A la découverte d'Access 2007 : Les pièces jointes par l'exemple

par Christophe WARIN

Date de publication : 04/07/2006

Dernière mise à jour :

Dans la continuité de l'article sur les champs multi-valués, je vous propose d'étudier deux autres nouveautés d'Access 2007 : Les pièces-jointes et les FileDialog

I - Introduction

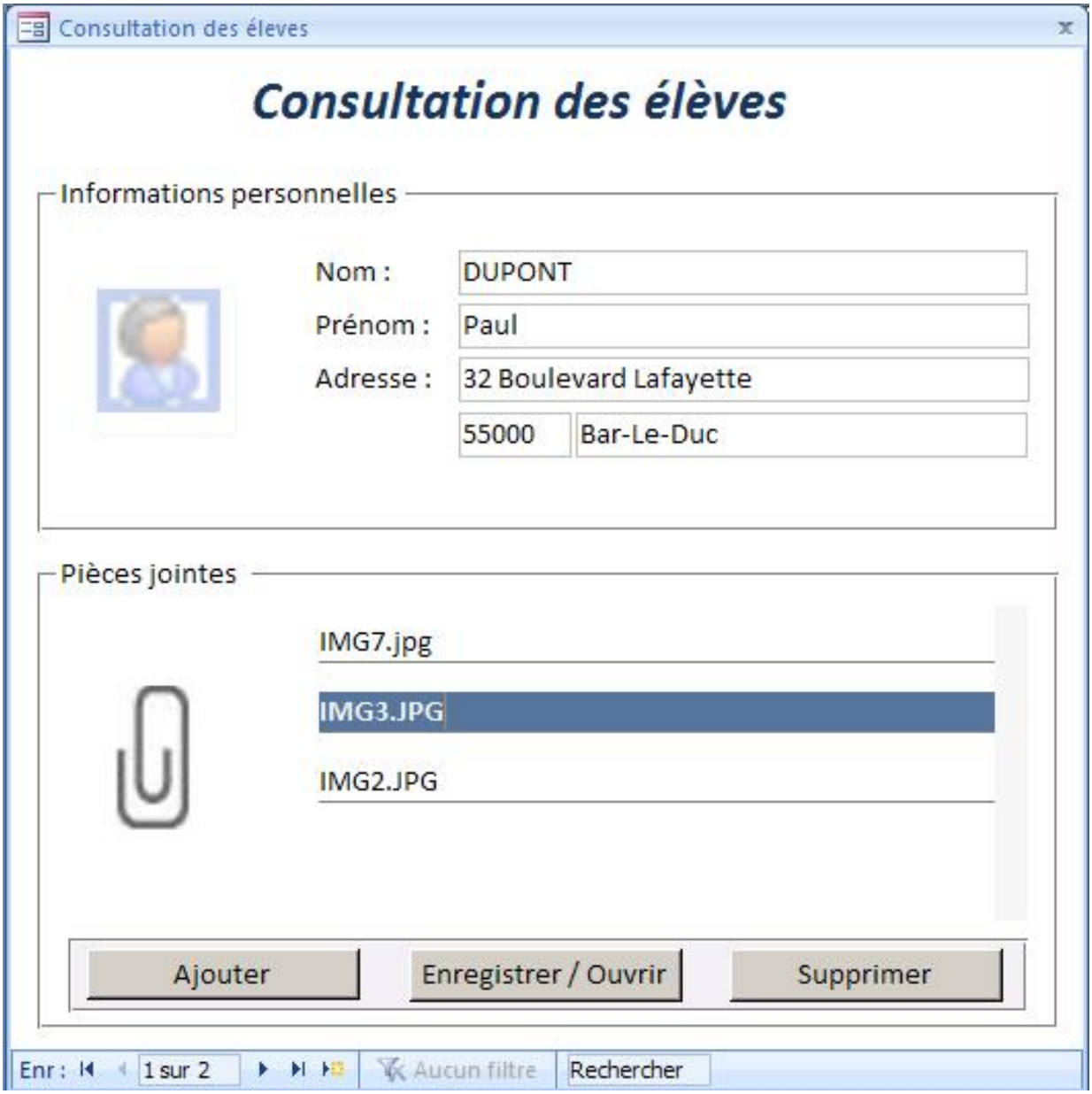
Deux fonctionnalités importantes viennent rejoindre le lot des nouveautés d'Access 2007. Attendues depuis longtemps, elles permettent respectivement l'ajout de pièces-jointes dans les enregistrements et l'affichage des fenêtres communes de Windows.

En attendant des articles beaucoup plus complets sur ces sujets, je vous propose de découvrir les nouvelles possibilités de programmation offertes via un exemple pratique.

II - Présentation de l'exemple

Il s'agit d'un formulaire de saisie d'élèves (Nom, prénom, etc.) permettant l'ajout de différents documents pour chacun d'eux.

Le résultat final est le suivant :



III - Techniques utilisées

Plusieurs solutions techniques sont mises en oeuvre à travers cet exemple :

- Le stockage des fichiers est assuré par la fonction pièce jointe du champ de la table.
- Les boutons **Ajouter** et **Enregistrer** affichent les boîtes de dialogue **Ouvrir** et **Enregistrer Sous** du système d'exploitation.
- La mise en surbrillance de l'élément sélectionné est réalisée avec la mise en forme conditionnelle.

Comme indiqué plus haut, la vocation de ce tutoriel est avant tout de vous faire découvrir ces nouvelles fonctionnalités que vous pouvez tester sur la version Béta2 d'Office 2007 (gratuite). Seules les propriétés et méthodes utilisées seront détaillées.

En ce qui concerne la gestion des pièces-jointes, Access fournit un assistant graphique très intuitif rendant la partie basse du formulaire ci-dessus inutile. Toutefois, cet exemple va vous permettre d'approcher les nouvelles commandes VBA dédiées aux fichiers.

IV - Structure de la base de données

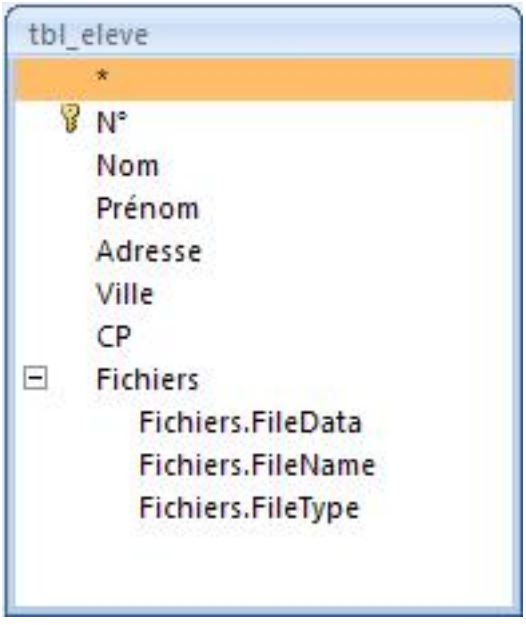
IV-A - Structure de la table

La base de données est composée d'une seule table nommée **tbl_eleve**. Sa structure est la suivante :

Nom du champ	Type de données
N°	NuméroAuto
Nom	Texte
Prénom	Texte
Adresse	Texte
Ville	Texte
CP	Texte
Fichiers	Pièce jointe

IV-B - Création des requêtes

La gestion des pièces-jointes repose sur des champs ***multi-valués***. Un moyen pour s'en apercevoir : créez une nouvelle requête sur la table **tbl_eleve** et basculez l'affichage en mode création :



Il s'agit bien là d'une structure avec un champ multi-valué : **Fichiers**. En fait lorsque le développeur décide d'ajouter un champ de type **Pièce-Jointe** à sa table, Access créée automatiquement l'arborescence correspondante. Pour comprendre le mécanisme mis en jeu dans ces champs, je vous conseille de lire ce tutoriel : A la découverte d'Access V12 : Les champs multi-valués.

Quelques remarques :

- Le champ **FileData** correspond aux données du fichier. Par la suite, nous verrons que VBA propose une méthode pour enregistrer le contenu d'un fichier dans une pièce-jointe. Le fichier est donc inclus dans la base de données et non pas simplement lié.
- **FileName** correspond au nom du fichier. Il doit être unique pour chaque élève.
- **FileType** représente le type du fichier. Il est automatiquement mis à jour lors de l'affectation de FileData

En tenant compte des informations pratiques données dans le tutoriel cité plus haut, il est facile d'établir les deux requêtes nécessaires au projet :

R01 Liste des élèves

```
SELECT N°, Nom, Prénom, Adresse, Ville, CP
FROM tbl_eleve;
```

R02 Liste Fichier élèves

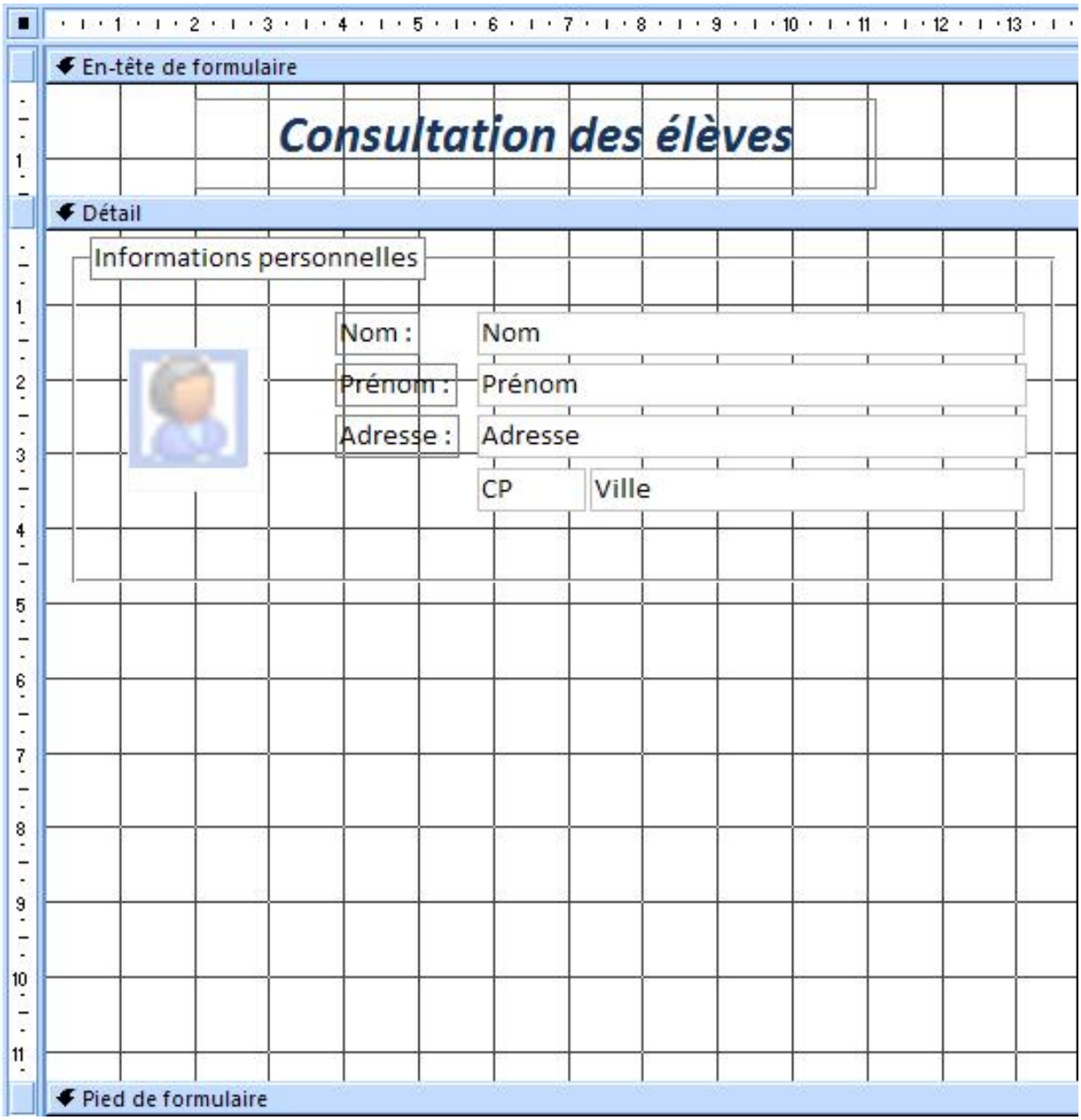
```
SELECT N°, [tbl_eleve].Fichiers.Value.Filename AS LesFichiers
FROM tbl_eleve;
```

R02 fournit le nom des fichiers associés à chaque élève.

V - L'interface graphique

V-A - Le formulaire principal

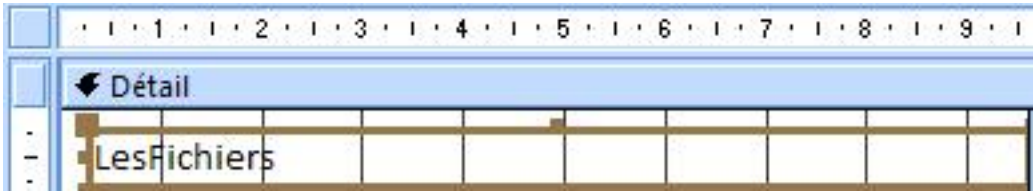
Créez un nouveau formulaire vide et disposez les objets de la sorte :



Ici, rien de compliqué ni de nouveau. La source du formulaire est la requête R01. Placez ensuite les 3 boutons nommés respectivement **cmdAjouter**, **cmdEnregistrer**, **cmdSupprimer**.

V-B - Le sous-formulaire

Créez un autre formulaire vide basé sur la requête R02

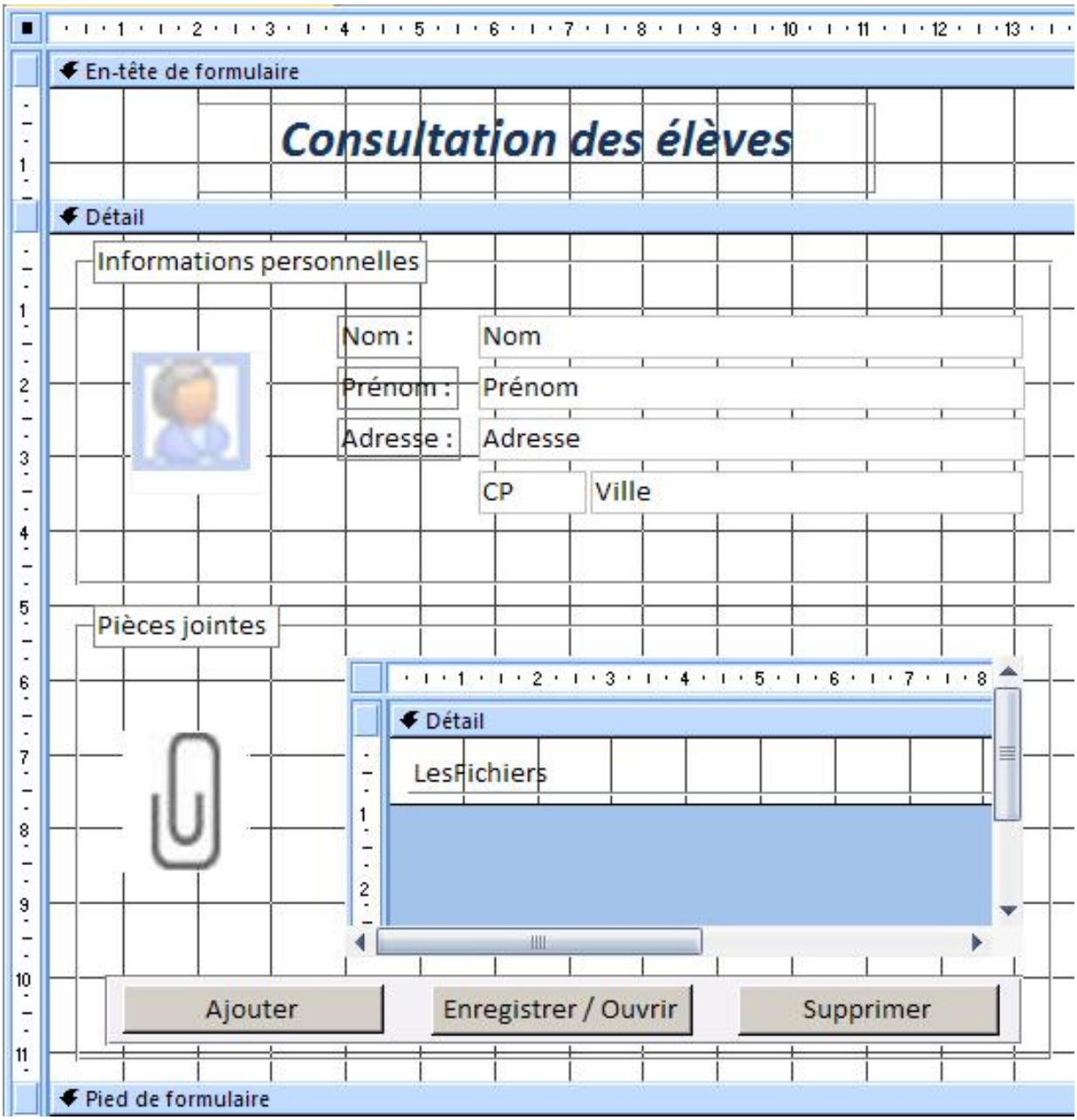


La zone de texte est nommée **txtFichier** (ce nom est important car il sera repris dans le code VBA)

Fixez les propriétés **Ajout autorisé**, **modification autorisée** et **suppression autorisée** à **Non**.

Revenez ensuite sur le formulaire principal, ajoutez y un cadre de sous-formulaire. Laissez-vous guider par l'assistant. La liaison champ père / champ fils est basée sur le champ **N°** des deux requêtes.

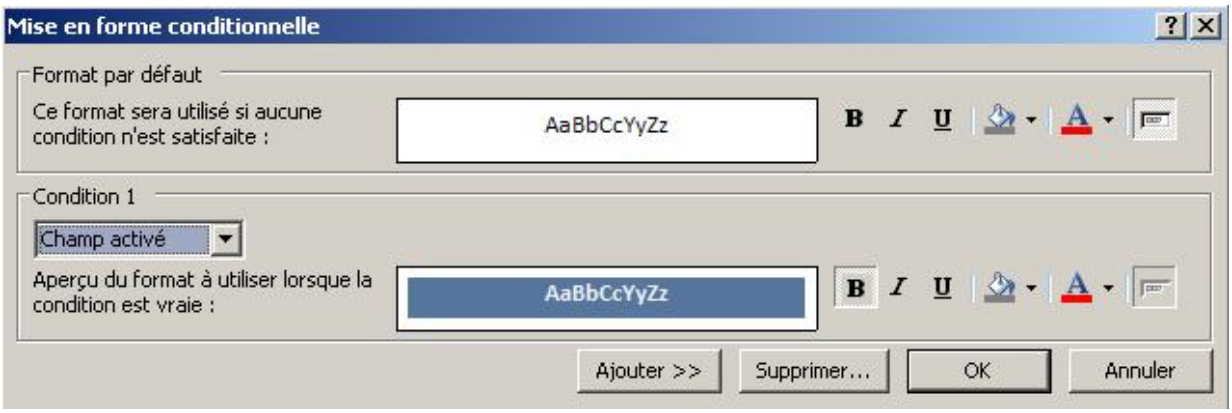
Résultat final :



V-C - Mise en forme conditionnelle

Pour que l'élément sélectionné dans le sous-formulaire change de couleur, il faut utiliser la mise en forme conditionnelle. Pour toute explication relative à cette notion, je vous recommande de lire mon tutoriel.

Cliquez sur le contrôle **txtFichier** puis sur le bouton **Mise en forme conditionnelle** dans le *ruban* **Créer**. La condition est de type **Champ Activé**.



VI - Programmation VBA

VI-A - Les pièces jointes

VI-A-1 - L'ajout

Comme indiqué dans le tutoriel sur les champs multi-valués, les pièces-jointes sont accessibles via un *recordset* retourné par la propriété **Value** du champ **Fichiers** de la table **tbl_eleve**.

Le squelette de la fonction **AjouterFichier** est donc semblable à :

```
Function AjouterFichier(strchemin As String, inteleve As Integer) As Boolean
On Error GoTo err
Dim oRst As DAO.Recordset
    'Ouvre un recordset sur les fichiers de l'éleve passé en paramètre
    Set oRst = CurrentDb.OpenRecordset("SELECT Fichiers FROM tbl_eleve WHERE [N°]=" & inteleve)
    With oRst.Fields(0).Value
        'Ajoute la pièce-jointe
        .AddNew
                '
                        'ici code qui va enregistrer les données du fichier
                        '
                        '
        .Update
    End With
    AjouterFichier = True
err:
End Function
```

La fonction accepte deux paramètres : le chemin du fichier à joindre et l'identifiant de l'élève.

Etant donné que **oRst.Fields(0).Value** retourne un recordset, il est possible de lui appliquer la méthode **AddNew** (même si VBA ne la propose pas dans la liste des méthodes disponibles). En résumé, il faut travailler avec **oRst.Fields(0).Value** comme s'il s'agissait d'un objet recordset.

En ce qui concerne le code qui va convertir le fichier en données à insérer dans la base, il se résume à une seule ligne fournit par le nouveau modèle DAO.

```
.Fields("FileData").LoadFromFile strchemin
```

Explications :

En introduisant les pièces jointes, Microsoft a été contraint d'écrire des nouvelles méthodes pour les champs DAO de ce type. Toutefois, ne souhaitant pas perturber l'existant, les développeurs d'Office ont en fait créé un nouvel objet de type **Field2** venant compléter l'objet **Field**. Le champ **FileData** étant concerné par cette modification, le code suivant est correct :

```
Dim oFld as DAO.Field2
Set oFld=.Fields("FileData")
```

Les objets **Field2** exposent 2 méthodes pour lire et écrire les fichiers joints

- **LoadFromFile** : Charge le fichier dans la base de données.
- **SaveToFile** : Sauvegarde la pièce-jointe dans un nouveau fichier.

Pour ajouter une pièce-jointe dont le chemin est **strChemin**, on aura donc :

```
Dim oFld as DAO.Field2
Set oFld=.Fields("FileData")
oFld.LoadFromFile strchemin
```

Soit en une seule ligne :

```
.Fields("FileData").LoadFromFile strchemin
```

La fonction **AjouterFichier** devient :

```
Function AjouterFichier(strchemin As String, inteleve As Integer) As Boolean
On Error GoTo err
Dim oRst As DAO.Recordset
    'Ouvre un recordset sur les fichiers de l'éleve passé en paramètre
    Set oRst = CurrentDb.OpenRecordset("SELECT Fichiers FROM tbl_eleve WHERE [N°]=" & inteleve)
    With oRst.Fields(0).Value
        'Ajoute la pièce jointe
        .AddNew
        'Lit le fichier
        .Fields("FileData").LoadFromFile strchemin
        .Update
    End With
    AjouterFichier = True

fin:
    Set oRst = Nothing
    Exit Function

err:
    'Gestion d'erreur
    Select Case err.Number
        Case 3024:
            MsgBox "Fichier inexistant", vbCritical
        Case 3820:
            MsgBox "Une autre pièce-jointe de ce nom existe déjà", vbCritical
        Case Else:
            MsgBox "Erreur inconnue", vbCritical
    End Select
    Resume fin
End Function
```

VI-A-2 - L'enregistrement

Sur le même principe que l'ajout mais avec la méthode **SaveToFile**, voici le code de la fonction **EnregistrerFichier** :

```
Function EnregistrerFichier(strNomFichier As String, inteleve As Integer, strNomDestination) As
Boolean
On Error GoTo err
Dim oRst As DAO.Recordset
    'Ouvre un recordset sur les fichiers de l'éleve passé en paramètre
     Set oRst = CurrentDb.OpenRecordset("SELECT Fichiers.FileData FROM tbl_eleve WHERE [N°]=" & _
            inteleve & " AND Fichiers.FileName=" & Chr(34) & strNomFichier & Chr(34))
    If Not oRst.EOF Then

        oRst.Fields(0).SaveToFile strNomDestination
        EnregistrerFichier = True
    End If

fin:
Set oRst = Nothing
Exit Function

err:
    Select Case err.Number
    Case 3839:
```

```
            MsgBox "Impossible d'écraser le fichier", vbCritical
        Case Else
            MsgBox "Erreur inconnue", vbCritical
        End Select
        Resume fin
    End Function
```

Cette fonction accepte un paramètre supplémentaire : le chemin du fichier dans lequel sera enregistrée la pièce jointe.

VI-A-3 - La suppression

Il serait possible, là aussi, d'utiliser un recordset sur le champ **Fichiers** et appliquer sa méthode **Delete**. Cependant, dans un souci pédagogique, j'ai préféré utiliser une requête SQL exécutée en VBA de la façon suivante :

```
    Function SupprimerFichier(strNomFichier As String, inteleve As Integer) As Boolean
    On Error GoTo err
        CurrentDb.Execute "DELETE [tbl_eleve].Fichiers.FileName " & _
                          "FROM tbl_eleve " & _
                          "WHERE [N°]=" & inteleve & _
                          " AND [tbl_eleve].Fichiers.FileName=" & Chr(34) & strNomFichier & Chr(34)


        SupprimerFichier = True
    err:
    End Function
```

La requête permet de supprimer les sous-enregistrements de **tbl_eleve.Fichier** pour l'élève N° **inteleve** où le nom du fichier est **strNomFichier**.

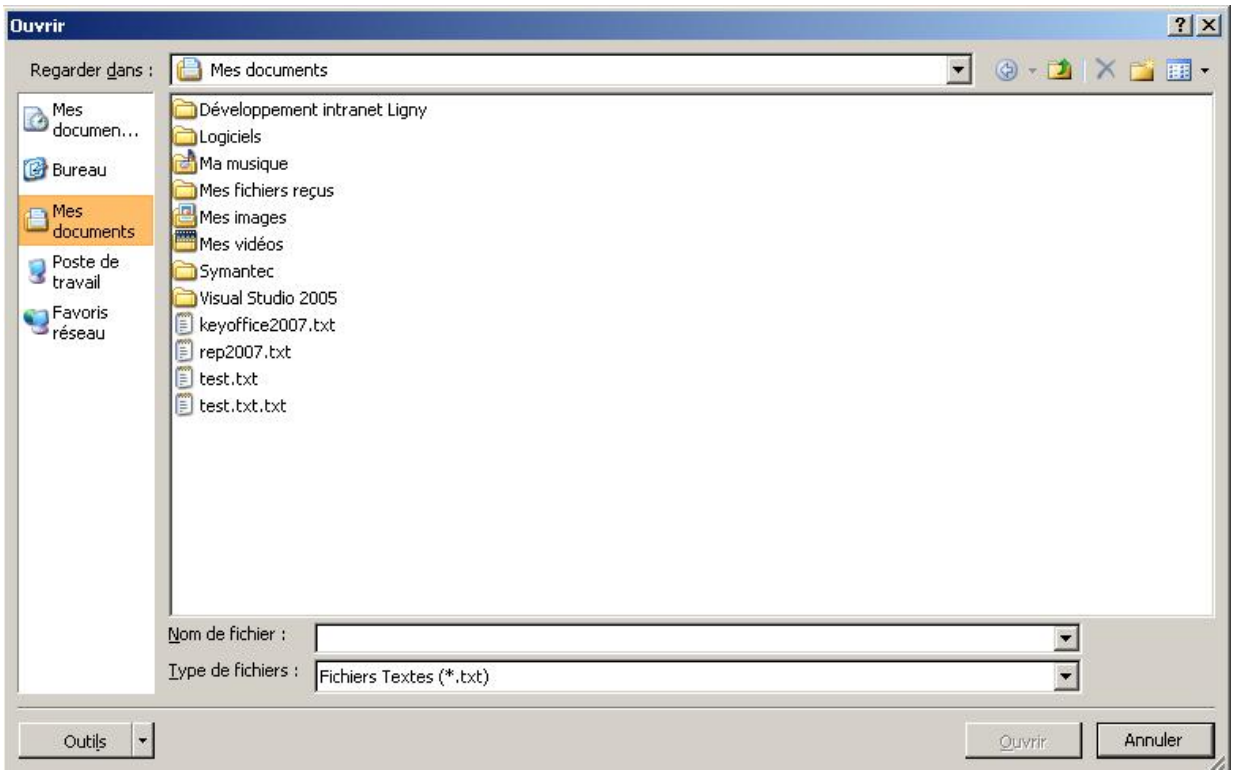
Le code du bouton **Supprimer** est le suivant :

```
    Private Sub cmdSupprimer_Click()
    Dim strNomFichier As String
    strNomFichier = Nz(SFormFichier.Form.txtfichier)
    If strNomFichier <> "" Then
        'Interroge l'utilisateur
        If MsgBox("Etes-vous sûr de vouloir supprimer : " & strNomFichier & " ?", _
                vbQuestion + vbYesNo + vbDefaultButton2, "Supprimer une pièce-jointe") = vbYes Then
            If SupprimerFichier(strNomFichier, Me.N°) Then
                'Rafraichit le formulaire
                Me.SFormFichier.Requery
            Else
                MsgBox "Impossible de supprimer la pièce-jointe", vbCritical, "Suppression de
    pièce-jointe"
            End If
        End If
    End If
    End Sub
```

VI-B - Les boîtes de dialogue

Tant attendus depuis plusieurs versions, les objets **FileDialog** font enfin leur apparition dans Access 2007. Ils permettent entre autres d'afficher les fenêtres Ouvrir et Enregistrer Sous.

*Pour que ces fenêtres fonctionnent vous devez ajouter la référence **Microsoft Office 12** à votre projet ou bien vous passez des constantes **msoFileDialogOpen** et **msoFileDialogSaveAs** et les remplacez par leur valeur respective : **1** et **2**.*



Pour créer une telle boîte, il faut appeler la méthode **Application.FileDialog(TypedeBoite)** qui retourne une instance de la fenêtre à afficher.

Pour une boîte Ouvrir :

```
Set oFD = Application.FileDialog(msoFileDialogOpen)
```

Pour une boîte Enregistrer Sous :

```
Set oFD = Application.FileDialog(msoFileDialogSaveAs)
```

Le nom du fichier sélectionné est disponible dans la collection **SelectedItems**.

Bien que l'ouverture soit similaire pour les deux types de **FileDialog**, les paramètres à définir ne sont pas les mêmes.

VI-B-1 - La fenêtre Ouvrir

Lorsque la fenêtre s'ouvre, il faut que la zone de texte **Nom de fichier** soit vide. Ceci est obtenu par la ligne de code suivante :

```
FileDialog.InitialFileName = ""
```

InitialFileName correspond donc au nom du fichier par défaut.

Il est possible de restreindre le type de fichier à ouvrir via la zone de liste **Type de Fichiers**. Ces types sont représentés par une collection dans l'objet **FileDialog**.

La méthode ***Filedialog.Filters.Add description, extension, position*** permet d'ajouter une extension à la liste.

Enfin, dans ce projet, l'utilisateur ne doit pas pouvoir sélectionner plusieurs fichiers. Il faut donc fixer la propriété **AllowMultiSelect** à **False**.

```
FileDialog.AllowMultiSelect = False
```

Voici le code du bouton **Ajouter** dans son intégralité :

```
Private Sub cmdAjouter_Click()
Dim strchemin As String
Dim oFD As Object
'Paramètre la fenêtre Ouvrir
Set oFD = Application.FileDialog(msoFileDialogOpen)
With oFD
    'Ajoute les filtres pour fichiers textes et tous
    With .Filters
        .Clear
        .Add "Fichiers Textes", "*.txt", 1
        .Add "Tous", "*.*", 2
    End With
    .InitialFileName = ""
    'Interdit la multi sélection
    .AllowMultiSelect = False
    'Affiche la fenêtre et vérifie qu'un fichier a bien été choisi
    If .Show Then
        strchemin = .SelectedItems(1)
        If AjouterFichier(strchemin, Me.N°) Then
            Me.SFormFichier.Form.Requery
        End If
    End If
End With
End Sub
```

VI-B-2 - La fenêtre Enregistrer Sous

Ici, il n'est pas nécessaire de tenir compte des filtres ou de la multi-selection. Il suffit juste de définir le nom par défaut du fichier. Ce qui donne :

```
Private Sub cmdEnregistrer_Click()
Dim strchemin As String
Dim strNomFichier As String
Dim oFD As Object
strNomFichier = Nz(SFormFichier.Form.txtfichier)
If strNomFichier <> "" Then

    'Paramètre la fenêtre Enregistrer
    Set oFD = Application.FileDialog(msoFileDialogSaveAs)
    With oFD
        'Affiche la fenêtre et vérifie qu'un fichier a bien été choisi
        .InitialFileName = strNomFichier
        If .Show Then
            strchemin = .SelectedItems(1)
            If EnregistrerFichier(strNomFichier, Me.N°, strchemin) Then
                If MsgBox("Voulez-vous ouvrir le fichier ?", vbQuestion + vbYesNo, "Enregistrement
d'une pièce jointe") = vbYes Then
                    ShellExecute Me.hwnd, "open", strchemin, "", CurrentProject.Path, 1
                Else
                    MsgBox "Enregistrement réussi", vbInformation, "Enregistrement d'une pièce
jointe"
                End If
            End If
        End If
    End With
```

```
    End If
    End Sub
```

La méthode **ShellExecute** est une fonction de l'API Windows. Elle permet d'ouvrir un fichier. Pour l'utiliser vous devez rajouter sa déclaration dans l'entête d'un module nommé **mduAPI** :

```
    Public Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
        (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, _
        ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

*Lorsque l'utilisateur clique sur le bouton **Annuler** d'une boîte de saisie, la fonction **Show** qui l'a appelée retourne **False**.*

VII - Conclusion

Comme vous l'avez constaté, Access 2007 amène de nouveaux réflexes à assimiler pour le développeur. La notion de champs multi-valués bien que destinée aux débutants risque d'en rebuter plus d'un.

De plus, même si l'insertion de fichier dans la base de données semble pratique, il ne faut pas oublier qu'elle va faire augmenter rapidement et considérablement la taille du fichier Access au détriment des performances et de la stabilité.

Aussi, personnellement, je pense que la séparation des données de la base et des fichiers comme avant reste une pratique à conserver. Le stockage du nom du fichier dans un simple champ texte reste le plus optimisé.