Formulaire de recherche prêt à l'emploi.(nouvelle version)

par Fabrice CONSTANS (autres articles)

Date de publication : 24/05/2005

Dernière mise à jour : 01/10/2009

La recherche est une fonction critique dans une base de données aussi disposer d'un système de recherche dans une application est souvent obligatoire.
Dans ce tutoriel nous allons brièvement passer en revue les différentes méthodes de recherche disponibles et réaliser un outil de recherche totalement générique dans le sens où il pourra s'intégrer facilement dans n'importe quelle application.

I - Avertissement

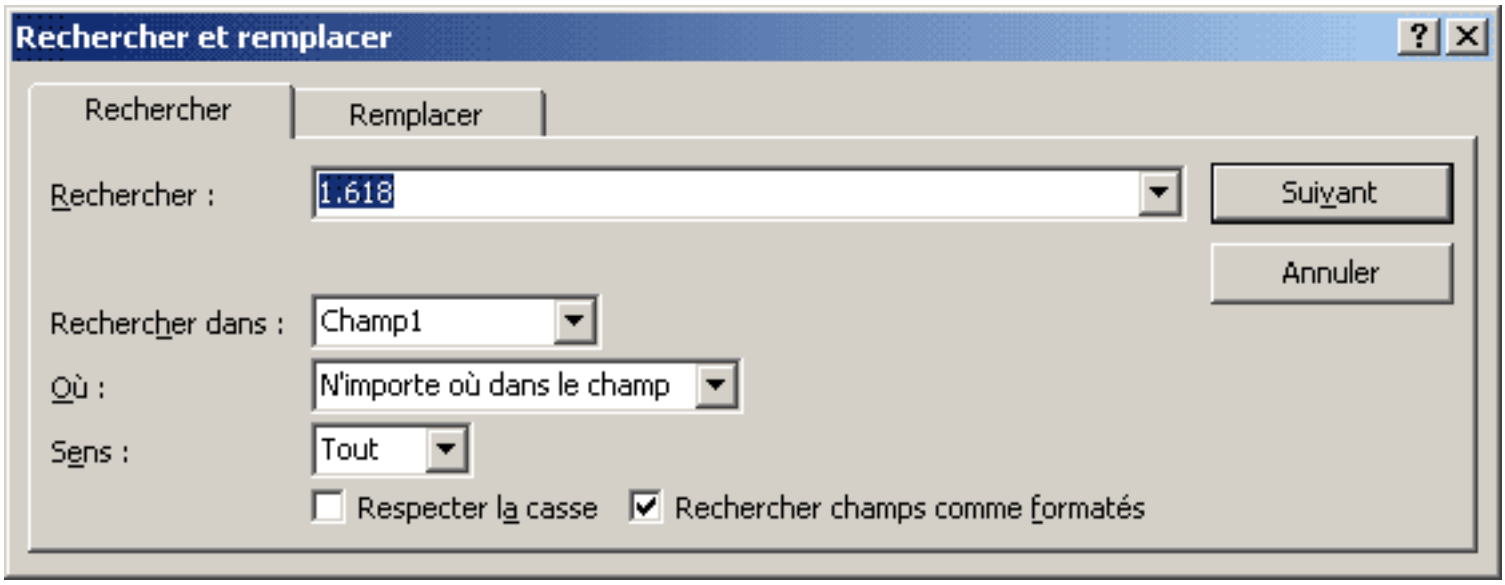
L'utilisation de la touche F1 est vivement conseillée à tous les stades de l'utilisation d'ACCESS. L'amélioration constante de l'aide en fait un partenaire de choix dans l'apprentissage permanent d'ACCESS. Personnellement, je ne peux m'en passer, ne serait-ce que pour mémoire.

II - Tour d'horizon des techniques de recherches classiques.

Dans une base de données, la recherche d'enregistrements est primordiale. Microsoft ACCESS dispose de plusieurs solutions accessibles par l'interface.

II.A - Commande Rechercher

La plus simple d'utilisation est sans nul doute la commande **Rechercher** du menu **Edition/Rechercher** que l'on peut appeler via l'icône **Jumelles** de la barre d'icônes **Base de données** ou encore grâce à **Ctrl+F**.

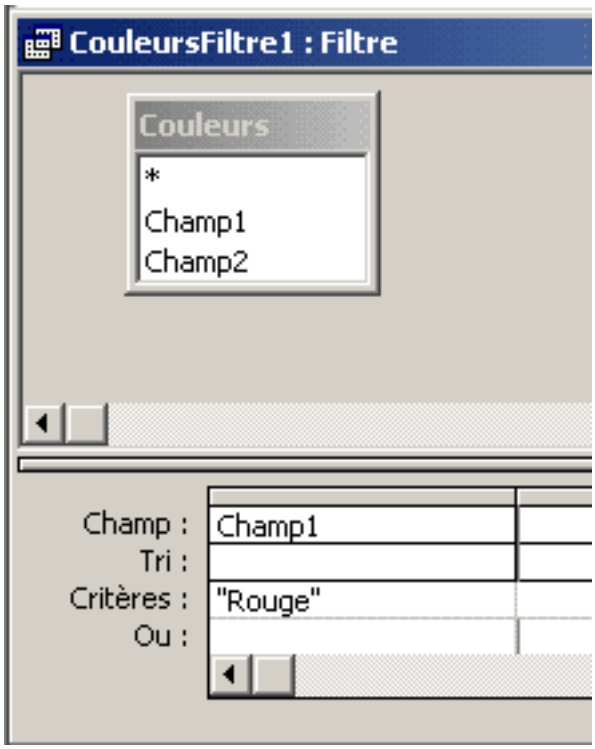


Ctrl+F une petite fenêtre bien pratique

Celle-ci effectue des recherches séquentielles à partir d'un critère unique. En d'autres termes, elle va parcourir un ensemble d'enregistrement (table ou requête) et s'arrêter sur le prochain enregistrement satisfaisant au critère.

II.B - Fonctionnalité Filtre

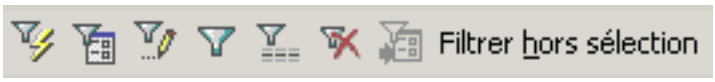
Le filtre accessible par le menu **Enregistrement/Filtre** permet de visualiser un groupe d'enregistrements suivant une ou plusieurs critères, un peu à la manière d'une requête. L'interface ressemble au QBE.



On jurerait le QBE.

Notez qu'un filtre peut être sauvegardé en tant que requête.

On peut également utiliser les boutons ci-dessous pour faire et défaire des filtres. Ce sont d'ailleurs mes préférés car leur utilisation est simple et intuitive.

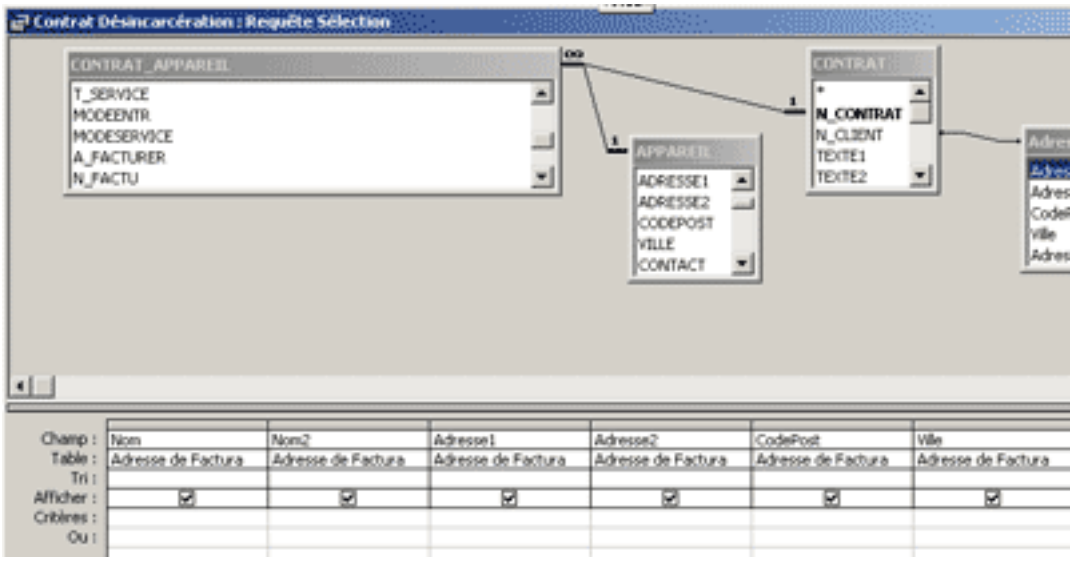


Une série de petits boutons à tester.

QBE : littéralement Query By Example, désigne l'utilitaire de construction des requêtes de l'interface ACCESS.

II.C - La requête par le QBE

La requête, comme le filtre, permet de visualiser un groupe d'enregistrements d'après un critère.



Le QBE...

Dans le cas d'une application fonctionnant avec un runtime, il est impossible de construire une requête à l'aide du QBE, de même qu'un utilisateur lambda n'a pas souvent la capacité ou l'envie de l'utiliser.
Ce tutoriel va décrire une solution de substitution totalement générique comme à mon habitude.

II.D - Autre solution

Si vous ne souhaitez pas utiliser plusieurs critères de recherche ou que vous n'opérez que sur une seule table je vous conseille plutôt l'excellente solution de **Cafeine** à cette **adresse**.

III - Définition des besoins et des contraintes

- Afficher une liste d'enregistrements sans connaissances du langage SQL.
- La solution doit pouvoir être utilisée à partir de la version Runtime d'ACCESS.

IV - Le formulaire

Comme dans toute fonctionnalité avancée nous aurons besoin d'un formulaire pour choisir nos actions et afficher le résultat.
Ce formulaire n'a pas besoin du sélecteur d'enregistrement, du séparateur d'enregistrement, des boutons de navigation ni des ascenseurs. Vous pouvez les désactiver dans les propriétés.

Propriété	Valeur
Légende	Recherche
Barre défilement	Aucune
Afficher sélecteur	Non
Boutons de déplacements	Non
Diviseurs d'enregistrements	Non

Dans un premier temps nous n'allons réaliser que le système de génération d'une instruction SQL simple. Ce système évoluera vers un véritable générateur doté de fonctionnalités performantes.
Commencez par insérer quelques contrôles :

Zo
lis
modif
lis
de
tab

Propriété	Valeur
Nom	table_origine
Origine valeurs	Collez le nom (d'une ou) de plusieurs tables de votre application séparées par un point virgule ;.
Légende	Quelle table effectuer la recherche ?

Zo
lis
modif
lis

Propriété	Valeur

Nom_champ	Nom du champ
Origine	Origine des champs
Source	Laissez cette propriété vide.
Légende	Légende de l'étiquette

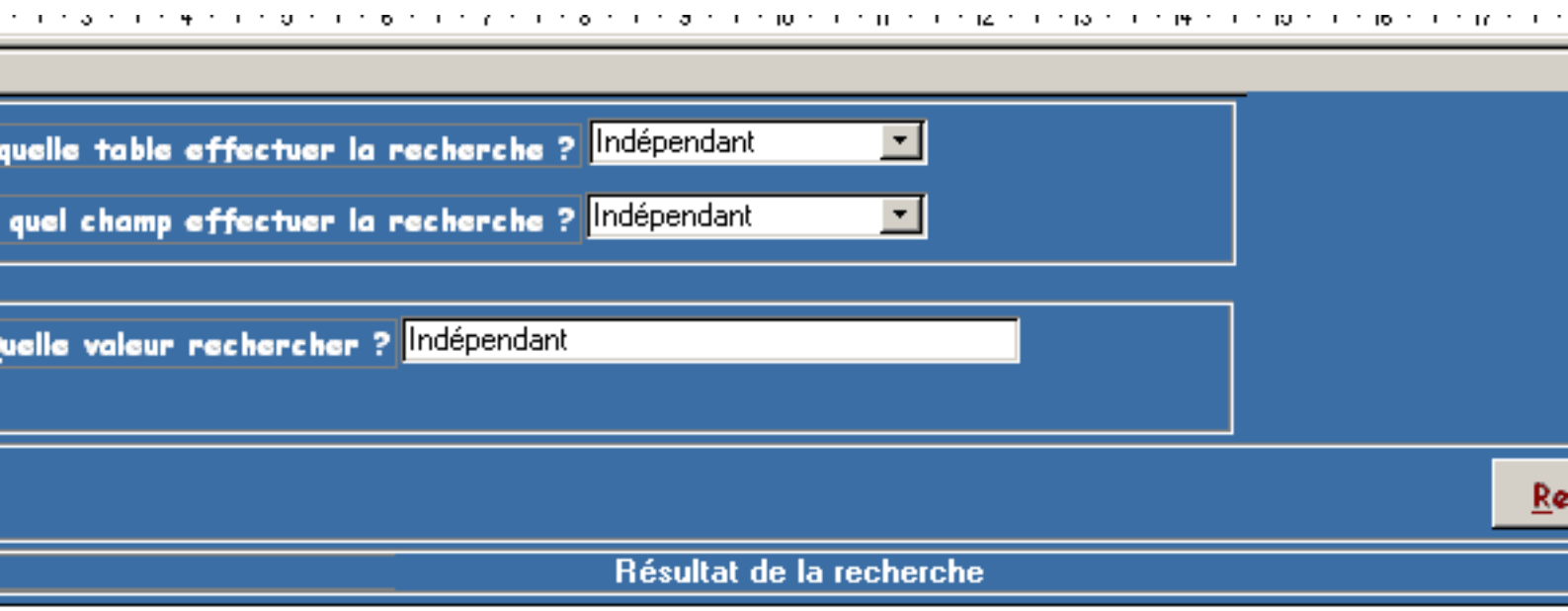
Propriété	Valeur
Nom	Nom du résultat
Origine	Origine des champs
Source	Laissez cette propriété vide.
Nbre colonnes	Nombre colonnes
En-tête	En-têtes colonnes
Légende	Légende de l'étiquette

Les deux contrôles suivants ne présentent aucune particularité dans leurs propriétés.

Propriété	Valeur
Nom	Nom du critere
Légende	Légende de l'étiquette

Propriété	Valeur

Avec quelques cadres bien ajustés cela devrait donner quelque chose comme ça...



Sobre mais fonctionnel...

V - Interaction des zones listes

Pour l'instant la zone liste **cbo_champ** contenant les champs n'affiche rien car nous n'avons pas défini la propriété correspondante. C'est tout à fait normal, celle-ci étant définie lors du choix de la table.
Sélectionnez le contrôle **cbo_table**,
choisissez la propriété **Après MAJ**,
cliquez sur l'icône de droite 
dans la fenêtre **Choisir générateur** cliquez sur **Générateur de code**
Le module s'ouvre sur l'évènement **cbo_Table_AfterUpdate()**
Entrez le code suivant :

A insérer

```
Me.cbo_champ.RowSource = Me.cbo_Table.Value
Me.cbo_champ.Requery
```

A chaque choix d'un nouvel item dans la liste des tables, celui-ci sera affecté à la propriété **Contenu** du contrôle **cbo_champ**. Le contrôle **cbo_champ** sera ensuite "recalculé" pour afficher les champs de la table sélectionnée.
Faites un petit essai... Ce n'est ni de la magie ni une technique du vaudou, simplement la définition des propriétés d'un contrôle par VBA.

Pensez à sauvegarder régulièrement le formulaire durant l'exercice.

VI - Le bouton Rechercher - Part I

Ce chapitre va nous permettre de comprendre la mécanique à mettre en place pour la création et l'affichage des résultats

VI-A - La théorie

Le bouton recherche est le moteur de notre formulaire. Il contient le code essentiel au fonctionnement de l'outil puisqu'il permet à l'aide des options préalablement choisies dans le formulaire de créer une chaîne SQL valide et de visualiser son résultat.

Tout au long de ce tutoriel, vous devez garder à l'esprit qu'une chaîne SQL est avant tout une suite de caractères mais qu'elle répond à un formalisme très strict.

Syntaxe générique

```
SELECT Table.* FROM Table;
```

Cette chaîne est ensuite modifiée par l'ajout de la clause WHERE.

Syntaxe générique

```
SELECT Table.* FROM Table WHERE Table.champ LIKE "moncritere";
```

VI-A-1 - La syntaxe SQL décryptée

Ce chapitre n'a pas pour vocation d'expliquer le langage SQL mais pour une meilleure compréhension du code par les néophytes, un petit décryptage s'impose.

Les mots clefs du langage SQL sont signalés en bleu.

Code	Explication
SELECT	Clause indiquant les champs à sélectionner.
Table.*	Tous les champs de la table nommée Table vont être sélectionnés, ceci est indiqué par l'étoile (*).
FROM Table	Clause indiquant que la table concernée se nomme Table.
WHERE	Clause indiquant que seul les enregistrements répondant au critère suivant seront affichés
Table.Champ Like "Mavaleur"	Condition ou le champ de la table Table nommé Champ doit être identique au mot "Mavaleur"
;	Le point virgule indique que la chaîne SQL est finie.

*Pour plus d'information sur le langage SQL veuillez consulter les tutoriaux présent sur **cette page.**.*

La recherche de texte par SQL ne tient pas compte des majuscules et minuscules.

VI-B - Le moteur de recherche en VBA

Après la théorie nous allons passer à la pratique en traitant uniquement les données de type Texte. Nous traiterons les autres types dans un des chapitres suivants.
Faites un clic droit sur le bouton **Rechercher**,
choisissez **Créer un évènement...**,
dans la fenêtre **Choisir générateur**,
cliquez sur **Générateur de code**
,
Le module s'ouvre sur l'évènement **Btn_Recherche_Click**. Entrez le code suivant :

A insérer

```
Dim strTable As String, strField As String, strCriteria As String, strSql As String
Dim Criter As Variant

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

' compose le critere de recherche
strCriteria = strTable & "." & strField & " Like """ & Me.txt_Critere & """"

' construit la requête sql
strSql = "SELECT DISTINCTROW " & strTable & ".*"
strSql = strSql & " FROM " & strTable
strSql = strSql & " WHERE ((" & strCriteria & "));"


Me.lst_Resultat.RowSource = strSql  ' affecte sql a lst_Resultat
Me.lst_Resultat.Requery             ' recalcule la liste
```

Il est important de remarquer que les 2 dernières lignes de ce code constituent le mécanisme du moteur de recherche.

Pour éviter les erreurs dans la chaîne SQL vous pouvez la copier à partir d'une requête de sélection préalablement ouverte dans le QBE.

Notre moteur de recherche est prêt. Pour l'instant il ne fonctionne qu'avec des données de type texte et pour une égalité.
Cette dernière affirmation n'est qu'apparente, le prochain chapitre le démontre.

VI-C - Un test de recherche

Choisissez une table et un champ texte et saisissez une valeur incomplète dans le critère.
Pour l'exemple j'ai volontairement choisi une série de mots simples.

La liste des mots

- Eau
- Auto
- Tableau
- Eaux
- Rideaux

Essayez tour à tour l'insertion de l'étoile * avant et après le mot et cliquez sur le bouton de recherche.
Vous obtiendrez les résultats suivants :

Critère	Résultat	Rapport
Eau	Eau	Egalité stricte
Eau*	Eau, Eaux	Commence par
*Eau	Tableau, Eau	Finit par
Eau	Tableau, Eau, Eaux, Rideaux	Contient

L'opérateur LIKE que nous avons employé peut, grâce à quelques caractères jokers substituer un ou plusieurs caractères dans la chaîne recherchée. C'est l'un des opérateurs le plus puissant disponible dans ACCESS. Il existe dans VBA, SQL et dans les expressions.

Joker	Effet
*	Un, plusieurs ou pas de caractères
?	Un caractère
[A-Z]	Un caractère de A à Z
[0-9]	Un caractère de 0 à 9.
[B,C,2]	Le caractère B, C ou 2.
#	Un chiffre

*Référez-vous toujours à l'aide en ligne pour les subtilités de l'opérateur **Like**.*
Certains jokers sont apparus dans les dernières versions d'ACCESS de plus le fonctionnement de ceux-ci sont tributaires de certaines options.

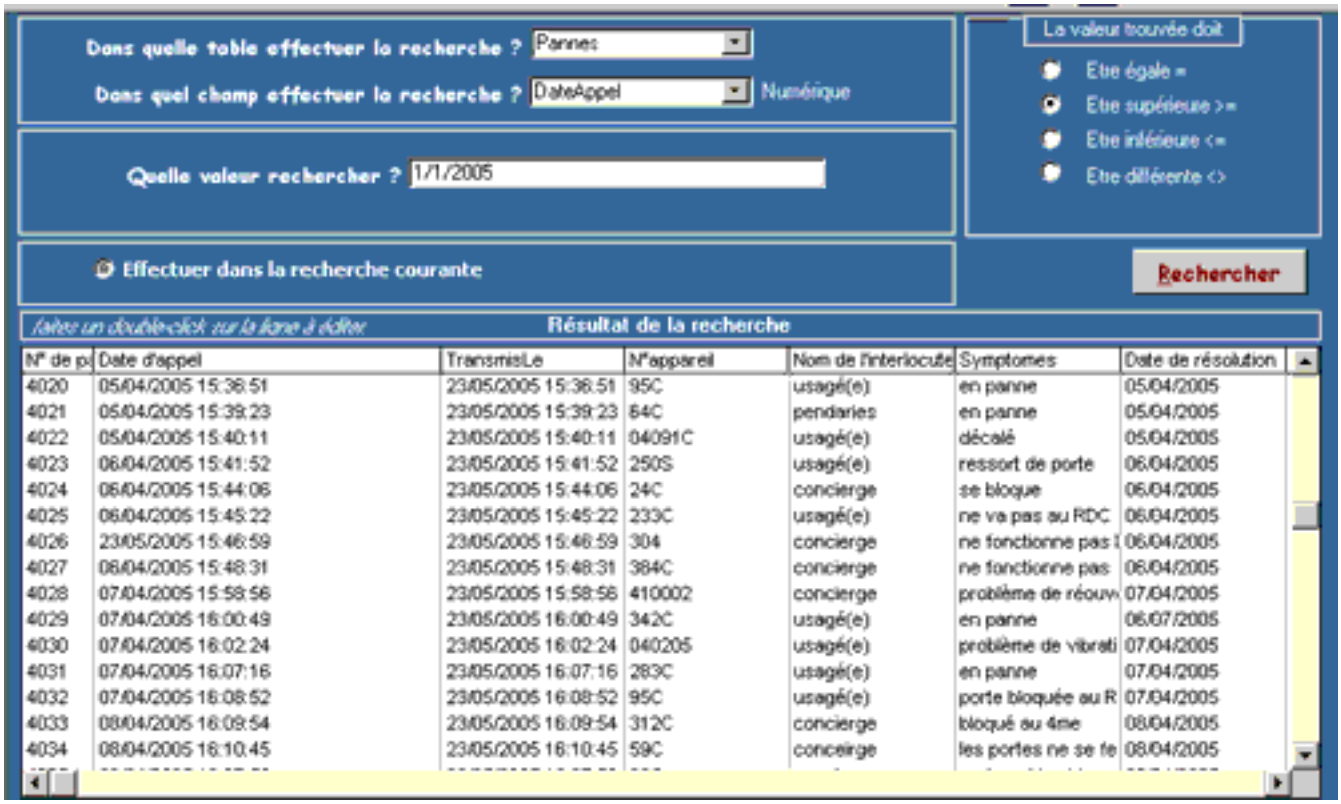
Vous pouvez consulter les articles suivants sur le MSDN. Ceux-ci s'appliquent à ACCESS 2000 mais sont toujours d'actualités.

Fundamental MS JET SQL for ACCESS 2000

Intermediate MS JET SQL for ACCESS 2000

Advanced MS JET SQL for ACCESS 2000

Comme vous l'avez constatez la recherche fonctionne mais seulement avec des champs de type Texte (Texte, Mémo...). Les jokers sont pratiques mais sont complexes pour un utilisateur.



Un exemple de recherche...

VII - Le bouton de recherche - Part II

Dans ce chapitre nous allons mettre en oeuvre la recherche avec plusieurs opérateurs.

VII-A - Les autres opérateurs

- Est strictement identique
- Commence par
- Contient
- Finit par
- Ne contient pas

Ces opérateurs sont les plus utilisés, nous pourrions en mettre plusieurs autres comme "ne commence pas par" ou encore "ne finit pas par" mais il faut bien en laisser un peu à votre réflexion.

VII-A-1 - Mise en place

Pour mettre en oeuvre cette fonctionnalité nous devons définir un groupe de cases d'options dans notre formulaire. Celui devrait se présenter comme ceci :

Un groupe d'option dédié au texte.

Gro d'opt ch d l'opé	Propriété	Valeur
	Nom	Recherche
	Valeur par défaut	3

Nom d'étiquette	Valeur
Etre strictement identique	1
Commencer par la valeur	2
Contenir la valeur	3
Finir par la valeur	4
Ne pas contenir la valeur	5

Très important ! Renommez les cases d'options de opt_Ope1 à opt_Ope5 et leur étiquette de lbl_Etiq1 à lbl_Etiq5 car nous y ferons appel ultérieurement.

La valeur par défaut sélectionnée à l'ouverture du formulaire est définie à 1. Cela correspond à **Etre strictement identique**.
Pour fixer une option différente vous devez agir sur la propriété **Valeur par défaut** du groupe d'option.
Entrez la valeur correspondant à la valeur de l'option.

VII-A-2 - Le code VBA

Pour mettre en place la gestion des opérateurs nous devons mettre en place une structure à base de **Select Case**.
Ouvrez le code du bouton **Recherche** et repérez la ligne suivante.

A remplacer

```
' compose le critere de recherche
strCriteria = strTable & "." & strField & " Like """ & Me.txt_Critere & """"
```

A sa place insérez le code suivant :

A insérer

```
            Select Case Me.opt_Recherche
                  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
                        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
```

La structure **Select Case** permet de composer la chaîne WHERE en jouant avec le joker *. Seule la dernière option (5) est différente puisqu'elle inclut l'opérateur de négation NOT.
Le traitement des champs de type texte est terminé.

VII-B - Les autres types de données

Au cours de ce chapitre nous allons traiter les 21 types de données existants. Cependant certains n'existent pas dans les tables Jet.
Ceux-ci seront regroupés en 3 catégories.

Les 3 catégories

1 Booléen,
2 Numérique et Date,
3 Texte, Mémo et Hyperlien

Pour chaque groupe les opérateurs, la syntaxe SQL et l'affichage changent.

VII-B-1 - Déterminer le type de données

Nous allons dans un premier temps déterminer le type de données. Pour cela nous devons interroger les propriétés du champ de la table choisie.

*Vous devez d'abord ajouter la référence suivante : **Microsoft DAO 3.6 Object Library***
*Dans la fenêtre Microsoft Visual Basic, utilisez le menu **Outils/Références.***

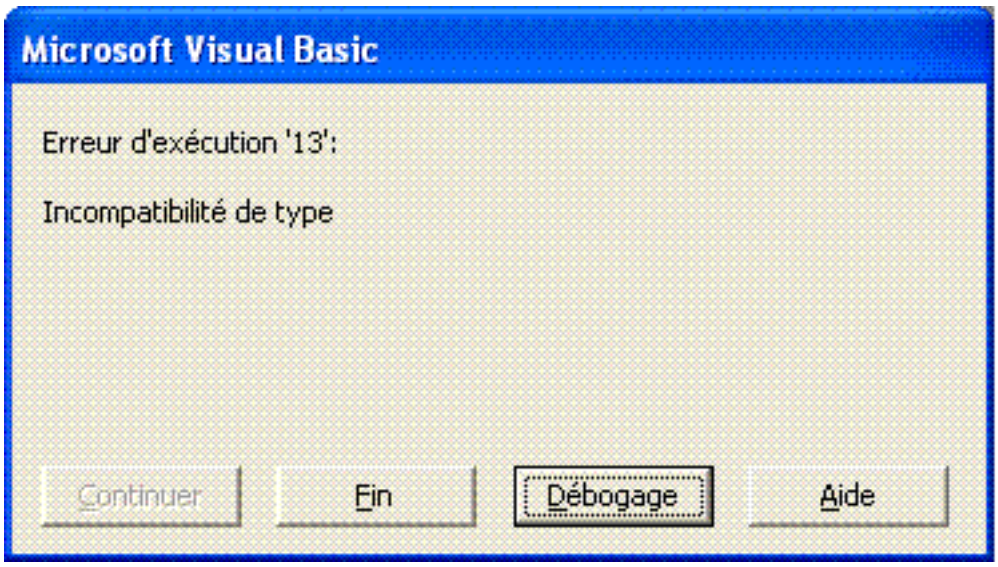
VII-B-1-a - Message d'erreur 13

Si vous utilisez une librairie ADO (**Microsoft ActiveX Data Objects x.x Library**) vous devez placer la DAO avant celle-ci dans la liste. Sinon rajoutez systématiquement à vos déclarations de variables DAO le préfixe **DAO**. Comme dans l'exemple ci-dessous.

Exemple

```
Dim tbl As DAO.TableDef
```

Dans le cas contraire vous serez confronté au message d'erreur suivant :



Le vilain message d'erreur

VII-B-1-b - Le Code

Insérez le code suivant dans un objet module que vous nommerez **Recherche.**

*Outre le fait de pouvoir appeler une fonction ou procédure de n'importe quel endroit de votre application, l'intérêt de placer du code dans un module est de pouvoir y faire appel à partir de la **fenêtre d'Exécution.***

A insérer dans un module

```
Function lf_GetTypeField(lfNameTbl As String, lfNameFld As String)
' 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
```

A insérer dans un module

```
  Set dbs = CurrentDb          ' ouvre la base courante
  Set tbl = dbs.TableDefs(lfNameTbl)   ' ouvre la définition table

  lf_GetTypeField = tbl.Fields(lfNameFld).Type   ' renvoie le type de champ

     Set tbl = Nothing     ' libération des objets
     Set dbs = Nothing

End Function
```

Vous pouvez tout de suite tester cette fonction grâce à la fenêtre **Exécution** présente dans l'interface Microsoft Visual Basic. Tapez **Ctrl+G** ou encore menu **Affichage/Fenêtre Exécution**. Lorsque le curseur clignote dans la fenêtre tapez la syntaxe suivante :

Pour test dans la fenêtre d'exécution

```
? lf_GetTypeField("Matable", "Monchamp")
```

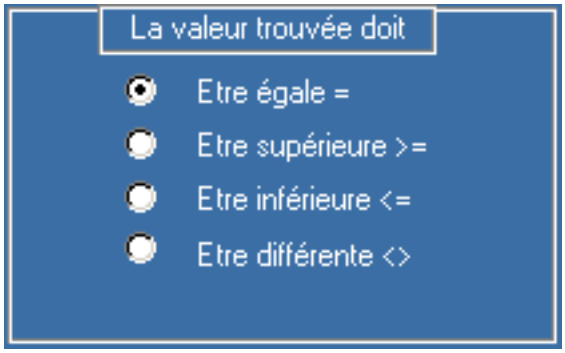
Attention ! Matable représente le nom d'une table existante, attachée ou non, de votre base de données actuelle et Monchamp le nom d'un champ de cette table.

Validez pour voir apparaître le numéro du type de données. C'est celui-ci que nous allons traiter dans notre code.

VII-B-2 - L'affichage

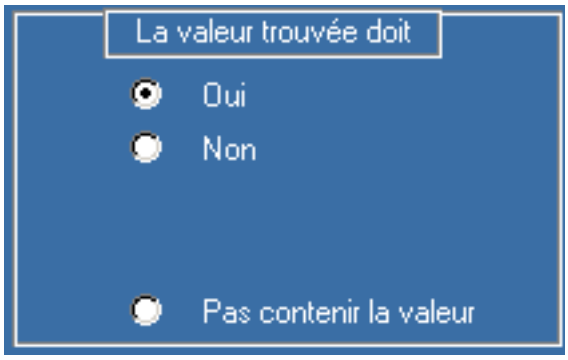
Pour que notre formulaire soit interactif, il convient de changer l'affichage des opérateurs suivant le type de données du champ choisi.

Ayant déjà traité le cas de données texte dans un des chapitres précédents nous n'avons plus que 2 cas à traiter qui sont présentés ci-dessous.



Numérique et date confondu...

Numérique, Date
Egal à =
Supérieur ou égal à >=
Inférieur ou égal à <=
Différent de <>



C'est le ni oui, ni non...

Booléen (oui/non)
Oui
Non
Rien (Null)

*Chaque type de champ peut contenir une valeur nulle. Celle-ci correspond à une absence de saisie. Dans le cas actuel une valeur ni à oui ni à non sera égale à NULL. Attention ! Ce fonctionnement est conditionné par la propriété **Valeur par défaut** du champ correspondant que vous pouvez définir dans la structure de la table.*

Nous pouvons afficher le type de champ au moment du choix. Insérez un étiquette à droite de la zone de liste déroulante **cbo_Champs**. Utilisez les propriétés ci-dessous.

Etiqu Affi l ty d cha	Propriété	Valeur
	Nom	lbl_TypeChamp
	Légende	
	Largeur	un espace

Vous pouvez remarquez que si vous ne saisissez pas de contenu elle disparait. Ceci peut être évité grâce à l'insertion d'un espace.

Pour cette partie il s'agit simplement de changer la légende (l'espace) de l'étiquette suivant le type de données. Le meilleur moment pour changer l'affichage sera après le choix du champ. L'évènement **Après MAJ** de la liste des champs semble tout désigné pour accomplir cette tâche.
Placez-vous sur votre formulaire **Recherche**
Passez en *mode Création.*
Cliquez sur le contrôle **cbo_champ**
Dans les propriétés placez-vous sur **Après MAJ** et utilisez l'icône de création d'évènement pour ouvrir la procédure correspondante.
Entrez le code suivant :

A insérer

```
If IsNull(Me.cbo_Table) Or IsNull(Me.cbo_Champ) Then
   Exit Sub     ' l'un des champs est vide
End If

' initialise les étiquettes de l'opérateur
```

A insérer

```
Me.lbl_Etiq1.Visible = True
Me.lbl_Etiq2.Visible = True
Me.lbl_Etiq3.Visible = True
Me.lbl_Etiq4.Visible = True
Me.lbl_Etiq5.Visible = True
Me.opt_Ope1.Visible = True
Me.opt_Ope2.Visible = True
Me.opt_Ope3.Visible = True
Me.opt_Ope4.Visible = True
Me.opt_Ope5.Visible = True
Me.txt_Critere.Visible = True


Select Case lf_GetTypeField(Me.cbo_Table, Me.cbo_Champ)  ' pour trouver le type du champs


    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

    Case dbByte To dbbinary, dbLongBinary, dbGUID to dbVarBinary, dbNumeric to dbTimeStamp
    ' Numériques / date
         Me.lbl_TypeChamp.Caption = "Numérique"
         Me.lbl_Etiq1.Caption = "Etre égale ="
         Me.lbl_Etiq2.Caption = "Etre supérieure >="
         Me.lbl_Etiq3.Caption = "Etre inférieure <="
         Me.lbl_Etiq4.Caption = "Etre différente <>"
         Me.lbl_Etiq5.Visible = False
         Me.opt_Ope5.Visible = False

    Case dbText, dbMemo, dbChar ' texte / mémo
         Me.lbl_TypeChamp.Caption = "Texte"
         Me.lbl_Etiq1.Caption = "Etre strictement identique"
         Me.lbl_Etiq2.Caption = "Commencer par la valeur"
         Me.lbl_Etiq3.Caption = "Contenir la valeur"
         Me.lbl_Etiq4.Caption = "Finir par la valeur"
         Me.lbl_Etiq5.Caption = "Pas contenir la valeur"

     Case Else
         Me.lbl_TypeChamp.Caption = "Cas non prévu " & lf_GetTypeField(Me.cbo_Table, Me.cbo_Champ)

End Select
```

Faites un essai. Normalement si vous sélectionnez tour à tour des champs n'ayant pas le même type vous devriez voir apparaître les différentes possibilités.

VII-B-3 - Le code VBA

Nous allons, toujours à l'aide du **Select Case**, traiter les 3 types de données. Les tests sont identiques à ceux permettant de déterminer l'affichage des cases d'options.
Tout d'abord nous devons déclarer quelques variables. Placer vous à la suite des déclarations (Dim) et insérez le code suivant.

Code à insérer

```
Dim intTypChamp As Integer
Dim intOpeChamp As Integer
```

Puis repérez la ligne de code suivante :

A repérer

```
Select Case Me.opt_Recherche
```

Remplacez-la par le code suivant :

A insérer

```
intTypChamp = lf_GetTypeField(strTable, strField)  ' pour trouver le type du champs ...
intOpeChamp = Me.opt_Recherche

Select Case intTypChamp

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

       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, ","))

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

       Case dbText, dbMemo, dbChar                     ' texte
            Select Case intOpeChamp
```

Repérez ensuite le code suivant.

A repérer

```
                Case 5 ' ne contient pas
                     strCriteria = "NOT (" & strTable & "." & strField & " Like ""*" &
strCriteria & "*"")"
          End Select
```

Après le **End Select** insérez-y le code suivant.

A insérer à la suite

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

Pour bien comprendre le code il faut expliquer quelques subtilités de VBA et SQL ACCESS.
SELECT CASE et clause TO (Case valeur1 To valeur2) :
La clause **To** permet de définir une plage à la place d'une valeur unique. Elle peut être combiné avec d'autres valeur ou plage à l'aide de la virgule **(,)**. Ainsi on pourra coder ***1 to 10, 12, 15 to 20*** ce qui déclenchera le Case pour les valeurs de 1 à 10, 12 et de 15 à 20.
Traitement Booléen :
Les valeurs booléennes (oui/non) sont représentées par 0 et -1. MS ACCESS et les bases JET stocke les booléens avec ces valeurs numériques. Avec SQL nous pouvons utiliser les constantes Yes et No. Ne pas confondre avec les constantes vbYes et vbNo de Visual Basic qui n'ont pas la même valeur. Notez également la possibilité d'avoir une valeur nulle
Traitement GUID :
Le GUID est le numéro identificateur unique d'un enregistrement dans une entité de réplication. Peu employé lors d'une recherche il est présent uniquement pour l'exhaustivité du traitement.
Traitement Numériques :
La virgule utilisée comme séparateur décimale est remplacée par un point. VBA et les bases de données utilisent le point.
Notez qu'il y a 2 syntaxes pour ce traitement. Celle placée en commentaire est destinée aux versions antérieures à l'apparition de la fonction **Replace()**.
Traitement Dates :
Pour que SQL utilise les dates en tant que telles, et non comme une suite de divisions, le signe / étant également l'opérateur de division, celles-ci doivent obligatoirement être entourées par des dièses (#).

Si un champ, qu'importe son type, n'est pas saisi, que sa valeur par défaut n'a pas été défini il sera rempli par le valeur Null.

VIII - Le bouton de recherche - Part III

VIII-A - Recherche multicritères par imbrication

Ce chapitre tant attendu traite de la recherche multicritères par imbrication ou tout simplement comment faire une recherche dans le résultat de la recherche précédente.
Pour cela nous avons simplement besoin d'une case d'option indépendante et d'un peu de code.
Insérez une case d'option indépendante.

Propriété	Valeur
Nom	Opt_RechCourante
Valeur par défaut	Vrai
Effectuer dans l'étiquette	recherche courante

Maintenant que le contrôle est en place nous pouvons passez au code. Celui-ci n'est pas lié à ce contrôle mais au bouton de recherche.

VIII-A-1 - Le code VBA

Cette fonction n'a rien d'extraordinaire puisqu'elle manipule la chaîne SQL déjà composée et présente dans la propriété Contenu (Rowsource en VBA) du contrôle Lst_resultat.
Recherchez le code suivant.

Code à remplacer

```
' construit la requête sql
strsql = "SELECT DISTINCTROW " & strTable & ".*"
strsql = strsql & " FROM " & strTable
strsql = strsql & " WHERE ((" & strCriteria & "));"
```

Remplacez-le par celui-ci.

Nouveau code

```
If Me.Opt_RechCourante And Not Len(Me.Lst_Resultat.RowSource) = 0 Then
   If Not Me.Lst_Resultat.RowSource Like "*FROM " & strTable & "*" Then
      MsgBox "La recherche précédente ne porte pas sur la même table que la recherche actuelle.", _
          vbExclamation + vbOKOnly, "Erreur"
      Exit Sub
   End If
   strsql = Left(Me.Lst_Resultat.RowSource, Len(Me.Lst_Resultat.RowSource) - 3)
   strsql = strsql & " AND " & strCriteria & "));"
Else
   ' construit la rq sql
   strsql = "SELECT DISTINCTROW " & strTable & ".*"
   strsql = strsql + " FROM " & strTable
   strsql = strsql + " WHERE ((" & strCriteria & "));"
End If
```

Si vous avez choisi une recherche récursive et que la recherche précédente concerne la même table, la chaîne SQL qui est stockée dans la propriété **Contenu** (Rowsource) du contrôle **Lst_Resultat** est récupérée. Les 3 derniers caractères "));" sont omis et nous rajoutons un **AND** suivi du nouveau critère.
Dans le cas ou ce n'est pas une recherche récursive, nous créons une nouvelle chaîne SQL.


Vous pouvez également traiter les différents opérateurs au moyen d'une liste déroulante ou encore d'un cadre de case d'options comme celui des opérateurs de comparaisons.

Voilà notre recherche récursive est fin prête. Faites quelques tests avant de passer au prochain chapitre.


N'hésitez pas à mettre des points d'arrêts dans le code (F9) pour suivre et comprendre le déroulement du programme. Utilisez la fenêtre Espion (Menu Affichage/Fenêtre Espions) pour observer les valeurs pendant l'exécution.

IX - Vers un formulaire totalement générique

Dans ce chapitre nous allons aborder un point important de l'intégration du formulaire de recherche. La composition automatique de la liste des tables.

IX-A - Les objets nécessaires

Veuillez ajouter la référence suivante à votre application :

Pour composer la liste des tables automatiquement nous devons créer une table de réception des informations. Cette table sera renseignée au lancement du formulaire et son contenu s'affichera dans la zone liste **cbo_table**.

Ta tbl_Tem	Nom du champ	Type	Longueur
	[illegible]		

IX-B - Le code VBA

Ouvrez le module **Recherche** qui doit déjà contenir la fonction **lf_GetTypeField()** à la suite insérez-y le code suivant.

A insérer

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

Dim qrs As TableDefs
Dim rst As Recordset

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

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

' rempli    la table temporaire
Set qrs = CurrentDb.TableDefs
Set rst = CurrentDb.OpenRecordset("tbl_TempLstTbl")

For i = 0 To qrs.Count - 1
    ' ecarte les tables temp et systeme
    If Not (qrs(i).Name Like "*Temp*") And Not (qrs(i).Name Like "Msys*") And _
    Not (qrs(i).Name Like "*tmp*") Then
       rst.AddNew
       rst.Fields(0) = qrs(i).Name
       rst.Update
    End If
Next
lf_GetTableList = rst.RecordCount

rst.Close
Set rst = Nothing
Set qrs = Nothing
DoCmd.SetWarnings True
End Function
```

C'est le code qui va réaliser l'inventaire de tous les noms de tables de votre base, attachées ou non à l'exclusion des tables systèmes (Msys), des tables Temporaires (Temp) et autres objets temporaires cachés (~tmp).

Vous pouvez constater que la table de réception (tbl_TempLstTbl) est vidée à chaque exécution et que la fonction renvoie le nombre de tables enregistrées dans la table (rst.recordcount) vous comprendrez pourquoi dans le prochain chapitre.

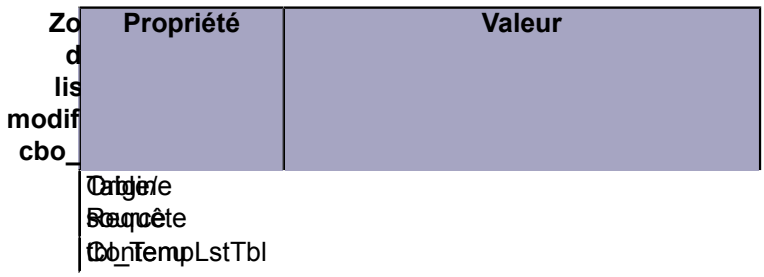
IX-C - Le formulaire

Après ce que nous avons vu au cours de ce tutoriel, l'intégration de cette fonctionnalité n'a rien de complexe.
Affichez les propriétés du formulaire, dans la propriété **Sur ouverture**. Insérez le code suivant.

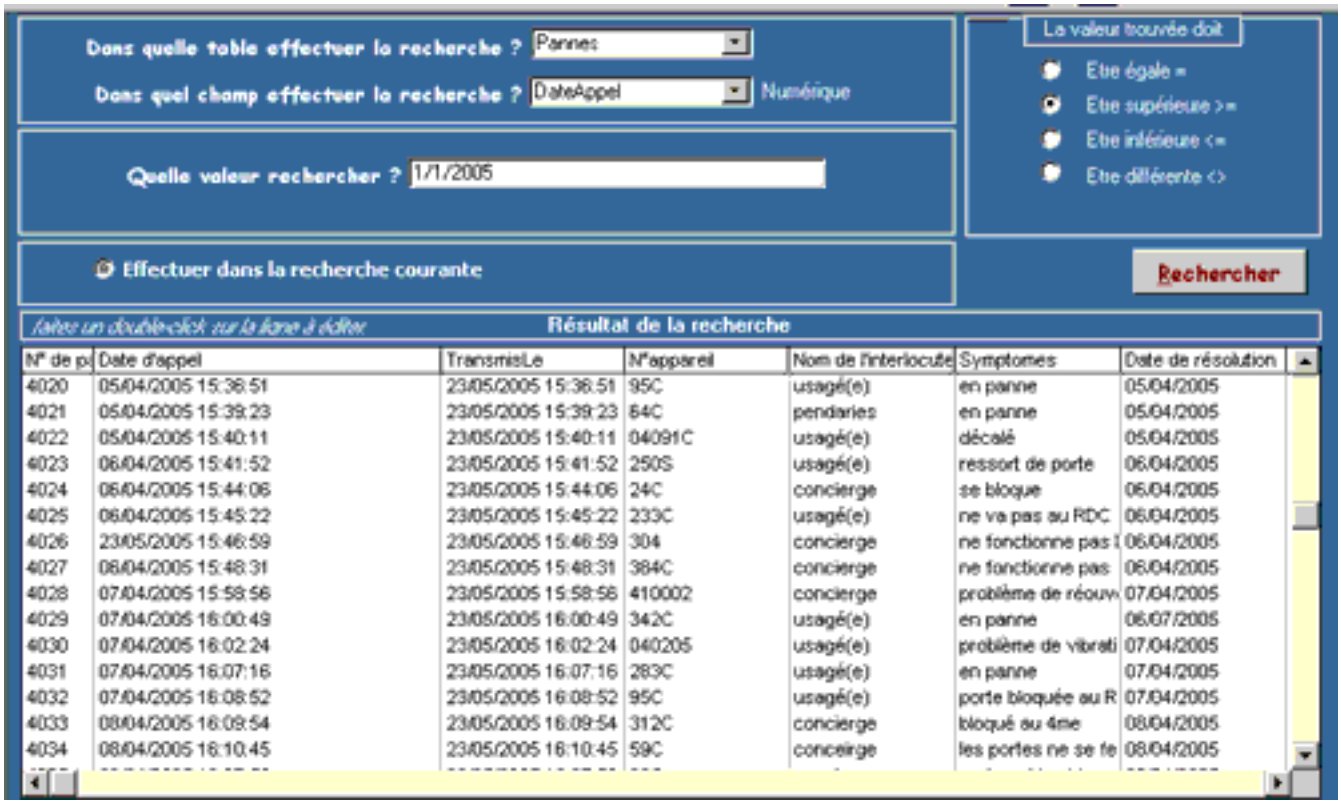
A insérer

```
' crée la liste des tables
If lf_GetTableList() = 0 Then
   MsgBox "Pas de tables dans cette application .", vbInformation + vbOKOnly, "Erreur"
   Cancel = True
End If
```

Réglez maintenant les propriétés de **cbo_table** suivant le tableau ci-dessous. Dans le chapitre précédent nous avons pu observer que la fonction renvoyait le nombre de tables inscrites. Cela nous permet de vérifier rapidement que l'on ne lance pas le formulaire pour rien.



Sauvez le formulaire et le module puis rouvrez-le.
Allez dans la liste des tables pour constater que l'ensemble des tables de votre base de données y figurent.
Voici à quoi devrait ressembler votre formulaire une fois en fonctionnement.



Le formulaire fini... enfin presque !

X - Intégration

C'est très simple et totalement portable d'une application à l'autre, cela peut même être utilisé indépendamment de toute application, reste pour cela à créer un formulaire d'attachement si vous utilisez un runtime.
Pour installer ce formulaire vous devez importer ces 3 objets :

- le formulaire **frm_Recherche**,
- la table **tbl_TempLstTbl**,
- le module indépendant **Recherche**.

Le mois prochain nous agrémenterons notre formulaire d'autres fonctionnalités comme le traitement des opérateurs logiques SQL, et bien d'autres choses.

XI - Remerciements

Je tiens à remercier : **Tofalu, Papy Turbo, Argyronet** pour le temps passé en relecture et correction.
Boulap pour les erreurs signalées dans le code. 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.