

Excel et les bases de données

Pourquoi travailler avec des bases de données

Si l'on se place du point de vue informatique, le travail des comptables nationaux présente deux caractéristiques essentielles :

- il porte sur des données nombreuses ;
- les comptes se présentent sous la forme de tableaux.

Pour travailler sur des données nombreuses, les bases de données sont l'outil idéal, pour travailler sur des tableaux il est difficile d'imaginer mieux qu'un tableur. Un logiciel qui permettrait de combiner ces deux outils serait donc particulièrement bien adapté au travail du comptable national. Or, il se trouve qu'Excel, qui est fondamentalement un tableur, permet également de travailler en relation avec des bases de données. Si l'on ajoute à cette propriété remarquable la capacité de programmation, on obtient un outil qui peut s'avérer extrêmement puissant entre les mains d'un comptable national expérimenté.

Excel peut donc travailler avec de nombreuses bases de données. Notons, cependant, que l'un des inconvénients des bases de données actuelles est l'inexistence d'un format de fichier universel, si bien qu'il est parfois difficile d'utiliser les résultats produits par un gestionnaire de base de données avec un autre gestionnaire de bases de données. Ceci peut s'avérer un inconvénient non négligeable pour les comptables nationaux amenés à travailler dans différents pays. Aussi, l'une des solutions possibles est l'utilisation de fichiers textes de format CSV.

Les fichiers CSV sont des fichiers texte contenant des données séparées par un point-virgule. Il s'agit ici du format européen car le format américain utilise la virgule comme séparateur (d'où le nom Comma Separator Values). Les fichiers CSV permettent de stocker des données qui pourront être utilisées aussi bien par Excel que par n'importe quel gestionnaire de bases de données.

Utiliser des fichiers CSV dans Excel présente quatre grands avantages :

- il est possible de travailler avec des fichiers contenant plus d'enregistrements que le nombre de lignes des feuilles de calcul Excel ;
- il est possible d'appliquer des requêtes SQL à ces fichiers, ce qui permet d'utiliser Excel comme un gestionnaire de bases de données ;
- ces fichiers sont, comparativement à d'autres formats, très compacts. C'est particulièrement vrai pour les fichiers organisés en hypercubes qui contiennent essentiellement des codes, c'est-à-dire des textes ;
- les fichiers permettent d'échanger facilement des données avec d'autres programmes car le format CSV est universel.

Les fichiers CSV n'ont cependant pas que des avantages. Ainsi, on ne peut pas les modifier aussi facilement avec des requêtes SQL que des fichiers utilisant d'autres formats. En effet, les enregistrements étant de longueur variable et écrits les uns à la suite

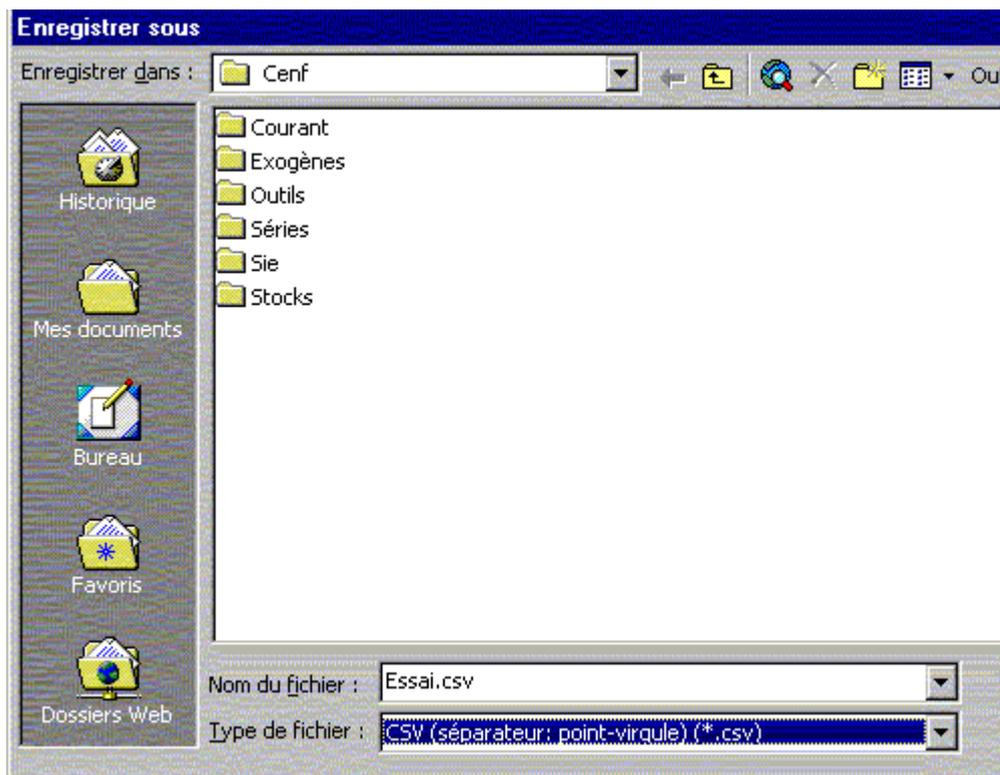
des autres, il n'est pas possible, par exemple, de remplacer un enregistrement par un autre plus long sans modifier totalement le fichier. Aussi, la méthode normale pour corriger un enregistrement dans un fichier CSV est-elle d'annuler l'ancien enregistrement par un enregistrement de signe contraire avant de passer le nouvel enregistrement. Cette contrainte n'est pas nécessairement un inconvénient en comptabilité nationale car elle oblige à conserver la trace des corrections, comme c'est le cas en comptabilité d'entreprise. L'autre inconvénient est l'existence de deux formats de fichiers CSV, le format d'origine avec la virgule comme séparateur de données et le point pour séparer la partie décimale des nombres, le format européen qui utilise le point-virgule comme séparateur de données et la virgule pour les décimales. Il sera donc nécessaire dans la définition de ces fichiers de spécifier quel est le séparateur décimal.

Créer des fichiers CSV

Nous ne traiterons ici que des fichiers CSV structurés en hypercubes. Le moyen le plus simple de créer un petit fichier CSV est de l'écrire dans une feuille Excel et de le sauvegarder au format CSV. La première ligne doit contenir les noms des variables et les suivantes les enregistrements. Par exemple, le fichier peut être constitué de la manière suivante :

	A	B	C	D	E
1	DA	AE	OP	VAL	
2	DA2000	AEGA01	OPP1	1200	
3	DA2000	AEGA02	OPP1	750	
4	DA2000	AEGA01	OPP2	680	
5	DA2000	AEGA02	OPP2	351	
6	DA2001	AEGA01	OPP1	1350	
7	DA2001	AEGA02	OPP1	832	
8	DA2001	AEGA01	OPP2	691	
9	DA2001	AEGA02	OPP2	372	
10					
11					

Il peut être sauvegardé au format CSV avec le menu *Enregistrer sous* en sélectionnant l'option *CSV (séparateur point-virgule)*.



Mais, le plus souvent, on dispose d'un tableau Excel que l'on souhaite transformer en fichier CSV structuré en hypercube. Supposons, par exemple, que nous souhaitions transformer le tableau ci-dessous en fichier CSV.

	A	B	C	D
1	OPB1			
2		DA2000	DA2001	
3	AEGA01	50	56	
4	AEGA02	100	105	
5	AEGA03	151	45	
6	AEGB01	28	34	
7	AEGB02	32	28	
8				
9				

L'hypercube possède trois dimensions : DA, OP, AE. On pourra générer dans le répertoire c:\User\Essais un fichier CSV nommé *Fichier.csv*, structuré en hypercube, à l'aide de la macro suivante :

```
(Général) CréerCSV
Sub CréerCSV()
Open "c:/user/Essais/Fichier.csv" For Output As #1
Print #1, "DA;OP;AE;VAL"
op = "OPB1"
Set f = Sheets("Essai")
For i = 3 To 7
    ae = f.Cells(i, 1)
    For j = 2 To 3
        da = f.Cells(2, j)
        va = f.Cells(i, j)
        enr = da & ";" & op & ";" & ae & ";" & va
        If va <> 0 Then Print #1, enr
    Next j
Next i
Close #1
End Sub
```

Dans cette macro, l'instruction *Open "c:/user/Essais/Fichier.csv" For Output As #1* ouvre dans le répertoire C:\User\Essais un fichier texte de nom *Fichier.csv*. La clause *Output* indique que l'on va écrire dans ce fichier texte, si un fichier du même nom existait déjà il serait écrasé. La clause *As #1* indique que le fichier sera repéré dans toute la macro par son numéro qui est ici 1. Nous aurions pu choisir pour numéro tout entier compris entre 1 et 511.

L'instruction *Print #1, "DA;OP;AE;VAL"* sert à écrire dans le fichier dont le numéro est 1, c'est-à-dire dans le fichier *Fichier.csv*, le texte *DA;OP;AE;VAL*. Ce texte est placé dans la première ligne du fichier et il permet de déterminer le nom des champs de données. Remarquons que les noms des champs sont séparés par un point-virgule car nous travaillons ici en format européen.

On va ensuite créer des variables correspondant à chacun des champs. Ainsi, on affecte à la variable *op* la valeur *OPB1* qui est commune à tous les enregistrements. On crée ensuite la variable objet *f* représentant la feuille *Essai* dans laquelle est placé le tableau, puis on lit, grâce à deux boucles imbriquées, les valeurs des variables *ae*, *da*, *va* représentant les différents champs. Remarquons que nous utilisons ici la variable *va* et non *val* car *val* est un mot réservé qui ne peut être utilisé comme nom de variable.

On va ensuite constituer le texte qui sera envoyé dans le fichier et qui correspond à une ligne de données. Les différentes variables sont concaténées grâce à l'opérateur *&*. Des points-virgules correspondant aux séparateurs de données sont insérés entre les variables. On a introduit la condition *If va <> 0* car il est inutile de saisir des enregistrements avec des valeurs nulles. A la fin, il est obligatoire de fermer le fichier par l'instruction *Close #1* pour pouvoir l'utiliser par la suite.

Regrouper des fichiers CSV

Il est souvent intéressant, lorsqu'on travaille avec des données issues de plusieurs sources, de pouvoir regrouper en un seul plusieurs fichiers CSV. Cela est particulièrement facile lorsque les fichiers ont la même structure. Supposons, par

exemple, que nous cherchions à regrouper le fichier *fichier.csv* avec les fichiers *fich1.csv* et *fich2.csv* présentés ci-dessous :

DA	OP	AE	VAL
DA2000	OPB1	AEGA01	1500
DA2001	OPB1	AEGA01	254
DA2000	OPB1	AEGA02	851
DA2001	OPB1	AEGA02	265

DA	OP	AE	VAL
DA2000	OPB1	AEGA01	650
DA2001	OPB1	AEGA01	850
DA2000	OPB1	AEGA02	430
DA2001	OPB1	AEGA02	142

Le programme ci-dessous permet de créer un fichier *fichtout.csv* qui regroupe les trois fichiers.

```

Sub Agrege ()
'
'Création d'une variable alphanumérique enr de longueur 100
'Nécessaire pour éviter que l'enregistrement soit tronqué
Dim enr As String * 100
'Création du fichier de sortie
Open "C:\User\Essais\fichtout.csv" For Output As #1
'Ouverture en mode lecture des fichiers à regrouper
Open "C:\User\Essais\fichier.csv" For Input As #2
Open "C:\User\Essais\fich1.csv" For Input As #3
Open "C:\User\Essais\fich2.csv" For Input As #4
'Ecriture des intitulés des variables
Print #1, "DA;OP;AE;VAL"
'Début de la boucle de lecture
For i = 2 To 4
' Lecture du premier enregistrement contenant les intitulés
Line Input #i, enr
' Lecture des autres enregistrements et écriture dans fichtout.
Do While Not EOF(i)
Line Input #i, enr
Print #1, Trim$(enr)
Loop
Close #i
Next i
Close #1
'
End Sub

```

Nous avons utilisé ici deux modes d'accès aux fichiers CSV, le mode écriture pour le fichier *fichtout.csv* et le mode lecture pour les fichiers *fich1.csv* et *fich2.csv*. Le mode écriture est caractérisé par *output* et le mode lecture par *input*. Il existe aussi le mode *append* pour ajouter des enregistrements à un fichier existant. La différence entre le

mode *output* et le mode *append* est que le mode *output* écrase le fichier existant et que le mode *append* conserve les enregistrements existants.

L'instruction *Line Input #i, enr* lit la ligne courante du fichier *i* et la place dans la variable *enr*, elle passe ensuite à l'enregistrement suivant. La boucle *DO While NOT EOF(i) ... Loop* permet de lire tous les enregistrements du fichier *i* jusqu'à sa fin. Les lignes du fichier *i* sont lues puis envoyées au fichier *l*. La fonction *Trim\$* est utilisée pour éliminer les espaces à la fin de la variable *enr*, ce qui est nécessaire car cette variable ayant été dimensionnée avec une longueur de 100, le programme ajoute des espaces au texte pour atteindre cette longueur.

Remarquons, dans ce programme, comment est gérée la première ligne qui correspond aux intitulés des variables. Elle est, en effet, envoyée tout d'abord dans le fichier de sortie, puis, dans chaque fichier de lecture, la première ligne est lue par l'instruction *Line input* mais n'est pas écrite. Dans le cas contraire, on se retrouverait avec plusieurs lignes d'intitulés parmi les données.

Nous trouvons le fichier *fichtout.csv* suivant :

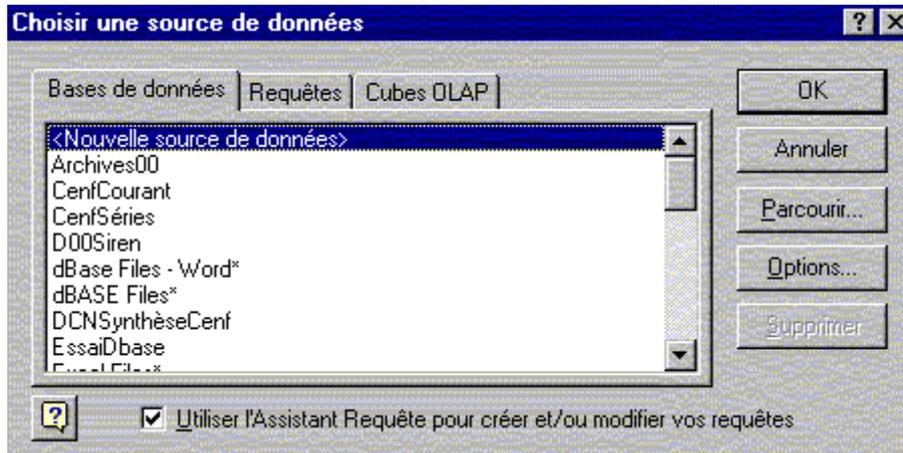
DA	OP	AE	VAL
DA2000	OPB1	AEGA01	10050
DA2001	OPB1	AEGA01	56
DA2000	OPB1	AEGA02	100
DA2001	OPB1	AEGA02	105
DA2000	OPB1	AEGA03	151
DA2001	OPB1	AEGA03	45
DA2000	OPB1	AEGB01	28
DA2001	OPB1	AEGB01	34
DA2000	OPB1	AEGB02	32
DA2001	OPB1	AEGB02	28
DA2000	OPB1	AEGA01	1500
DA2001	OPB1	AEGA01	254
DA2000	OPB1	AEGA02	851
DA2001	OPB1	AEGA02	265
DA2000	OPB1	AEGA01	650
DA2001	OPB1	AEGA01	850
DA2000	OPB1	AEGA02	430
DA2001	OPB1	AEGA02	142

Lire des fichiers CSV avec des tableaux croisés dynamiques

Le moyen le plus simple pour lire un petit fichier CSV est de l'ouvrir avec Excel, dans ce cas il apparaîtra avec les champs placés dans des colonnes distinctes. Mais les fichiers CSV qui contiennent plus d'enregistrements que le nombre de lignes d'Excel ne peuvent être lus de cette manière, il faut donc utiliser d'autres méthodes et parmi elles l'utilisation d'un tableau croisé dynamique apparaît comme l'une des plus simples. Pour le montrer, nous allons donc lire notre fichier *Fichier.csv* avec un tableau croisé dynamique.

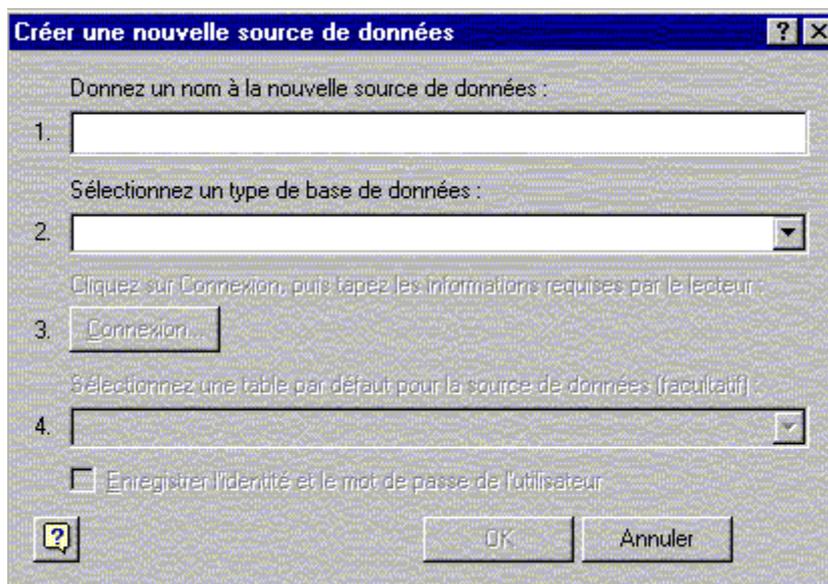
Création d'une connexion

Pour utiliser un tableau croisé dynamique avec une source de données externe, il faut disposer d'une connexion avec cette source. La première fois, il faudra la créer et, pour cela, il faut aller dans le menu *Données* d'Excel et sélectionner *Autres sources*, puis *Depuis Microsoft Query*. Apparaît alors le cadre suivant qui peut présenter une liste plus ou moins longue de noms.

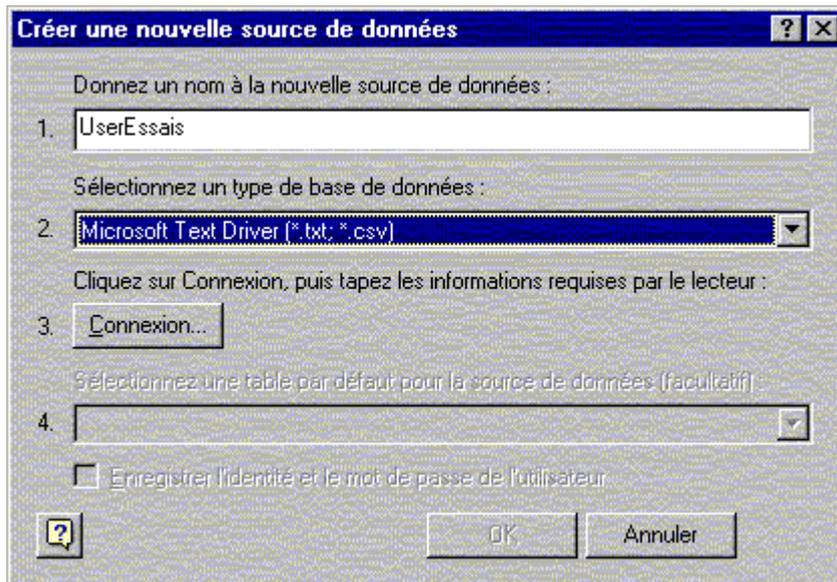


Chaque nom de la liste correspond à une source de données caractérisée par le répertoire dans lequel se trouvent les données et leur type, par exemple Access, dans notre cas il s'agira de base de données de type fichier texte CSV. La première fois, il faut définir un nom correspondant à la source que nous voulons utiliser. Puisque nous ne travaillons ici qu'avec des fichiers de type CSV, il est judicieux que le nom choisi fasse référence au répertoire sur lequel se trouve les données. Par exemple, puisque le fichier *Fichier.csv* se trouve dans le répertoire *C:\User\Essais* nous choisirons le nom *UserEssais* pour nous connecter.

Sélectionnons donc *Nouvelle source de données* et cliquons sur *OK*. Le cadre suivant apparaît :



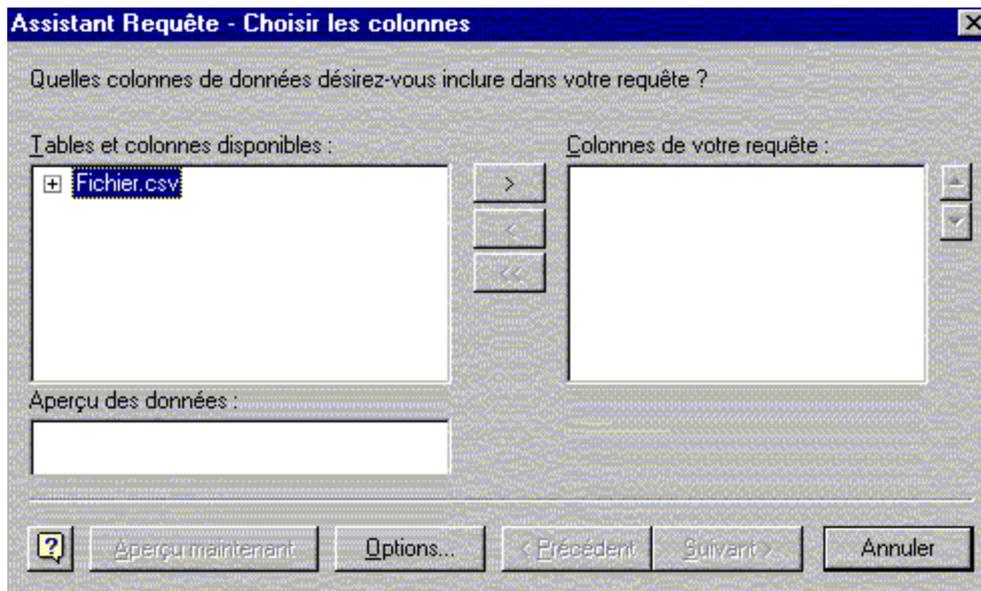
Entrons le nom retenu dans la première ligne puis sélectionnons l'option *Microsoft Text Driver (*.txt;*.csv)* comme suit :



Cliques sur le bouton *Connexion* puis décochons *Utilisation du répertoire en cours*
Cliques sur le bouton *Sélectionner le répertoire*, nous pouvons alors choisir d'abord le lecteur puis le répertoire.



Cliques sur *OK* puis à nouveau sur *OK* deux fois, nous voyons apparaître le cadre suivant :



Dans notre exemple le fichier *Fichier.csv* est le seul disponible mais ce n'est pas nécessairement le cas et il faut alors sélectionner le fichier que nous voulons lire.

Cliquons sur *Options*, apparaît alors un cadre dans lequel nous cliquons sur *Définir Format*.

Dans la liste des fichiers, sélectionnons le fichier CSV que nous voulons utiliser, ici *Fichier.csv*. Cochoons la case indiquant que le nom des colonnes se trouve dans la première ligne, puis dans *Format* sélectionnons *Personnalisé* et entrons le point-virgule comme séparateur. Dans la partie droite du cadre, cliquons sur *Deviner*. Nous voyons alors apparaître la liste des champs du fichier *Fichier.csv*. Pour chaque champ, nous pouvons définir son type et, pour les champs correspondant à du texte, sa longueur. Ensuite, il faut répondre OK à chaque cadre qui apparaît jusqu'à revenir à l'assistant Query pour choisir les colonnes. Cliquons sur la flèche pour sélectionner les champs du fichier CSV que nous voulons utiliser.

Cliquons alors sur *Suivant* nous voyons alors apparaître un cadre qui nous permet de filtrer les données. Si nous désirons conserver toutes les données nous cliquons sur *Suivant* puis à nouveau sur *Suivant* car trier préalablement les données n'a aucun intérêt dans un tableau croisé dynamique. On clique enfin sur *Terminer* puis sur *Suivant*. On arrive alors sur le cadre suivant :



En cliquant sur le bouton *Terminer*, on se retrouve dans le tableau croisé dynamique.

Fichiers DSN et schema.ini

En créant une connexion, Excel crée en fait deux fichiers texte, un fichier avec le nom de la connexion suivi de l'extension *.dsn* et le fichier *schema.ini*. Dans notre exemple, le fichier DSN s'appelle *UserEssais.dsn*, il est situé dans le répertoire C:\Users\AppData\Roaming\Microsoft\Queries. Il se présente comme suit :

```
[ODBC]
DBQ=C:\USERS\ESSAIS
DefaultDir=C:\USERS\ESSAIS
Driver={Microsoft Text Driver (*.txt; *.csv)}
DriverId=27
FIL=text
MaxBufferSize=2048
MaxScanRows=8
PageTimeout=5
SafeTransactions=0
Threads=3
UserCommitSync=Yes
```

Ce fichier indique que la connexion se fait avec un fichier texte, *DriverId=27* indique que le séparateur est le point-virgule, *DBQ* indique le nom du répertoire où se situe le fichier CSV que nous voulons lire. Il est possible de créer son propre fichier DSN avec un éditeur de texte, par exemple le bloc-notes de Windows, de l'adapter à ses besoins et de le placer là où on le souhaite.

Le fichier *schema.ini* se trouve, et doit impérativement se trouver, dans le même répertoire que les fichiers CSV. Il se présente sous la forme suivante :

```
[Fichier.csv]
ColNameHeader=True
Format=Delimited(;)
MaxScanRows=0
CharacterSet=ANSI
Col1=DA Char Width 255
Col2=OP Char Width 255
```

```
Col3=AE Char Width 255
Col4=VAL Float
[essai.csv]
ColNameHeader=True
Format=Delimited(;)
MaxScanRows=0
CharacterSet=ANSI
Col1=DA Char Width 255
Col2=SI Char Width 255
Col3=PR Char Width 255
Col4=VA Float
```

Dans cet exemple, nous avons la description de deux fichiers CSV. Tous les fichiers que nous voulons utiliser doivent impérativement se trouver dans le fichier schema.ini, les uns à la suite des autres, comme dans l'exemple. Les différents types de variables sont les suivants :

```
Char : caractère
Currency : monnaie
Date: date
Float : décimal double précision
Integer : entier long
LongChar : caractère long
Short : entier court
Single : décimal simple précision
```

Là encore, on peut créer soi-même son fichier schema.ini avec un éditeur de texte et le placer dans le même répertoire que les fichiers CSV. Créer soi-même les fichiers DSN et schem.ini peut être une bonne solution pour être certain de contrôler la connexion et la définition des champs des fichiers.

Créer un tableau croisé dynamique avec une connexion existante

Plaçons-nous donc dans une feuille Excel et dans le menu *Données* choisissons l'option *Rapport de tableau croisé dynamique*. Cochons alors *Source de données externe* comme ci-dessous :



Cliquons sur *Suivant* puis sur *Lire les données*. Apparaît alors un cadre qui montre la liste des noms de connexions créées avec l'assistant. Si l'on a créé soit-même un fichier DSN, il faut cliquer sur le bouton *Chercher* et naviguer dans les répertoires pour le trouver. En cliquant sur le nom de la connexion on fait apparaître le cadre de création d'un tableau croisé dynamique. En cliquant sur le nom du fichier DSN on voit apparaître la liste des fichiers contenus dans le répertoire où se trouvent les fichiers CSV. En cliquant sur l'un d'eux, on se retrouve dans le cadre de création d'un tableau croisé dynamique.

Quand on travaille avec de gros fichiers CSV il est intéressant, dans le tableau croisé dynamique, de décocher l'option *Enregistrer les données et la mise en forme*. En effet, lorsque cette option est sélectionnée les données sont enregistrées avec le classeur Excel, ce qui peut donner un classeur occupant une mémoire importante. Décocher cette option permet de rendre le classeur Excel indépendant des données du fichier CSV.

Travailler avec des requêtes SQL dans un tableau croisé dynamique

L'un des grands intérêts de l'utilisation des fichiers CSV avec les tableaux croisés dynamiques est d'ouvrir l'accès aux requêtes SQL et donc au monde des bases de données relationnelles. Supposons, par exemple, que nous cherchions à regrouper les activités du fichier *Fichier.csv* en utilisant la table de passage ci-dessous qui montre comment passer du niveau G de la nomenclature au niveau F :

	A	B
1	AEG	AEF
2	AEGA01	AEFA0
3	AEGA02	AEFA0
4	AEGA03	AEFA0
5	AEGB01	AEFB0
6	AEGB02	AEFB0
7		

Enregistrons cette table au format CSV dans le répertoire *Essais* sous le nom *TablePas.csv* puis fermons le classeur Excel. Ouvrons le fichier *schema.ini* avec un éditeur de texte et ajoutons la définition du nouveau fichier :

```
[TablePas.csv]
ColNameHeader=True
Format=Delimited(;)
MaxScanRows=0
CharacterSet=ANSI
Col1=AEG Char Width 20
Col2=AEF Char Width 20
```

Cliquons maintenant sur le tableau croisé dynamique créé avec le fichier *Fichier.csv*. Dans le menu, allons dans le groupe *Options* du tableau croisé dynamique et sélectionnons *Changer la source de données*, puis cliquons sur *Propriétés de connexion*. Dans le cadre qui apparaît, allons dans *Définition*. Nous arrivons alors au cadre qui nous permettra de travailler avec les deux fichiers CSV et cela de deux manières différentes.

La première manière consiste à modifier la requête SQL se trouvant dans le cadre *Commande*. On pourra ainsi écrire la requête SQL suivante :

```
SELECT f.DA, f.OP, f.AE, t.AEF, f.VAL
FROM Fichier.csv f, TablePas.csv t
WHERE f.AE=t.AEG
```

Après avoir cliqué sur *OK*, nous pouvons modifier le tableau croisé dynamique en y intégrant la dimension AEF.

La seconde méthode consiste à cliquer sur le bouton *QUERY* puis sur *OK*. Nous parvenons alors dans Microsoft Query. Dans le menu, cliquons sur *table* pour pouvoir ajouter le fichier *TablePas.csv* :

Lancer la requête à partir de UserEssais

DA	OP	AE	VAL	AEG
DA2000	OPB1	AEGA01	10050	AEGA01
DA2000	OPB1	AEGA01	10050	AEGA02
DA2000	OPB1	AEGA01	10050	AEGA03
DA2000	OPB1	AEGA01	10050	AEB01
DA2000	OPB1	AEGA01	10050	AEB02
DA2001	OPB1	AEGA01	56	AEGA01
DA2001	OPB1	AEGA01	56	AEGA02
DA2001	OPB1	AEGA01	56	AEGA03
DA2001	OPB1	AEGA01	56	AEB01
DA2001	OPB1	AEGA01	56	AEB02
DA2000	OPB1	AEGA02	100	AEGA01
DA2000	OPB1	AEGA02	100	AEGA02
DA2000	OPB1	AEGA02	100	AEGA03
DA2000	OPB1	AEGA02	100	AEB01
DA2000	OPB1	AEGA02	100	AEB02

Enreg: 1

Nous allons établir un lien entre le champ *AE* du fichier *Fichier* et le champ *AEG* du fichier *TablePas* car ces deux champs se correspondent. Pour cela, nous sélectionnons le champ *AEG* et nous le faisons glisser sur *AE* de manière à faire apparaître un lien entre les deux. Si nous cliquons sur l'icône *SQL*, nous accédons à la requête SQL que nous sommes en train de créer et que nous pouvons éventuellement modifier.

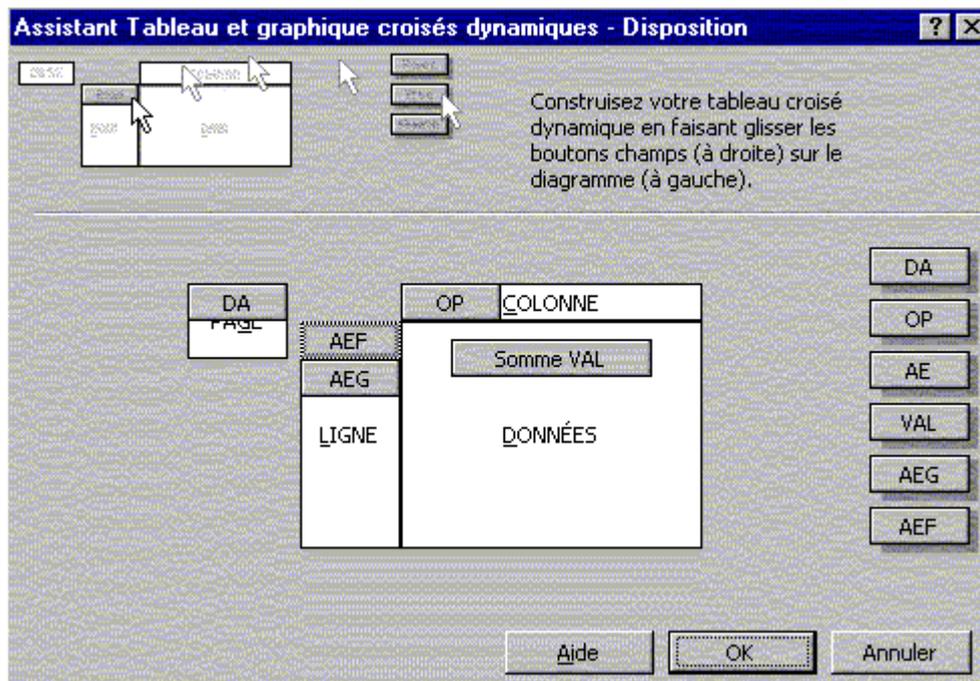
SQL

```

Instruction SQL :
SELECT Fichier.DA, Fichier.OP, Fichier.AE, Fichier.VAL,
TablePas.AEG, TablePas.AEF
FROM Fichier.csv Fichier, TablePas.csv TablePas
WHERE TablePas.AEG = Fichier.AE
  
```

OK
Annuler

En cliquant sur *OK* puis, dans le menu *Fichier* en sélectionnant *Renvoyer les données vers Microsoft Excel* puis *Suivant* nous retombons sur les menus des tableaux croisés dynamiques. Nous pouvons, par exemple, retenir la disposition suivante :



Nous obtenons le tableau ci-dessous qui nous donne une agrégation au niveau F de la nomenclature.

1	DA	(Tous)		
2				
3	Somme VAL		OP	
4	AEF	AEG	OPB1	Total
5	AEFA0	AEGA01	10106	10106
6		AEGA02	205	205
7		AEGA03	196	196
8	Somme AEFA0		10507	10507
9	AEFB0	AEGB01	62	62
10		AEGB02	60	60
11	Somme AEFB0		122	122
12	Total		10629	10629
13				

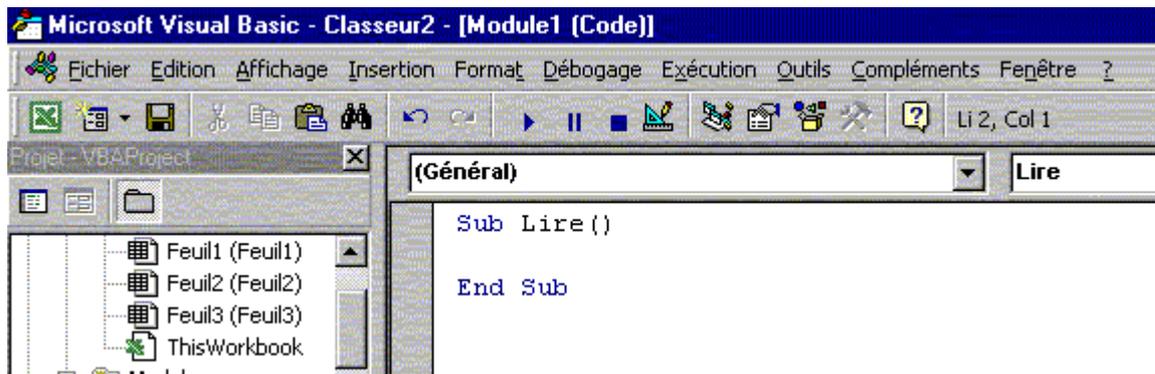
Nous pouvons maintenant travailler avec ce tableau dynamique en sélectionnant, par exemple, une année.

Travailler avec des requêtes SQL dans Visual Basic

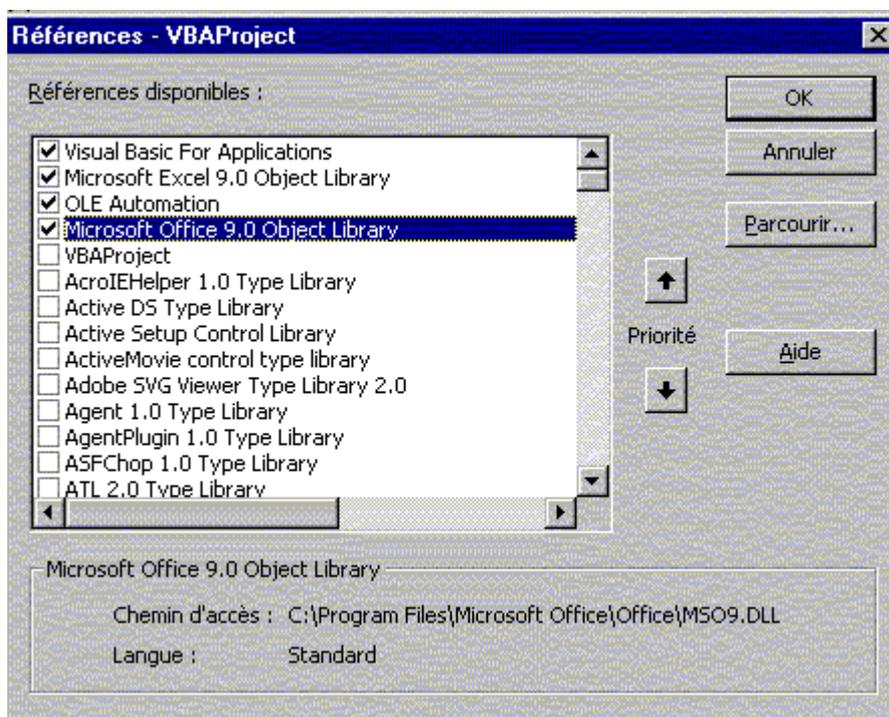
Il est possible de travailler sur des fichiers CSV sans passer par les tableaux croisés dynamiques. Pour cela, on peut utiliser les possibilités d'interrogation des bases de données que donne Visual Basic. Avant toute chose il convient de paramétrer Visual Basic. Si le groupe *Développeur* n'est pas disponible au menu, il faut aller dans *Fichiers*, puis dans *Options* et *Personnaliser*. Dans la partie droite du cadre, il faut alors cocher *Développeur* puis cliquer sur *OK*.

Supposons que nous voulions créer la macro *Lire* pour lire le fichier *Fichier.csv*. Dans le groupe *Développeur* cliquons sur *Macros*. Dans le cadre *Macro*, tapons le nom de la

macro puis cliquons sur *Créer*. Nous arrivons dans l'éditeur Visual Basic qui se présente ainsi :



Allons dans le menu *Outils* et choisissons l'option *Références*, le cadre ci-dessous s'ouvre alors :



Nous devons sélectionner dans la liste les éléments suivants :

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

Dans notre macro, nous devons définir la connexion à notre source de données et un objet Recordset, objet qui est destiné à recevoir les enregistrements générés par notre requête SQL. Nous allons donc définir deux variables, l'une destinée à définir la connexion et que nous appellerons, par exemple, *Connex*, et l'autre pour définir le recordset et que nous appellerons *Record*. Pour cela nous utiliserons les instructions suivantes :

```
Dim Connex As ADODB.Connection
Dim Record As ADODB.Recordset
```

```
Set Connex = New ADODB.Connection
Set Record = New ADODB.Recordset
```

Puisqu'une connexion se définit par le nom du répertoire où se trouvent les données, il est pratique de placer le nom de ce répertoire dans une variable que nous appellerons *RepertNom*. La connexion sera définie de la manière suivante :

```
Repertnom = "C:\User\Essais\"

Connex.ConnectionString = _
"ODBC;DBQ=" & Repertnom & ";DefaultDir=C:\;" & _
"Driver={Microsoft Text Driver (*.txt; *.csv)};" & _
"DriverId=27;Extensions=txt, csv, tab, asc;FIL=text;MaxBufferS"

Connex.Open
```

Le signe " _ " (espace puis espace souligné) est utilisé pour présenter sur plusieurs lignes une instruction trop longue pour tenir sur une seule, son utilisation est facultative et ne sert qu'à améliorer la lisibilité du programme. On utilise *ConnectionString* pour définir le répertoire et le type de base de données utilisé, ici des bases CSV. L'instruction *Open* permet d'ouvrir la base de données afin de permettre son utilisation. Il nous faut maintenant écrire la requête SQL qui nous permettra de lire le fichier CSV. Pour des raisons de lisibilité nous allons décomposer la requête en ses différentes composantes que nous allons placer dans des variables. Nous allons ensuite la lancer de la manière suivante :

```
sel = " SELECT f.DA,f.AE,f.OP,f.VAL "
fro = " FROM fichier.csv f "
Record.Open sel & fro, Connex
```

L'instruction *Record.Open* ouvre le recordset *Record* et exécute la requête placée dans la chaîne de caractères définie par la concaténation des variables *sel* et *fro*, la base de données utilisée étant celle définie par la connexion *Connex*. Remarquons que nous avons laissé des espaces à la fin des variables *sel* et *fro* afin que la concaténation n'agrège pas des instructions. Le résultat de la requête, c'est-à-dire les enregistrements recherchés, a été placé dans l'objet recordset *Record* que nous devons lire. Supposons que nous cherchions à afficher le résultat dans la feuille *Données*, nous lirons le recordset de la manière suivante :

```
Set f = Sheets("Données")
Record.MoveFirst
i = 1
Do While Not Record.EOF
    f.Cells(i, 1) = Record("DA")
    f.Cells(i, 2) = Record("AE")
    f.Cells(i, 3) = Record("OP")
    f.Cells(i, 4) = Record("VAL")
    i = i + 1
    Record.MoveNext
```

Loop
Record.Close
Connex.Close

Pour lire le recorset on se place d'abord sur le premier enregistrement en utilisant l'instruction *MoveFirst* puis on fait une boucle qui va lire les enregistrements suivants jusqu'à la fin qui est signalée par *Record.EOF* (EOF voulant dire End of file). *Record("DA")* renvoie la valeur du champ *DA*, *Record.MoveNext* permet de passer à l'enregistrement suivant, *Loop* marque la fin de la boucle, *Record.Close* ferme le recordset et *Connex.Close* ferme la connexion. Si on lance ce programme on obtient sur la feuille *Données* les enregistrements de *Fichier.csv*. Notons que les intitulés des champs n'apparaissent pas car ils ne font pas partie des enregistrements.

	A	B	C	D
1	DA2000	AEGA01	OPB1	10050
2	DA2001	AEGA01	OPB1	56
3	DA2000	AEGA02	OPB1	100
4	DA2001	AEGA02	OPB1	105
5	DA2000	AEGA03	OPB1	151
6	DA2001	AEGA03	OPB1	45
7	DA2000	AEGB01	OPB1	28
8	DA2001	AEGB01	OPB1	34
9	DA2000	AEGB02	OPB1	32
10	DA2001	AEGB02	OPB1	28
11				

Le programme complet est le suivant :

```
Sub Lire()  
Dim Connex As ADODB.Connection  
Dim Record As ADODB.Recordset  
  
Set Connex = New ADODB.Connection  
Set Record = New ADODB.Recordset  
  
Repertnom = "C:\User\Essais\  
  
Connex.ConnectionString = _  
"ODBC;DBQ=" & Repertnom & ";DefaultDir=C\;" & _  
"Driver={Microsoft Text Driver (*.txt; *.csv)};" & _  
"DriverId=27;Extensions=txt,csv,tab,asc;FIL=text;MaxBufferS"  
  
Connex.Open  
sel = " SELECT f.DA,f.AE,f.OP,f.VAL "  
fro = " FROM fichier.csv f "  
Record.Open sel & fro, Connex  
  
Set f = Sheets("Données")  
Record.MoveFirst  
i = 1  
Do While Not Record.EOF  
    f.Cells(i, 1) = Record("DA")  
    f.Cells(i, 2) = Record("AE")
```

```

f.Cells(i, 3) = Record("OP")
f.Cells(i, 4) = Record("VAL")
i = i + 1
Record.MoveNext
Loop
Record.Close
Connex.Close

End Sub

```

Présentations personnalisées

On peut également faire une présentation plus sophistiquée. Par exemple, supposons que nous voulions remplir le tableau suivant avec les résultats de la requête :

	2000	2001
A01		
A02		
A03		
B01		
B02		

Nous pouvons commencer par nommer les cellules correspondant aux titres en lignes et en colonnes avec les noms des champs correspondants :

		DA2000	DA2001
		2000	2001
AEGA01	A01		
AEGA02	A02		
AEGA03	A03		
AEGB01	B01		
AEGB02	B02		

Pour cela, après avoir sélectionné les bonnes références dans le menu *Outils* de Visual Basic, on peut écrire le programme suivant :

```

Sub Lire()
'Nécessite d'avoir sélectionné dans Tools References les options suivantes :
'Microsoft ActiveX DataObjects 6.1 Library
'Microsoft ActiveX DataObjects Recordset 6.0 Library
'Microsoft ADO Ext. 6.0 for DDL and Security

Dim Connex As ADODB.Connection
Dim Record As ADODB.Recordset
Dim cmdCommand As ADODB.Command

Rep = "C:\User\Essais\" ' Nom du répertoire où se trouve la base de données

Set f = Workbooks("PrésentationFichierCSV.xlsm").Sheets("Présentation")

Application.ScreenUpdating = False

```

```

f.Range("Résultats").ClearContents
' Ouvre la connexion
Set Connex = New ADODB.Connection
Set Record = New ADODB.Recordset

Connex.ConnectionString = _
"ODBC;DBQ=" & Rep & ";DefaultDir=C:\; " & _
"Driver={Microsoft Text Driver (*.txt; *.csv)}; " & _
"DriverId=27;Extensions=txt,csv,tab,asc;FIL=text;MaxBufferS"

Connex.Open
sel = " SELECT f.DA,f.AE,f.OP,f.VAL "
fro = " FROM fichier.csv f "
Record.Open sel & fro, Connex

Record.MoveFirst

Do While Not Record.EOF
    da = Record("DA") & "_"
    ae = Record("AE")
    op = Record("OP")
    va = Record("VAL")
    l = Range(ae).Row
    c = Range(da).Column
    f.Cells(l, c) = va
    Record.MoveNext
Loop

Record.Close

Connex.Close

Set cmdCommand = Nothing

Application.ScreenUpdating = True

End Sub

```

Nous obtenons :

		DA2000	DA2001
		2000	2001
AEGA01	A01	50	56
AEGA02	A02	100	105
AEGA03	A03	151	45
AEGB01	B01	28	34
AEGB02	B02	32	28

Dans le programme, la boucle permet de lire les données du fichier *Fichier.csv* et de les placer dans le tableau de présentation. La variable *da* reprend le contenu du champ DA et ajoute "_" parce que le nom "DA2000" n'est pas un nom acceptable dans Excel car il peut être confondu avec la référence d'une cellule, il a été remplacé par "DA2000_".

La variable l reprend le numéro de ligne de la plage dont le nom est contenu dans la variable ae, la variable c reprend le numéro de colonne de la plage dont le nom est contenu dans la variable da, la valeur est placée dans la cellule définie par la ligne l et la colonne c.

Notons dans ce programme que la plage contenant les résultats a été effacée au préalable par l'instruction ClearContents. Notons également que l'instruction Application.ScreenUpdating a été utilisée pour bloquer l'affichage pendant l'exécution du programme afin de ne pas la ralentir par des mises à jour d'écran répétées.

Générer un fichier CSV à partir d'une requête SQL

Il est possible avec Visual Basic de générer un fichier CSV à partir d'une requête SQL. Il suffit pour cela de décrire au préalable dans *schema.ini* tous les fichiers utilisés.

Supposons que nous cherchions à créer à partir du fichier *Fichier.csv* un fichier CSV qui ne contienne que les valeurs agrégées au niveau F. Le programme se présentera alors de la manière suivante :

```
Sub Agreger()  
'Nécessite d'avoir sélectionné dans Tools References les options suivantes :  
'Microsoft ActiveX DataObjects 6.1 Library  
'Microsoft ActiveX DataObjects Recordset 6.0 Library  
'Microsoft ADO Ext. 6.0 for DDL and Security  
  
Dim Connex As ADODB.Connection  
Dim cmdCommand As ADODB.Command  
Dim Ctexte As String * 990  
  
Set Connex = New ADODB.Connection  
  
Rep = "C:\User\Essais" ' Nom du répertoire où se trouve la base de données  
  
With Connex  
    .ConnectionString = _  
    "ODBC;DBQ=" & Rep & ";DefaultDir=C:\; " & _  
    "Driver={Microsoft Text Driver (*.txt; *.csv)}; " & _  
    "DriverId=27;Extensions=txt,csv,tab,asc;FIL=text;MaxBufferS"  
End With  
Connex.Open  
  
Open Rep & "\Agreg.csv" For Output As #1  
Print #1, "DA;AEF;OP;VAL"  
Close #1  
  
sel = " SELECT f.DA,t.AEF,f.OP,sum(f.VAL) as VAL "  
fro = " FROM fichier.csv f, TablePas.csv t "  
whe = " WHERE f.AE=t.AEG "  
gro = " GROUP BY f.DA,t.AEF,f.OP "  
Ctexte = " INSERT INTO Agreg.csv " & sel & fro & whe & gro  
  
Set cmdCommand = New ADODB.Command  
Set cmdCommand.ActiveConnection = Connex
```

```

With cmdCommand
    .CommandText = Ctexte
    .CommandType = adCmdText
    .Execute
End With

Connex.Close
Set cmdCommand = Nothing

End Sub

```

Une requête SQL ne peut générer un fichier CSV que dans le répertoire défini par la connexion. Pour générer un fichier CSV en dehors de ce répertoire, on peut combiner deux techniques, la première consiste à placer le résultat de la requête dans un recordset, la seconde consiste à créer le fichier CSV en lisant le recordset.

Supposons, comme précédemment, que nous cherchions à créer à partir du fichier *Fichier.csv* un fichier CSV qui ne contienne que les valeurs agrégées au niveau F. Le programme se présentera alors de la manière suivante :

```

Sub Lire()
Dim Connex As ADODB.Connection
Dim Record As ADODB.Recordset

Set Connex = New ADODB.Connection
Set Record = New ADODB.Recordset

Repertnom = "C:\User\Essais\"

Connex.ConnectionString = _
"ODBC;DBQ=" & Repertnom & ";DefaultDir=C:\; " & _
"Driver={Microsoft Text Driver (*.txt; *.csv)}; " & _
"DriverId=27;Extensions=txt,csv,tab,asc;FIL=text;MaxBufferS"

Connex.Open
sel = " SELECT f.DA,t.AEF,f.OP,sum(f.VAL) as VAL "
fro = " FROM fichier.csv f, TablePas.csv t "
whe = " WHERE f.AE=t.AEG "
gro = " GROUP BY f.DA,t.AEF,f.OP "
Record.Open sel & fro & whe & gro, Connex

Open "C:\User\Essais1\Agreg.csv" For Output As #1
Print #1, "DA;AE;OP;VAL"
Set f = Sheets("Données")
Record.MoveFirst
Do While Not Record.EOF
    da = Record("DA")
    aef = Record("AEF")
    op = Record("OP")
    va = Record("VAL")
    enr = da & ";" & aef & ";" & op & ";" & va
    If va <> 0 Then Print #1, enr
    Record.MoveNext

```

```

Loop
Record.Close
Connex.Close
Close #1

End Sub

```

Nous obtenons alors le fichier suivant :

	1	2	3	4
1	DA	AE	OP	VAL
2	DA2000	AEFA0	OPB1	10301
3	DA2000	AEFB0	OPB1	60
4	DA2001	AEFA0	OPB1	206
5	DA2001	AEFB0	OPB1	62
6				
7				

Travailler avec des bases Access

Il est également possible de travailler avec de véritables bases de données, par exemple de type Access. Pour cela, le mieux est certainement de créer la base de données et ses tables en utilisant le logiciel Access puis de travailler avec elles à partir d'Excel. La manière de travailler sera, toutefois, légèrement différente de celle que nous avons présentée pour les fichiers CSV. L'exemple ci-dessous correspond au chargement d'une table Access structurée en hypercube à partir d'un tableau Excel.

```

Sub SaisieQuest()
'Suppose avoir sélectionné dans Tools/References :
'Microsoft ActiveX Data Objects 2.8 Library
'Microsoft ActiveX Data Object Recordsets 2.8 Library
'Microsoft ADO Ext for DDL and Security

Dim Ctexte As String * 300
Dim cnnConn As ADODB.Connection
Dim rstRecordset As ADODB.Recordset
Dim cmdCommand As ADODB.Command

Rep = "H:\MyDocuments\QuestGNI" ' Nom du répertoire où se trouve la base de données
NomBase = "questGNI.mdb" ' Nom de la base de données

' Ouvre la connexion.
Set cnnConn = New ADODB.Connection
With cnnConn
.ConnectionString = _
"Provider=Microsoft.Jet.OLEDB.4.0"
.Open Rep & NomBase
End With

Set cmdCommand = New ADODB.Command
Set cmdCommand.ActiveConnection = cnnConn
Set rstRecordset = New ADODB.Recordset

```

```

-----
anq = Sheets("Saisie").Cells(3, 2) ' Année du questionnaire
RepD = "O:\CN\GNP-GNI questionnaire " & anq & "\received" ' Nom du répertoire de données

For Each pays In Array("Pays1", "Pays2", "Pays3") ' Liste des pays
For k = 8 To 38
fic = RepD & "q" & anq & pays & ".xls" ' Nom du fichier du questionnaire
Workbooks.Open Filename:=fic ' Ouvre le questionnaire

-----

' Commence par vider la table QDATA avant d'ajouter les nouveaux enregistrements
' La table est vidée pour le pays et l'année saisies
With cmdCommand
.CommandText = "delete from QDATA where PAYS='" & pays & "' and ANQ='" & anq & "'"
.CommandType = adCmdText
.Execute
End With

Set f = Sheets("Quest" & anq & " (" & pays & ")") ' Feuille des données de l'année du questionnaire
For i = 7 To 44
coli = f.Cells(i, 1)
If coli <> 0 Then
coesa = f.Cells(i, 3)
For j = 4 To 9 anda = Mid(f.Cells(4, j), 1, 4) 'Saisie des années de la série avec limitation de l'année
à 4 caractères
va = Val(" " & f.Cells(i, j)) 'Saisie des valeurs avec élimination des espaces entrés comme des 0
Ctexte = "insert into QDATA values('" & pays & "','" & anq & "','" & anda & "','" & coli &
',' & coesa & "','" & va & ")"
If va <> 0 Then
With cmdCommand
.CommandText = Ctexte
.CommandType = adCmdText
.Execute
End With
End If
Next j
End If
Next i
ActiveWindow.Close ' Ferme le questionnaire

'Next pays
' Ferme la connexion et vide les variables
cnnConn.Close
Set cmdCommand = Nothing
Set rstRecordset = Nothing
Set cnnConn = Nothing

End Sub

```

Ce texte n'engage que son auteur : Francis Malherbe

