Formulaire de recherche prêt à l'emploi 2ème partie.

par Fabrice CONSTANS (autres articles)

Date de publication : 4/11/2005

Dernière mise à jour : 01/10/2009

Cet article est le deuxième de la série consacrée à la recherche dans Microsoft ACCESS. Au terme du **précédent article** nous avions créé un formulaire de recherche générique. Dans celui-ci nous allons agrémenter notre formulaire de plusieurs fonctions importantes.

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 - Conseils importants sur la programmation

Suite à des remarques sur le tutoriel précédent, je tiens à donner quelques conseils sur la manière de nommer certains objets dans Microsoft ACCESS comme les formulaires, états, tables et champs.
Même si l'utilisation d'**espaces** dans les noms de tables et de champs est autorisée dans Microsoft ACCESS, il est fortement **déconseillé** de le faire. En effet chaque fois qu'un nom comporte au moins un espace il faut l'entourer de crochets []. Dans le QBE (générateur de requête) ces crochets peuvent être insérés automatiquement. Mais le QBE peut interpréter également le nom comme une chaîne de caractères ou au pire provoquer une erreur.
Dans le code VBA il faut toujours le faire manuellement ce qui ralenti l'écriture du code et peut causer des erreurs.

Remplacez l'espace par l'underscore _.
Lisez l'excellent tutoriel d' ***Argyronet*** *sur les conventions d'écriture dans VBA.*

Pour éviter des problèmes avec les espaces et autres caractères veuillez appliquer cette modification dans **Cmd_recherche_click()**

Code à modifier

```
strTable = Me.cbo_Table          ' recupère le nom de la table
strField = Me.cbo_Champ          ' recupère le nom du champ
```

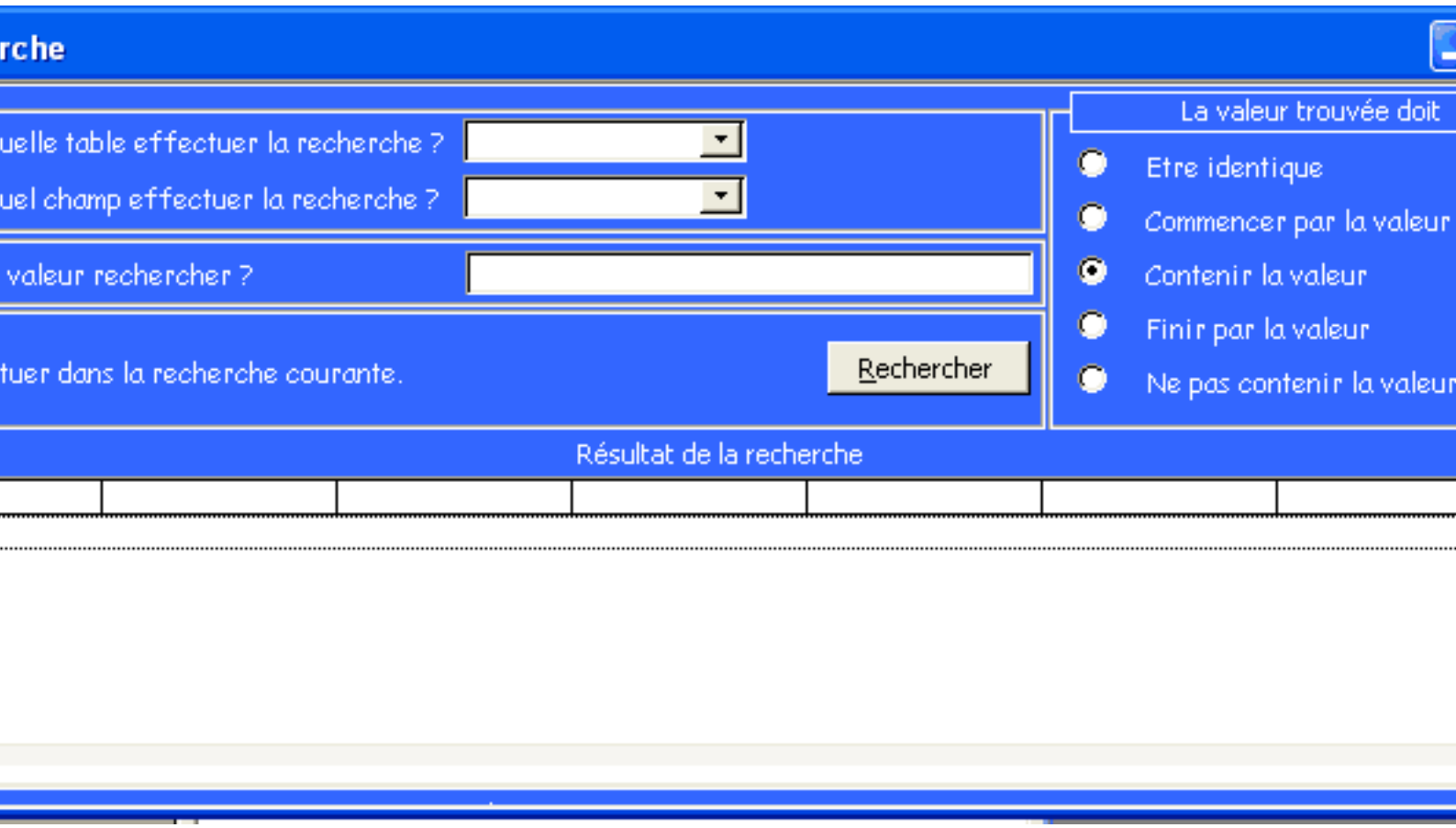
Code modifié

```
strTable = "[" & Me.cbo_Table & "]"          ' recupère le nom de la table
strField = "[" & Me.cbo_Champ & "]"          ' recupère le nom du champ
```

III - Présentation des nouvelles fonctionnalités

Les fonctionnalités ajoutées sont les suivantes :

1 Visualisation du code SQL généré
2 Gestion du code SQL
3 Traitement des opérateurs logiques principaux (AND, OR)
4 Traitement de la valeur Null
5 Ouverture d'un formulaire de visualisation
6 Choix des champs à visualiser dans la liste de résultat



L'état précédent du formulaire.

IV - Comment éviter les sorties intempestives en mode run-time

En environnement Run-time la moindre erreur non traitée se solde par une sortie sauvage. Dans l'état actuel de notre formulaire si un utilisateur clique sur le bouton Rechercher sans avoir défini la table ou le champ, la sortie est inévitable.
Pour l'éviter il vaut mieux faire quelques tests.
Ouvrez le code sur bouton **Rechercher**.
Recherchez le code suivant qui est en début de procédure.

Code à repérer

```
    Dim Criter As Variant
```

A la suite insérez le code de contrôle suivant.

Code à insérer

```
    If IsNull(Me.cbo_table) Or IsNull(Me.cbo_champ) Then

   MsgBox "Vous devez renseigner la table et le champ pour effectuer une recherche !", vbExclamation + vbOKOnly, "
       Exit Sub
    End If
```

Ici nous contrôlons que les 2 listes de choix (table et champ) contiennent bien une sélection. Sinon nous adressons un petit message à l'utilisateur et sortons de la procédure.

V - Visualisation/exécution du code SQL généré

Notre moteur de recherche étant opérationnel nous pouvons regarder et pourquoi pas manipuler la chaîne SQL générée. Avec un contrôle texte pour visualiser et modifier la chaîne SQL et un bouton pour l'exécuter nous pourrons facilement mettre en oeuvre cette fonctionnalité.

IV.A - Visualiser le code SQL

Rajoutez le contrôle suivant :

Zo tex Affic d l syn SC	Propriété	Valeur
	Nom	txt_ChaineSQL
	Partie	Partie
	Barre de défilement	Barre de défilement
	Légende	Légende
	Défaut	SQL
	Etiquette	l'étiquette

Prévoyez un contrôle de la largeur de la liste de résultat **lst_resultat** et de 3 à 4 lignes pour avoir un bon confort de lecture.

*La propriété **Rowsource** d'un contrôle est limitée à 1024 caractères.*
Un objet requête peut contenir une chaîne SQL d'environ 64 000 caractères.
Un contrôle texte peut contenir 65 535 caractères au maximum.
Nous ne pouvons donc pas générer des chaînes SQL de plus 1024 caractères.

Sélectionnez le bouton **Rechercher**
Editez l'évènement **Sur clic**
Placez-vous à la fin du code juste avant le **End Sub**.
Insérez le code suivant :

Code à insérer

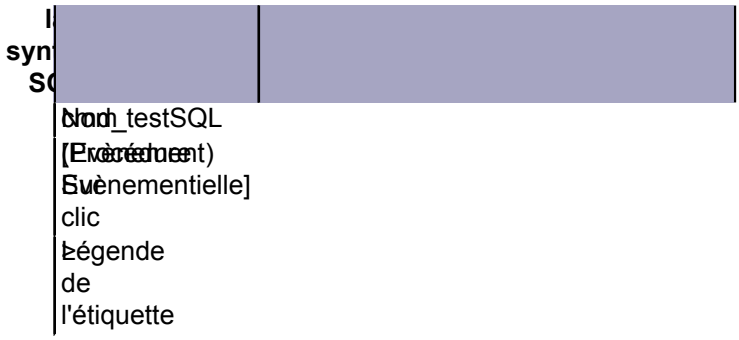
```
    Me.txt_chaineSQL.Value = strSql           ' affiche le code
```

Faites un petit test... Toujours pas de vaudou ni de magie. Nous affectons simplement la chaîne SQL contenu dans strSQL dans notre contrôle texte.

IV-C - Tester un code modifié

Toujours à l'aide de quelques instructions nous allons démontrer que nous pouvons visualiser le résultat d'une chaîne SQL modifiée.
Rajoutez le contrôle suivant :

Bou d comm Te	Propriété	Valeur



Dans la procédure événementielle **Sur Clic**, insérez le code suivant :

Code à insérer

```
' visualise la query
If IsNull(Me.txt_chaineSQL) Then Exit Sub     ' pas de chaine SQL
Me.lst_resultat.RowSource = Me.txt_chaineSQL  ' Affecte la chaine pour visualisation
Me.lst_resultat.Requery                       ' reffraichit la liste
```

Sauvegardez et passez en mode Formulaire pour faire un test.
Faites une recherche simple, sur un champ texte en utilisant l'option **Contenir la valeur**

VI - Gestion des chaînes SQL

Nous sommes souvent ammenés à faire les mêmes recherches et il peut être très pratique de pouvoir rappeler des chaînes SQL précédemment créées, surtout si celles-ci comportent plusieurs conditions.
Au stade actuel nous pouvons facilement concevoir un système de gestion des chaînes SQL.
Nous allons utiliser l'objet **Requête** pour stocker nos chaînes, cela nous permettra de pouvoir générer directement des requêtes.
Pour une gestion simple nous aurons besoin d'une zone de liste déroulante (combo) pour afficher la liste des requêtes, d'un bouton de sauvegarde et un de suppression.

Notez que nous pourrions également utiliser le stockage dans un champ Mémo d'une table.

V-A - Table et contrôles

Nous allons utiliser le même système qui nous a servi à faire l'inventaire des tables pour réaliser celui des requêtes.
Pour cela nous avons besoin de créer une table, par chance la structure de celle-ci est identique à **tbl_TempLstTbl**.
Nous allons en faire une copie.
Dans la liste des tables, sélectionnez la table **tbl_TempLstTbl** faites un **Ctrl+C** ou **clic-droit/copier** pour faire la copie de l'objet puis un **Ctrl+V** ou **clic-droit/coller** pour faire apparaitre le panneau suivant.

Le panneau

Entrez le nom suivant **tbl_TempLstQry** et choisissez le bouton d'option **Structure seulement**.
Validez à l'aide du bouton **Ok** ou appuyer sur la touche **Entrée**.
La table est créée. Nous pouvons passer aux contrôles.
Commencez par une zone de liste déroulante. Créez-la suivant les propriétés suivantes :

Zo lis modif lis	Propriété	Valeur

de requ		
	Nom	Nom_Query
	Catégorie	Requête
	Source	Requête
	Contrôle	Cb_TempLstQry
	Légende	Requêtes de l'étiquette

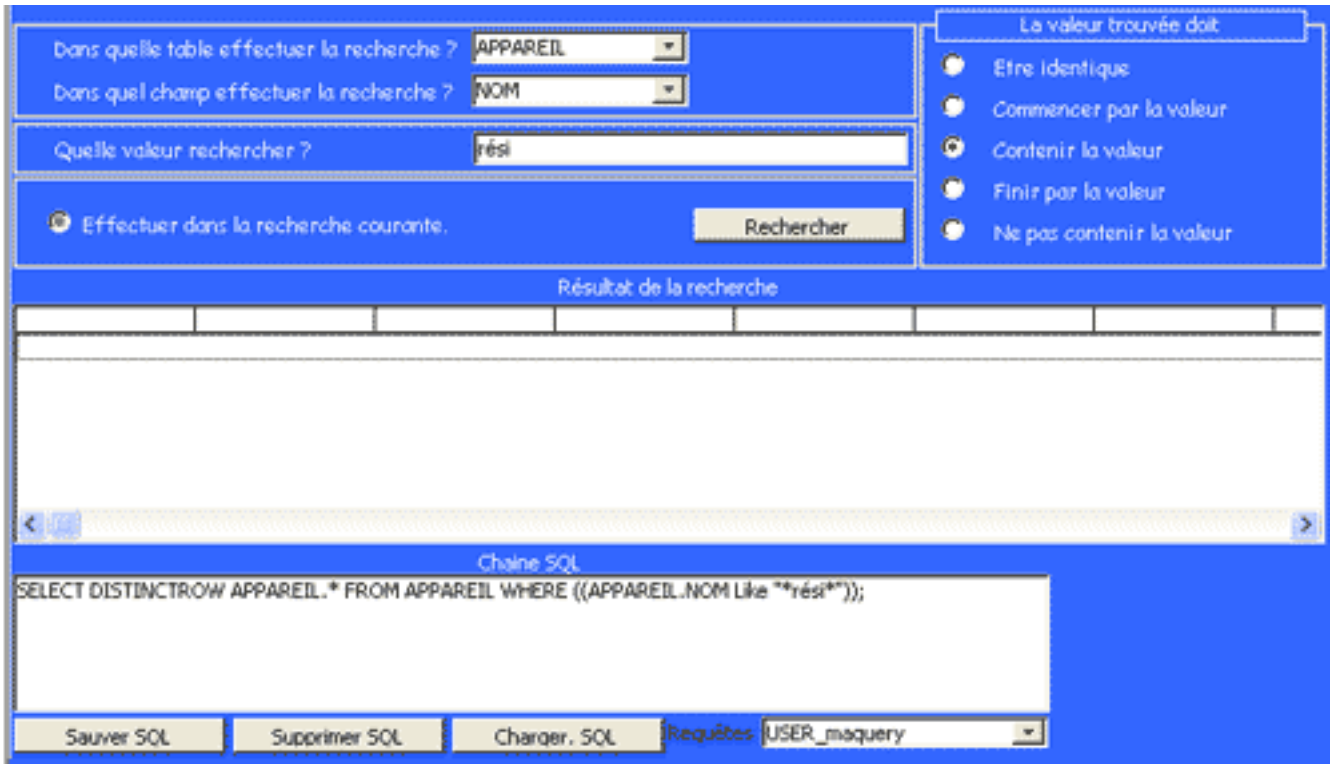
Créez ensuite les 3 boutons de commande nécessaires, Sauver SQL, Supprimer SQL et Charger SQL.

Bou d comm Sau SQ	Propriété	Valeur
	Nom	Nom_saveSQL
	Légende	Sauver SQL
	[Procédure Événementielle]	Sur clic

Bou d comm Supp SQ	Propriété	Valeur
	Nom	Nom_suppSQL
	Légende	Supprimer SQL
	[Procédure Événementielle]	Sur clic

Bou d comm Cha SQ	Propriété	Valeur
	Nom	Nom_LoadSQL
	Légende	Charger SQL
	[Procédure Événementielle]	Sur clic

Voici notre formulaire avec les nouveaux contrôles.



Avec les nouveaux contrôles...

V-B - Les codes VBA

V-B-1 - Alimentation de la liste des requête

La zone de liste modifiable **cbo_Query** est alimentée de la même manière que pour la zone liste **cbo_Table** qui contient les noms de tables . Nous modifions la fonction **lf_GetTableList()** pour faire l'inventaire des requêtes. Insérez le code suivant :

Code à insérer

```
Function lf_GetQueryList()
' renseigne la table tbl_TemplstQry

Dim qrs As QueryDefs
Dim rst As Recordset

Dim strSql As String
Dim i As Integer

' efface la table temporaire
DoCmd.SetWarnings False
strSql = "Delete tbl_TempLstQry.*"
strSql = strSql + " FROM tbl_TempLstQry;"
DoCmd.RunSQL strSql

' rempli    la table temporaire
Set qrs = CurrentDb.QueryDefs
Set rst = CurrentDb.OpenRecordset("tbl_TempLstQry")

For i = 0 To qrs.Count - 1
    ' Ne prend que le query USER_....
    If qrs(i).Name Like "USER_*" Then
       rst.AddNew
       rst.Fields(0) = qrs(i).Name
       rst.Update
```

Code à insérer
```
    End If
Next

lf_GetQueryList = rst.RecordCount

rst.Close
Set rst = Nothing
Set qrs = Nothing
DoCmd.SetWarnings True

End Function
```

Dans ce code nous pouvons constater que :
Nous remplaçons l'objet **Tabledefs** par **Querydefs**.
Nous remplaçons la table **tbl_TempLstTbl** par **tbl_TempLstQry**.
Nous ne recensons que les requêtes commençant par "USER_". Ceci permet de faire la différence entre les requêtes utilisateurs et celles qui pourraient être présente dans l'application.
Cette liste doit être toujours à jour, nous devons donc l'alimenter à l'ouverture du formulaire puis après chaque action d'ajout et de suppression de requête "USER_".
Ouvrez le code du formulaire et dans la procédure **Form_Open** (propriété Sur Ouverture) insérez avant le **End Sub** le code suivant :

Code à insérer
```
lf_GetQueryList       'alimente la table pour cbo_query
me.cbo_query.Requery  'raffraichit la liste
```

V-B-2 - Sauvegarde de la chaîne SQL

La sauvegarde ne représente pas une grosse difficulté technique puisqu'il existe une méthode, cependant la procédure n'est pas constituée que de la sauvegarde.

Elle doit :

- demander le nom de la nouvelle requête à l'utilisateur
- vérifier que ce nom ne soit pas déjà attribué
- sauvegarder la chaîne sous la forme d'une requête.

Créer le code évenementiel **Sur clic** pour le bouton **cmd_saveSQL** insérez le code suivant.

Code à insérer
```
' controle d'existance
If IsNull(Me.txt_chaineSQL) Then    ' chaine vide
   MsgBox "La chaine SQL est vide. Sauvegarde inutile.", vbExclamation + vbOKOnly, "erreur"
   Exit Sub
End If

Dim strQuery As String

' demande le nom de la nouvelle query
strQuery = "USER_" & InputBox("Insérez le nom de la requête à créer (245 caractères max.):" _
           & vbCrLf & "Ce nom sera précédé de USER_ .", "Nom de requête")
' vérifie la non présence
If IsNull(DLookup("Nom", "tbl_TempLstQry", "Nom=""" & strQuery & """")) Then
   ' sauve la query
   CurrentDb.CreateQueryDef strQuery, Me.txt_chaineSQL
   lf_GetQueryList 'alimente la table pour cbo_query
   Me.cbo_Query.Requery
   MsgBox "Sauvegarde de " & strQuery & " effectuée.", vbInformation + vbOKOnly, "information"
Else
   MsgBox "Ce nom de requête existe déjà.", vbExclamation + vbOKOnly, "Erreur"
```

Code à insérer
```
End If
```

Dans ce code 4 parties particulières sont à commenter :
Inputbox() : Cette fonction affiche une fenêtre comprenant un bouton OK, un bouton Annuler, une zone de saisie et une zone de texte. Elle invite l'utilisateur à saisir une valeur. Celle-ci est renvoyée lors de l'appui sur le bouton OK.
vbcrlf: Cette constante contient le code retour chariot (cr ou carriage return) et le saut de ligne (lf ou line feed), elle permet dans un texte de passer à la ligne.
Dlookup("champ","table","condition") : Cette fonction VBA existe également sous le nom de **Rechdom("champ";"table";"condition")** pour l'interface ACCESS. Elle renvoie une valeur unique recherchée dans une table ou une requête.

- *champ* est le champ contenant la valeur à renvoyer.
- *table* est la table ou la requête de sélection dans laquelle la recherche est effectuée.
- *condition* est une condition Where sans le mot clef WHERE.

CurrentDb.CreateQueryDef : Cette méthode permet de créer une requête à partir d'une chaîne de caractère SQL valide.
CurrentDB est l'objet database courant.
Pour plus d'information veuillez consulter l'aide ACCESS.

V-B-3 - Chargement SQL

Si le chargement d'une syntaxe SQL contenu dans une requête ne comporte pas de problème majeur, la chaîne SQL est cependant composée de caractères parasites.
En effet la syntaxe est "polluée" par un saut de ligne constitué des codes Carriage Return (13) et Line Feed (10).
Ouvrez les propriétés du bouton **Charger SQL**.
Créez un évènement sur la propriété **Sur clic**.
Insérez le code suivant :

Code à insérer
```
' charge la query
Dim strQuery As String

' élimine de cr lf (version 2000 et  +)
strQuery = Replace(CurrentDb.QueryDefs(Me.cbo_Query).Sql, Chr(13) & Chr(10), " ")

' élimine de cr lf (toutes versions)
'strQuery = CurrentDb.QueryDefs(Me.cbo_Query).Sql
'Dim lngPos As Long
'While (InStr(1, strQuery, Chr(13) & Chr(10)) > 0)
'     lngPos = InStr(1, strQuery, Chr(13) & Chr(10))
'     strQuery = Left(strQuery, lngPos - 1) & " " & Right(strQuery, Len(strQuery) - lngPos - 1)
'Wend

' met la chaine dans la zone texte
Me.txt_chaineSQL = strQuery
' met la chaine pour afficher le résultat
Me.lst_resultat.RowSource = strQuery
```

Nous remplaçons le CR et LF par un espace grâce à la fonction Replace. (disponible à partir de la version 2000 uniquement).

V-B-4 - Suppression de requête

La suppression ne pose aucun problème particulier, il faut simplement veiller à demander une confirmation à l'utilisateur et à rafraîchir la liste de requête. Insérez simplement le code suivant dans la procédure événementielle du bouton **Supprimer SQL**.

Code à insérer

```
If MsgBox("Voulez-vous vraiment supprimer la requête " & Me.cbo_Query & " ?", _
   vbExclamation + vbOKCancel, "Avertissement") = vbOK Then

   DoCmd.DeleteObject acQuery, Me.cbo_Query   'efface la query
   lf_GetQueryList          ' recompose la liste
   Me.cbo_Query.requery     ' raffraichit la liste
End If
```

VII - Traitement d'autres opérateurs logiques

Après l'ajout de ces nouvelles fonctionnalités nous allons étoffer les possibilités de la recherche imbriquée en ajoutant la gestion d'un autre opérateur logique.
Commencez par créer un contrôle zone liste déroulante avec les propriétés suivantes :

Zo lis modif lis de opéra logi	Propriété	Valeur
	Nom	cbo_operateur
	Origine	
	Source	
	Contenu	"AND";"OR"
	Valeur par défaut	"AND"
	Opérateur logique	
	Légende de l'étiquette	

Très important : Renommez l'étiquette en lbl_operateur car nous aurons besoin de nous y référer ultérieurement.

Le code est très simple à modifier puisqu'il suffit de remplacer le AND existant par la référence de la zone de liste. Repérez la ligne suivante vers la fin de la procédure du bouton **Rechercher**.

Code à remplacer

```
    strSql = strSql & " AND " & strCriteria & "));"
```

Remplacez-le par celui-ci.

Nouveau code

```
    strSql = strSql & " " & Me.cbo_operateur & " " & strCriteria & "));"
```

Voilà c'est tout pour le traitement des opérateurs. Remarquez que nous n'utilisons que les opérateurs les plus courant. Il y a cependant une petite chose à rajouter pour que notre fonctionnalité soit totalement fini. Il s'agit d'afficher ou de cacher la liste **cbo_operateur** suivant l'état du bouton d'option **opt_rechcourante**.

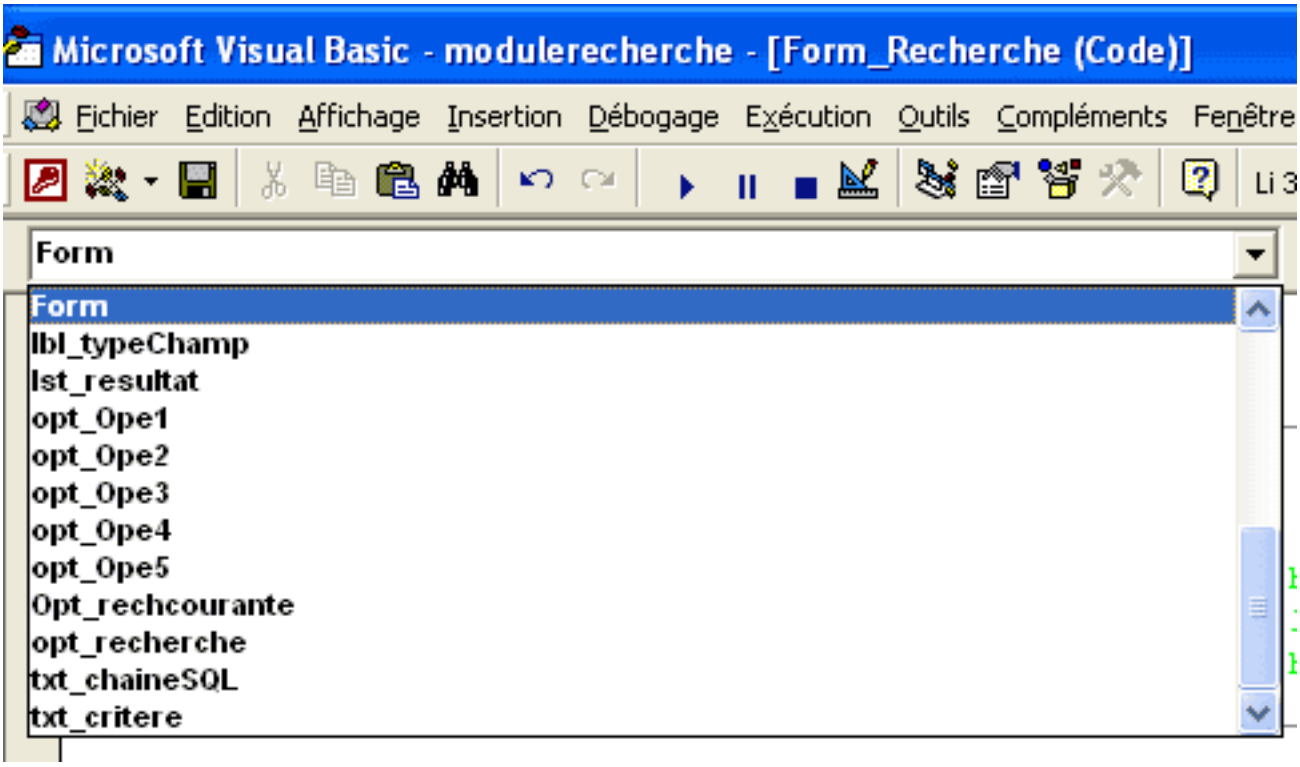
Opt_Rech lis de opéra	Propriété	Valeur
	Valeur par défaut	
	[Sur Clic]	[Procédure Evènementielle]

Insérez le code suivant dans la procédure de l'évènement **Sur Clic**:

Code à insérer

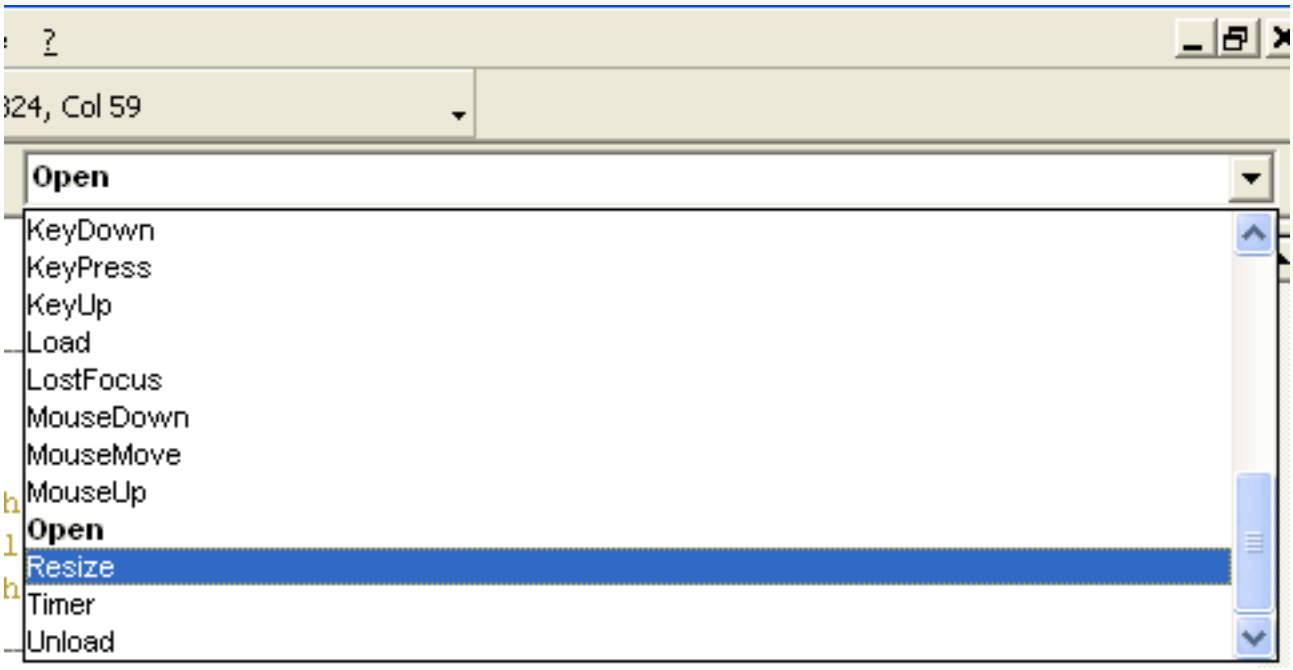
```
    If Me.Opt_rechcourante = True Then ' recherche courante activée
       Me.cbo_operateur.Visible = True
       Me.lbl_operateur.Visible = True
    Else
       Me.cbo_operateur.Visible = False
       Me.lbl_operateur.Visible = False
    End If
```

Positionnez-vous sur la procédure **Form_open**.
Il y a un moyen très rapide pour se déplacer d'une procédure à l'autre sans passer par la fenêtre des propriétés. Dans la barre de menu située au dessus de la zone contenant le code VBA il y a 2 zones liste déroulante. Celle de gauche indique l'ensemble des objets gérés par le module, dans notre cas il s'agit du formulaire de recherche.



Notez que tous les items sont en gras.

Cliquez sur **Form**
Celle de droite contient l'ensemble des évènements de l'objet sélectionné dans la liste de gauche.



Ceux en gras existent dans le code.

Cliquez sur **Open**.
Le curseur se positionne sur la procédure **Form_Open** qui correspond à la propriété événementielle **Sur ouverture** du formulaire.
Insérez le code suivant :

Code à insérer

```
Opt_rechcourante_Click    'cache la zone liste et l'étiquette
```

Ici nous faisons appel à la procédure précédemment créée qui cache où fait apparaître la zone liste et son étiquette. Faites un test, sauvez le formulaire, fermez-le puis réouvrez-le. Cliquez ensuite sur le bouton d'option de la recherche courante à plusieurs reprises...

Nuit...

Jour...

VIII - Traitement des NULL

Tant attendue, cette fonctionnalité nécessite des modifications dans les boutons d'options de recherche et dans le traitement des types de champs du bouton de recherche.
Un test sur la zone texte **txt_critere** nous permettra de déterminer si l'utilisateur recherche les Null.

VIII-A - L'affichage des boutons d'options.

Seules les options de recherche des valeurs booléennes (Oui/non) doivent être modifiées. Nous découvrirons pourquoi dans le traitement de la construction de la chaîne SQL.
Ouvrez le code VBA **cbo_champ AfterUpdate** comme nous l'avons appris dans le chapitre précédent.

Le code d'origine

```
Case Is = dbBoolean     ' Booléen
     Me.lbl_TypeChamp.Caption = "Oui/Non"
        Me.lbl_Etiq1.Caption = "Oui"
        Me.lbl_Etiq2.Caption = "Non"
        Me.lbl_Etiq3.Visible = False ' cache car inusité dans ce cas
        Me.lbl_Etiq4.Visible = False ' idem
        Me.lbl_Etiq5.Visible = False ' idem
        Me.opt_Ope3.Visible = False
        Me.opt_Ope4.Visible = False
        Me.opt_Ope5.Visible = False
        Me.txt_Critere.Visible = False    ' pas de critere
```

Vous remarquez que les options 3 et 4 sont inutilisées. Nous allons nous en servir pour traiter la valeur **Null** et **Non Null**.
Faites les modifications suivantes.

Le code une fois modifié

```
   Case Is = dbBoolean     ' Booléen
        Me.lbl_typeChamp.Caption = "Oui/Non"
        Me.lbl_etiq1.Caption = "Oui"
        Me.lbl_etiq2.Caption = "Non"
        Me.lbl_etiq3.Caption = "Est Null"
        Me.lbl_etiq4.Caption = "N'est pas Null"
        Me.lbl_etiq5.Visible = False
        Me.opt_Ope5.Visible = False
        Me.txt_critere.Visible = False    ' pas de critere
```

Nous affichons dans les étiquettes 3 et 4 leur texte respectif. Passons au traitement.

VIII-B - Le traitement

Nous devons modifier directement la procédure **cmd_recherche_Click**. Affichez-la en sélectionnant **cmd_recherche** dans la liste de gauche. Comme il n'y a qu'une procédure pour ce contrôle, nous y accédons directement, pas besoin de faire un choix dans la liste de droite.

VIII-B-1 - Les booléens (oui/non)

Recherchez le code suivant.

Le code d'origine

```
      Case dbBoolean                    ' bool
           If intOpeChamp = 1 Then      ' oui
```

Le code d'origine

```
            strCriteria = strTable & "." & strField & "=-1"
         ElseIf intOpeChamp = 2 Then       ' non
            strCriteria = strTable & "." & strField & "=0"
         Else
            strCriteria = strTable & "." & strField & "=Null"
         End If
```

Voici la modification à faire. Nous avons remplacé les tests conditionnels *(if, elseif, else et endif)* par une structure *Select Case*.

Le code une fois modifié

```
      Case dbBoolean                        ' bool
         Select Case intOpeChamp
            Case 1    ' oui
                strCriteria = strTable & "." & strField & "=-1"
            Case 2    ' non
                strCriteria = strTable & "." & strField & "=0"
            Case 3
                strCriteria = "ISNULL(" & strTable & "." & strField & ")"
            Case 4
                strCriteria = "NOT ISNULL(" & strTable & "." & strField & ")"
          End Select
```

VIII-B-2 - Les numériques et dates

La modification est plus importante pour le numérique. Repérez le code suivant, il se trouve immédiatement après le traitement booléen.

Le code d'origine

```
      Case dbByte To dbBinary, dbLongBinary, dbBigInt To dbVarBinary, dbNumeric To dbTimeStamp

         ' traite les numeriques
  strCriteria = Me.txt_Critere
         ' traite la virgule si elle existe
         If InStr(1, Me.txt_Critere, ",") > 0 Then
strCriteria = Replace(Me.txt_Critere, ",", ".", 1)

  ' ---------------- pour les versions antérieure à la 2000 ----------------
         'If InStr(1, Me.txt_critere, ",") > 0 Then _
  '    strCriteria = Left(Me.txt_critere, InStr(1, Me.txt_critere, ",") - 1) _
  '    & "." & Right(Me.txt_critere, InStr(1, Me.txt_critere, ","))
         ' ------------------------------------------------------------------------
  ' type champ = date
         If intTypChamp = dbDate And IsDate(Me.txt_Critere) Then  strCriteria = "#" &
Me.txt_Critere & "#"
         ' rajoute les dièses

         If Not IsNull(Me.txt_Critere) Then
            Select Case intOpeChamp                    ' numerique, date
                Case 1 ' =
                     strCriteria = strTable & "." & strField & "=" & strCriteria

                Case 2 ' >=
                     strCriteria = strTable & "." & strField & ">=" & strCriteria

                Case 3 ' <=
                     strCriteria = strTable & "." & strField & "<=" & strCriteria

                Case 4 '<>
                     strCriteria = strTable & "." & strField & "<>" & strCriteria
            End Select
         End If
```

Faites les modifications suivantes.

Le code une fois modifié

```
       Case dbByte To dbBinary, dbLongBinary, dbBigInt To dbVarBinary, dbNumeric To dbTimeStamp
            ' traite les numeriques
  If Not IsNull(Me.txt_critere) Then   ' si le null n'est pas la valeur à traiter
              strCriteria = Me.txt_critere
              ' traite la virgule si elle existe
              If InStr(1, Me.txt_critere, ",") > 0 Then
strCriteria = Replace(Me.txt_critere, ",", ".", 1)

     '----------------------- pour les versions antérieure à la 2000
              'If InStr(1, Me.txt_critere, ",") > 0 Then _
     ' strCriteria = Left(Me.txt_critere, InStr(1, Me.txt_critere, ",") - 1) _
     ' & "." & Right(Me.txt_critere, InStr(1, Me.txt_critere, ","))
     '-------------------------------------------------------------

     ' type champ = date
              If intTypChamp = dbDate And IsDate(Me.txt_critere) Then strCriteria = "#" &
Me.txt_critere & "#"

     ' rajoute les dièses
           End If

           Select Case intOpeChamp                    ' numerique, date
                Case 1 ' =
                     If IsNull(Me.txt_critere) Then
                        strCriteria = "ISNULL(" & strTable & "." & strField & ")"
                     Else
                        strCriteria = strTable & "." & strField & "=" & strCriteria
                     End If
                Case 2 ' >=
                     strCriteria = strTable & "." & strField & ">=" & strCriteria

                Case 3 ' <=
                     strCriteria = strTable & "." & strField & "<=" & strCriteria

                Case 4 '<>
                     If IsNull(Me.txt_critere) Then
                        strCriteria = "NOT ISNULL(" & strTable & "." & strField & ")"
                     Else
                        strCriteria = strTable & "." & strField & "<>" & strCriteria
                     End If
           End Select
```

Dans le début du code, nous pouvons remarquer que les modifications à apporter au critère - le remplacement de la virgule par le point et l'ajout des dièses - ne sont à faire que pour des valeurs autres que Null. Sans ce test préalable des erreurs se produiront.
Dans la seconde partie du code nous traitons les Null dans les options 1 (=) et 4 (<>) toujours en testant le contenu de la zone texte **txt_critere**.

VIII-B-3 - Les textes

Le traitement texte est similaire à celui du numérique.
Recherchez le code suivant, il vient immédiatement après le traitement précédent.

Code d'origine

```
       Case dbText, dbMemo, dbChar                    ' texte
            Select Case intOpeChamp
                Case 1 ' strictement egal
                       strCriteria = strTable & "." & strField & " Like """ & Me.txt_critere & """"
                Case 2 ' commence par
                    strCriteria = strTable & "." & strField & " Like """ & Me.txt_critere & "*"""
                Case 3 ' contient
```

Code d'origine

```
                    strCriteria = strTable & "." & strField & " Like ""*" & Me.txt_critere & "*"""
                Case 4 ' fini par
                    strCriteria = strTable & "." & strField & " Like ""*" & Me.txt_critere & """"
                Case 5 ' ne contient pas
                    strCriteria = "NOT " & strTable & "." & strField & " Like """ &
Me.txt_critere & """"
            End Select
```

Faites les modifications suivantes.

Code une fois modifié

```
        Case dbText, dbMemo, dbChar                     ' texte
            Select Case intOpeChamp
                Case 1 ' strictement egal
                    If IsNull(Me.txt_critere) Then
                        strCriteria = "ISNULL(" & strTable & "." & strField & ")"
                    Else
                        strCriteria = strTable & "." & strField & " Like """ & Me.txt_critere & """"
                    End If
                Case 2 ' commence par
                    strCriteria = strTable & "." & strField & " Like """ & Me.txt_critere & "*"""
                Case 3 ' contient
                    strCriteria = strTable & "." & strField & " Like ""*" & Me.txt_critere & "*"""
                Case 4 ' fini par
                    strCriteria = strTable & "." & strField & " Like ""*" & Me.txt_critere & """"
                Case 5 ' ne contient pas
                    If IsNull(Me.txt_critere) Then
                        strCriteria = "NOT ISNULL(" & strTable & "." & strField & ")"
                    Else
                        strCriteria = "NOT " & strTable & "." & strField & " Like """ &
Me.txt_critere & """"
                    End If
            End Select
```

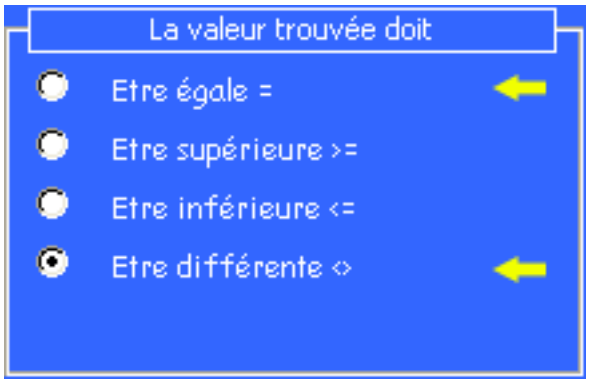
Nous nous sommes servi des options 1 et 4 pour traiter respectivement la rechercher du Null et de non Null.
Un utilisateur qui souhaite rechercher les Null ou non Null se contentera de laisser la zone texte vide et jouera avec l'option **Etre identique à la valeur** (option 1) et **Ne pas contenir la valeur** (option 4).
Regardez le chapitre suivant traitant du mode d'emploi.

VII-C - Mode d'emploi pour le Null

Pas de remarque sur le traitement des valeurs Oui/non, il suffit de cliquer sur l'un des deux boutons d'options.

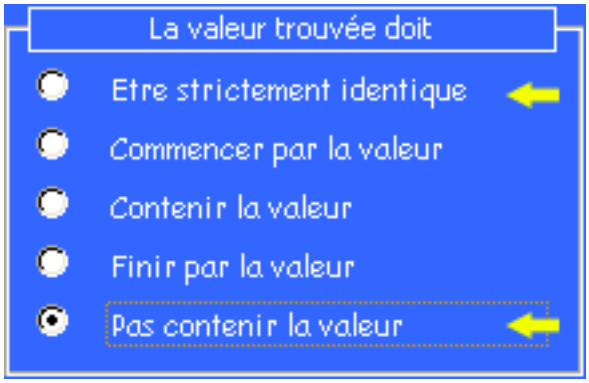
Juste un choix...

Pour les valeurs numériques/dates il ne faut rien saisir dans la zone texte critère et faire l'un des 2 choix.



Vous avez le choix... dans la date.

Pour le texte le mode d'emploi est le même que pour les valeurs numérique.



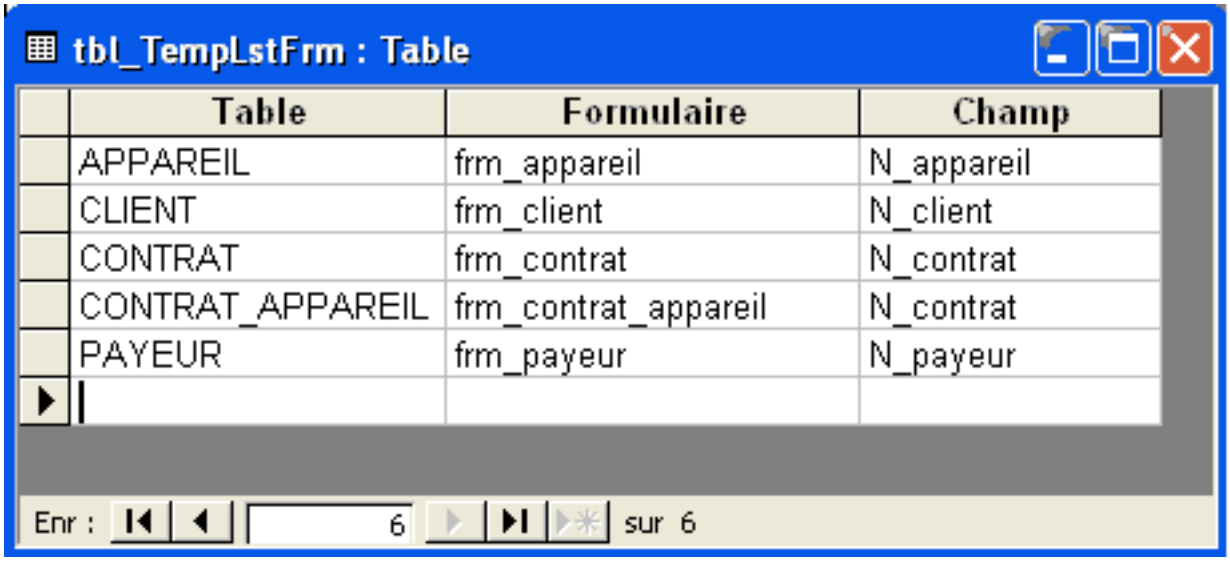
Encore un choix à faire.

IX - Formulaire d'édition

Un formulaire de recherche est intéressant que si l'on peut se servir de cette recherche pour éditer un enregistrement ou même un groupe d'enregistrement.
Le problème est de déterminer quel formulaire utiliser pour l'édition. Vu le nombre potentiel très important de formulaires que peut contenir une application et l'incapacité de faire la différence entre formulaires et sous-formulaires, nous sommes obligés de réaliser un table de référence.
Celle-ci devra être rempli manuellement suivant la structure suivante.

Table tbl_Temp	Nom du champ	Type	Longueur
	Table	Texte	50
	Formulaire	Texte	50
	Champ	Texte	50

Le champ **Table** contient la nom de la table de recherche courante. **Formulaire** contient le nom du formulaire à ouvrir et **Champ** le nom du champ clef permettant l'identification de l'enregistrement. Cette méthode n'est peut être pas la plus optimisée cependant elle est facile à comprendre et à mettre en oeuvre.



Un exemple de la table saisie...

IX-B - Le code VBA

Le code suivant est à insérer dans l'événement **Sur Double-click** de la liste contenant le résultat **lst_resultat**.

Code à insérer

```
Private Sub lst_resultat_DblClick(Cancel As Integer)
Dim rst As Recordset
Dim strCriteria As String

Set rst = CurrentDb.OpenRecordset("tbl_TempLstFrm", 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 de formulaire. Veuillez renseigner la table des paramètres.", _
          vbCritical + vbOKOnly, "formulaire de Recherche"
   Exit Sub
Else                     ' trouvé
   If lf_GetTypeField(Me.cbo_table, rst.Fields("Champ")) = dbText Then  'la clef est Texte
```

Code à insérer

```
        strCriteria = rst.Fields("Champ") & "='" & Me.lst_resultat & "'"
    Else                                                                    'la clef est numérique
        strCriteria = rst.Fields("Champ") & "=" & Me.lst_resultat
    End If
    DoCmd.OpenForm rst.Fields("Formulaire"), acNormal, , strCriteria
End If

End Sub
```

Dans ce code nous faisons une recherche du nom de la table sélectionnée dans le recordset de la table de référence **tbl_TempLstForm**. N'ayant qu'un besoin de consultation nous l'avons ouvert en **Snapshot** Si la table n'est pas inscrite un message explicite est renvoyé, sinon le code détermine s'il s'agit d'un champ numérique ou texte avec la fonction **If_GetTypeField** que nous avons mise en place dans le premier volet de ce tutoriel. Une fois les différents paramètres connus nous pouvons lancer la commande d'ouverture du formulaire **Docmd.openform**.

X - Le compteur

Le compteur permet de connaître le nombre d'enregistrement contenu dans la liste **lst_resultat** ainsi que le total de la table.
Commencez par créer une étiquette conforme aux propriétés suivantes :

Etiq	Propriété	Valeur
	Nom	lbl_nbRecord
	Légende	0/0
	Aligner	droite

Pour l'alignement vous pouvez également utiliser l'icône de mise en page *Aligné à droite*.

X-A - Le code VBA

Ce code utilise plusieurs fonctions et propriétés natives. Placez-vous à la fin de la procédure **cmd_recherche_Click()**. C'est celle du bouton de recherche.

Code à insérer

```
    Me.lbl_nbRecord.Caption = IIf(Me.lst_resultat.ListCount _
     <= 1, 0, Me.lst_resultat.ListCount - 1) & "/" & _
  DCount(Me.cbo_champ, Me.cbo_table)

End sub ' cette ligne existe déjà il ne faut pas la saisir.
```

Le détail du code est le suivant :
Me.lbl_nbRecord.Caption = : affecte dans la légende du contrôle.
Me.lst_resultat.ListCount : retourne le nombre de ligne de la liste.
Un petite explication d'impose concernant le calcul du compteur. Comme tout index d'objets en VBA il commence par 0, il faut systèmatiquement enlever 1 au résultat.
DCount(Me.cbo_champ, Me.cbo_table) donne le nombre total d'enregistrement de la table choisie dans la liste **cbo_table**.
Attention au **Dcount** celui-ci est un peut gourmand en ressource. Sur de très grosses tables et des machine sous-dimensionnées cela peut occasionner des temps de réponse assez long.

XI - Sélection de champs

Pour l'instant le module de recherche affiche la totalité des champs de la table choisie. Ceci nuit à la lisibilité, nous allons donc faire en sorte de corriger ce problème en mettant en place une liste de choix des champs.

XI-A - Contrôle liste

Propriété	Valeur
Nom	lst_champs
Origine source	champs
Contenu	cette propriété vide
Sélection multiple	Etendue
Légende	Champs à afficher

Une fois cette liste placée à droite des contrôles existant et sur toute la hauteur du formulaire nous pouvons passer au code VBA.

XI-B - Le Code VBA

XI-B-1 - Remplir la liste

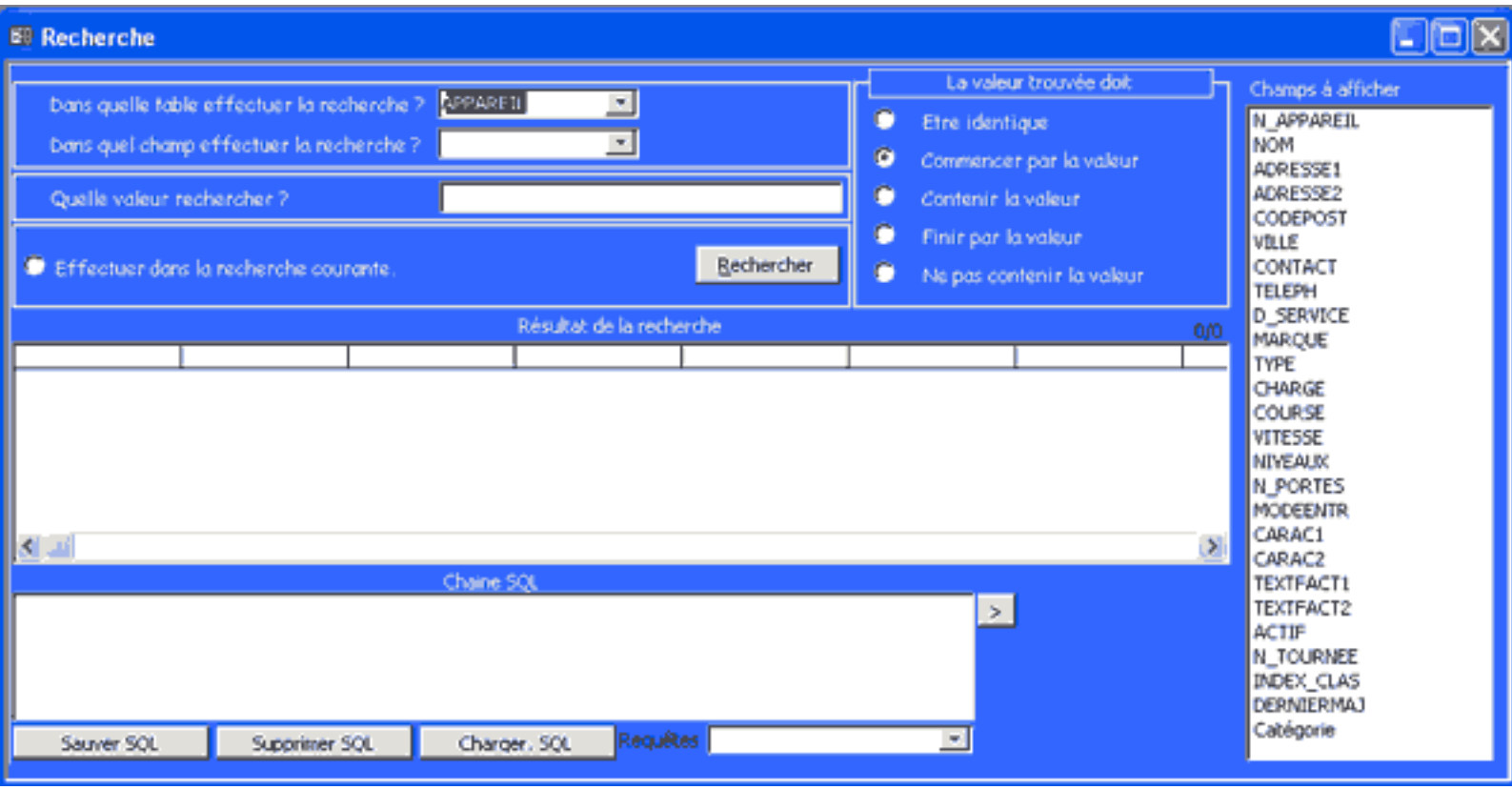
En premier lieu il convient de remplir cette liste avec les champs de la table sélectionnée. De la même manière que pour la liste du champs servant à sélectionner le champ pour la recherche.
Placez-vous à la fin de la procédure **cbo_table_AfterUpdate()** et insérez les lignes suivantes.

Code à insérer

```
    Me.lst_champs.RowSource = Me.cbo_table.Value
    Me.lst_champs.Requery

End Sub    ' cette ligne existe déjà il ne faut pas la saisir.
```

Vous pouvez observer que les lignes sont pratiquement identiques à celles permettant de renseigner la liste des champs **Cbo_Champ**.
Enregistrez le formulaire et passez en mode utilisation.
Sélectionnez une table dans la liste des tables pour voir la liste se remplir.



Ô la belle liste...

XI-B-2 - Récupérer la sélection

Une fois la liste rempli ouvrez la procédure **cmd_recherche_Click()**. Repérez les lignes suivantes :

Code d'origine

```
        Case Else
            MsgBox "Cas non prévu."
            Exit Sub
    End Select

' construit la requête sql
If Me.Opt_rechcourante And Not Len(Me.lst_resultat.RowSource) = 0 Then
```

Insérez le code suivant après le **End Select**.

Code à insérer

```
' 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
```

Explication sur le code précédent.
A l'aide d'une boucle **FOR...NEXT** nous balayons le contenu de la liste **lst_champs** et testons la propriété **Selected** de chaque ligne.
Si celle-ci est vraie nous copions la valeur de la ligne, donc le nom du champ entouré de crochets, dans la variable texte **strChamps** en insérant à la fin une virgule et un espace, séparateur habituel des champs en langage SQL.
Une fois la liste parcouru nous enlevons les 2 derniers caractères de la variable correspondant à la virgule et l'espace de la fin. En effet ceux-ci sont inutiles puisqu'il n'y a plus de champs à la suite mais la fin de la chaîne SQL.
Un test pour savoir s'il y a eu une sélection **Len(strChamps)=0** nous permettra d'alterner une liste des champs avec une sélection totale des champs de la table. Autrement-dit la syntaxe SQL **.*** que nous utilisons depuis le début.

XI-B-3 - Exploiter la sélection

Encore une petite modification pour profiter du résultat de notre nouvelle fonctionnalité ; l'insertion de la liste des champs dans le code SQL.
Commencez par repérer le code suivant :

Code à rechercher

```
    ' construit la rq sql
    strSql = "SELECT DISTINCTROW " & strTable
```

Faites la modification comme suit :

Code modifié

```
    ' construit la rq sql
    strSql = "SELECT DISTINCTROW " & strChamps
```

La fonctionnalité est opérationnelle, faites quelques essais.

XI-C - Comment utiliser la sélection multiple ?

L'utilisation de la sélection multiple dans une zone de liste se fait au moyen de la souris et des touches *Shift* et *Control*.

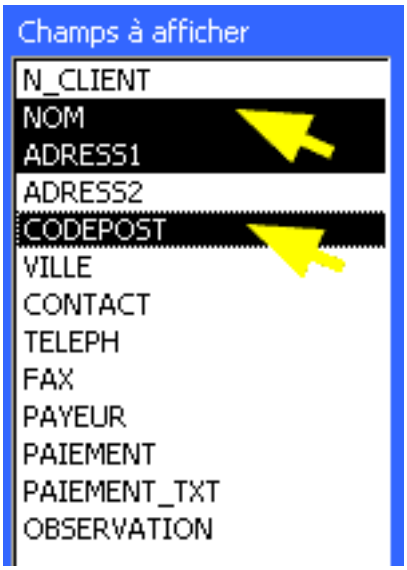
Sélection continue

Sélection continue Shift - méthode 1

- Maintenez la touche *Shift* enfoncée et cliquez sur **Nom**.
- Faites glisser la souris vers le bas pour atteindre **Paiement**.
- Relâchez la touche *Shift* et le bouton de la souris.

Sélection continue Shift - méthode 2

- Maintenez la touche *Shift* enfoncée et cliquez sur **Nom**.
- Relâchez la touche *Shift* et le bouton de la souris.
- Maintenez la touche *Shift* enfoncée et cliquez sur **Paiement**.
- Relâchez la touche *Shift* et le bouton de la souris.



Sélection discontinue

Sélection discontinue Control

- Maintenez la touche *Control* enfoncée
- Cliquez sur **Nom, Adress1** et **Codepost**
- Relâchez la touche *Control* et le bouton de la souris.

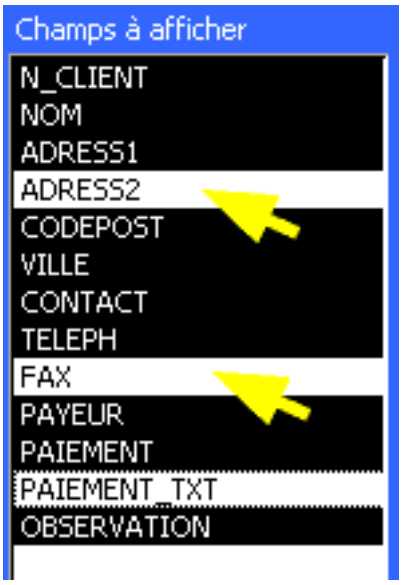
Vous pouvez mélanger les 2 méthodes pour alterner sélection continue et discontinue.

Pour l'exercice suivant sélectionnez l'intégralité de la liste.

Dé-sélection continue Shift

- Maintenez la touche *Shift* enfoncée et cliquez sur **Contact**.
- Relâchez la touche *Shift* et le bouton de la souris.

Si vous sélectionnez la liste du bas vers le haut la dé-sélection se fera dans l'ordre inverse.



Dé-sélection discontinue

Dé-sélection Control

- Maintenez la touche *Control* enfoncée et cliquez sur **Adress2**, **Fax** et **Paiement_Txt**.
- Relâchez la touche *Control* et le bouton de la souris.

XII - Conclusion

Le 2ème volet est terminé. Dans le 3ème et dernier volet (Tout à une fin) nous verrons quelques fonctionnalités supplémentaires comme l'export Excel, le réglage dynamique des largeurs de colonnes et le choix d'états pour l'impression.

XIII - Remerciements

Je tiens à remercier : **Maxence Hubiche** pour le temps passé en relecture et correction.
Les nombreux Devnautes qui m'ont apporté leurs judicieuses remarques sur la première partie. À l'équipe de **Developpez.com** pour la qualité du site.
À **Nono40** pour son super éditeur XML qui se bonifie avec le temps comme un vieux Pommard.
Je présente mes plus plates excuses à ceux que j'aurais omis de remercier.