Communication entre Access et Excel

par Charles A.

Date de publication : 01/09/2005

Dernière mise à jour : 01/09/2005

Comment faire communiquer Access et Excel

I - Introduction

Microsoft Access est intégré au pack de bureautique Microsoft Office.

Microsoft Excel est le tableur le plus utilisé du marché, il ne requiert pas de compétences de programmation et est d'une souplesse incomparable en matière de graphiques et de présentations.

C'est pourquoi, les applications Access les plus intelligemment conçues ont parfois recours à des échanges avec Excel.

Dans ce tutorial, nous allons aborder les différentes méthodes de communications entre les deux produits et étudier à quels cas de figures elles s'appliquent le mieux.

II - Présentation

Notre propos ici n'est pas de traiter des importations des fichiers Excel, mais bien des exportations.

Nous partons du principe que nous disposons d'une robuste application Access offrant toutes les fonctionnalités attendues.

Nous constatons cependant certaines limites inhérentes à Access parmi d'autres :

. manque de souplesse des états

. impossibilité de modifier la présentation ou les données dans un état ou un formulaire

. impossibilité d'effectuer des calculs à la marge

. non disponibilité d'Access sur tous les postes d'une entreprise

. diffusions par mail trop rigides avec Access

. manque d'ergonomie des moyens d'accès au moteur graphique d'Access

. impossibilité de "toucher", "jouer", "simuler" avec les données comme dans Excel

Tous ces points mettent en exergue le besoin de communication entre le SGBDR et le Tableur.

Voici les différentes méthodes :

. le copier/coller : l'ouverture d'une table, requête ou formulaire en mode feuille de données autorise la copie dans Excel. Nous ne détaillerons pas ici cette méthode de part sa trivialité.

. Copie (OutputTo) : Un objet d'Access est copié dans un format étranger, dont Excel

. Export simple (TransferSpreadsheet) : Une requête ou une table est envoyée dans .XLS

. Automation : Access prend le contrôle d'une instance d'Excel et manipule des objets du tableur.

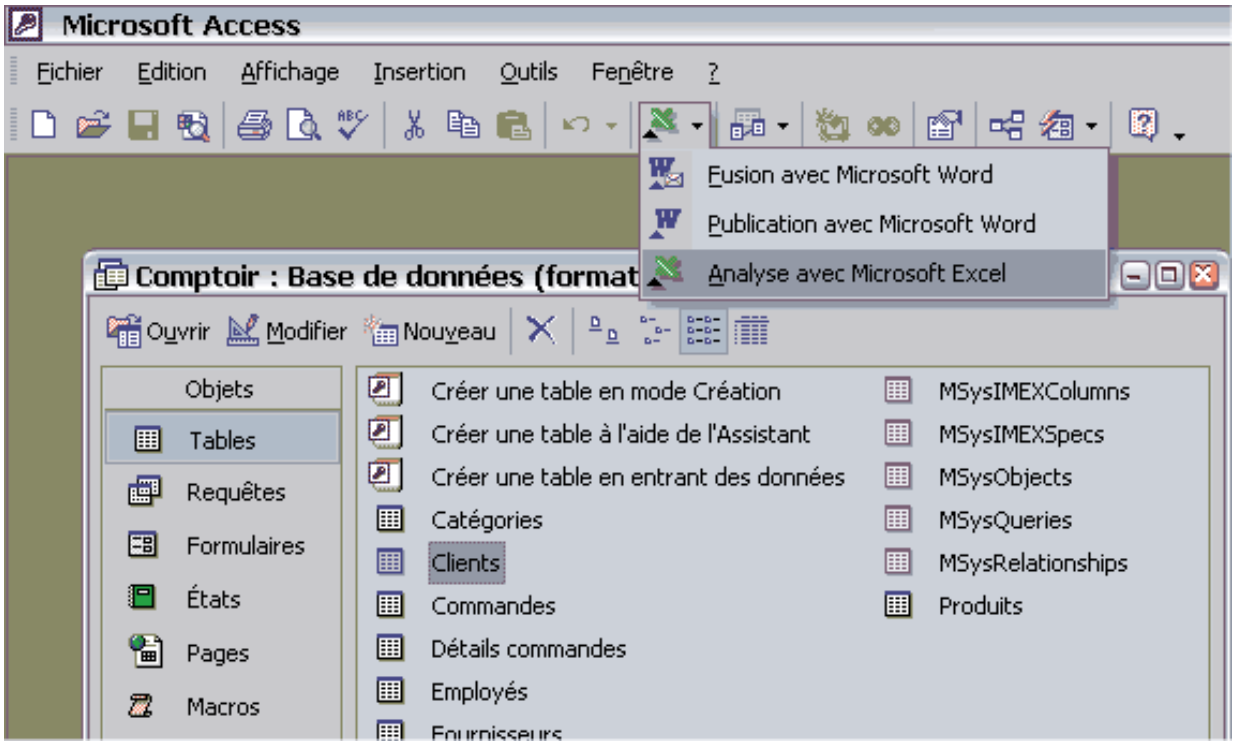
. Données externes : grâce à Microsoft Query, Excel se connecte à la base et rapatrie les données

. Plate-Forme ADO : Excel via VBA se connecte en ADO à une base.

Le but de cette démarche est, au delà, d'une simple énumération des méthodes, une analyse des forces et faiblesses de chacune, ce qui permet à terme de savoir vers laquelle s'orienter en fonction des besoins.

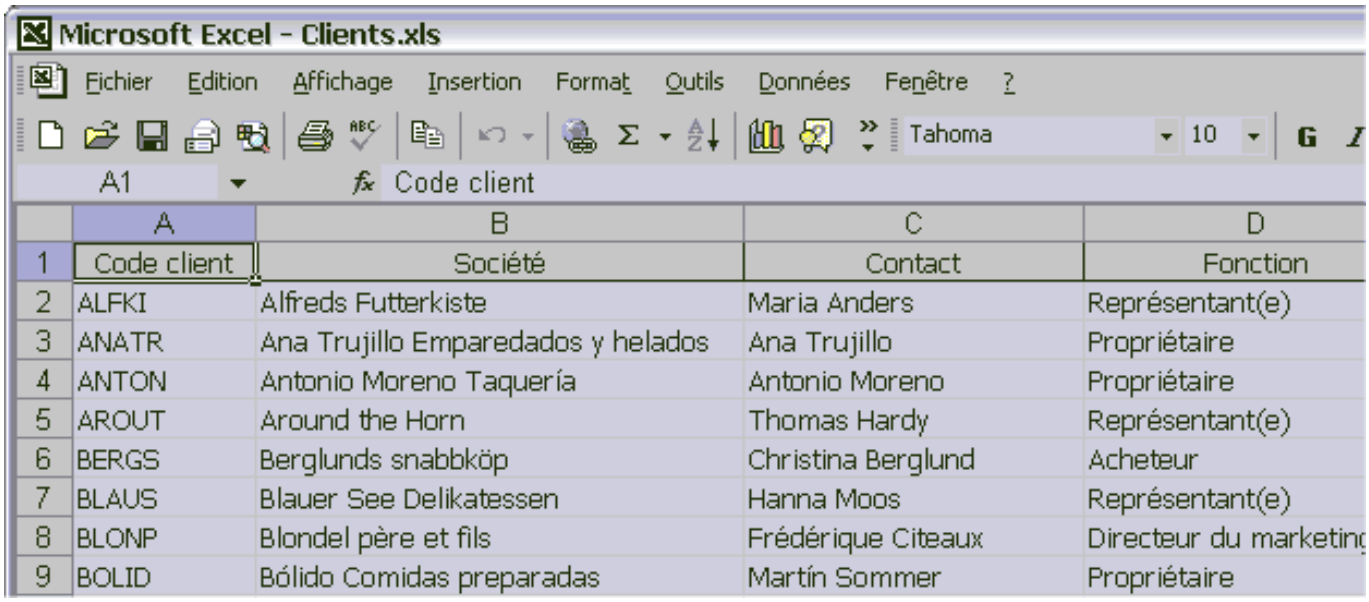
III - La copie

III-A - Barre d'outil



La méthode est de sélectionner un objet dans la fenêtre de base de données et de cliquer sur l'export Excel.

Le résultat est le suivant :



Microsoft Excel - Clients.xls

Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?

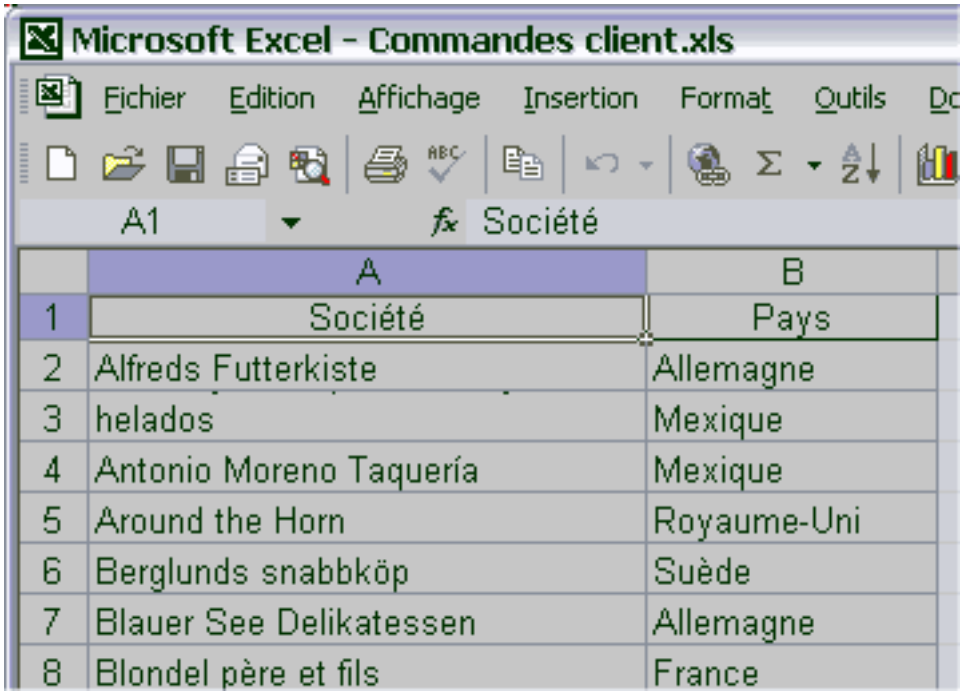
Tahoma 10 G I

A1 Code client

	A	B	C	D
1	Code client	Société	Contact	Fonction
2	ALFKI	Alfreds Futterkiste	Maria Anders	Représentant(e)
3	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Propriétaire
4	ANTON	Antonio Moreno Taquería	Antonio Moreno	Propriétaire
5	AROUT	Around the Horn	Thomas Hardy	Représentant(e)
6	BERGS	Berglunds snabbköp	Christina Berglund	Acheteur
7	BLAUS	Blauer See Delikatessen	Hanna Moos	Représentant(e)
8	BLONP	Blondel père et fils	Frédérique Citeaux	Directeur du marketing
9	BOLID	Bólido Comidas preparadas	Martín Sommer	Propriétaire

En exportant un formulaire, nous obtenons le recordset qui sous-tend ce formulaire, nous n'obtenons en revanche rien des sous formulaires.

Voici l'exportation du formulaire : "Commandes client".



Microsoft Excel - Commandes client.xls

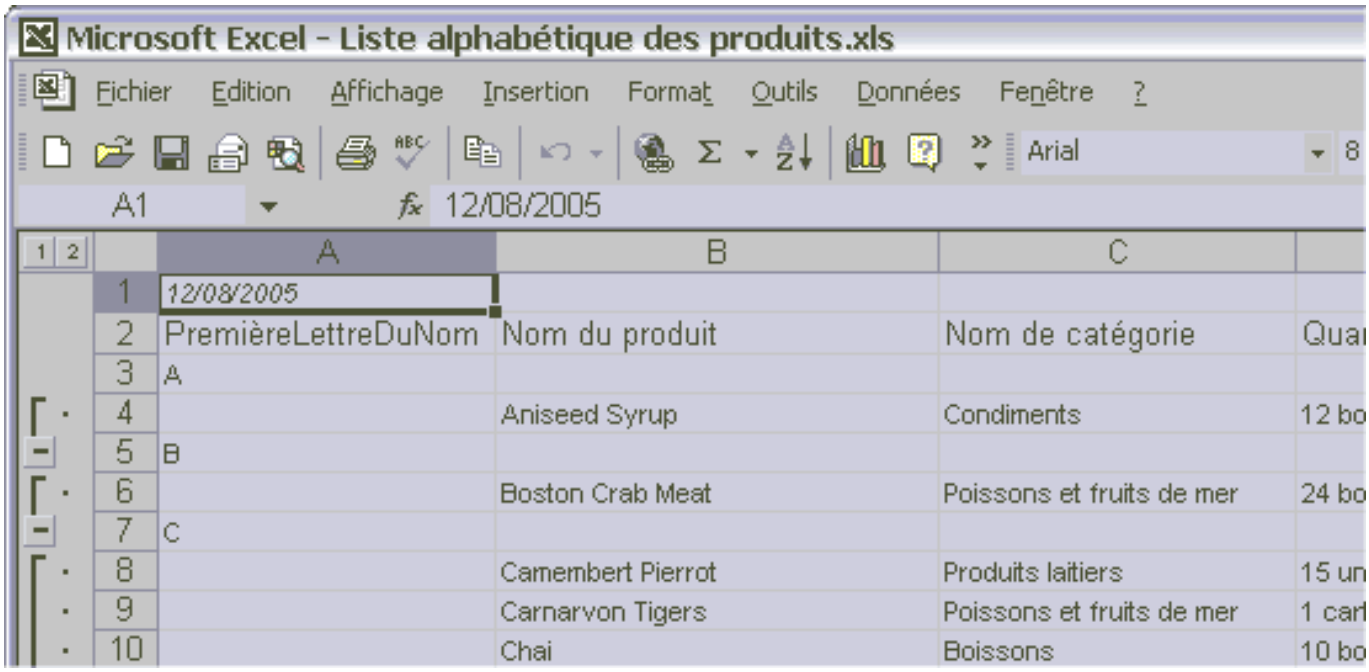
Fichier Edition Affichage Insertion Format Outils Do

A1 Société

	A	B
1	Société	Pays
2	Alfreds Futterkiste	Allemagne
3	helados	Mexique
4	Antonio Moreno Taquería	Mexique
5	Around the Horn	Royaume-Uni
6	Berglunds snabbköp	Suède
7	Blauer See Delikatessen	Allemagne
8	Blondel père et fils	France

L'exportation d'un état tente de restituer au mieux la disposition de l'état, notamment en adoptant un plan automatique en adéquation avec les regroupements créés dans l'état.

Les enrichissements de texte, de disposition et de mise en page sont bien sûr absents de la partie.



III-B - Code

J'écarte d'office les macros qui sont à mon avis à proscrire dans un développement d'une application Access, pour des raisons qui ont déjà été maintes fois abordées sur le forum.

La commande de code VBA est DoCmd.OutpuTo, dont voici la syntaxe :

expression.OutputTo (ObjectType [Entier constante d'objet Access],

ObjectName [Variant nom de l'objet à copier],

OutputFormat [Variant constante de format d'export],

OutputFile [Variant chaine désignant le nom du fichier],

AutoStart [Variant de lancement de l'application associée],

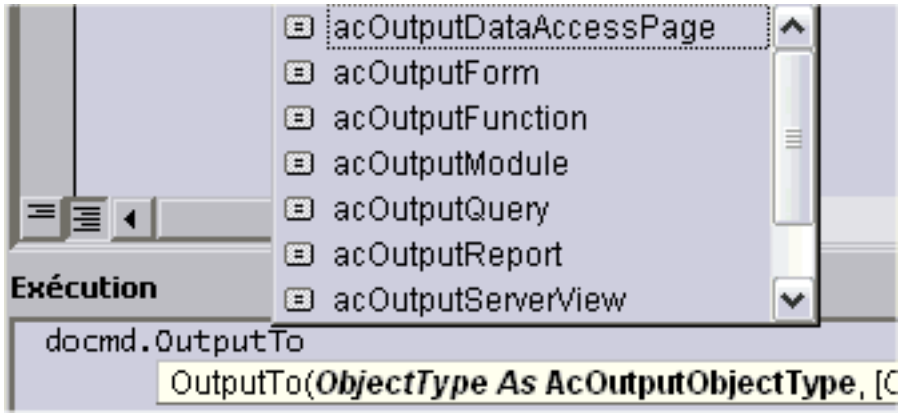
TemplateFile [Variant chemin d'accès du modèle, uniquement HTML, HTX ou ASP],

Encoding [Variant type d'encodage de jeu de caractères])

Un exemple de fonction de copie

```
Function ExportExcel()

DoCmd.OutputTo acOutputForm, _
                "Clients", _
                acSpreadsheetTypeExcel9, _
                "Clients.xls", _
                True
End Function
```

Ne nous inquiétons pas des Constantes : acOutputForm ou acSpreadsheetTypeExcel9, leur saisie est facilitée :



III-C - Avantages et Limites

Elle requiert que le classeur Excel soit fermé, si ce dernier n'existe pas, elle le crée.

Les avantages de cette méthode :

. rapidité : pas besoin de programmation

. accessibilité : le bouton dans la barre d'outil est extrêmement pratique, même le plus débutant est capable d'exporter une table vers Excel.

. facilité de programmation : une seule ligne de code !

Répertorions maintenant ce que l'export excel simple ne permet pas de faire :

. Impossible de préciser un nom d'onglet, ou une plage de cellules.

. Aucun formatage n'est possible

. Structure tabulaire, sauf dans l'export d'un état : pas de possiblité de découper des zones

IV - L'export simple

Cette fois, la commande n'est pas accessible par les menus, ni les barres d'outils.

IV-A - Le code

La commande est spécifique cette fois aux tableurs :

La commande de code VBA est DoCmd.TransferSpreadsheet, dont voici la syntaxe :

expression.TransferSpreadsheet (TypeTransfert [Entier constante d'objet exporté],

TypeFeuille [Entier constante de version excel],

NomTable [String désignant l'objet],

NomFichier [String Nom du fichier Excel],

ContientNomsChamps [Booléen pour l'affichage du nom des champs],

Étendue [!! String non disponible en export !!],

UtiliserOA [Variant non documenté])

Fonction de d'export simple

```
Function TransfertExportExcel()

DoCmd.TransferSpreadsheet acExport, _
                    acSpreadsheetTypeExcel9, _
                    "Clients", _
                    "Clients.xls", _
                    True
End Function
```

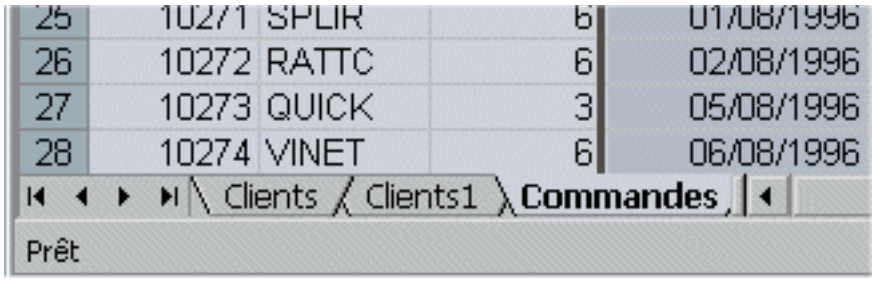
IV-B - Avantages et Limites

Avantages :

. facilité de programmation : une seule ligne de code

. création d'un nouvel onglet : en pointant plusieurs fois sur le fichier Excel, la méthode TransferSpreadsheet crée un nouvel onglet pour chaque export

. rapidité de génération



Limites :

. Impossible de préciser un nom d'onglet, ou une plage de cellules.

. Aucun formatage n'est possible

. Structure tabulaire, sauf dans l'export d'un état : pas de possiblité de découper des zones

Nota Bene : Les données de type texte sont exportées avec un ' en premier caractère à la différence de l'export par copie.

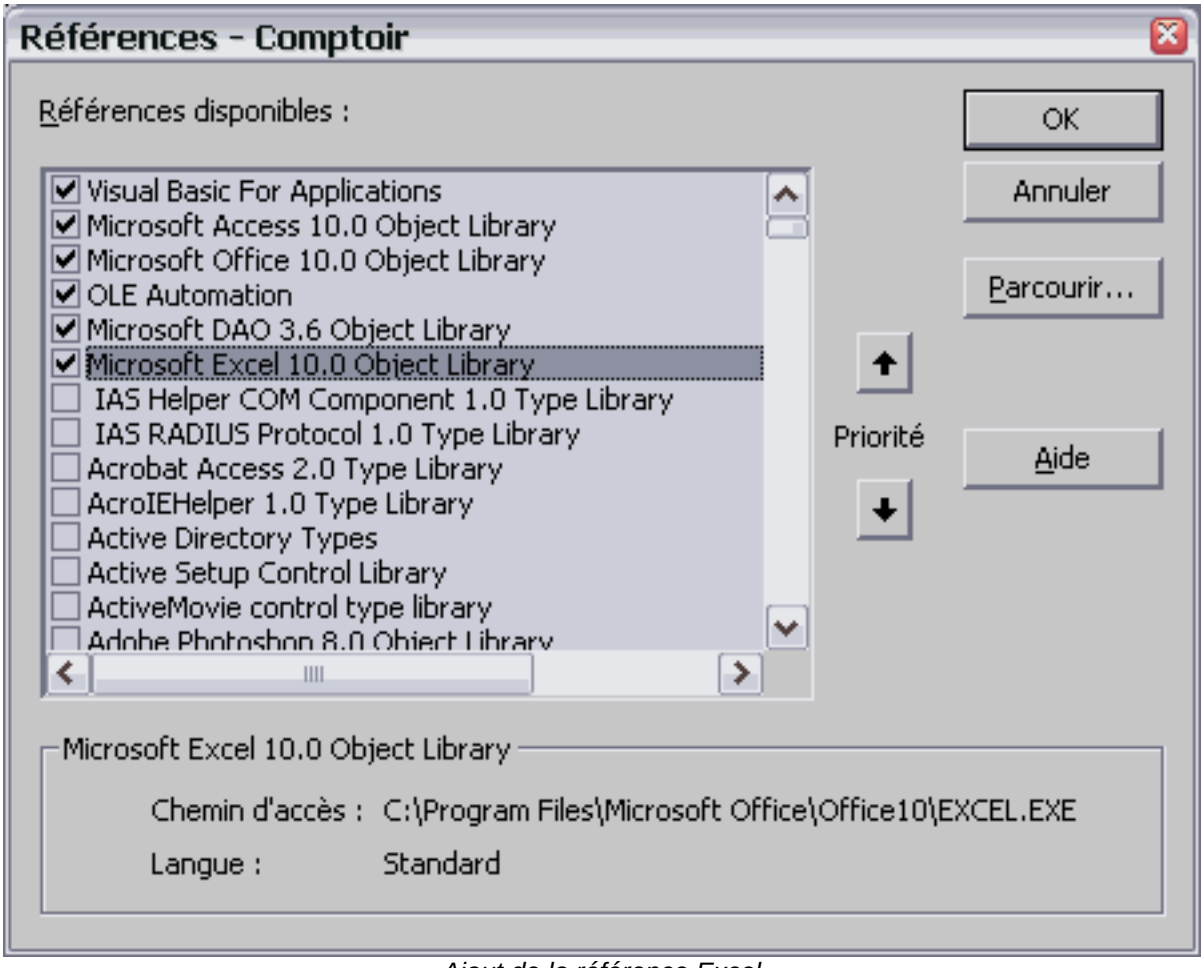
V - Automation

Pour plus d'informations je vous invite à consulter les questions de la FAQ qui traitent d'Excel.

Il est également intéressant de consulter cette source de Tofalu.

V-A - Le code

Pour pouvoir faire fonctionner Excel depuis Access, il faut intégrer à l'application la bibliothèque des objets du Tableur.



Ajout de la référence Excel

Fonction d'export via Automation

```
Function TransfertExcelAutomation()

    Dim xlApp As Excel.Application
    Dim xlSheet As Excel.Worksheet
    Dim xlBook As Excel.Workbook
```

Fonction d'export via Automation

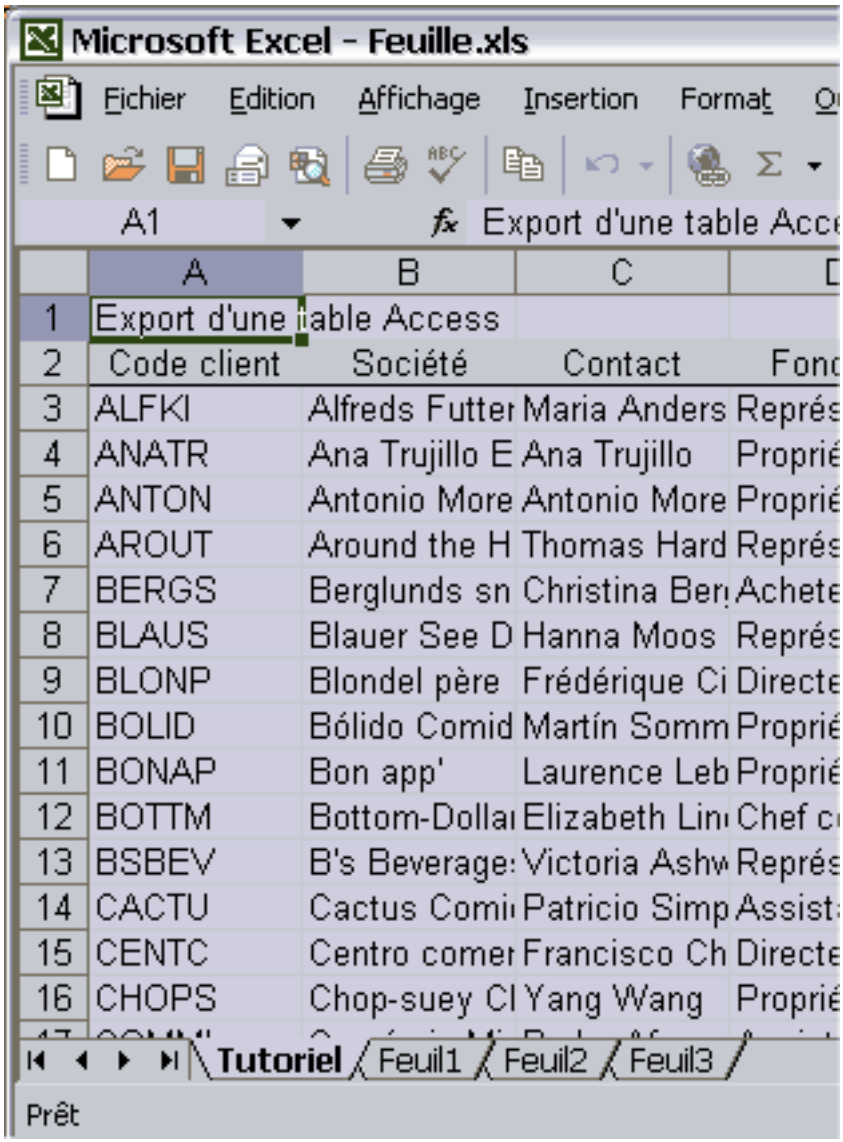
```
        Dim I As Long, J As Long
        Dim t0 As Long, t1 As Long

        t0 = Timer
        Dim rec As Recordset

        Set rec = CurrentDb.OpenRecordset("Clients", dbOpenSnapshot)

        'Initialisations
        Set xlApp = CreateObject("Excel.Application")
        Set xlBook = xlApp.Workbooks.Add

        'Ajouter une feuille de calcul
        Set xlSheet = xlBook.Worksheets.Add
        xlSheet.Name = "Tutoriel"

        ' le titre
        '  écriture dans la cellule de ligne 1 et de colonne 1
        xlSheet.Cells(1, 1) = "Export d'une table Access"


        ' les entetes
        '  .Fields(Index).Name renvoie le nom du champ
        For J = 0 To rec.Fields.Count - 1
            xlSheet.Cells(2, J + 1) = rec.Fields(J).Name
            ' Nous appliquons des enrichissements de format aux cellules
            With xlSheet.Cells(2, J + 1)
                .Interior.ColorIndex = 15
                .Interior.Pattern = xlSolid
                .Borders(xlEdgeBottom).LineStyle = xlContinuous
                .Borders(xlEdgeBottom).Weight = xlThin
                .Borders(xlEdgeBottom).ColorIndex = xlAutomatic
                .HorizontalAlignment = xlCenter
            End With
        Next J

        ' recopie des données à partir de la ligne 3
        I = 3
        Do While Not rec.EOF
            For J = 0 To rec.Fields.Count - 1
                ' .Fields(Index).Type renvoie le type du champ
                '   si c'est un Texte (dbText) nous insérons "'" pour
                '   qu'il soit reconnu par Excel comme du Texte
                If rec.Fields(J).Type = dbText Then
                    xlSheet.Cells(I, J + 1) = "'" & rec.Fields(J)
                Else
                    xlSheet.Cells(I, J + 1) = rec.Fields(J)
                End If
            Next J
            I = I + 1
            rec.MoveNext
        Loop

        ' code de fermeture et libération des objets
        xlBook.SaveAs "D:\Temp\Feuille.xls"
        xlApp.Quit
        rec.Close
        Set rec = Nothing
        Set xlSheet = Nothing
        Set xlBook = Nothing
        Set xlApp = Nothing

        t1 = Timer
        Debug.Print I & " enregistrements", Format(t1 - t0, "0") & " secondes"

    End Function
```

Nous obtenons en fenêtre immédiate (Ctrl + G)

Exécution

```
    TransfertExcelAutomation
     94 enregistrements          1 secondes
```

Et voici le résultat obtenu :



V-B - Avantages et Limites

L'implémentation de l'automation présente des avantages certains :

. Meilleur contrôle de l'export

. Contrôle sur les plages de cellules et la mise en forme

. Possibilité d'intégrer du VBA Excel grâce à la bibliothèque de références.

Elle a revanche un certain nombre de désavantages :

. Difficulté de programmation : sans que le codage soit complexe, il est bien entendu plus ardu que les fonctions vues précédement.

. Faire attention à la gestion des instances Excel

. Relative lenteur de l'export : plus lent que TransferSpreadsheet ou OutpuTo.

. Problèmes d'évolutivité : si le but est d'entrer des valeurs dans des cellules précises, il faut revoir le code en cas d'insertion de ligne.

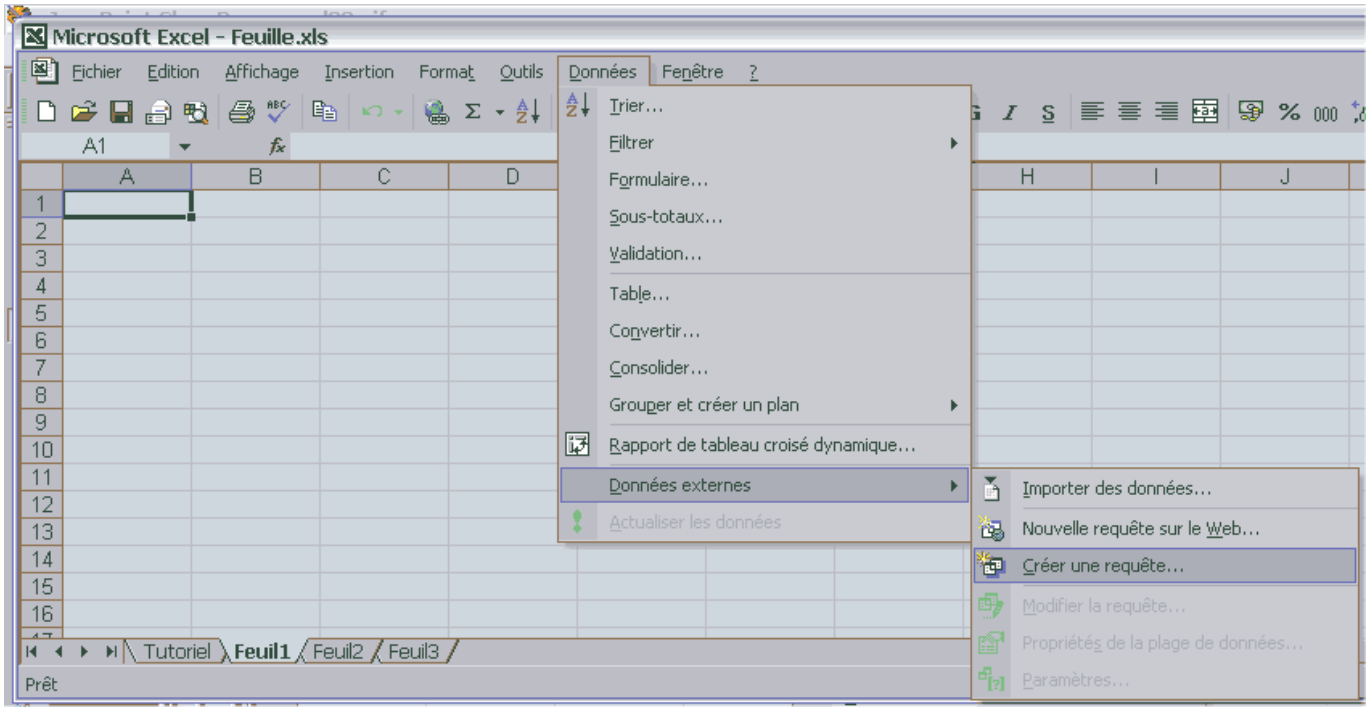
VI - Microsoft Query

Microsoft Query n'est pas installé par défaut dans certaines versions d'office, assurez vous qu'il est bien installé.

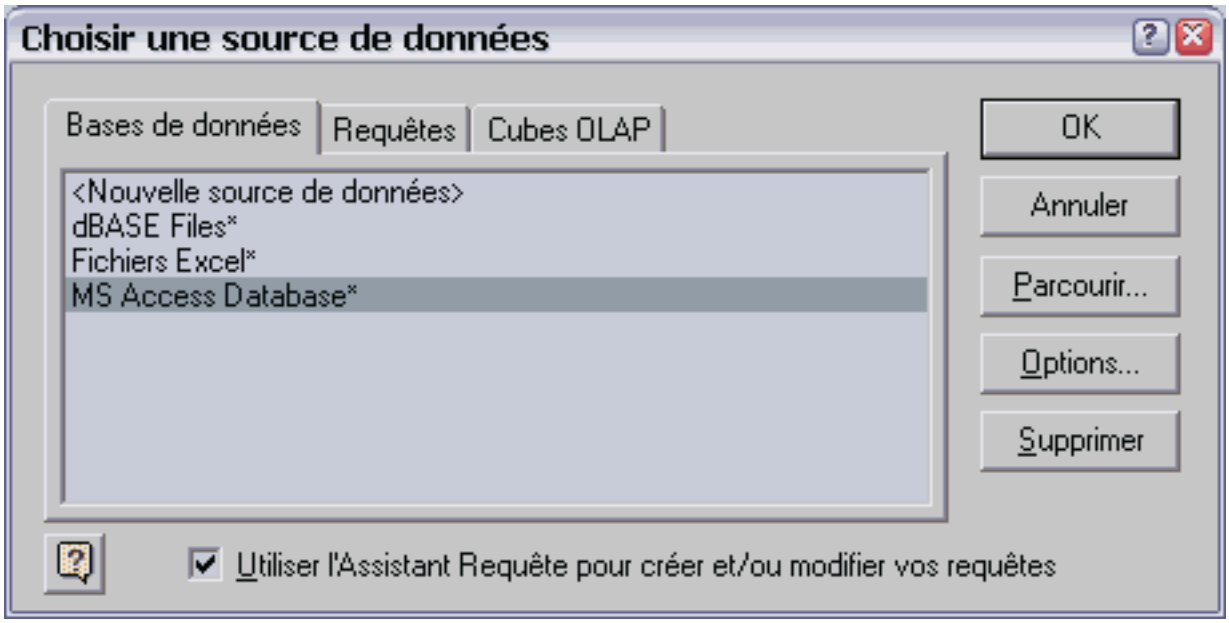
VI-A - La méthode

Cette fois, le travail se fait depuis Microsoft Excel, et Microsoft Query nous assiste tout au long de la procédure.

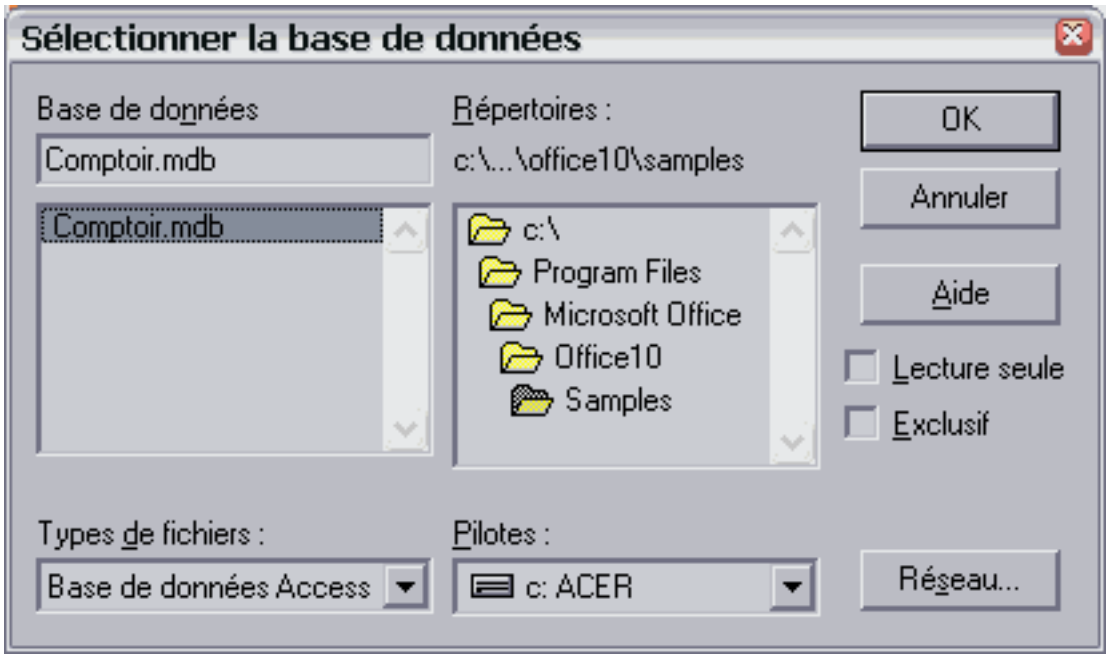
Via le menu : Données > Données Externes ...



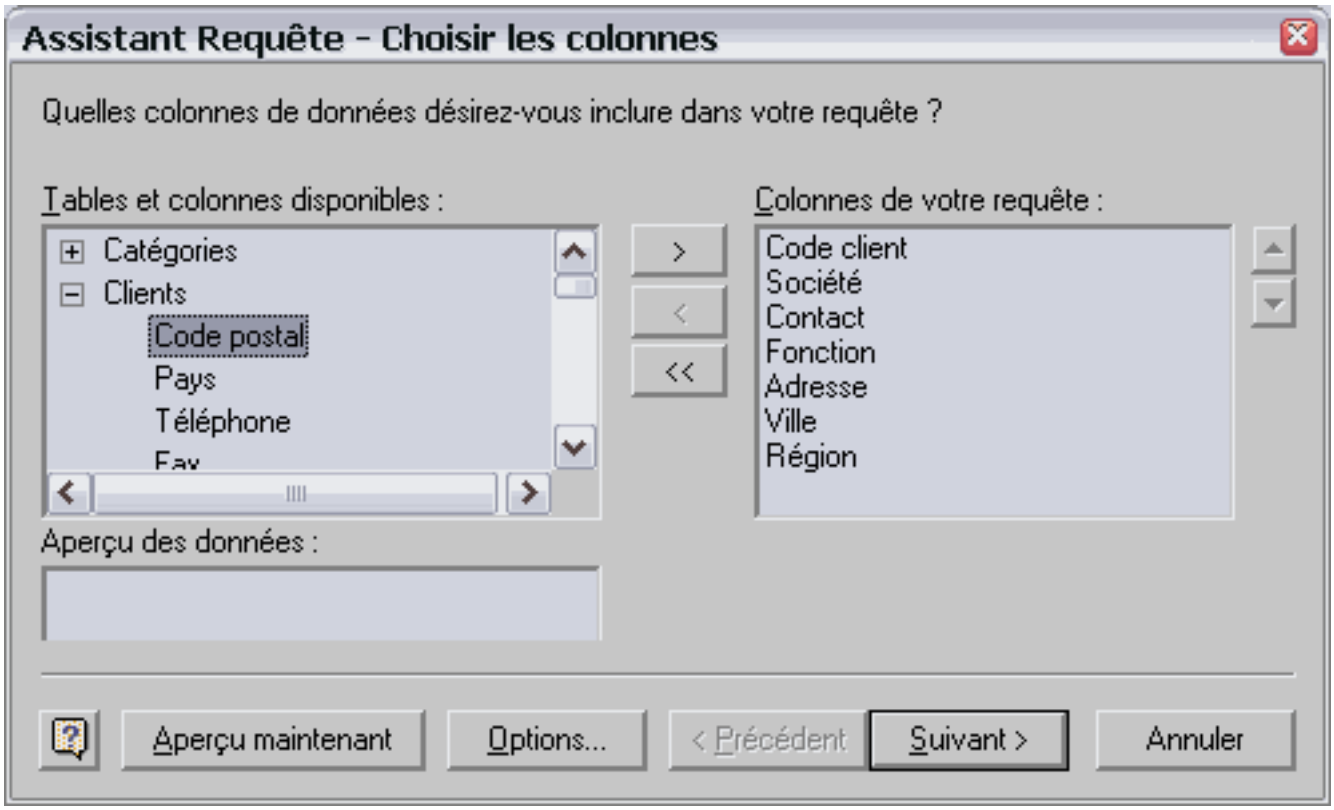
Nous sélectionnons une source de données au format Access (*.MDB)



Nous pointons sur la base Comptoir.mdb

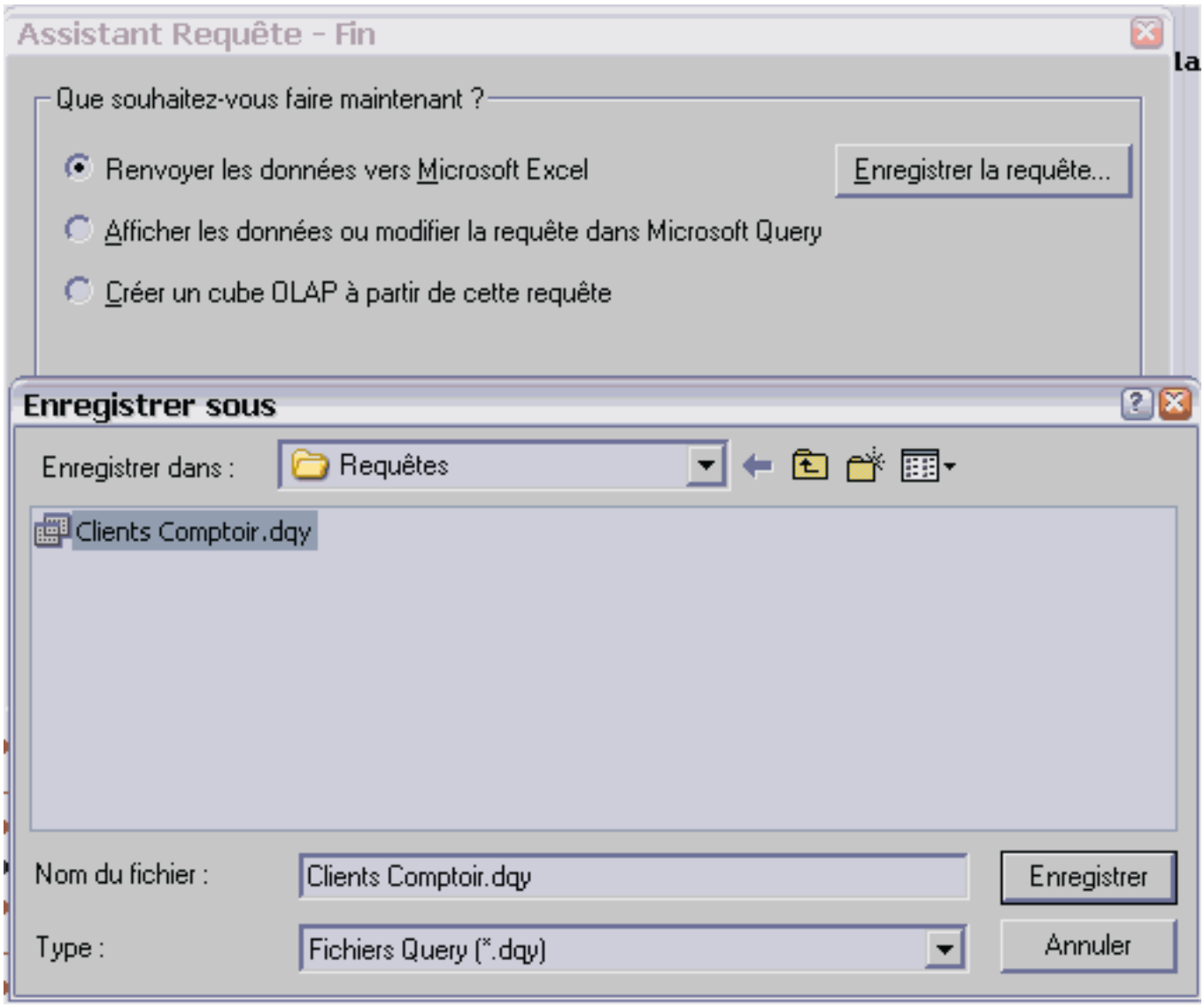


Il nous appartient ensuite de choisir une table ou une requête, puis de sélectionner les champs qui nous intéressent.

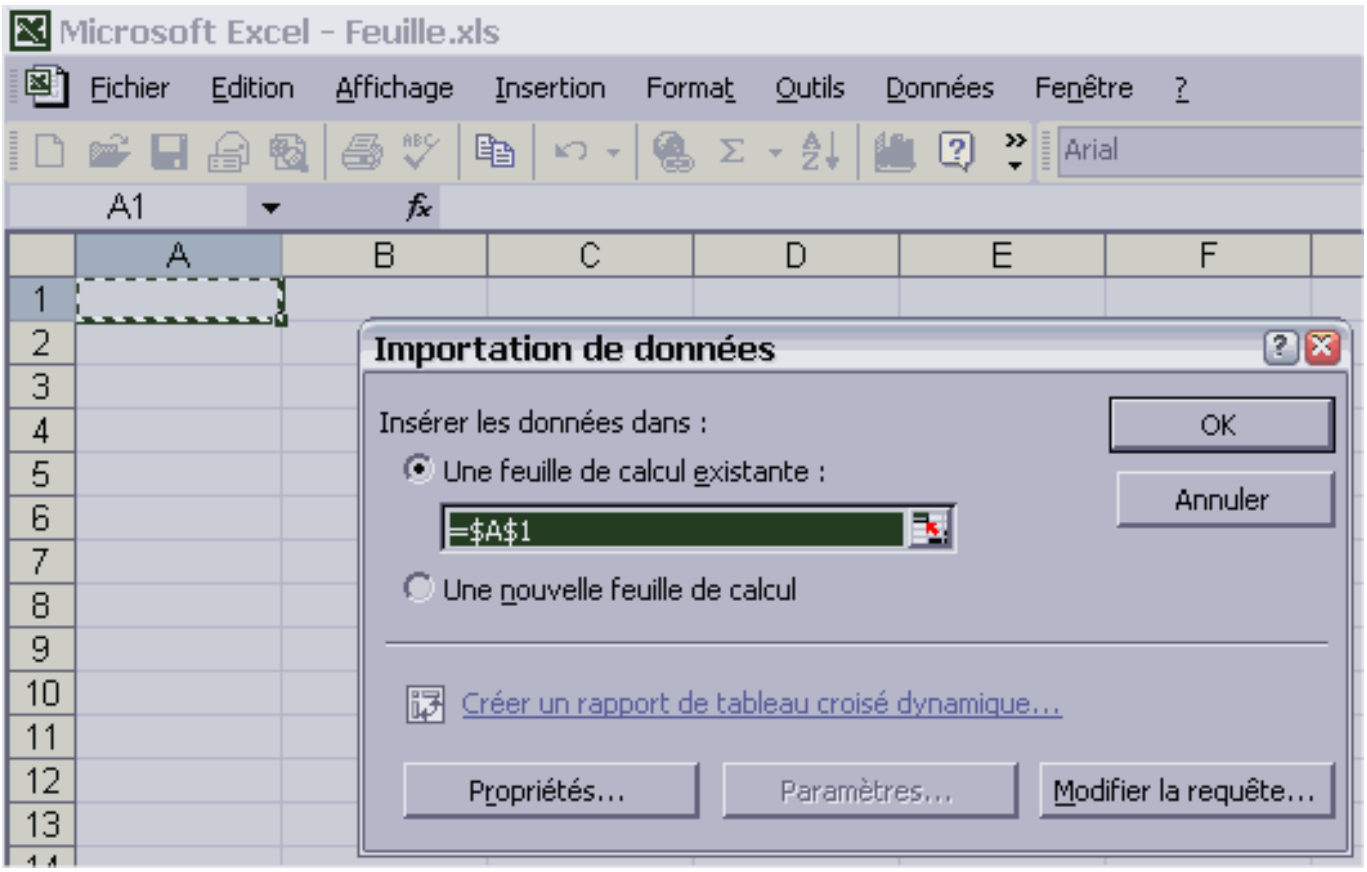


Nous pouvons appliquer des tris et des sélections que nous n'allons pas examiner ici, puisque l'interface est assez explicite et intuitive.

Passons à l'enregistrement de la requête.



Nous sélectionnons une plage qui va recevoir nos données :



Et le tour est joué !

VI-B - Avantages et Limites

Cette méthode comporte de nombreux avantages :

. Aucune programmation nécessaire

. Mise à jour des données depuis Excel par un simple bouton

. Choix de la plage de réception et de l'onglet.

. Possibilité de d'éditer et de créer plusieurs requêtes.

Elle a en contrepartie quelques limites :

. Impossibilité de sauvegarder des enrichissements de présentation et de formats

. Requêteur de MS Query moins puissant que celui d'Access.

. Export sous forme tabulaire

VII - Plate-forme ADO

ADO permet d'accéder, entre autres, aux bases Access depuis VB et VBA.

S'il n'a pas la souplesse de DAO, il peut en revanche répondre à la plupart des besoins.

Pour plus d'informations sur DAO, je vous conseille le très bon travail de Christophe WARIN.

Vous saurez tout sur ADO grâce au remarquable document de J-M Rabilloud.

VII-A - Une méthode originale

A l'exception de l'automation, toutes les méthodes d'export vers Excel souffrent d'une forte rigidité.

Tous les exports sont faits sous forme tabulaire et avec des requêtes figées.

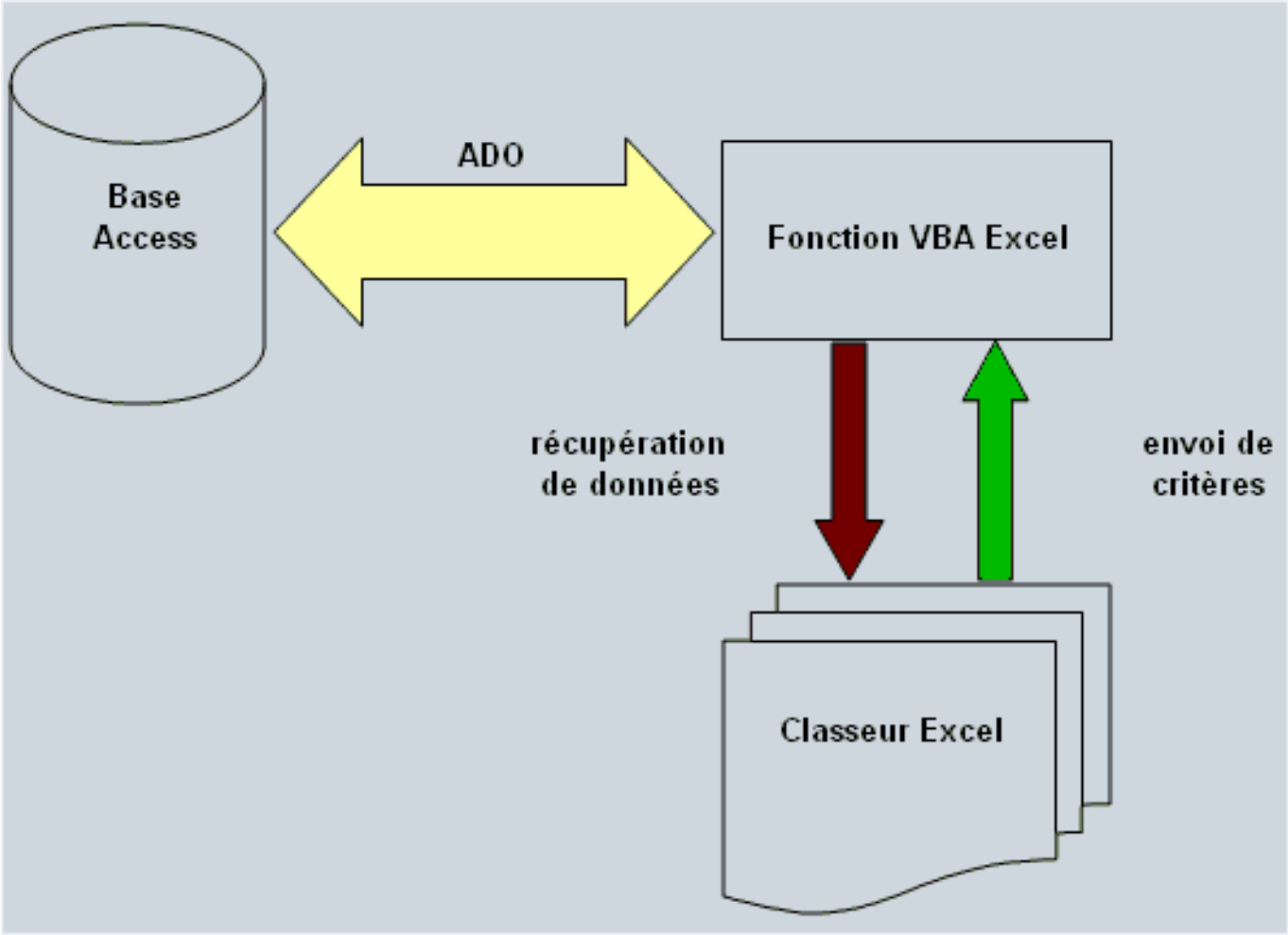
En outre l'automation requiert la présence de MS Access sur le poste sur lequel elle s'exécute, et elle relève d'une progammation délicate.

Notre but ici, est d'offrir une communication à la demande avec Access qui soit plus souple.

Nous pourrions via l'ADO offrir a peu près les mêmes fonctionnalités que par l'automation, à savoir un remplissage cellule par cellule d'une table / requête. Mais cela n'apporterait rien de plus.

La méthode que je vous propose nécessite une programmation préalable, mais laisse à l'utilisateur plus de latitude pour interroger Access.

Voici un diagramme du fonctionnement :



VII-B - Cas pratique

Nous voulons depuis Excel interroger les commandes passées dans la base comptoir, par employé et par mois.

Nous construisons une requête dans la base de données Comptoir :

requête qryXLSlookup
```
SELECT Employés.[N° employé], Employés.Nom, Employés.Prénom,
   Commandes.[N° commande], Commandes.[Code client], Commandes.[Date commande], [Détails
commandes].[Prix unitaire],
   [Détails commandes].Quantité
FROM (Employés INNER JOIN Commandes ON Employés.[N° employé] = Commandes.[N° employé])
   INNER JOIN [Détails commandes] ON Commandes.[N° commande] = [Détails commandes].[N° commande];
```

VII-C - Le code

Nous devons là aussi impérativement ajouter la référence ADO dans Excel, la procédure est la même que dans Access.

Il faut se placer sur un module : Menu **Outils** > **Références ...**

La référence s'appelle : Microsoft ActiveX Data Object.

```
Public cnx As ADODB.Connection

Sub auto_open()
    ' La sub auto_open possède la propriété d'être automatiquement
        '  exécutée à l'ouverture du classeur Excel
        '  à l'identique : auto_close est exécutée sur la fermeture
    Dim strPath As String

    ' Seule contrainte une cellule nommée strPath
    '  doit être présente dans le classeur et
    '  renvoyer sur le chemin de l'appli
    '  en l'occurence Comptoir.mdb
    Application.Goto Reference:="StrPath"

    strPath = ActiveCell

    ' Nous testons si le fichier est accessible
    If Len(Dir(strPath)) > 0 Then
        ' Déclaration de la variable de connexion
        Set cnx = New ADODB.Connection

        ' Connexion à la base
        ConnectDB cnx, strPath
    Else
        MsgBox "La base n'a pas pu être trouvée" & vbCrLf & _
                strPath & vbCrLf & _
                "n'est pas un chemin valide.", vbCritical + vbOKOnly
    End If

End Sub

Sub ConnectDB(ByRef cnx As ADODB.Connection, ByVal strPath As String)

    'Définition du pilote de connexion
    cnx.Provider = "Microsoft.Jet.Oledb.4.0"
    'Définition de la chaîne de connexion
    cnx.ConnectionString = strPath
    'Ouverture de la base de données
    cnx.Open

End Sub

Public Function xRetrieve(Optional ByVal NomEmployé As String = vbNullString, _
                          Optional ByVal Mois As Date = 0, _
                          Optional ByVal Quarterly As Boolean = False)
' Chaine de caractère : nom de l'employé ou cellule qui contient cette information
' Date : date qui va indiquer le mois de la requête  ou cellule qui contient cette information
' Booléen : Si vrai => informations trimestrielles, Si faux => informations mensuelles


    Dim rec As New ADODB.Recordset
    Dim strSQL As String


    'Redaction du SQL
    strSQL = "SELECT Sum([Prix unitaire] * [Quantité]) AS MONTANT " & _
             "FROM [qryXLSlookup] WHERE 1=1"

    ' rappelons que les chaines de caractères en SQL sont à entourer de ''
    ' /!\ toute insertion de chaine dans un SQL comporte un danger pour les données
    '     nous pourrions fort bien ici contrôler le contenu pour neutraliser la
    '     la présence de mots clés placés involontairement ou par malveillance
    If Len(NomEmployé) > 0 Then
        strSQL = strSQL & " And ([Nom] = '" & NomEmployé & "')"
    End If

    ' rappelons que les dates en SQL sont à mettre au format US
    If Mois > 0 Then
        strSQL = strSQL & " And ([Date Commande] Between #" & _
```

```
                Format(MoisInf(Mois, Quarterly), "mm/dd/yyyy") & "# And #" & _
                Format(MoisSup(Mois, Quarterly), "mm/dd/yyyy") & "#)"
    End If

    Dim rst As New ADODB.Recordset

    rst.Open strSQL, cnx

    On Error GoTo errH01
    rst.MoveFirst

    xRetrieve = CDbl(rst("MONTANT"))

    rst.Close
    Set rst = Nothing
    Exit Function

errH01:
    ' Nous sommes dans un tableur excel,
    '  nous ne cherchons pas à analyser les éventuelles erreurs
    '  nous rendons la main au tableur.
    Err.Clear
    xRetrieve = 0
    rst.Close
    Set rst = Nothing


End Function


Function MoisInf(ByVal dat As Date, ByVal blnQter As Boolean) As Date
' Fonction qui permet d'obtenir la première date du mois / trimestre

If Not blnQter Then
    MoisInf = CDate("01/" & Month(dat) & "/" & Year(dat))
Else
    MoisInf = CDate("01/" & ((Month(dat) \ 4) * 3) + 1 & "/" & Year(dat))
End If

End Function

Function MoisSup(ByVal dat As Date, ByVal blnQter As Boolean) As Date
' Fonction qui permet d'obtenir la dernière date du mois / trimestre

If Not blnQter Then
    MoisSup = DateAdd("m", 1, MoisInf(dat, blnQter)) - 1
Else
    MoisSup = DateAdd("m", 3, MoisInf(dat, blnQter)) - 1
End If
End Function
```

Remarque

La Sub auto_open possède l'avantageuse propriété d'être exécutée automatiquement à l'ouverture du classeur Excel.

Veillez à modifier les paramètres de sécurité d'Excel, pour qu'il vous demande à l'ouverture du classeur si vous souhaitez ou non activer les macros.

*Menu **Outils** >*

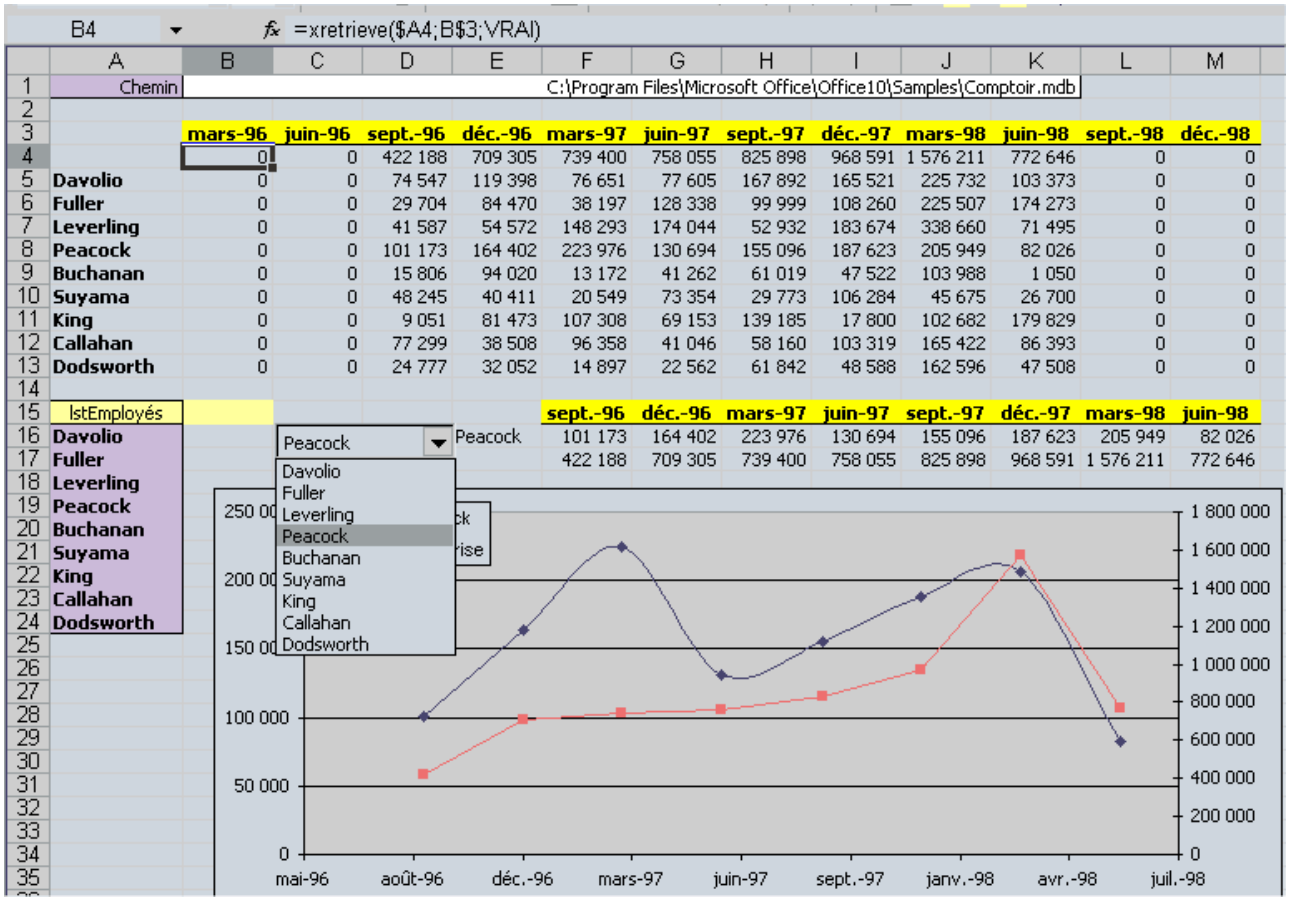
***Options ...** >*

*Onglet **Sécurité** >*

*Bouton **Sécurité des macros ...** >*

Niveau de sécurité moyen

VII-D - Démonstration



Chaque cellule reçoit une formule.

Sur l'exemple ci-dessus, la formule de la cellule **B4** est : **=xretrieve($A4;B$3;VRAI)**

Nous retrouvons là, les arguments de la fonction issue du code.

Les **$** sont là pour fixer les lignes ou colonnes astucieusement afin de remplir le tableau par simple recopie vers la droite et vers le bas.

Le tableau peut ensuite servir de support à des graphiques, des analyses différentes.

Le tableau est dynamique, toute modification d'une cellule de paramètre (qui est argument de la fonction xRetrieve) entraine un recalcul de la feuille et une mise à jour immédiate.

VII-E - Avantages et Limites

Les avantages de cette méthode :

. Souplesse extrême : possibilité de n'obtenir qu'une cellule, de choisir la feuille, la cellule, la plage. Les données ne sont pas sous forme tabulaire, elles sont à disposition de l'utilisateur.

. Mises à jour dynamiques : toute modification d'une cellule de paramètre met à jour les cellules dépendantes

. Simplicité et intégration bureautique : pour l'utilisateur, il s'agit d'une simple formule à l'identique d'une somme.

. Enrichissements et formats totalement libres.

Les limites existent néanmoins :

. Performances faibles : chaque cellule est une requête, ne convient pas à des tableaux de grande taille.

. Niveau synthétique seulement : cette méthode n'est pas pertinente pour récupérer des données de détail et qui ont un nombre inconnu de lignes.

VIII - Conclusions

Un petit diagramme pour résumer :

	Niveau de codage	Facilité d'utilisation	Rapidité	Souplesse	Logiciel de travail	Indications
Copie		★★★★★	★★★★★	★	Access	=> Export massif => Niveau débutant => Données à retravailler manuellement
Export Simple	★	★★★★★	★★★★★	★	Access	=> Export massif => Niveau débutant => Données à retravailler manuellement ou en vue d'éventuelles liaisons
Automation	★★★	★★	★★	★★★★	Access	=> Export précis => Niveau avancé => Données de présentations, agrégées
Microsoft Query	★	★★★	★★★★	★★	Excel	=> Export massif => Niveau avancé => Données à retravailler manuellement ou en vue d'éventuelles liaisons
ADO xRetrieve	★★★★	★★★★	★	★★★★★	Excel	=> Export données agrégées / synthétiques => Niveau avancé pour le programmeur / débutant pour l'utilisateur => Présentations, impressions => Besoin de mise à jour dynamiques

Il n'y a pas de mauvaise méthode, il faut simplement trouver celle qui convient à votre cas.

La méthode xRetrieve dont je me fais l'avocat m'a rendu de grands services dans nombre de mes projets mais n'est pas universelle.

Merci de m'avoir lu ;)