REPOUSSER LES LIMITES DES FORMULAIRES EN MODE CONTINU

par Charles A.

Date de publication : 09/06/2005

Dernière mise à jour : 07/15/2005

Challenge de programmation : réaliser un formulaire continu affichant des images externes.

I - Introduction

Cet article tutoriel n'est pas destiné aux débutants mais aux utilisateurs moyens et avancés de Microsoft Access.

Sur le forum de nombreuses questions tournent autour des formulaires continus.

Ces formulaires qui offrent d'intéressantes possibilités présentent en contrepartie des contraintes dont il est difficile de s'affranchir.

La principale contrainte qui heurte tant d'utilisateurs est que dans un formulaire continu chaque contrôle n'existe qu'une fois en contradiction avec l'affichage.

Cet affichage multiple est trompeur, car si la valeur d'un contrôle lié à une source de données change à chaque enregistrement il n'en va pas de même pour tous les contrôles indépendants.

Dans le cas de mon tutoriel de gestion des images, il est impossible d'avoir un formulaire continu qui affiche les photos, on aura bien une liste mais la photo sera la même pour tous.

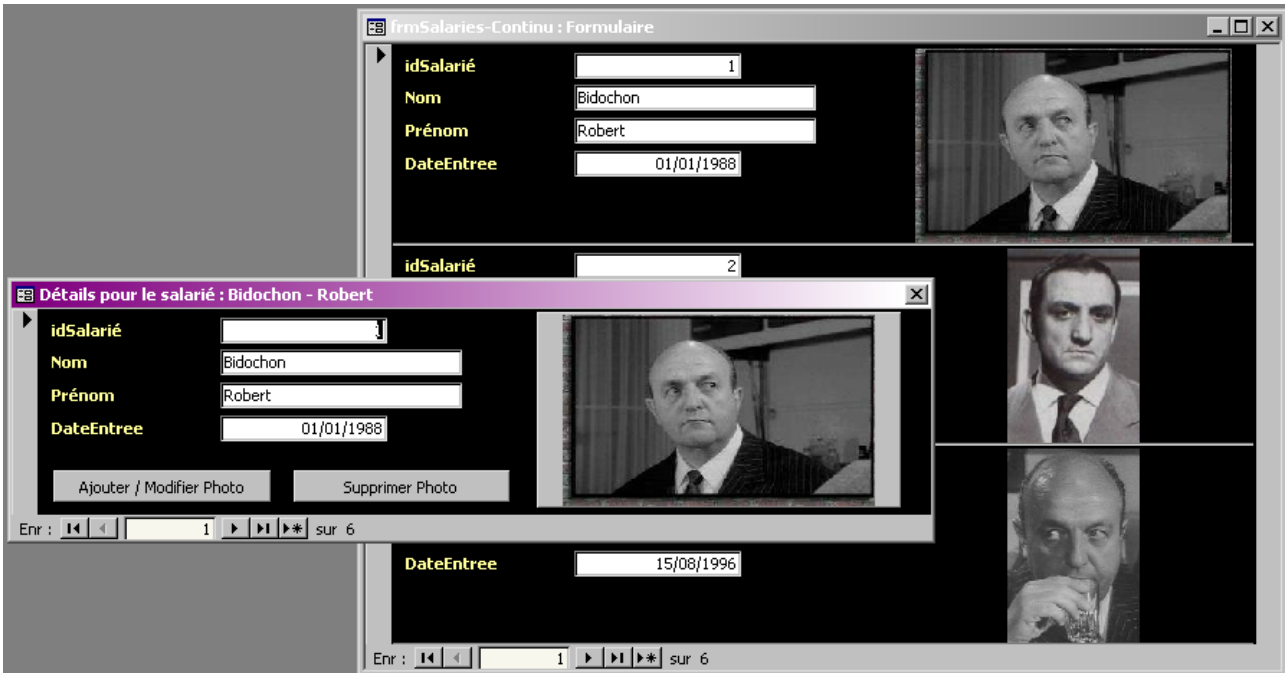
Ce tutoriel propose une solution pour y parvenir malgré tout.

Toutefois, je reste bien conscient que ce developpement n'est pas applicable à tous les cas, loin de là, mais il ouvre certaines pistes et a été pour moi un plaisant petit défi de programmation.

Il me permet surtout d'aborder ici de nombreux thèmes susceptibles d'intéresser les développeurs Access.

II - Rendu final de l'application

A gauche le formulaire issu du tutoriel Photos, et à droite le formulaire "pseudo-continu" construit par l'application que nous allons étudier.



III - Méthodologie

A - Démultiplier la source des données

Pour pallier aux défauts du mode continu, nous allons utiliser un formulaire en mode "normal", et donner l'impression à l'utilisateur qu'il travaille sur un formulaire continu.

L'idée est de démultiplier la source de données pour simuler un formulaire continu. Ainsi, si nous avons un formulaire avec : txtNom, txtPrenom, cmbVille nous allons construire une source de données avec : txtNom0, txtNom1, txtNom2 ... autant que nous souhaitons avoir de lignes sur notre formulaire continu.

La source de données passe de

tblSalaries : Table

	idSalarié	Nom	Prénom	DateEntree	Photo
▶	1	Bidochon	Robert-Marcel	01/01/1988	C:\Documents and Settin
	2	Meunier	Tudor	14/07/1992	C:\Documents and Settin
	3	Hamouque	Nadine	15/08/1996	C:\Documents and Settin
	4	Céquilnépamort	Nadine	11/11/2000	C:\Documents and Settin
	5	Ledivinenfant	René	25/12/2003	C:\Documents and Settin
	7	Dehurlevent	Léo	31/12/1999	C:\Documents and Settin
*	néroAuto)				

en

tblContinue : Table

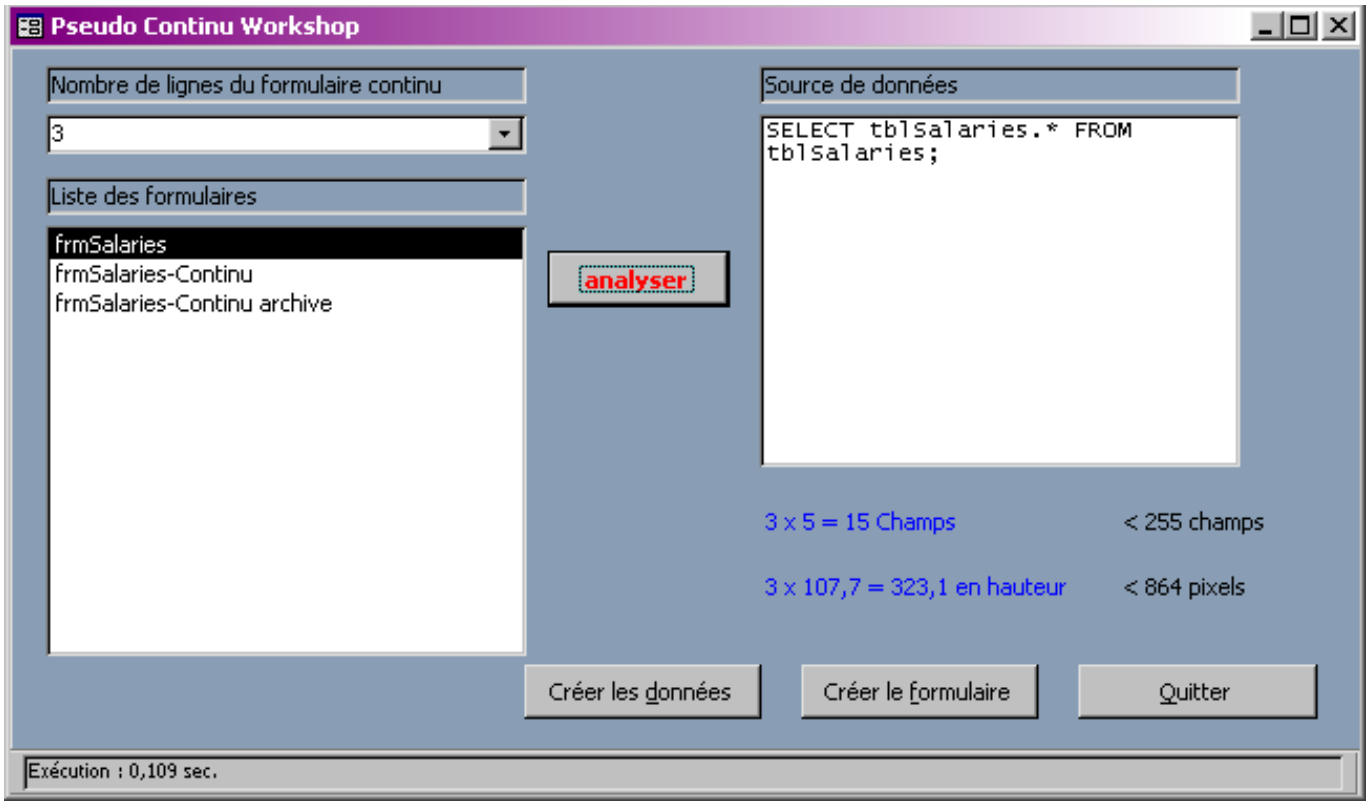
	idSala	NomC	Prénc	DateE	Photc	idSala	Nom1	Prénc	DateE	Photc	idSala	Nom2	Prénc	DateE	Photc	idSala	Nom3	Prénc	DateE	Photc
▶	1	Bidoc	Robei	/1988	C:\Do	2	Meun	Tudor	/1992	C:\Do	3	Hamc	Nadin	/1996	C:\Do	4	Céqui	Nadin	/2000	C:\Do
	2	Meun	Tudor	/1992	C:\Do	3	Hamc	Nadin	/1996	C:\Do	4	Céqui	Nadin	/2000	C:\Do	5	Ledivi	René	/2003	C:\Do
	3	Hamc	Nadin	/1996	C:\Do	4	Céqui	Nadin	/2000	C:\Do	5	Ledivi	René	/2003	C:\Do	7	Dehur	Léo	/1999	C:\Do
	4	Céqui	Nadin	/2000	C:\Do	5	Ledivi	René	/2003	C:\Do	7	Dehur	Léo	/1999	C:\Do					
	5	Ledivi	René	/2003	C:\Do	7	Dehur	Léo	/1999	C:\Do										
	7	Dehur	Léo	/1999	C:\Do															
*																				

B - Analyser le formulaire existant

L'étude portera sur deux critères essentiels : la hauteur finale du formulaire et le nombre de champs possible. Le nombre de champs d'une table Access ne peut excéder 255 caractères. L'application va récupérer le nombre de

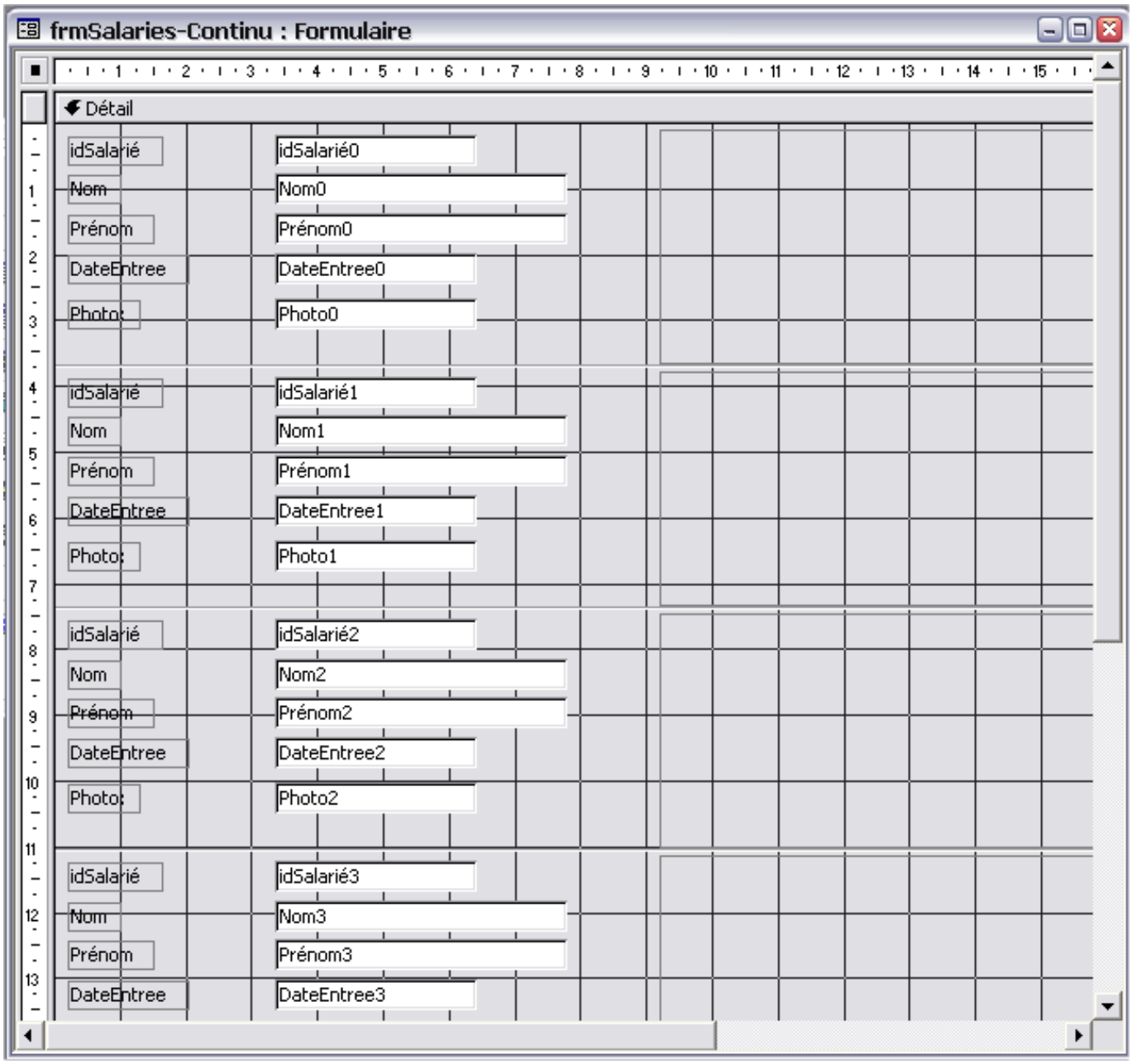
champs de la requête sous-jacente du formulaire et le multiplier par le nombre de lignes souhaitées.

Par une fonction API, nous allons récupérer la résolution verticale du PC et la comparer à la hauteur du formulaire (ici la zone de détail) multipliée par le nombre de lignes souhaitées.



C - Créer le formulaire

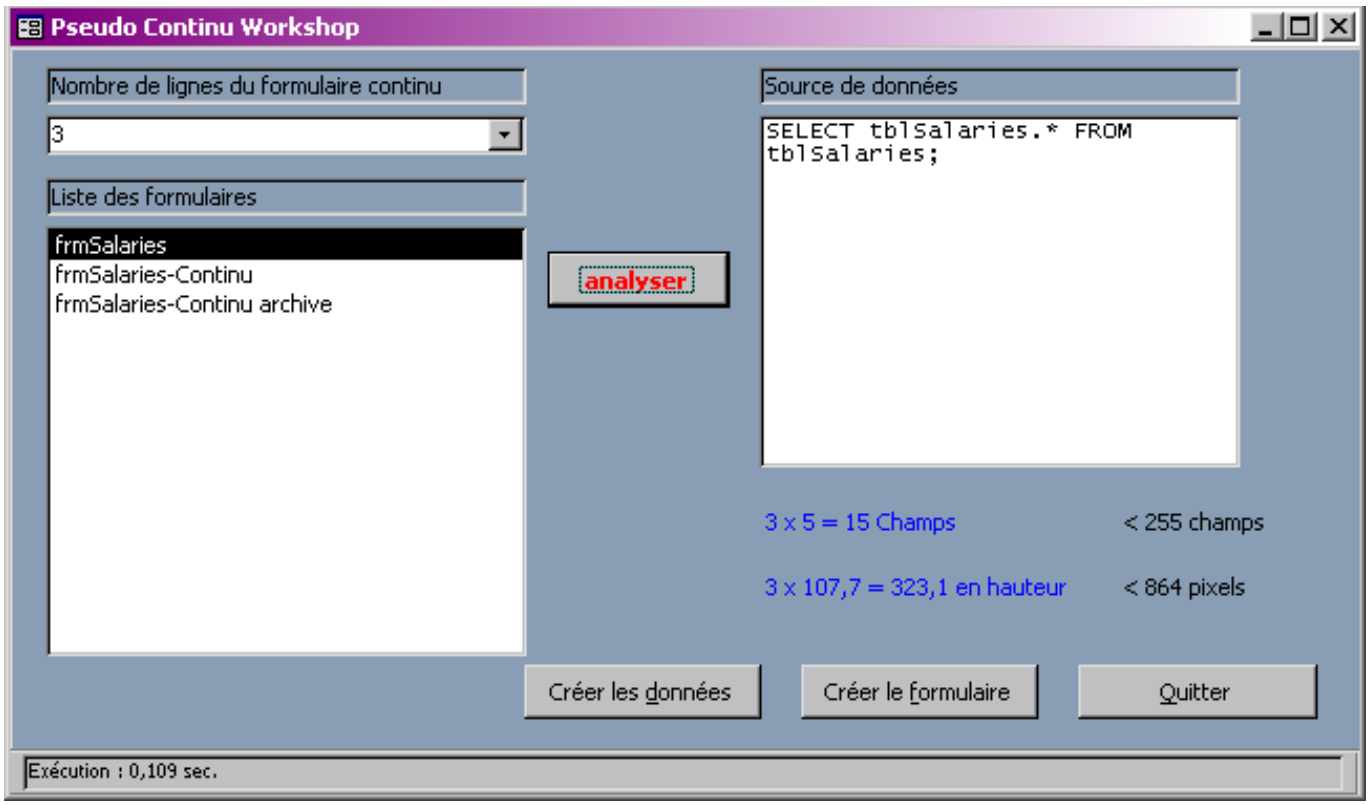
L'application va prendre en charge la construction automatique du formulaire, l'attribution de données et de code de mise à jour des changements.



IV - Analyse du formulaire

A - Nombre de champs de la données source

L'analyse du formulaire se fait au travers ... d'un formulaire, en voici une copie d'écran



Nous créons un bouton pour analyser le formulaire

```
Private Sub cmdAnalyse_Click()

Dim rec As Recordset
Dim NbFields As Integer

' nous n'analyserons pas le formulaire lui meme
'
If Me.lstForms <> Me.Name Then
    ' ouverture du formulaire en mode caché
    DoCmd.OpenForm Me.lstForms, acNormal, , , , acHidden

    ' récupération du recordset (OBJET) sous-jacent au formulaire
    Set rec = Forms(Me.lstForms).RecordsetClone
    ' récupération de la source de données (CHAINE)
    Me.lblSQL.Caption = Forms(Me.lstForms).RecordSource

    ' Utilisation de la propriété .Count de la collection Fields.
```

```
        ' ce nombre sera à multiplier par le nombre de lignes souhaitées
        ' pour calculer le nombre de la champs de la future table démultipliée
        NbFields = rec.Fields.Count

        ' fermeture du recordset et libération de mémoire
        rec.Close
        Set rec = Nothing

        ' Mise à jour des label d'analyse
        Me.lblComment.Caption = Me.cmbNb & " x " & NbFields & " = " & Me.cmbNb * NbFields & " <= 255
    Champs maximum"
        Me.lblScreen.Caption = Me.cmbNb & " x " & Forms(Me.lstForms).Détail.Height / 20 & " = " & _
                                                                Me.cmbNb * Forms(Me.lstForms).Détail.Height
    / 20 & " en hauteur <= " & _
                                GetResolution("V") & " résolution verticale"

        ' Fermeture du formulaire sans sauvegarde
        DoCmd.Close acForm, Me.lstForms, acSaveNo


    Else
        Me.lblSQL.Caption = ""
        Me.lblComment.Caption = "Opération impossible sur le formulaire courant"
        Me.lblScreen.Caption = "Opération impossible sur le formulaire courant"
    End If
```

B - Liste des formulaires

Contrairement aux Tables et Requêtes nous ne pouvons directement accéder aux objets report et form sans les ouvrir.

C'est pourquoi nous les ouvrons en mode caché pour récupérer les informations nécessaires à l'analyse.

Pour alimenter la liste des formulaires, nous utilisons un SQL sur des tables systèmes, nous aurions également pu passer par les Containers.

Pour information voici la liste des containers ...

```
Function listContainers()
Dim i As Integer
For i = 0 To CurrentDb.Containers.Count - 1
    Debug.Print "Container " & i & " : " & CurrentDb.Containers(i).Name
Next i
End Function
```

Exécution

```
listcontainers
Container 0 : DataAccessPages
Container 1 : Databases
Container 2 : Forms
Container 3 : Modules
Container 4 : Relationships
Container 5 : Reports
Container 6 : Scripts
Container 7 : SysRel
Container 8 : Tables
```

La table MsysObjects est une table système qui recense les objets de la base de données, nous la filtrons avec la constante dédiée aux formulaires -32768.

Nous attribuons le SQL suivant à la liste des formulaires.

```
SELECT msysobjects.Id, msysobjects.Name
FROM msysobjects
WHERE (((msysobjects.Type)=-32768));
```

C - Connaître la résolution verticale du futur formulaire

Dernier point d'intérêt, nous récupérons la résolution verticale du système grâce à la fonction GetResolution() dont voici le code

```
Type RECTANGLE
   x1 As Long
   y1 As Long
   x2 As Long
   y2 As Long
End Type


Declare Function GetDesktopWindow Lib "User32" () As Long
Declare Function GetWindowRect Lib "User32" (ByVal hWindow As Long, R As RECTANGLE) As Long


Function GetResolution(ByVal H_V As String) As Long
   Dim R As RECTANGLE
   Dim hWindow As Long
   Dim RC As Long
   Dim WindowResolution As String
   hWindow = GetDesktopWindow()
   RC = GetWindowRect(hWindow, R)
   If H_V = "h" Then
    GetResolution = (R.x2 - R.x1)
   Else
    GetResolution = (R.y2 - R.y1)
   End If
End Function
```

Nous pouvons maintenant passer à la démultiplication des données, accessible depuis le bouton créer données sur notre formulaire.

V - Démultiplication des données

A - Méthode

Cette opération est automatisée par la fonction DemultiplieSource.

Le code est commenté mais en voici la logique.

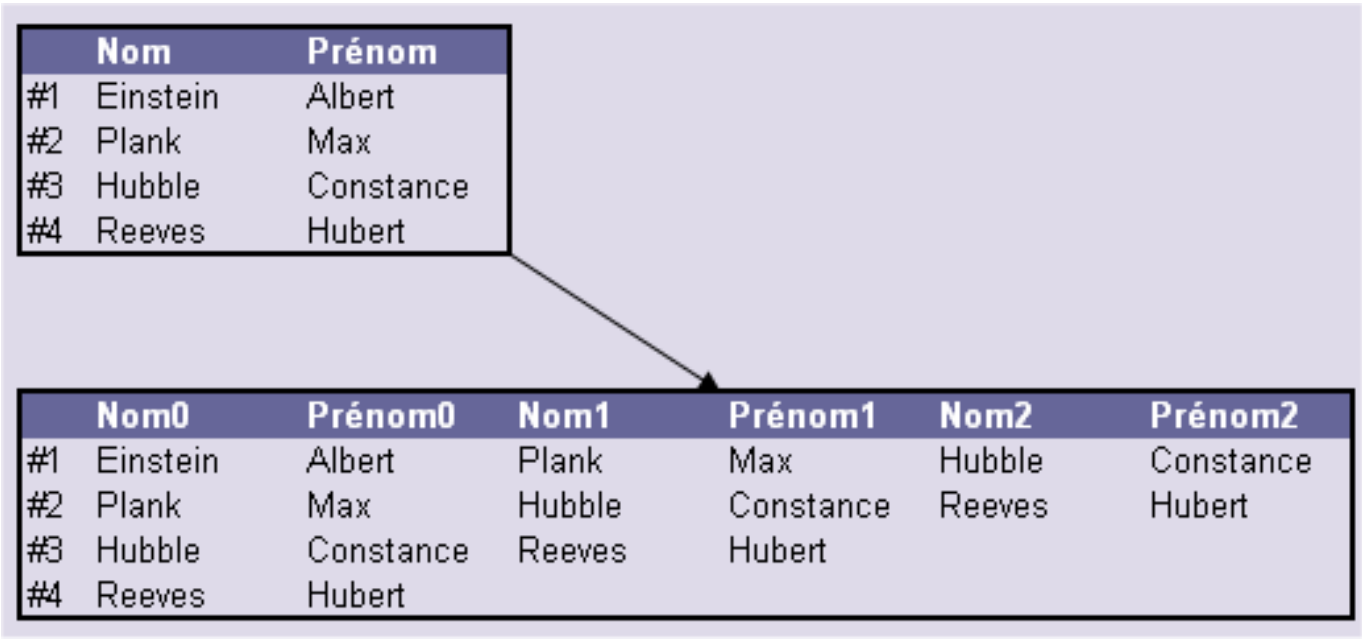
. Vérification que la limite des 255 champs n'est pas atteinte.

. Création de la structure de la table, copiée sur la source de données.

. Alimentation de la nouvelle table, Champ => Champ0, Nom => Nom0 etc.

. Mise à jour de la table.

Le but final est d'obtenir ceci :



B - Le code

Le code suivant opère ces différentes étapes.

```
Function DemultiplieSource(ByVal strTable As String, ByVal intNb As Integer)
' strTable est le nom de la requête qui est source de données à démultiplier
' intNb est le nombre de lignes souhaitées pour le futur formulaire pseudo continu
```

```
    Dim newTbl As TableDef
    Dim fld As Field
    Dim recSource, recDest As Recordset
    Dim i, j As Integer

    On Error GoTo DS01

    'Création des champs
    ' la fonction qui appelle ce code se charge de détruire la table "tblContinue"
    ' Vérification que la limite des 255 champs n'est pas atteinte
    If CurrentDb.TableDefs(strTable).Fields.Count * intNb < 255 Then
        'Création de la table
        Set newTbl = CurrentDb.CreateTableDef("tblContinue")
        'Boucle sur le nombre de lignes du futur formulaire
        For j = 0 To intNb - 1
            'Boucle sur les champs de la donnée source
            ' utilisation intensive de l'objet Fields et des propriétés de ces champs
            For i = 0 To CurrentDb.TableDefs(strTable).Fields.Count - 1
                ' nous nommons le Champ comme l'orginal plus un chiffre Nom devient Nom0, Nom1 ...
                Set fld = newTbl.CreateField(CurrentDb.TableDefs(strTable).Fields(i).Name & j, _
                                             CurrentDb.TableDefs(strTable).Fields(i).Type, _
                                             CurrentDb.TableDefs(strTable).Fields(i).Size)
                ' la méthode .Append ajoute le champ défini lors du Set à la collection Fields
                newTbl.Fields.Append fld
            Next i
        Next j
       ' la méthode .Append ajoute la nouvelle table définie lors du Set à la collection TableDefs
        CurrentDb.TableDefs.Append newTbl


        'Remplissage des données
        Set recSource = CurrentDb.OpenRecordset(strTable, dbOpenDynaset)
        Set recDest = CurrentDb.OpenRecordset("tblContinue", dbOpenDynaset)
        ' ouverture des deux sources de données

        If Not recSource.EOF Then
            ' première passe recopie des champs
            ' recopie des champs à leur place originale
            '   Nom de l'enregistrement 1 sera Nom0 de l'enregistrement 1
            recSource.MoveFirst
            Do While Not recSource.EOF
                recDest.AddNew
                    For j = 0 To recSource.Fields.Count - 1
                        If Len(recSource.Fields(j)) > 0 Then
                            recDest.Fields(j) = recSource.Fields(j)
                        Else
                            recDest.Fields(j) = 0
                        End If
                    Next j
                recDest.Update
                recSource.MoveNext
            Loop

            ' passes suivantes avec décalage
            '   Nom de l'enregistrement 4 sera Nom1 de l'enregistrement 3
            '                                  Nom2 de l'enregistrement 2
            '                                  Nom3 de l'enregistrement 1
            For i = 1 To intNb - 1
                recSource.MoveFirst
                recSource.Move i
                recDest.MoveFirst

                Do While Not recSource.EOF
                    ' la méthode Edit rend modifiable l'enregistrement
                    recDest.Edit
                        For j = 0 To recSource.Fields.Count - 1
                            If Len(recSource.Fields(j)) > 0 Then
                                recDest.Fields(j + (i * recSource.Fields.Count)) = recSource.Fields(j)
                            Else
                                recDest.Fields(j + (i * recSource.Fields.Count)) = 0
                            End If
                        Next j
                    ' la méthode Update sauvegarde les modifications
                    recDest.Update

                    recDest.MoveNext
                    recSource.MoveNext
                Loop
            Next i
```

```
        End If

        ' fermeture des jeux d'enregistrement et libération des ressources
        recSource.Close
        recDest.Close
        Set recSource = Nothing
        Set recDest = Nothing

    Else
        MsgBox "le nombre de champs de la source à passer en mode pseudo continu dépasse 255, " & _
                    "merci de diminuer le nombre de champ ou le nombre de lignes", _
                vbOKOnly + vbCritical
    End If


    Exit Function

    DS01:
    Select Case Err.Number
        Case 3010
            Resume Next
        Case Else
            MsgBox Err.Number & " : " & Err.Description, vbCritical + vbOKOnly
    End Select
    Err.Clear

    End Function
```

C - Points d'intérêt

1 - Comment créer une table

```
    Dim newTbl As TableDef
    Set newTbl = CurrentDb.CreateTableDef("NomDeLaTable")
    CurrentDb.TableDefs.Append newTbl
```

2 - Comment créer un champ

```
    Dim fld As Field
    Set fld = UnObjetTable.CreateField("Nom", acText, 50)
    UnObjetTable.Fields.Append fld
```

Le reste du code consiste à de la manipulation de Recordset dont le maniement est expliqué ailleurs sur developpez.com

La source de données est prête, passons à la création du formulaire.

VI - Création du formulaire

A - Méthode

. ouverture du formulaire source, et parallèlement création du nouveau formulaire.

. récupération de la source de données, démultiplication

. calcul des dimensions du formulaire

. création des contrôles ligne par ligne

. attribution de la source demultipliée

. attribution du module de code de mise à jour

. sauvegarde et nommage du formulaire

B - Le code

```
Function PseudoContinu(ByVal strForm As String, _
                       ByVal intNbLignes As Integer, _
                       Optional ByVal AddForm As Boolean = True)
'
' strForm       :   le nom du formulaire à rendre pseudo continu
' intNbLignes   :   le nombre de lignes du futur formulaire
' addForm       :   demande ou non la création du formulaire
'                   si AddForm = False, la fonction ne fait que créer les données
'                   si AddForm = True, la fonction crée les données et le formulaire


Dim newForm As Form
Dim newCtl As Control
Dim ctl As Control
Dim prp As Property
Dim qry As QueryDef
Dim rec As Recordset
Dim mdl As Module

Dim i As Integer
Dim strSource As String
Dim MaxHeight, MaxWidth, intCount As Long

intCount = 1
PseudoContinuNb = intNbLignes

'Nettoyage des tables
If ExistTable("tblContinue") Then
    DoCmd.SetWarnings False
    DoCmd.DeleteObject acTable, "tblContinue"
    DoCmd.SetWarnings True
End If
```

```
' Ouverture du formulaire à cloner
DoCmd.OpenForm strForm, acDesign, , , , acHidden

' Récupération de la source de données
strSource = Forms(strForm).RecordSource

If InStr(strSource, " ") > 0 Then
    ' il y a un espace dans la source ce qui indique qu'il s'agit d'un SQL et non d'une table ou
requête
    'création de la requete puis creation de la table temporaire
    DoCmd.SetWarnings False
    ' si la requête existe : attribution du SQL
    If ExistQuery("rqtPseudoContinu") Then
        CurrentDb.QueryDefs("rqtPseudoContinu").SQL = strSource
    Else
    ' création de la requête et attribution du SQL
        Set qry = CurrentDb.CreateQueryDef("rqtPseudoContinu", strSource)
        CurrentDb.QueryDefs.Append qry
    End If

    ' nous exécutons une requête création de table pour transformer la source en table
"tblPseudoContinu"
    DoCmd.RunSQL "SELECT * INTO tblPseudoContinu FROM [rqtPseudoContinu]"
    DoCmd.SetWarnings True
    DemultiplieSource "tblPseudoContinu", intNbLignes
    DoCmd.RunSQL "DROP TABLE tblPseudoContinu"
Else
    'indique que la source n'est qu'une table
    'création de la table temporaire
    DemultiplieSource strSource, intNbLignes
End If


If AddForm Then
    ' Création du formulaire pseudo continu
    Set newForm = CreateForm

    ' Trouver le point le plus bas du formulaire
    ' Nous balayons les contrôles Boutons exceptés pour connaître la taille maximale d'une ligne de
notre futur formulaire
    For Each ctl In Forms(strForm).Controls
        If ctl.ControlType <> acCommandButton Then
            If ctl.Top + ctl.Height > MaxHeight Then
                MaxHeight = ctl.Top + ctl.Height
            End If
            If ctl.Left + ctl.Width > MaxWidth Then
                MaxWidth = ctl.Left + ctl.Width
            End If
        End If
    Next ctl

    ' Attribution des dimensions du nouveau formulaire
    ' La largeur est définie par le formulaire : Form.Width
    ' La hauteur est définie par la zone de détail
    newForm.Width = MaxWidth + 10
    newForm.Détail.Height = (MaxHeight + 10) * intNbLignes

    For i = 0 To intNbLignes - 1
        For Each ctl In Forms(strForm).Controls
            Select Case ctl.ControlType
                Case acCommandButton

                Case Else
                    Set newCtl = CreateControl(newForm.Name, _
                                               ctl.ControlType, _
                                               acDetail, , , _
                                               ctl.Left, _
                                               ctl.Top + (MaxHeight + 10) * i, _
                                               ctl.Width, _
                                               ctl.Height)
                    newCtl.Visible = ctl.Visible
                    Select Case ctl.ControlType
                        Case acTextBox, acComboBox, acListBox
                        newCtl.ControlSource = ctl.ControlSource & i

                        Case Else

                    End Select
                    newCtl.Name = ctl.Name & i
```

```
                    Select Case ctl.ControlType
                        Case acLabel
                            newCtl.Caption = ctl.Caption
                        Case Else
                    End Select
                End Select
            Next ctl
            ' Création de la barre de démarquation
            Set newCtl = CreateControl(newForm.Name, acLine, acDetail, , , _
                                       5, _
                                       (MaxHeight + 10) * i + MaxHeight + 2, _
                                       MaxWidth - 10, _
                                       0)
            newCtl.Name = "drwBarre" & i
            newCtl.SpecialEffect = 2
        Next i

        'attribution des données
        newForm.RecordSource = "tblContinue"

        'attribution du formulaire de code
        newForm.HasModule = True
        'attention le fichier contient le code standard doit être dans le chemin de l'appli
        'c'est ce module qui permettra de mettre à jour les modifications
        newForm.Module.AddFromFile CurrentProject.Path & "\modFormContinu.cls"

        'sauvegarde
        newForm.Tag = strForm & "-pseudo continu" & intNbLignes
        DoCmd.Close acForm, newForm.Name, acSaveYes
    End If

    'fermeture du formulaire source, inutile de sauvegarder
    DoCmd.Close acForm, strForm, acSaveNo

    'renommage du formulaire
    For i = 0 To CurrentDb.Containers(2).Documents.Count - 1
        If CurrentDb.Containers(2).Documents(i).DateCreated > Now - 10 / 24 / 3600 Then
            'document créé depuis moins de 10 secondes c'est le bon
            DoCmd.Rename strForm & "-Continu", acForm, CurrentDb.Containers(2).Documents(i).Name
        End If
    Next i


    End Function
```

C - Points d'intérêt

1 - Savoir si une table ou une requête existe

Ce code nous permet d'utiliser les fonctions très pratiques de test d'existence de table et requête.

```
    Function ExistQuery(ByVal strQry As String) As Boolean

    On Error GoTo EQ01
    If CurrentDb.QueryDefs(strQry).Name = strQry Then
        ExistQuery = True
        Exit Function
    Else
        ExistQuery = False
    End If
    EQ01:
    Select Case Err.Number
        Case 3265
            ExistQuery = False
        Case Else
            ExistQuery = False
    End Select
    Err.Clear

    End Function

    Function ExistTable(ByVal strTbl As String) As Boolean
```

```
On Error GoTo EQ01
If CurrentDb.TableDefs(strTbl).Name = strTbl Then
    ExistTable = True
    Exit Function
Else
    ExistTable = False
End If
EQ01:
Select Case Err.Number
    Case 3265
        ExistTable = False
    Case Else
        ExistTable = False
End Select
Err.Clear

End Function
```

2 - Manipuler un objet QueryDef

QueryDef est la définition d'une requête.

QueryDefs est la collection des objets requête.

Créer une requête avec CreateQueryDef et .Append

```
Dim qry As QueryDef
Set qry = CurrentDb.CreateQueryDef("NomDeLaRequete", ChaineSQL)
CurrentDb.QueryDefs.Append qry
```

Attribuer un SQL à une requête

```
CurrentDb.QueryDefs("NomDeLaRequete").SQL = ChaineSQL
```

3 - Manipuler un objet Form

Créer un formulaire

```
Dim newForm as Form
Set newForm = CreateForm
```

Parcourir les contrôles d'un formulaire **OUVERT**.

```
Dim ctl As Control
    For Each ctl In Forms("MonFormulaire").Controls
                ' votre code
    Next ctl
```

4 - Créer des contrôles

```
Dim newCtl as Control
Set newCtl = CreateControl("MonForm", acTextBox, "ZoneduForm", , , 0, 0, 100, 30)
' O => Gauche, 0 => Haut, 100 => Largeur, 30 => Hauteur
newCtl.Name = "MonControle"
```

Attribuer une source

```
newCtl.ControlSource = "Nomd-unChampdeLaSourceLieeAuFormulaire"
```

5 - Attribuer un module à un formulaire

Nous avons de côté un module sous forme de fichier texte que notre code intègre au formulaire.

Son fonctionnement sera décrit dans la partie suivante

```
ObjetFormulaire.HasModule = True
ObjetFormulaire.Module.AddFromFile VariableChainePointantSurUnFichierTexte
```

Nous avons attribué un module de code qui permettra de mettre à jours les enregistrements dans la table d'origine qui n'est maintenant plus liée à notre formulaire.

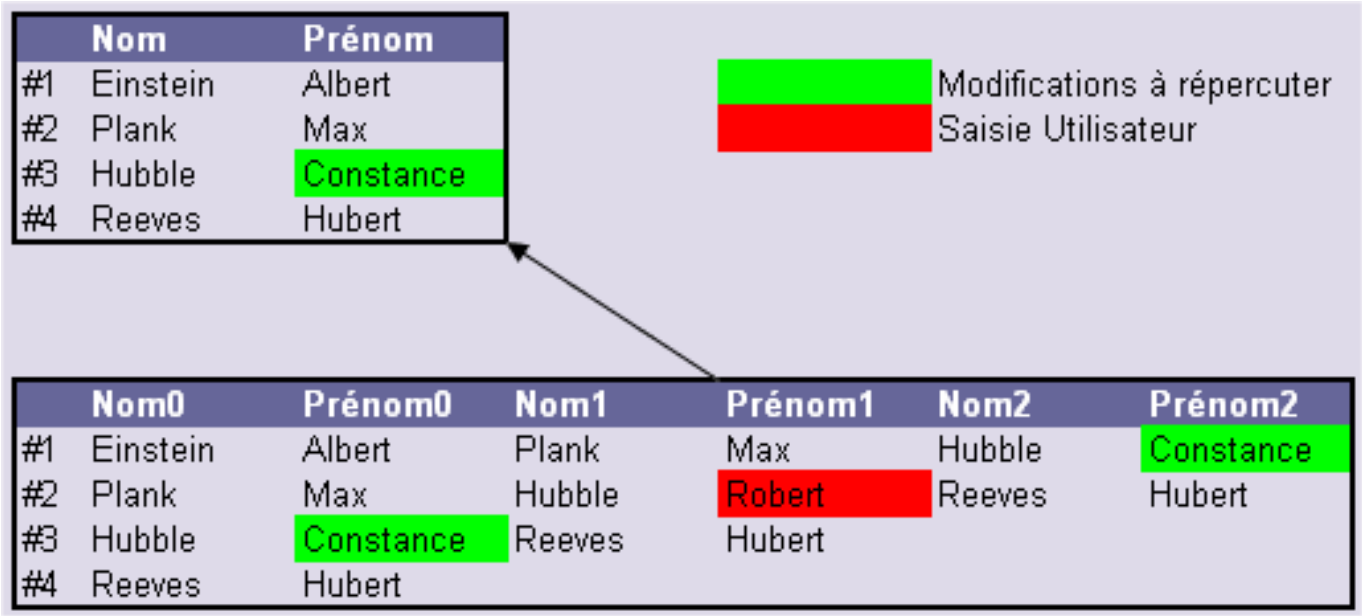
VII - Fonctions de mise à jour des données

Le formulaire fonctionne, mais si l'utilisateur modifie des données, la table source n'est pas mise à jour, nous allons remédier à cela.

A - Méthode

Il faut impérativement qu'une modification d'un enregistrement soit répercutée sur la table source, mais aussi sur les enregistrements démultipliés.

Comme précisé sur le schéma ci-dessous



. traquer les modifications dans les contrôles du formulaire pseudo continu

. mettre à jour la table source

. mettre à jour les enregistrements suivants et précédents

B - Le code

```
Private Sub Form_Load()
Dim ctl As Control

For Each ctl In Me.Controls
    Select Case ctl.ControlType
        Case acTextBox, acListBox, acComboBox, acCheckBox, _
```

```
             acOptionButton, acOptionGroup
            'représente la quasi majorité des cas possibles
            ' à adapter si besoin
            ctl.BeforeUpdate = "=GoUpdate('" & ctl.Name & "', " & CurrentRecord & ")"
        Case Else
            'ne rien faire

    End Select
Next ctl

Set ctl = Nothing

End Sub


Function GoUpdate(ByVal ctlName As String, _
                  ByVal CurrRec As Long)

Dim i, j As Integer
Dim intRelative As Integer
Dim intCont As Integer
Dim lngMaxRow, lngMinRow As Long
Dim rec As Recordset

CurrRec = Me.CurrentRecord


' récupère le nombre maximum de contrôles
intCont = CInt(Right(Me.Tag, 1))

' mise à jour des enregistrements précédents et suivants
Set rec = Me.RecordsetClone
intRelative = CInt(Right(ctlName, 1))
lngMaxRow = intRelative + CurrRec
lngMinRow = Max(1, lngMaxRow - intCont + 1)


rec.MoveFirst
rec.Move lngMinRow - 1
For j = Min(intCont - 1, lngMaxRow - 1) To 0 Step -1
    If j <> intRelative Then
        rec.Edit
        rec.Fields(Left(ctlName, Len(ctlName) - 1) & j).Value = Me.Controls(ctlName)
        rec.Update
    End If
    rec.MoveNext
Next j
rec.Close

' mise à jour de la table source
Set rec = CurrentDb.OpenRecordset("rqtPseudoContinu", dbOpenDynaset)
rec.MoveFirst
rec.Move lngMaxRow - 1
rec.Edit
    rec.Fields(Left(ctlName, Len(ctlName) - 1)).Value = Me.Controls(ctlName)
rec.Update
rec.Close


Set rec = Nothing

End Function


Function Max(ByVal n1 As Long, ByVal n2 As Long) As Long
If n1 >= n2 Then
    Max = n1
Else
    Max = n2
End If
End Function


Function Min(ByVal n1 As Long, ByVal n2 As Long) As Long
If n1 < n2 Then
    Min = n1
Else
    Min = n2
End If
```

```
End Function
```

C - Points d'intérêt

1 - Traquer les modifications

```
ctl.BeforeUpdate = "=GoUpdate('" & ctl.Name & "', " & CurrentRecord & ")"
```

Nous attribuons une fonction sur l'événement BeforeUpdate d'un contrôle.

Cette attribution de code en balayant dynamiquement les contrôles au chargement du formulaire (OnLoad) permet de ne pas avoir à écrire toutes les fonctions Private Sub MonControle_BeforeUpdate() qui seraient impossible à prévoir toutes.

CurrentRecord permet de connaître le numéro de l'enregistrement courant, indispensable pour le retrouver dans la table source.

2 - Effectuer les mises à jour

Nous manipulons le recordset du formulaire grâce à RecordSetClone, qui est le clone du recordset du formulaire. Je pense qu'il est peu utile ici de détailler la manière dont le code se déplace dans les jeux d'enregistrement.

VIII - Application au tutoriel Photos

Nous avons vu comment notre application pouvait construire totalement un formulaire pseudo-continu.

Nous allons maintenant l'exploiter dans le cadre de notre précédent tutoriel de gestion de photos.

En adaptant quelque peu le code nous allons pouvoir obtenir un formulaire qui a l'apparence d'être continu et qui permet néanmoins d'afficher une photo différente par "ligne".

```
Private Sub Form_Current()
Dim ctl As Control

' Gestion des erreurs
'On Error GoTo Catch02

' si la photo n'est pas définie, on affiche la photo blank.jpg
For Each ctl In Me.Controls
    If ctl.ControlType = acImage Then
        If Len(Me.Controls("Photo" & Right(ctl.Name, 1))) > 0 Then
            ctl.Picture = Me.Controls("Photo" & Right(ctl.Name, 1))
        Else
            ctl.Picture = CurrentProject.Path & "\images\blank.jpg"
        End If
    DisplayPhoto ctl.Name
    End If
Next ctl


Exit Sub

Catch02:
Select Case Err.Number
    Case 2114
        'Cas d'un type de fichier photo non supporté ...
        MsgBox "Le format de l'image n'est supporté par le contrôle image Picture", vbCritical +
vbOKOnly, "Application Photos"
    Case 2220
        'Cas d'un emplacement non valide du fichier images
        MsgBox "Le fichier image n'a pas été trouvé à l'emplacement indiqué", vbCritical + vbOKOnly,
"Application Photos"
    Case Else
        ' tout autre cas d'erreur
        MsgBox "Erreur inattendue : " & Err.Number & vbCrLf & Err.Description, vbCritical +
vbOKOnly, "Application Photos"
End Select
Err.Clear

End Sub


Sub DisplayPhoto(ByVal ctlImage As String)
' Traitement en fonction de la taille de l'image

' regarde si la hauteur de l'image dépasse celle du controle Picture
If Me.Controls(ctlImage).ImageHeight > Me.Controls(ctlImage).Height Then
    ' met le controle en mode zoom
    Me.Controls(ctlImage).SizeMode = 3
Else
    ' met le contrôle en mode respect de la taille originale
    Me.Controls(ctlImage).SizeMode = 0
End If

' si la largeur dépasse et qu'on est en mode taille réelle ...
If (Me.Controls(ctlImage).ImageWidth > Me.Controls(ctlImage).Width) And
(Me.Controls(ctlImage).SizeMode) = 0 Then
    ' on met en mode zoom
    Me.Controls(ctlImage).SizeMode = 3
End If
```

```
End Sub
```

IX - Conclusions

Notre application fonctionne et permet une navigation de type continu tout en affichant bien la photo liée à chaque salarié.

Cet article-tutoriel montre qu'on peut toujours arriver à ses fins sur Access même si la dépense d'énergie n'est pas toujours appropriée à la demande des utilisateurs.

D'autres solutions pour remédier aux limites des formulaires continus existent, certaines vous seront bientôt présentées ici.

D'ici là bon codage, et merci de m'avoir lu jusqu'au bout ;)