Formulaire de recherche prêt à l'emploi 3ème partie.

par Fabrice CONSTANS (autres articles)

Date de publication : 13/07/2006

Dernière mise à jour : 01/10/2009

Cet article est le troisième et dernier de la série consacrée à la recherche dans Microsoft ACCESS. Il fait suite aux deux précédents articles : **formulaire de recherche générique** et **Fonctions supplémentaires**. Ce dernier volet ajoutera plusieurs fonctions très attendues par bon nombre d'entre vous.

I - Avertissement

L'utilisation de la touche F1 est vivement conseillée à tous les stades de l'utilisation d'ACCESS. L'amélioration constante de l'aide en fait un partenaire de choix dans l'apprentissage permanent d'ACCESS. Personnellement, je ne peux m'en passer, ne serait-ce que pour mémoire.

II - Correction sur le code précédent

Dans le code précédent il existe une erreur découlant de la correction des crochets à utiliser pour les noms de tables et de champs contenant des espaces.
Cette erreur se manifeste par l'affichage systématique du message suivant :

```
La recherche précédente ne porte pas sur la même table que la recherche actuelle.
```

Allez dans le code du bouton de recherche et repérez le code suivant :

Code à repérer

```
If Not Me.lst_resultat.RowSource Like "*FROM " & strTable & "*" Then
```

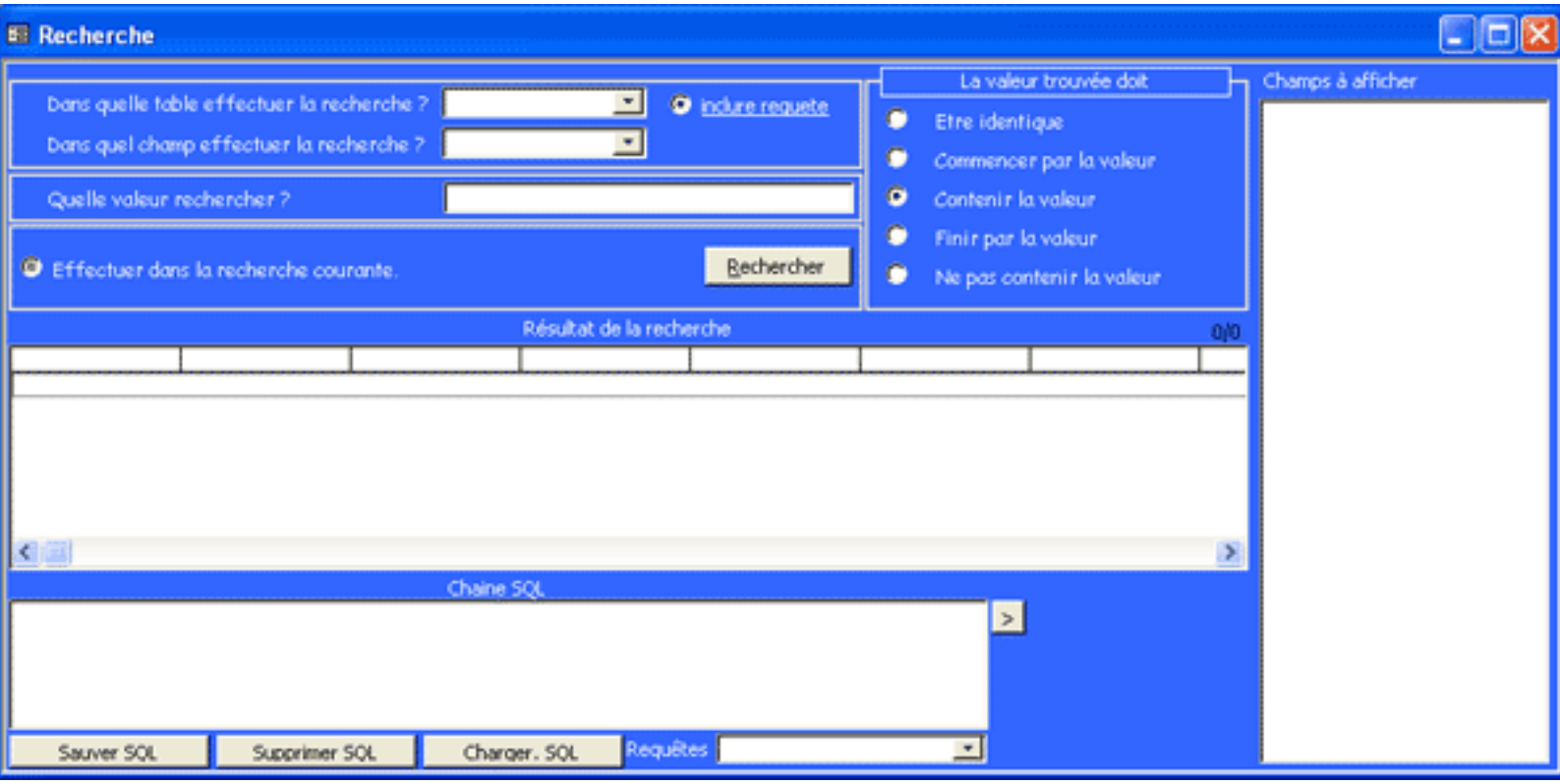
Code à insérer à la place

```
    Dim ctrl_table As String
    ctrl_table = Left(strTable, Len(strTable) - 1)
    ctrl_table = Right(ctrl_table, Len(ctrl_table) - 1)
    If Not Me.lst_resultat.RowSource Like "*FROM [[]" & ctrl_table & "*" Then
```

Les crochets font partie des caractères utilisés par l'opérateur Like ils permettent de rechercher des caractères contenus entre 2 bornes.
Exemple :
[A-E] recherche tous les caractères entre A et E inclus.
Il fallait donc les retirer de la variable **strTable** (le left et le right) puis indiquez dans la recherche que nous souhaitions trouver un crochet ouvrant "**[**". Ce que nous avons fait en écrivant "**[[]**".
Encore une autre bonne raison de ne pas utiliser d'espaces dans les noms d'objets, champs, tables, contrôles etc. !

III - Présentation des nouvelles fonctionnalités

1 Recherche multi-tables liées.
2 Export des données vers Microsoft EXCEL
3 Réglage automatique des largeurs de colonnes
4 Choix de l'état d'impression



L'état précédent du formulaire.

IV - Recherche multi-tables liées

Un grand nombre d'entre vous attendait une solution de recherche multi-tables, c'est ce que nous allons mettre en place au cours de ce chapitre.
Nous pourrions explorer les relations de la base courante puis les traduire en code SQL mais même si cette solution est la plus professionnelle souvent il vaut mieux consacrer son temps de développement pour des choses incontournables. Comme vous l'aurez compris nous allons une nouvelle fois contourner le problème.

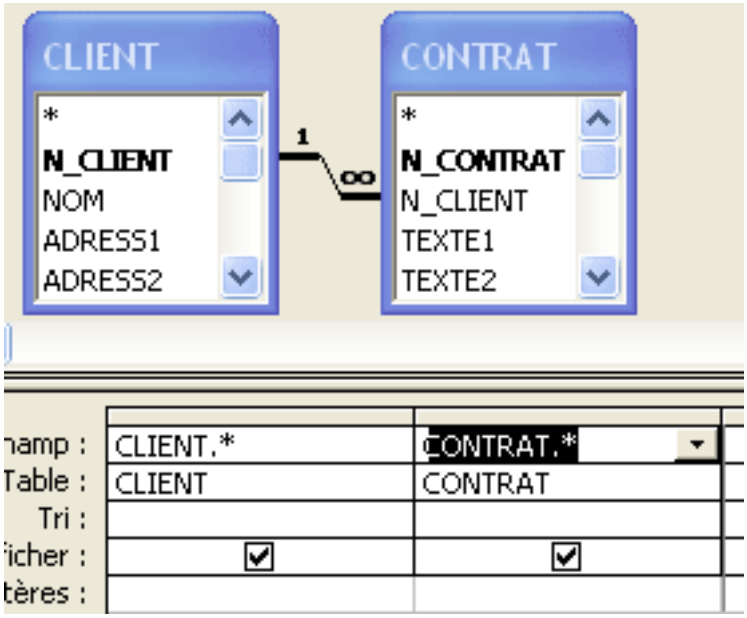
IV-A - Description de la solution

Pour cette solution le développeur ou l'administrateur, vous même en l'occurrence, devra créer une requête pour chaque relation. Nous utiliserons ces requêtes en tant que source comme s'il s'agissait d'une table.
Comme cette fonctionnalité peut ne pas intéresser l'utilisateur final, nous la rendrons optionnelle.
Nous utiliserons du code déjà fait que nous adapterons pour la circonstance. A vos claviers !

IV-B - Préparation - La requête source

Nous allons dans un premier temps créer une requête contenant la relation. Celle-ci nous servira pour les tests.

- Ouvrez votre application, si ce n'est déjà fait.
- Dans la fenêtre **Base de données** cliquez sur **Requêtes** dans la zone **Objets**
- Cliquez sur **Créer une requête en mode Création**
- La fenêtre **Ajouter une table** s'ouvre
- Sélectionnez 2 tables normalement liées dans votre schéma relationnel
- La liaison apparaît en même temps que les tables, dans le cas contraire vérifiez votre schéma relationnel ou sélectionnez des tables possédant une liaison valide
- Faites un glisser/poser du premier item de la liste des 2 tables, l'item est représenté par une étoile (*).



Cela devrait donner ceci... excepté les noms de tables et de champs.

- Enregistrez la requête sous le nom suivant : RQS_TableMere_TableEnfant

Notez que *TableMère* et *TableEnfant* doivent être remplacés par le nom des tables respectives. Le préfixe **RQS_** sera le repère de sélection qui nous permettra de capitaliser les requêtes concernées.
L'item * représente l'ensemble des champs. Cette requête affichera tous les champs des 2 tables. Vous pouvez choisir de sélectionner que les champs sur lesquels vous souhaitez effectuer les recherches dans ces requêtes.

TRES IMPORTANT ! Dans les requêtes mettant en relation plusieurs tables il arrive que des champs portent le même nom. Assurez-vous qu'une telle situation ne se produira pas pour les champs clefs utilisés pour l'ouverture des formulaires. La notation TABLE.CHAMP n'est pas compatible avec la fonction d'ouverture des formulaires.

IV-C - Le contrôle d'option

Nous allons insérer un bouton d'option qui permettra à l'utilisateur de choisir des tables et des requêtes ou seulement des tables dans la liste.

- Ouvrez le formulaire de recherche en mode création,
- Sélectionnez dans la barre d'outils des contrôles le bouton d'option,
- Insérez-le à droite de la liste des tables.

Réglez les propriétés du contrôle comme suit :

Bou d'opt Séle requ	Propriété	Valeur
	Nom	optInclureRequete
	Valeur par défaut	0
	Légende	Inclure requêtes

IV-D - Le code

Pour le code nous allons réutiliser un code que nous connaissons bien puisqu'il s'agit de celui qui capitalise les requêtes utilisateurs (préfixe **USER_**).

- Ouvrez la fenêtre du code du formulaire
- Recherchez la fonction **lf_GetQueryList()**
- Faites un copier/coller
- Renommez la fonction en **lf_GetQueryListOnTable()**
- Faites les modifications suivantes (ou recopiez le code):

Code à insérer/modifier

```
Function lf_GetQueryListOnTable()
' renseigne la table tbl_TemplstTbl

Dim qrs As QueryDefs
Dim rst As Recordset

Dim strSql As String
Dim i As Integer, j As Integer

'------ MODIFICATION ---------------------
' On n'efface pas les enregistrements
'DoCmd.SetWarnings False
'strSql = "Delete tbl_TempLstTbl.*"
'strSql = strSql + " FROM tbl_TempLstTbl;"
'DoCmd.RunSQL strSql
'------ FIN MODIFICATION -----------------
```

Code à insérer/modifier

```
' rempli la table temporaire
Set qrs = CurrentDb.QueryDefs
'------ MODIFICATION --------------------
Set rst = CurrentDb.OpenRecordset("tbl_TempLstTbl")
'------ FIN MODIFICATION ----------------

For i = 0 To qrs.Count - 1
'------ MODIFICATION --------------------
    ' ne prend que les requete qui commence par RQS_
    If qrs(i).Name Like "RQS_*" Then
'------ FIN MODIFICATION ----------------
        rst.AddNew
        rst.Fields(0) = qrs(i).Name
        rst.Update
    End If
Next
'------ MODIFICATION --------------------
lf_GetQueryListOnTable = rst.RecordCount
'------ FIN MODIFICATION ----------------
rst.Close
Set rst = Nothing
Set qrs = Nothing
'DoCmd.SetWarnings True

End Function
```

Ce n'est pas la seule modification à apporter puisque jusqu'à présent nous avions travaillé avec des tables (objets TableDefs). Maintenant que nous exploitons à la fois des tables et des requêtes nous devons modifier le code qui permet de trouver le type du champ de la recherche.

Code à modifier

```
Function lf_GetTypeField(lfNameTbl As String, lfNameFld As String)
' utilise la référence Microsoft DAO 3.6 Object Library
' Renvoie le numéro du type du champ
'lfNameTbl = nom de la table
'lfNameFld = nom du champ

    Dim dbs As Database             ' Objet de la base
    Dim tbl As TableDef             ' Objet de définition de table
'----------- MODIFICATION ---------------
    Dim qrd As QueryDef    ' Objet de définition de requete
'----------- FIN MODIFICATION -----------
    Set dbs = CurrentDb             ' ouvre la base courante

'----------- MODIFICATION ---------------
    If lfNameTbl like "*RQS_*" Then
       Set qrd = dbs.QueryDefs(lfNameTbl)            ' ouvre la définition table
       lf_GetTypeField = qrd.Fields(lfNameFld).TYPE  ' renvoie le type de champ
    Else
       Set tbl = dbs.TableDefs(lfNameTbl)            ' ouvre la définition table
       lf_GetTypeField = tbl.Fields(lfNameFld).TYPE  ' renvoie le type de champ
    End If
'----------- FIN MODIFICATION -----------

    Set tbl = Nothing               ' libération des objets
'----------- MODIFICATION ---------------
    Set qrd = Nothing
'----------- FIN MODIFICATION -----------
    Set dbs = Nothing

End Function
```

Maintenant nous allons nous occuper du lancement de cette fonction.

- Toujours dans le code du formulaire recherchez la procédure **Form_Open**
- Faites les modifications suivantes

Code à modifier

```
Private Sub Form_Open(Cancel As Integer)

' crée la liste des tables
'----------- MODIFICATION ---------------
If lf_GetTableList() = 0 And lf_GetQueryListOnTable() = 0 Then
   MsgBox "Pas de tables ou de requêtes dans cette application .", vbInformation + vbOKOnly, "Erreur"
'----------- FIN MODIFICATION -----------

   Cancel = True
End If

lf_GetQueryList 'alimente la table pour cbo_query

Opt_RechCourante_Click    'cache la zone liste et l'étiquette

End Sub
```

Dans l'état actuel la nouvelle fonctionnalité est opérationnelle. Il manque à mettre le code sur le contrôle **opt_inclureRequete**.

- Allez dans les propriétés du bouton d'option
- Repérez la propriété **Sur Clic**
- Créez une procédure événementielle
- Insérez le code suivant.

Code à insérer

```
Private Sub opt_inclureRequete_Click()

If Me.opt_inclureRequete Then
   lf_GetTableList
   lf_GetQueryListOnTable
Else
   lf_GetTableList
End If

Me.cbo_table.Requery
End Sub
```

Lorsque l'utilisateur clique sur le bouton d'option la liste des tables est recréée et on y ajoute la liste des requêtes **RQS_** si le bouton d'option est coché.

La fonctionnalité est opérationnelle, il ne manque plus qu'à créer les requêtes suivant vos besoins.
Nous aurions pu également utiliser le **Container** des relations pour trouver et mettre en oeuvre cette fonctionnalité.

V - Export EXCEL

Nous avons souvent besoin d'exporter des données d'une base Microsoft ACCESS vers d'autres produits Office comme EXCEL. EXCEL permet de manipuler facilement les données pour obtenir des tableaux de bord. Beaucoup d'utilisateurs de produits Office utilisent le copier/coller pour les transferts.
Nous allons introduire dans le module de recherche une procédure d'exportation vers EXCEL.

*Attention cette fonction nécessite OBLIGATOIREMENT la présence d'EXCEL installé sur tous les postes ou cette application sera installée. Vous devez impérativement ajouter la référence **Microsoft EXCEL x.x Object Library**. Ou x.x est la version de votre EXCEL. 8.0 pour 97, 9.0 pour 2000 etc.*

V-A - Description de la solution

Il existe plusieurs méthodes pour exporter des données. Le plus simple mais également le plus rigide consiste à utiliser les commandes d'exportations. Elles sont disponibles via la commande **Docmd**.

- *DoCmd.OutputTo typeobjet[, nomobjet][, formatdesortie][, fichierdecopie][, lancementautomatique][, fichiermodèle]*
  Exporte n'importe quel objet de la base de données courante vers de nombreux formats génériques.
- *DoCmd.TransferSpreadsheet [typetransfert][, typefeuille], nomtable, nomfichier [, contientnomschamps][, étendue]*
  Exporte une table vers EXCEL ou Lotus. Cette commande permet également d'importer ou de lier des données depuis ces produits.
- *DoCmd.SendObject [typeobjet][, nomobjet][, formatdesortie][, à][, cc][, ccc][, sujet][, textemessage][, modifiermessage][, fichiermodèle]*
  Exporte un objet de la base de données courante dans plusieurs formats vers un mail.

V-B - Le contrôle

Pour lancer l'export nous avons besoin d'un simple bouton de commande.

Bou d comm Exp ve EX	Propriété	Valeur
	Nom	Cmd_Export
	Légende	Export vers EXCEL
	[Procédure Événementielle]	

V-C - Le code

Dans un premier temps il faut vérifier que l'export est valide. Nous pourrions très bien tester que le contenu de la propriété **Source** de la liste du résultat (**me.Lst_resultat.rowsource**) est bien rempli, mais nous ne saurions pas si la liste contient des lignes. Nous allons contrôler plutôt que la liste contient des enregistrements grâce à la propriété **ListCount**.
Cette propriété est disponible en lecture seule sous VBA pour les zones de liste et zones de liste modifiable.
Nous faisons ensuite appel à la fonction **lf_Export2EXCEL** avec comme paramètre le contenu de Rowsource de la liste des résultats. Nous envoyons donc directement la chaîne SQL à la fonction.

Code à insérer

```
Private Sub cmd_Export_Click()
    ' ne contient pas d'enregistrement
    If Me.lst_resultat.ListCount = 1 Then
       MsgBox "Pas de données à exporter.", vbOKOnly, "Export EXCEL"
       Exit Sub
    End If
    ' Exporte vers EXCEL
    lf_Export2EXCEL Me.lst_resultat.RowSource
End Sub
```

Vous remarquerez que notre test indique que s'il y a une ligne présente dans la liste **If Me.lst_resultat.ListCount = 1 Then** nous considérons qu'il n'y a pas de données. Pourquoi ?
Tout simplement parce que notre liste contient une ligne pour les noms de champs et celle-ci est incluse dans le décompte. (voir l'aide Microsoft ACCESS - ListCount et ColumnHeads
La fonction suivante concerne l'export lui-même. Notez le paramètre optionnel **Optional strNameFile** vous pouvez passer un nom de fichier particulier à la place de celui par défaut.

Code à insérer

```
Function lf_Export2EXCEL(strSQL, Optional strNameFile As String)

    On Error GoTo Err_lf_Export2EXCEL

    ' le sablier à On
    DoCmd.Hourglass True

    ' vérifie que le fichier xls n'existe pas dans le chemin
    If Len(strNameFile) = 0 Then strNameFile = "Export.xls"

    strNameFile = Environ("USERPROFILE") & "\Mes Documents\" & strNameFile

    If Len(Dir(strNameFile)) = 0 Then     ' teste si le fichier existe

        ' crée la requete Temp avec la SQL select
        CurrentDb.CreateQueryDef "Temp", strSQL

        '   Crée une sortie au format EXCEL
        DoCmd.OutputTo acOutputQuery, "Temp", acFormatXLS, strNameFile, True
        '   supprime la query Temp
        CurrentDb.QueryDefs.Delete "Temp"

    Else   ' le fichier existe on écrit à sa suite

        Dim oExcel As Excel.Application ' l'application (évite l'erreur 462)
        Dim oFeuille As Worksheet       ' la feuille
        Dim oWork As Workbook           ' le workbook

        Dim rst As Recordset            ' le recordset ACCESS
        Dim l As Long, i As Long, c As Long        ' pour les déplacements dans la feuille
        ' ouvre instance Excel
        Set oExcel = New Excel.Application
        ' feuille invisible
        oExcel.Visible = False
        ' ouvre le fichier

        Set oWork = oExcel.Workbooks.Open(strNameFile)

        ' active la 1ere feuille
        Set oFeuille = oExcel.ActiveSheet

        '   recupère le n° de la dernière ligne rempli + 1
        l = oFeuille.Cells.SpecialCells(xlCellTypeLastCell).Row

        '   ouvre la requete avec la SQL select
        Set rst = CurrentDb.OpenRecordset(strSQL, dbOpenSnapshot, dbForwardOnly)

        ' compte le nombre de champs à copier (initialise le compteur c )
```

Code à insérer

```
        c = rst.Fields.Count
        ' rajoute 1
        If l > 1 Then l = l + 1

        If MsgBox("Souhaitez-vous nettoyer le fichier EXCEL ?", vbYesNo, "Export EXCEL") = vbYes Then
            ' option nettoyage de feuille ouverte
            oFeuille.Rows("1:65536").ClearContents
            oFeuille.Rows("1:65536").ClearFormats
            oFeuille.Rows("1:65536").Clear
            ' debut de fichier
            l = 1
        End If

        If MsgBox("Souhaitez-vous insérer les noms des champs ?", vbYesNo, "Export EXCEL") = vbYes Then
            '   copie le nom des champs sur la première ligne
            For i = 1 To c
                oFeuille.Cells(l, i) = rst(i - 1).Properties("Caption")
                oFeuille.Cells(l, i).Interior.Color = RGB(192, 192, 192) ' c'est le gris Excel
            Next i          '   traite chaque record
            ' ligne suivante
            l = l + 1
        End If

        ' copie le recordset
        oFeuille.Cells(l, 1).CopyFromRecordset rst

        '--------ACCESS-------------
        '   ferme le recordset libère l'objet
        rst.Close
        Set rst = Nothing

        '--------EXCEL--------------
        '   ajuste les cellules
        oFeuille.Rows.AutoFit
        '   rend la feuille visible
        oExcel.Application.Visible = True
        '   active la fenetre principale EXCEL
        oExcel.Windows(1).Visible = True
        '   sauve la feuille EXCEL
        oWork.Close (True)
        oExcel.Quit
        '   ferme l'objet xls
        Set oFeuille = Nothing
        Set oWork = Nothing
        Set oExcel = Nothing
    End If

Exit_lf_Export2EXCEL:
' le sablier à off
    DoCmd.Hourglass False
    Exit Function

Err_lf_Export2EXCEL:
    If Err.Number = 3012 Then
        CurrentDb.QueryDefs.Delete "Temp"
        Resume
    End If
    If Err.Number = 3270 Then  ' remplace le Caption par le Name
       oFeuille.Cells(l, i) = rst(i - 1).Properties("Name")
       Resume Next
    End If
    MsgBox Err.Number & " " & Err.Description, vbCritical, "Erreur"
    '------- EXCEL ---------
    oExcel.Visible = True
    Set oFeuille = Nothing
    Set oWork = Nothing
    Set oExcel = Nothing
    DoCmd.Hourglass False

End Function
```

Cette fonction travaille de 2 manières. La première est utilisée dans le cas ou le fichier par défaut **Export.xls** n'existe pas, l'export se fait par la commande **Docmd.OutputTo**. Celle-ci a l'avantage de créer le fichier au format EXCEL.
La deuxième est utilisée lorsque le fichier existe. Dans ce cas nous utilisons l'automation (Pilotage d'EXCEL à partir d'ACCESS). Cela nous permet de repérer la dernière ligne renseignée **.Cells.SpecialCells(xlCellTypeLastCell).Row** et d'insérer nos enregistrements à la suite **- Méthode CopyFromRecordset -**.
Il y a 2 options, le nettoyage du fichier **.Rows("1:65536").ClearContents** et la copie des noms de champs **.Cells(l, i) = rst(i - 1).Name**. Bien entendu cette fonction n'est pas figée vous pouvez la faire évoluer comme vous le souhaitez.

VI - Largeur de colonnes dynamique

Si le réglage des propriétés des contrôles est simple à réaliser sous Microsoft ACCESS, les dimensions et le positionnement de ceux-ci est un peu plus compliqué.

VI-A - Description de la solution

Nous allons utiliser la boucle de la sélection des champs pour mettre en place notre fonction. Le but est de calculer pour chaque champ sélectionné le nombre maximal de caractères que la requête va afficher. Ce nombre sera converti en centimètre puisque c'est l'unité utilisée pour les largeurs de listes. Le résultat sera affecté à la propriété de la liste résultat.

VI-B - Le Code

Le code est inclus au bouton de recherche.

1 Ouvrez le code du bouton de **Recherche**
2 Recherchez le code suivant.

Code à modifier

```
    ' debut de selection des champs
    Dim strChamps As String
    Dim entCurrLigne As Integer

 For entCurrLigne = 0 To Me.lst_champs.ListCount - 1
        If Me.lst_champs.Selected(entCurrLigne) Then
            strChamps = strChamps & "[" & Me.lst_champs.Column(0, entCurrLigne) & "], "
  End If
    Next entCurrLigne

    If Len(strChamps) = 0 Then
        strChamps = strTable & ".*"
    Else
        strChamps = Left(strChamps, Len(strChamps) - 2)
    End If
    ' fin de selection des champs
```

1 Opérez les modifications suivantes

Code modifié

```
    ' debut de selection des champs
    Dim strChamps As String
    Dim entCurrLigne As Integer

'----------- MODIFICATION ---------------
    Dim strLenCol As String    ' la variable
'----------- FIN MODIFICATION -----------

 For entCurrLigne = 0 To Me.lst_champs.ListCount - 1
        If Me.lst_champs.Selected(entCurrLigne) Then
            strChamps = strChamps & "[" & Me.lst_champs.Column(0, entCurrLigne) & "], "

'----------- MODIFICATION ---------------
   ' Largeur de colonne dynamique
            If Not strLenCol = "" Then strLenCol = strLenCol & "; "
            strLenCol = strLenCol & Round((Nz(DMax(Eval("""len([" & _
                        Me.lst_champs.Column(0, entCurrLigne) & "])"""), _
                        strTable, strCriteria), 0) * 130) / 571, 2) & " cm"

            ' méthode WizHook.TwipsFromFont Voir tuto de Cafeine
```

Code modifié

```
            'strLenCol = strLenCol & Round(GetTextLength(Me.lst_resultat, _
                        String(Nz(DMax(Eval("""len([" & Me.lst_champs.Column(0,
 entCurrLigne) & "])"""), _
                        strTable, strCriteria), 0), "u"), False) / 571, 2) & " cm"

            ' fin Largeur de colonne dynamique
'----------- FIN MODIFICATION -----------

  End If
    Next entCurrLigne

'----------- MODIFICATION ---------------
    Me.lst_resultat.ColumnWidths = strLenCol    ' Affecte Largeur de colonne dynamique
'----------- FIN MODIFICATION -----------

    If Len(strChamps) = 0 Then
        strChamps = strTable & ".*"
    Else
        strChamps = Left(strChamps, Len(strChamps) - 2)
    End If
    ' fin de selection des champs
```

Cette fonction comporte 3 parties que nous détaillons ci-dessous.

1 La variable **strLenCol** va contenir les longueurs calculées. Son contenu est de type String (texte) et contient ce type de chaine **"2,52 cm;3 cm;0,05 cm"**
2 Lors du traitement de chaque champ le contenu de la variable est testé, si celui-ci n'est pas vide le séparateur ; est ajouté à la fin de la variable.
La longueur du plus long résultat est calculé puis converti en cm et ajouté à la suite de la variable.
3 Enfin nous affectons la variable à la propriété **ColumnWidths** de notre liste de résultat **lst_resultat**

Si la majorité du code introduit ne comporte pas de difficultés la ligne qui calcule la longueur mérite d'être détaillée. Non seulement celle-ci trouve le nombre de caractères de la valeur la plus longue de la sélection en cours, mais elle convertie également ce nombre de caractères en centimètre.
Le tableau ci-dessous détaille chaque fonction de l'expression.
Nz(DMax(Eval("""len(" & Me.lst_champs.Column(0, entCurrLigne) & ")""") , strTable, strCriteria),0)

Dét de fonc	Nom	Description
	Round(*expression*, *nombre de décimale*)	Renvoie un nombre de décimales spécifiées.
	Nz(*expression*, *valeur retournée*)	Renvoie une valeur si expression est Null. Elle est équivalente à

Iif(Isnull(expression),Valeur retournée,expression) dans les versions antérieure à 2000.
Dmax(*"Champ"*,*"Table"*,*"Critère"*) la plus grande valeur de **"Champ"** de la table **"Table"** suivant le critère **"Critère"**. Notez que "Champ" peut être une expression retournant un valeur suivant le contenu d'un champ. **"Table"** est un nom de table ou de requête et critère utilise la syntaxe de la clause **Where**

des requêtes
Eval(*expression*) et renvoie la valeur de l'expression. La valeur retournée est de type texte ou numérique.
Len(*expression*) le nombre de caractères de l'expression.

Me.lst_champs.Column(0, entCurrLigne) renvoie le nom du champ en cours de traitement, il faut l'entourer de crochets [] pour éviter des erreurs avec des noms de champs comportant des caractères spéciaux ou des espaces.

*Remarquez la construction particulière du paramètre **"Champ"** de la fonction **Dmax()**.*

Si nous observions la ligne au moment de son exécution nous pourrions voir ceci :
DMax("Len([Monchamp])","MaTable","le critère courant")
La fonction **Dmax** peut donc renvoyer la valeur la plus grande d'un champ mais également d'une expression calculée pour un champ. Dans notre cas c'est le nombre de caractères de la plus longue valeur.
Une fois ce nombre connu il est soumis au calcul "*** 130 / 571**" pour trouver sa largeur en centimètre. Nous commençons par le multiplier par 130 pour trouver sa largeur en **Twips**. Multiplié par 130 (De 130 à 160 c'est la largeur moyenne d'un caractère de taille 10 en Twips pour un écran 1024/768) puis divisé par 571 (rapport de conversion Twips/Centimétre) qui trouvera approximativement son équivalent en centimètre. Il s'agit d'une approximation du fait de plusieurs facteurs :

- Le Twips dépend de la résolution de l'écran.
- Les caractères des polices proportionnelles n'ont pas tous la même largeur. Un i est moins large qu'un o et les majuscules sont généralement plus large que les minuscules.
- La taille des caractères utilisée dans la liste résultat.
- Certain type de données sont codées d'une manière dans la table et affichées différemment : c'est le cas du bouléen (Oui/Non) par exemple :
  Dans la table il est codé 0 ou -1 et à l'affichage ACCESS le transforme automatiquement (pour une meilleure lecture) en Oui et Non, la largeur est donc inférieure au contenu affiché.

Vous pouvez jouer avec la valeur 130 pour élargir ou rétrécir les colonnes ou encore utiliser la méthode de **Caféine** dans son excellent tutoriel **Formulaire Auto-Extensible** Dans ce cas remplacez le code par celui-ci :

```
        ' méthode WizHook.TwipsFromFont Voir tuto de Cafeine
        strLenCol = strLenCol & Round(GetTextLength(Me.lst_resultat, _
```

```
                        String(Nz(DMax(Eval("""len([" & Me.lst_champs.Column(0,
entCurrLigne) & "])"""), _
                        strTable, strCriteria), 0), "u"), False) / 571, 2) & " cm"
```

N'oubliez pas d'insérer également la fonction **GetTextLenght** dans le module. Pour cette fonction, son explication et ses paramètres veuillez vous référer au tutoriel de Caféine.

VII - Les états

Après avoir vu les formulaires nous allons mettre en oeuvre les états. Pour cela nous avons besoin de quelques états si vous n'en avez pas vous pouvez les créer en vous aidant de l'assistant.

*La source (propriété **RowSource**) de l'état doit être la même que l'item de la liste **cbo_table**.*

VII-A - Table, contrôle ; les besoins

Cette fonctionnalité ressemble à celle utilisée pour les formulaires c'est pour cela que nous allons créer une table très ressemblante à **tbl_TempLstFrm**.

Ta tbl_Tem	Nom du champ	Type	Longueur
	75 Te 75 Te		

Dans cette table insérez le nom de chaque état et de la source correspondante.

La source peut être une table ou une requête

Nous avons également besoin d'un bouton de commande pour ouvrir l'état.

Bou d comm Ou l'é	Propriété	Valeur
	Nom [Pro Sur clic Légende de l'étiquette	cmd_imprime [Procédure Evènementielle]

Nous n'avons pas besoin de liste pour le choix de l'état puisque celui-ci est sélectionné automatiquement, cependant vous pouvez facilement modifier le code pour intégrer une liste d'états.

VII-B - Le Code

Pour cette fonctionalité l'approche est différente de celle mise en place pour les formulaires. En effet pour ces derniers nous avions utilisé le champ clef pour visualiser/éditer un seul enregistrement. Dans le cas présent nous allons utiliser la condition Where de notre chaine SQL pour l'appliquer au formulaire.

Vous pouvez très bien appliquer cette méthode aux formulaires et même donner le choix à l'utilisateur pour l'un ou l'autre des résultats.

Dans l'événement **Sur Clic** du bouton de commande **cmd_imprime** insérez le code suivant.

Code à insérer

```
Private Sub cmd_imprime_Click()
    Dim rst As Recordset
    Dim strCriteria As String

    Set rst = CurrentDb.OpenRecordset("tbl_TempLstRpt", dbOpenSnapshot)
    ' recherche les informations de la table
    rst.FindFirst ("Table='" & Me.cbo_table & "'")

    If rst.NoMatch Then     ' non trouvé
        MsgBox "Cette table ne possède pas d'état. " & _
               "Veuillez renseigner la table des paramètres.", _
               vbCritical + vbOKOnly, "formulaire de Recherche"
        Exit Sub
    Else    ' trouvé
       DoCmd.OpenReport rst.Fields("Etat"), acViewPreview, , lf_GetSqlWhere
    End If

    Set rst = Nothing

End Sub
```

N'oubliez pas la fonction **lf_GetSqlWhere** qui récupère la clause **Where** de la syntaxe SQL.

Code à insérer

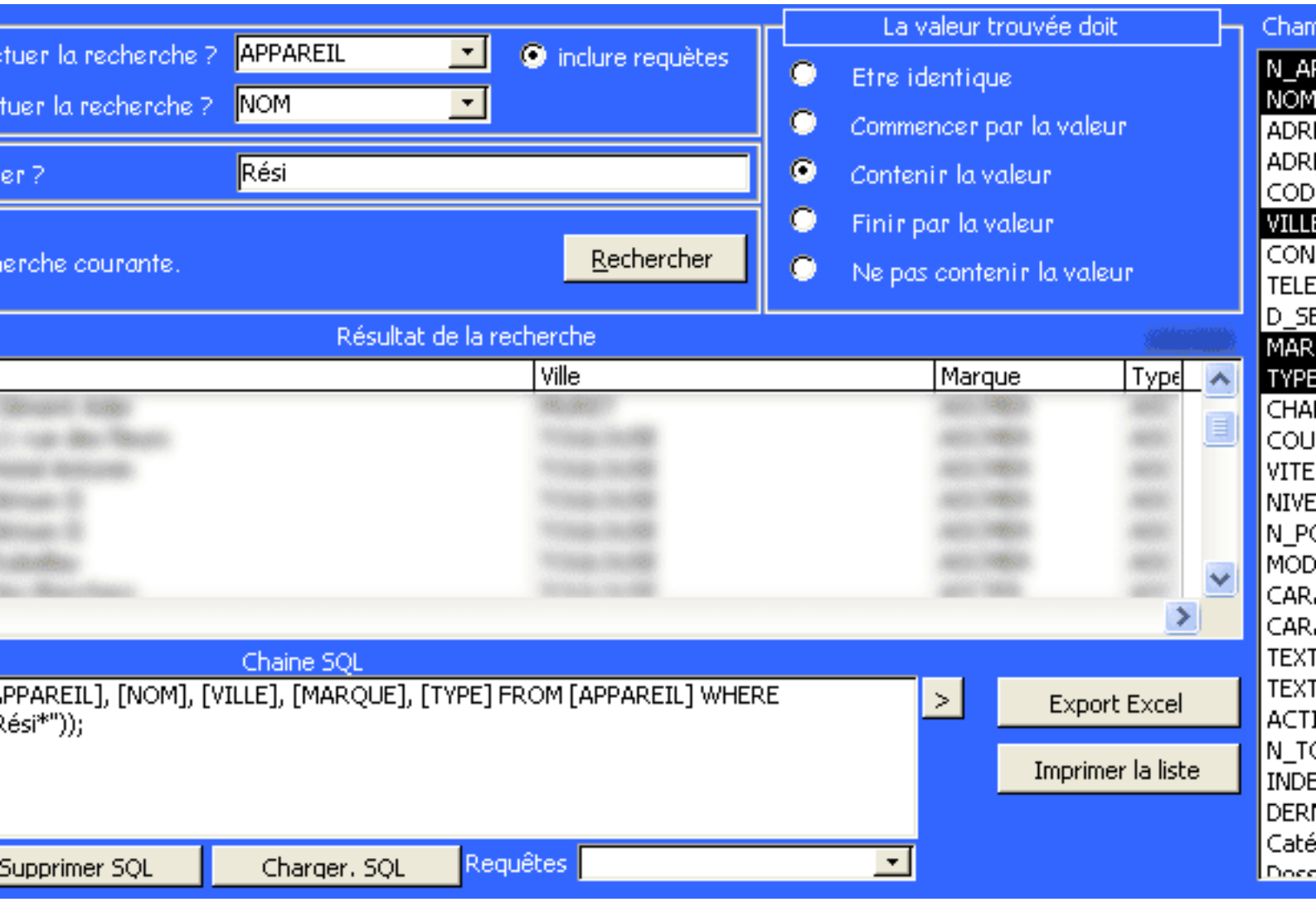
```
Function lf_GetSqlWhere()
Dim strWhere As String
Dim strSQL As String

 strSQL = Me.lst_resultat.RowSource
 ' récupère à partir des doubles paranthèses
 strWhere = Right(strSQL, Len(strSQL) - InStrRev(strSQL, "(("))
 ' supprime les caractères inutile de la fin
 strWhere = Left(strWhere, Len(strWhere) - 2)

 'on renvoi le résultat
 lf_GetSqlWhere = strWhere
End Function
```

Notez que nous faisons référence à Me donc cette fonction doit être incluse dans le module du formulaire.

Vous pouvez réutiliser cette fonction directement dans la syntaxe d'ouverture du formulaire comme pour la syntaxe d'ouverture de l'état.



Voilà... C'est fini (JL AUBERT)

VIII - Conclusion

Le 3ème volet de la Recherche est terminé. Nous avons pu constater que Microsoft ACCESS est loin d'être un produit limité et qu'avec un peu d'imagination nous pouvons réaliser beaucoup de choses. N'hésitez pas à nous soumettre vos évolutions sur ce formulaire de recherche.

IX - Remerciements

Je tiens à remercier : **Caféine** pour son tutoriel **Formulaire Auto-Extensible** qui m'a donné une méthode plus académique pour traiter les largeurs de champ.
Dolphy35 pour le temps passé en relecture et correction.
Les nombreux Devnautes qui m'ont apporté leurs judicieuses remarques sur les 2 premières parties.
À l'équipe de **Developpez.com** pour la qualité du site.
À **Nono40** pour son super éditeur XML et ses superbes évolutions.
Je présente mes plus plates excuses à ceux que j'aurais omis de remercier.