Nous pouvons vérifier que les données ont bien été chargées dans la table *Ventes*.

L'objet Recordset

Nous pouvons également lire dans Excel des données contenues dans la base de données. Par exemple, nous allons afficher dans la feuille de calcul *Données* les enregistrements de la table *Ventes*.

Pour cela nous devons créer dans notre programme un objet *Recordset* qui recueillera les données provenant de la table *Ventes*. Avant tout, nous devons configurer Visual Basic et dans le menu *Outils/Références* nous devons cocher *Microsoft ActiveX Data Object Recordsets 6.0 Library*.

Si nous appelons *Record* notre Recordset, nous devons le créer de la manière suivante :

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

Nous allons maintenant charger notre objet Recordset grâce à une requête SQL :

```
Record.Open " SELECT * FROM Ventes ", Connect
```

Pour lire notre Recordset et afficher les données dans la feuille de calcul *Données*, nous allons procéder ainsi :

```
Record.MoveFirst
i = 1
Do While Not Record.EOF
    f.Cells(i + 10, 1) = Record("Produit")
    f.Cells(i + 10, 2) = Record("Vendeur")
    f.Cells(i + 10, 3) = Record("Valeur")
    i = i + 1
    Record.MoveNext
Loop
```

Nous pouvons constater que les données affichées dans la feuille de calcul sont bien les mêmes que celles que nous avions saisies.

Le programme complet est le suivant :

```
Sub Ecrire()
'Suppose avoir sélectionné dans Outils/Références :
'Microsoft ActiveX Data Objects 6.1 Library
'Microsoft ActiveX Data Object Recordsets 6.0 Library
'Microsoft Ado Ext 6.0 for DDL and Security

Dim Connect As ADODB.Connection
Dim Record As ADODB.Recordset
Dim Commande As ADODB.Command
Set Connect = New ADODB.Connection

With Connect
    .ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\Essais\Commerce.accdb"
    .Open
End With

Set Commande = New ADODB.Command
Set Commande.ActiveConnection = Connect
With Commande
    .CommandText = "DELETE FROM Ventes "
    .CommandType = adCmdText
    .Execute
End With
Set f = ThisWorkbook.Sheets("Données")
For i = 1 To 8
    produit = f.Cells(i, 1)
    vendeur = f.Cells(i, 2)
    va = f.Cells(i, 3)
    valeur = Replace(va, ",", ".")
    With Commande
        .CommandText = "INSERT INTO Ventes VALUES(" & "'" & produit
& "','" & vendeur & "'," & valeur & ")"
        .CommandType = adCmdText
        .Execute
    End With
Next i
Set Record = New ADODB.Recordset
Record.Open " SELECT * FROM Ventes ", Connect
Record.MoveFirst

i = 1
Do While Not Record.EOF
    f.Cells(i + 10, 1) = Record("Produit")
    f.Cells(i + 10, 2) = Record("Vendeur")
    f.Cells(i + 10, 3) = Record("Valeur")
```

```
    i = i + 1
    Record.MoveNext
Loop

Connect.Close

End Sub
```

Base Access protégée par un mot de passe

Pour se connecter à une base Access protégée par un mot de passe, il faut ouvrir la connexion en précisant le mot de passe. Cela se fait de la manière suivante :

```
Dim Connect As ADODB.Connection
Set Connect = New ADODB.Connection
With Connect
    .ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=" & BaseRSP & "; Jet OLEDB:Database Password=ABCD;"
    .Open
End With
```

Où *ABCD* est le mot de passe.

Rappelons que pour définir un mot de passe sur une base Access, il faut l'ouvrir en mode exclusif. Pour cela, il faut aller dans le menu *Ouvrir*, sélectionner *Parcourir*, choisir la base que l'on souhaite ouvrir puis, dans la liste déroulante *Ouvrir* sélectionner *Ouvrir en exclusif*. La base s'ouvre alors en mode exclusif.

Il faut alors aller dans le menu *Fichier*, sélectionner *Chiffrer avec mot de passe* puis entrer le mot de passe choisi.

Connexions ADO avec des fichiers CSV

Il est possible de travailler sur des fichiers CSV en utilisant des connexions ADO, cela permet de pouvoir bénéficier, tout au moins partiellement, de la puissance de SQL.

La principale différence avec les connexions à une base de données Access réside dans la chaîne de connexion. La source ne fait, en effet, plus référence à une base mais au dossier dans lequel se trouvent les fichiers CSV auxquels on veut se connecter. Il faut également préciser

le type des fichiers grâce à l'option *Extended Properties*. Celle-ci se présente, par exemple, ainsi :

```
Extended Properties="text;HDR=NO;FORMAT=Delimited(;)"
```

Ce texte doit être inséré dans la chaîne de connexion, comme il contient des guillemets ceux-ci doivent être doublés pour pouvoir être reconnus comme tels à l'intérieur de la chaîne. On pourra écrire, par exemple :

```
RepertNom = "C:\Essais\"
Set Connect = New ADODB.Connection
propriete = " ; Extended
Properties=""text;HDR=NO;FORMAT=Delimited(;)"""
With Connect
    .ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=" & RepertNom & propriete
    .Open
End With
```

HDR indique si le fichier CSV contient ou non les titres des colonnes en première ligne. Dans notre exemple, ce n'est pas le cas, si ce l'était, il faudrait écrire *HDR=YES*. Dans le format, nous avons précisé que les fichiers CSV utilisent le point-virgule comme séparateur. Le format par défaut est la virgule.

Le fichier Schema.ini

Lorsqu'on utilise des connexions ADO avec des fichiers CSV, il est très recommandé, voire nécessaire, d'utiliser un fichier texte *Schema.ini* qui décrit la structure des fichiers CSV utilisés. Il se présente, par exemple, sous la forme suivante :

```
[EssaiCSV.csv]
Format=Delimited(;)
DecimalSymbol=,
Col1=OP Char Width 25
Col2=VAR Char Width 25
Col3=BR Char Width 25
Col4=VA Float
[Libel.csv]
Format=Delimited(;)
```

Ici, nous avons décrit deux fichiers CSV : EssaiCSV.csv et Libel.csv. Toutes les fichiers doivent être décrits dans le même fichier *Schema.ini* et apparaître les uns en-dessous des autres. Le nom du fichier est indiqué entre crochets.

Pour le fichier *EssaiCSV.csv*, nous avons précisé qu'il utilisait le point-virgule comme séparateur et la virgule comme symbole décimal. Nous avons ensuite décrit ses colonnes en précisant pour chaque colonne son nom et son format. Les formats possibles sont les suivants :

Char (un texte)
Float (un nombre réel)
Integer (un nombre entier court)
LongChar (un mémo)
Date (une date)

Pour le fichier *Libel.csv*, seul son séparateur est précisé, Excel va alors déterminer le format de chacune des colonnes en lisant les premières lignes. Cela peut s'avérer dangereux si le format peut être interprété différemment selon les lignes. Par exemple, si les premières lignes contiennent des entiers et les dernières des nombres décimaux, Excel va comprendre que le format est *Integer* et il va supprimer la partie décimale des derniers nombres.

Le fichier Schema.ini est un fichier texte, il peut être créé avec le bloc-notes de Windows. Il doit impérativement se trouver dans le même dossier que les fichiers CSV qu'il décrit.

Travailler avec plusieurs fichiers CSV

L'un des principaux avantages d'utiliser une connexion ADO pour travailler avec des fichiers CSV est de pouvoir travailler avec plusieurs fichiers et de faire des jointure avec SQL. Par exemple, supposons que nous disposions des deux fichiers CSV : *EssaiCSV.csv* et *Libel.csv* décrits dans le fichier *Schema.ini* et qu'ils se présentent ainsi :

```
OP;VAR;BR;VA
B.1;CSD;BR1;1034,45
B.2;CSD;BR1;654
P.1;TRE;BR1;754
B.1;CSD;BR2;1034,45
B.2;CSD;BR2;654
P.1;TRE;BR2;7540
B.1;CSD;BR3;1034,45
B.2;CSD;BR3;654
P.1;TRE;BR3;754
B.1;CSD;BR4;1034,45
B.2;CSD;BR4;654
P.1;TRE;BR4;754
B.1;CSD;BR5;1034,45
B.2;CSD;BR5;654
P.1;TRE;BR1;754
B.1;CSD;BR1;1034,45
```

```
B.2;CSD;BR1;654
P.1;TRE;BR1;754
B.1;CSD;BR1;1034,45
B.2;CSD;BR1;654
P.1;TRE;BR1;754
B.1;CSD;BR1;1034,45
B.2;CSD;BR1;654
P.1;TRE;BR1;754,6785
```

Et :

```
OP;LIBEL
B.1;Valeur ajoutée
B.2;Excédent brut d'exploitation
P.1;Production
```

Le programme suivant réalise la jointure des deux fichiers et affiche les résultats dans la feuille *Données* :

```
Sub JointureCSV() 'Suppose avoir sélectionné dans Outils/Références :
'Microsoft ActiveX Data Objects 6.1 Library
'Microsoft ActiveX Data Object Recordsets 6.0 Library
'Microsoft Ado Ext 6.0 for DDL and Security

Dim Connect As ADODB.Connection
Dim Record As ADODB.Recordset

RepertNom = "C:\Essais\"

Set Connect = New ADODB.Connection
propriete = " ; Extended
Properties=""text;HDR=YES;FORMAT=Delimited(;)"""
With Connect
    .ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=" & RepertNom & propriete
    .Open
End With

Set Record = New ADODB.Recordset

Record.Open "SELECT e.OP, l.LIBEL, e.VAR, e.BR, e.VA FROM
EssaiCSV.csv e, Libel.csv l where e.OP=l.OP", Connect

Set f = ThisWorkbook.Sheets("Données")

Record.MoveFirst
i = 1
Do While Not Record.EOF
```

```
        f.Cells(i, 1) = Record("OP")
        f.Cells(i, 2) = Record("LIBEL")
        f.Cells(i, 3) = Record("VAR")
        f.Cells(i, 4) = Record("BR")
        f.Cells(i, 5) = Record("VA")
        i = i + 1
        Record.MoveNext
    Loop
    Record.Close
    Connect.Close
    End Sub
```

On obtient :

	A	B	C	D	E
1	B.1	Valeur ajoutée	CSD	BR5	1034,45
2	B.1	Valeur ajoutée	CSD	BR1	1034,45
3	B.1	Valeur ajoutée	CSD	BR2	1034,45
4	B.1	Valeur ajoutée	CSD	BR1	1034,45
5	B.1	Valeur ajoutée	CSD	BR3	1034,45
6	B.1	Valeur ajoutée	CSD	BR1	1034,45
7	B.1	Valeur ajoutée	CSD	BR4	1034,45
8	B.1	Valeur ajoutée	CSD	BR1	1034,45
9	B.2	Excédent brut d'exploit	CSD	BR1	654
10	B.2	Excédent brut d'exploit	CSD	BR2	654
11	B.2	Excédent brut d'exploit	CSD	BR3	654
12	B.2	Excédent brut d'exploit	CSD	BR4	654
13	B.2	Excédent brut d'exploit	CSD	BR5	654
14	B.2	Excédent brut d'exploit	CSD	BR1	654
15	B.2	Excédent brut d'exploit	CSD	BR1	654
16	B.2	Excédent brut d'exploit	CSD	BR1	654
17	P.1	Production	TRE	BR1	754,6785
18	P.1	Production	TRE	BR1	754
19	P.1	Production	TRE	BR3	754
20	P.1	Production	TRE	BR1	754
21	P.1	Production	TRE	BR2	7540
22	P.1	Production	TRE	BR1	754
23	P.1	Production	TRE	BR1	754
24	P.1	Production	TRE	BR4	754

La commande COMMAND

On peut également utiliser la commande *Command* pour effectuer des requêtes SQL. Par exemple, dans le programme précédent, on peut insérer la table *Ventes1* dans la table *Ventes* de la manière suivante :

```
Dim Commande As ADODB.Command
Set Commande = New ADODB.Command
With Commande
    .ActiveConnection = Connect
    .CommandText = " INSERT INTO Ventes.csv SELECT * FROM
Ventes1.csv "
    .CommandType = adCmdText
    .Execute
End With
```

Pour que ce programme fonctionne, il faut au préalable avoir défini la structure du fichier *Ventes1.csv* dans le fichier texte *Schema.ini*.

La principale limitation avec les fichiers CSV est qu'on ne peut pas modifier des enregistrements spécifiques avec la commande UPDATE ou les supprimer avec la commande DELETE FROM car les fichiers CSV sont des fichiers séquentiels et que tout nouvel enregistrement est placé à la fin du fichier.

Ce n'est pas forcément un problème pour les comptables nationaux car il est préférable que, comme les comptables d'entreprise, ils n'effacent jamais de données et que, pour effectuer une correction, ils commencent par introduire un enregistrement avec des valeurs de signe opposé à l'enregistrement qu'ils veulent modifier.

Cela dit, nous avons vu au chapitre précédent qu'il est toujours possible de modifier ou de supprimer des enregistrements dans un fichier CSV en utilisant les instructions Line Input # et Print #.

Auteur : Francis Malherbe