Initiation - Débogage : requêtes écrites par VBA

par Charles A.

Date de publication : 01/10/2005

Dernière mise à jour : 01/10/2005

Thèmes abordés : . débogage d'une requête . manipulation des requêtes et chaînes de caractères Niveau requis : débutant

I - Objectif

Ce très court tutoriel a pour objectif de donner une méthode de débogage lors de requêtes écrites par du code VBA.

Nous nous situons dans le cas fréquent où le développeur souhaite ouvrir un recordset / modifier une requête / modifier un Recordsource par le Code.

Le codage VBA ne permet pas de vérifier la validité d'une expression SQL qui ne sera évaluée qu'à l'exécution.

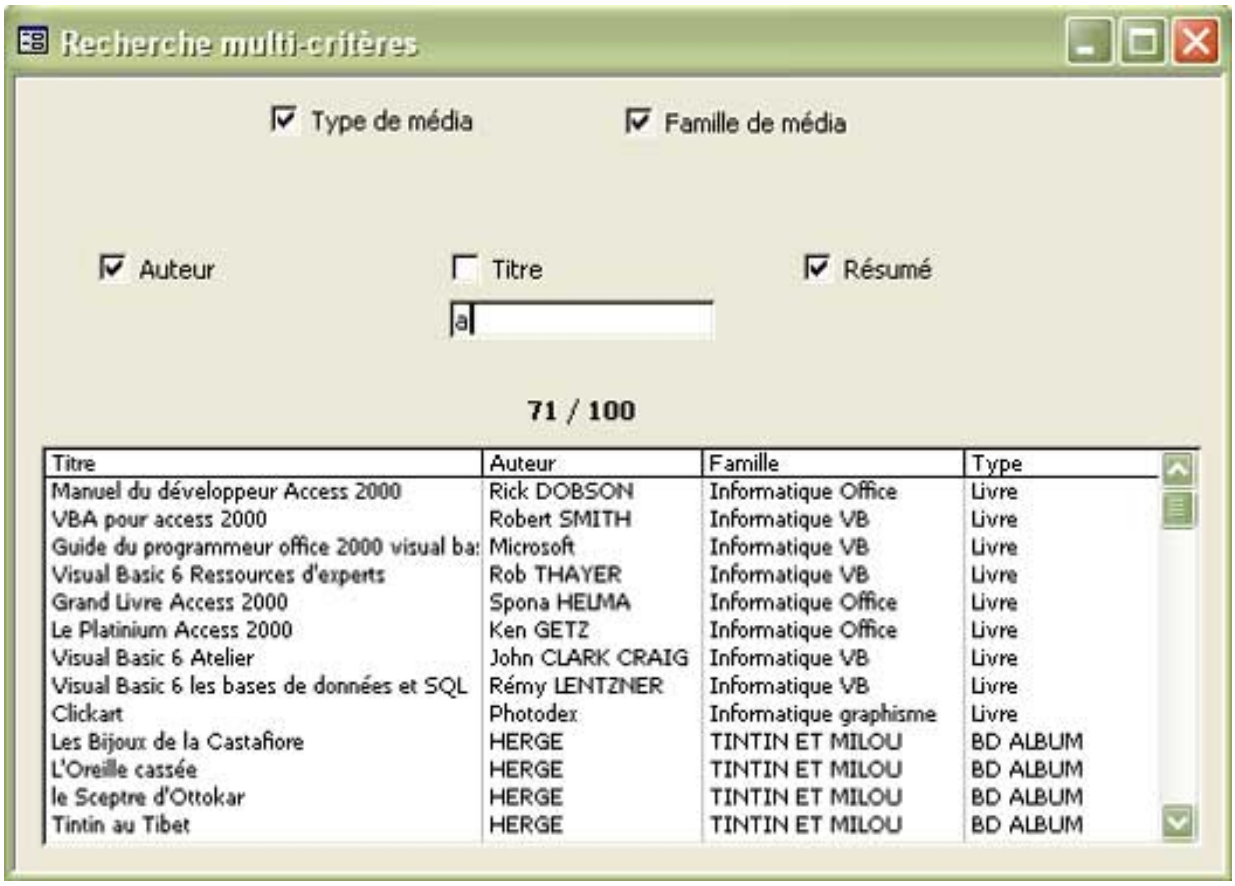
II - Constat de bogue (bug)

Nous avons une procédure de recherche multicritères (cf. mon tutoriel) qui hélas ne fonctionne pas.

(Général) RefreshQuery

```
Private Sub RefreshQuery()
Dim SQL As String
Dim SQLWhere As String

  SQL = "SELECT CodMedia, Titre, Auteur, Famille, Type FROM Medias Where Medias!CodMedia <> 0 "
  If Not Me.chkAuteur Then
      SQL = SQL & "And Medias!Auteur like '*" & Me.txtRechAuteur & "*' "
  End If
  If Not Me.chkFamille Then
      SQL = SQL & "And Medias!Famille = '" & Me.cmbRechFamille & "' "
  End If
  If Not Me.chkResume Then
      SQL = SQL & "And Medias!Résumé like '*" & Me.txtRechResume & "*' "
  End If
  If Not Me.chkTitre Then
      SQL = SQL & "And Medias!Titre like '*" & Me.txtRechTitre & "*' "
  End If
  If Not Me.chkType Then
      SQL = SQL & "And Medias!Type = '" & Me.cmbRechType & "' "
  End If
  SQLWhere = Trim(Right(SQL, Len(SQL) - InStr(SQL, "Where ") - Len("Where ") + 1))
  SQL = SQL & ";"
  Me.lblStats.Caption = DCount("*", "Medias", SQLWhere) & " / " & DCount("*", "Medias")
  Me.lstResults.RowSource = SQL
  Me.lstResults.Requery

End Sub
```

Et lorsque nous modifions un critère de recherche rien ne se passe, le débogage serait identique si nous avions un message d'erreur.



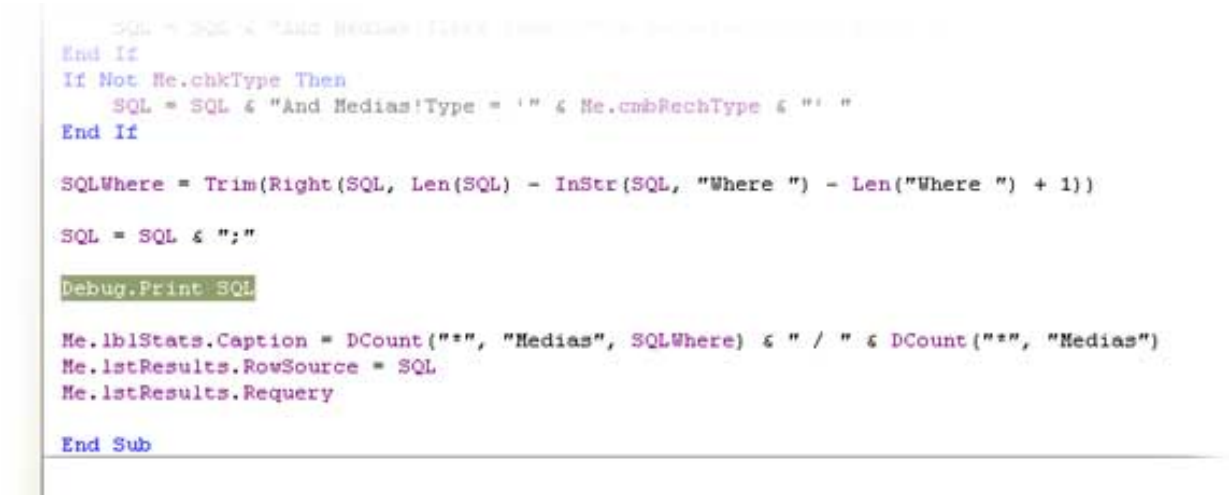
Que faire ?

III - Utilisation du Debug.Print

Nous insérons dans le code, après la dernière mise à jour de la chaîne SQL, la ligne suivante :

```
Debug.Print SQL
```

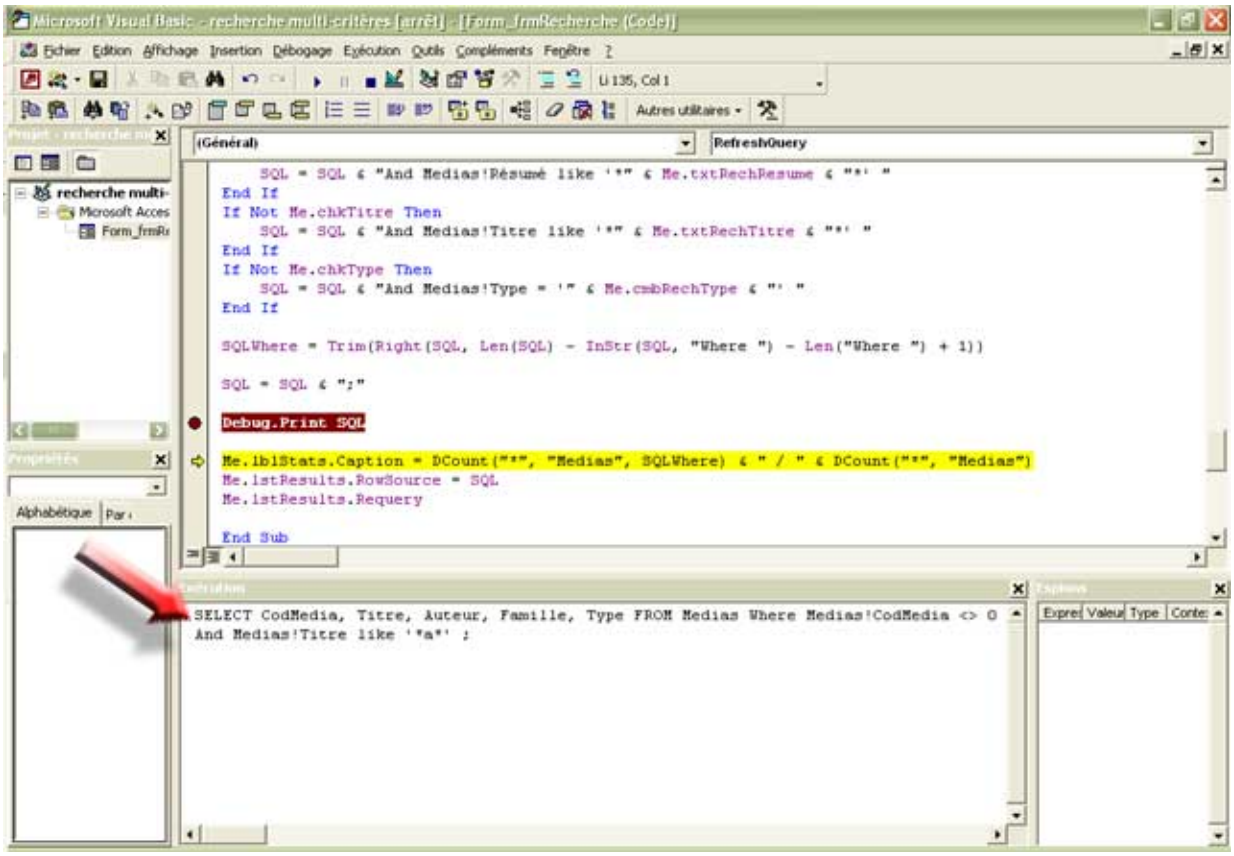
Nous l'insérons à cet endroit :



```
End If
If Not Me.chkType Then
    SQL = SQL & "And Medias!Type = '" & Me.cmbRechType & "' "
End If

SQLWhere = Trim(Right(SQL, Len(SQL) - InStr(SQL, "Where ") - Len("Where ") + 1))

SQL = SQL & ";"

Debug.Print SQL

Me.lblStats.Caption = DCount("*", "Medias", SQLWhere) & " / " & DCount("*", "Medias")
Me.lstResults.RowSource = SQL
Me.lstResults.Requery

End Sub
```

Ce qui signifie que nous demandons à l'objet Debug d'afficher la valeur de notre variable SQL.

Cet affichage se fera dans la fenêtre Exécution d'Access, elle-même située dans la fenêtre VBA.



Pour les versions 97 et inférieures, la fenêtre débogage est une fenêtre indépendante, pour les versions 2000 et ultérieures, elle apparaît dans VBA.

Dans tous les cas : **Ctrl + G** permet de la mettre au premier plan.

Si vous êtes familier avec le SQL ou que l'erreur est criante, le Debug.Print peut suffire à trouver l'erreur.

Cependant dans certains cas, la lecture du SQL ne permet un débogage immédiat, nous allons donc passer par une requête.

L'utilisation de **Debug.Print** fait partie de l'éventail des outils de débogage, la partie suivante va tenter d'en brosser les principaux.

IV - Généralisation à l'utilisation des outils de débogage

Cette partie vise à présenter de manière synthétique l'éventail des outils d'aide au débogage.

IV-A - Définition de point d'arrêt

Les points d'arrêts sont utiles dans le cas d'une procédure au bogue muet.

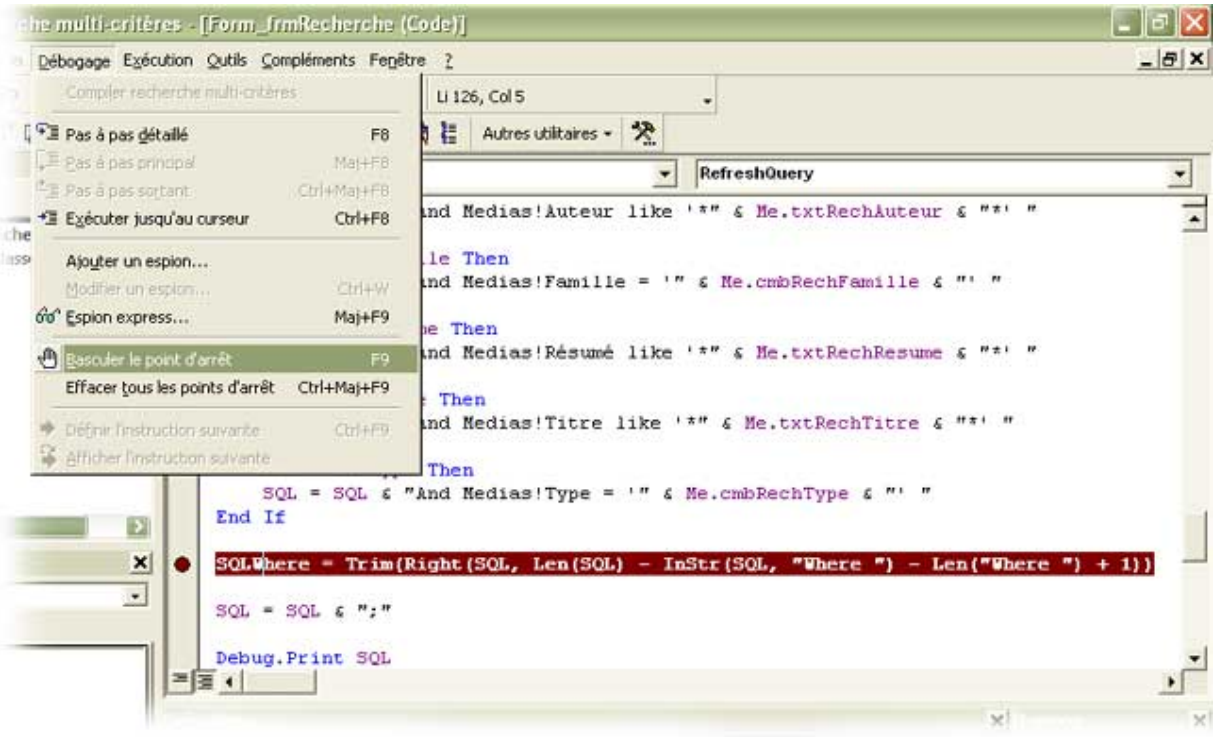
C'est à dire, un traitement qui ne produit aucune erreur visible, mais qui ne donne pas le résultat escompté.

Pour insérer un point d'arrêt, vous pouvez au choix :

. cliquer dans la marge grise

. appuyer sur **F9**

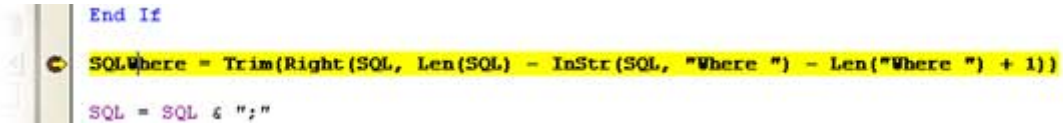
. le faire dans le menu Débogage



Visuellement, la ligne est surlignée en rouge, avec un point rouge dans la marge grise.

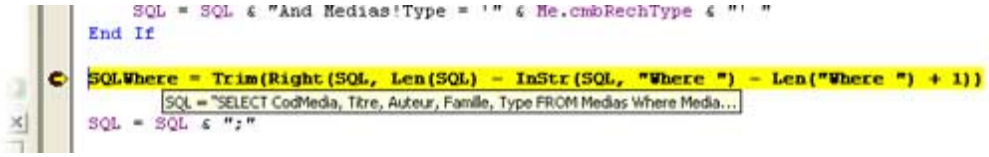
Cela signifie que lors de l'exécution le code va s'arrêter sur cette ligne, soit **avant** l'exécution de celle-ci.

La ligne va être surlignée en jaune, comme sur l'image suivante :

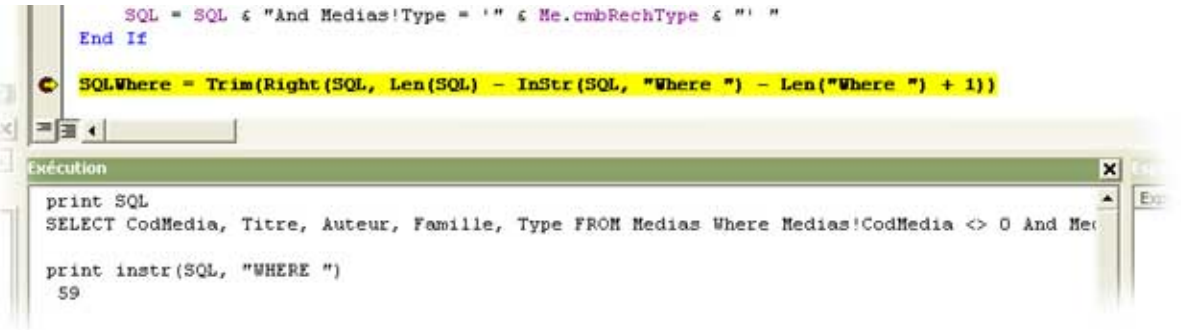


Access vous rend la main pour effectuer votre débogage, et ce de plusieurs manières différentes.

Vous pouvez survoler avec la souris, le nom des variables, dans notre cas, la souris survole la Variable **SQL**.

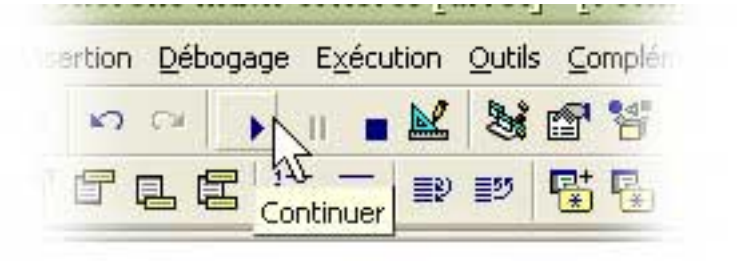


Mais vous pouvez tout aussi bien évaluer des expressions dans la fenêtre exécution : avec la fontion Print suivie du nom de la variable.



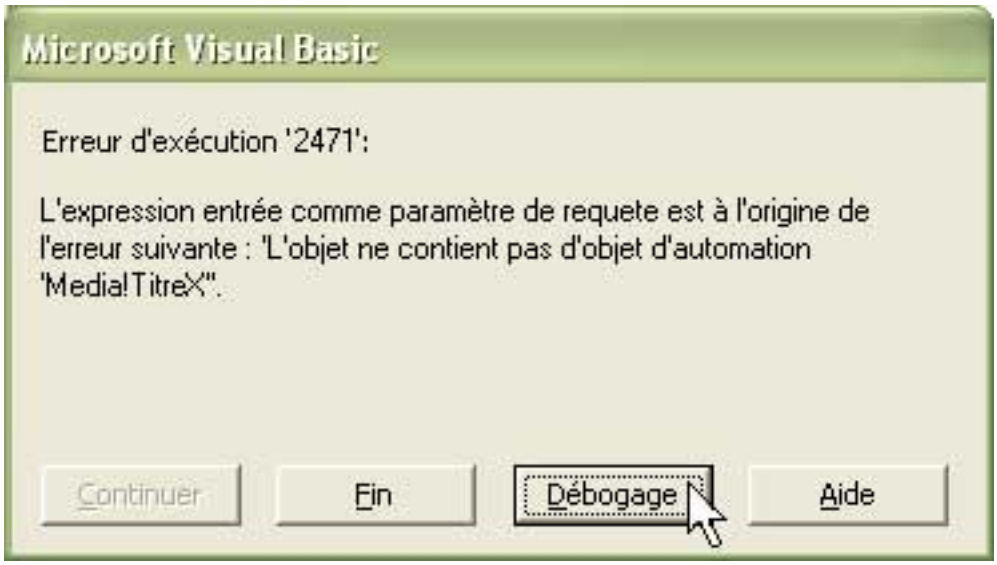
Vous pouvez également corriger à chaud (pendant l'exécution) le code qui suit votre point d'arrêt.

Pour relancer le traitement après vos modifications, il vous suffit de cliquer sur le bouton "continuer" en forme de lecture, ou d'appuyer sur la touche **F5** :



IV-B - Le mode pas à pas

Ce mode est accessible depuis un point d'arrêt, ou un bogue rencontré sur lequel vous avez cliqué sur **Débogage**.



Le mode est accessible par le menu ou par la touche **F8**.

Ce mode est utile quand vous n'avez pas identifié la ligne de code qui pose problème.

Vous allez pouvoir parcourir chaque ligne de votre code pour l'évaluer.

Dans les faits, Access va nous rendre la main sur chaque ligne en la surlignant en jaune.

Ce qui nous laisse la possiblité de modifier le code de la ligne surlignée ou le code des lignes suivantes.

Pour passer à la ligne suivante, il suffit d'appuyer sur **F8**.

IV-C - Les espions

Sous ce titre "aventureux" se cache une fonctionnalité de débogage qui permet d'afficher directement la valeur d'une expression.

Cet affichage a lieu dans une fenêtre dédiée (les affichages varient selon que l'application soit sous 97 ou sous 2000 et ultérieures).

Cas pratique :

. nous définissons un espion sur l'expression SQL

. nous définissons un point d'arrêt sur la première ligne de code

. nous allons faire du pas à pas pour regarder comment notre espion évolue.

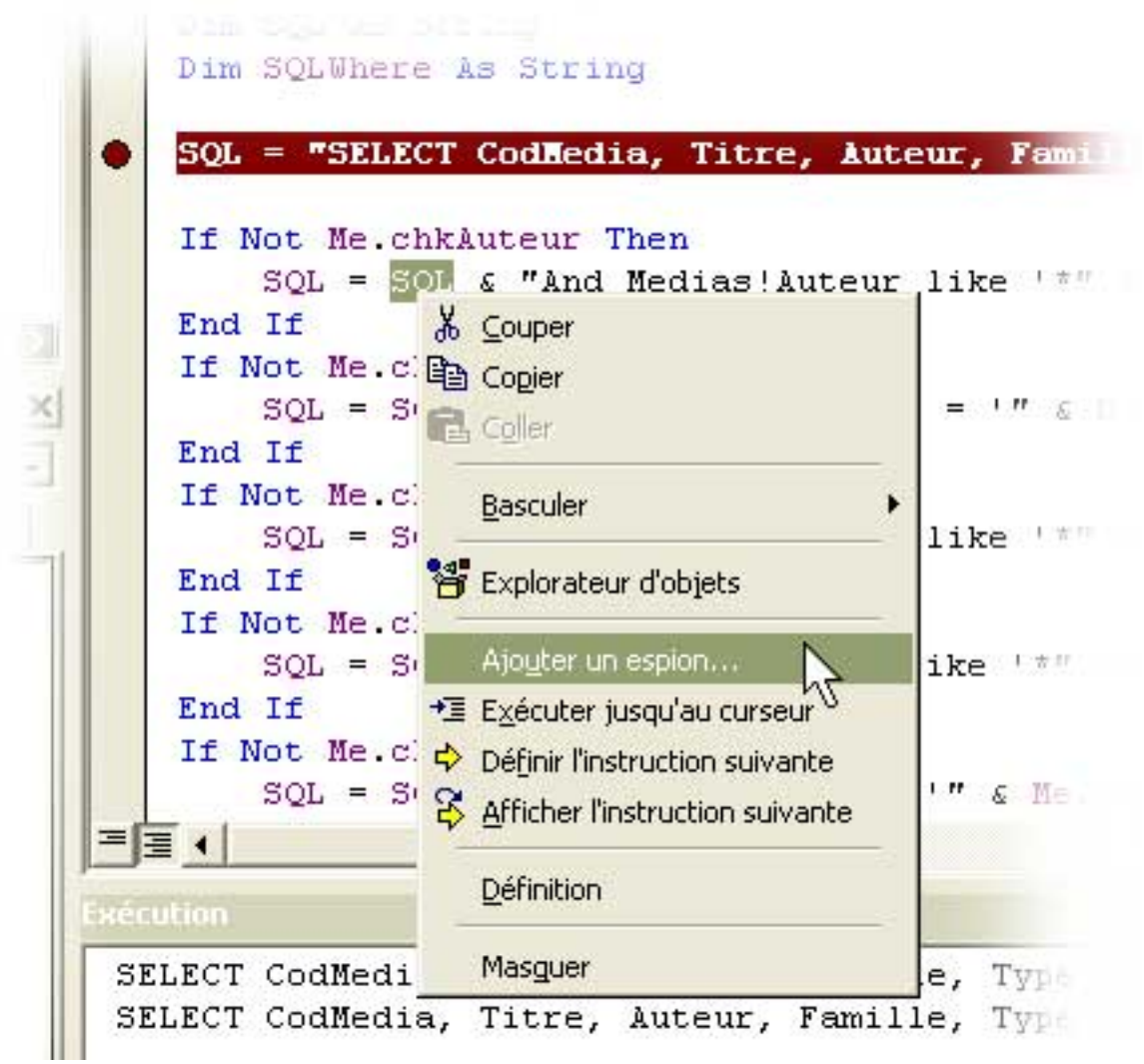
Pour définir un espion :

1- nous sélectionnons une expression :

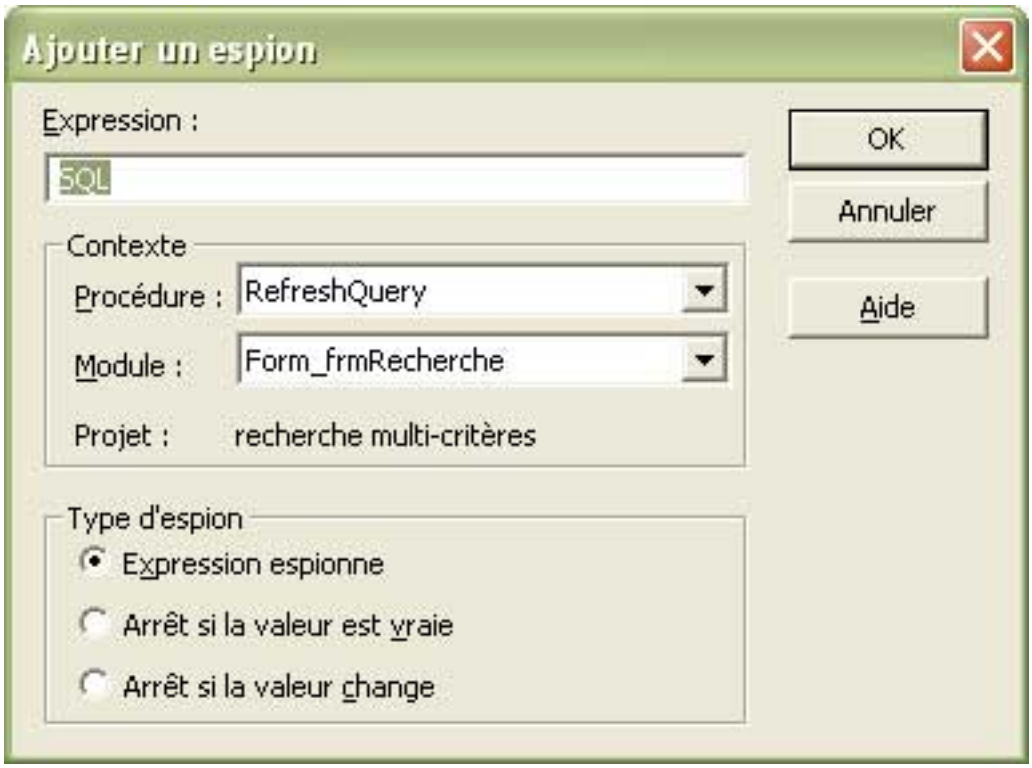
. soit un nom de variable

. soit une portion de code : égalité qui renverra un True ou False, ou une fonction

2- Clic droit : Ajouter un espion



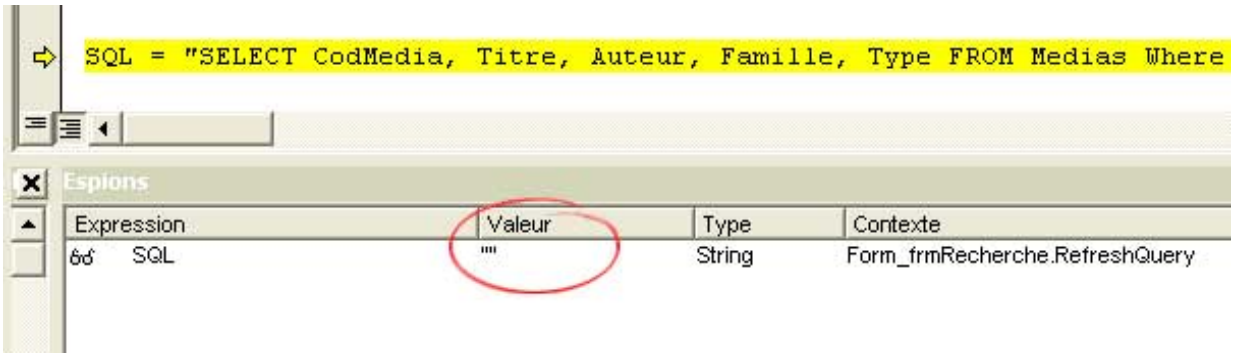
puis



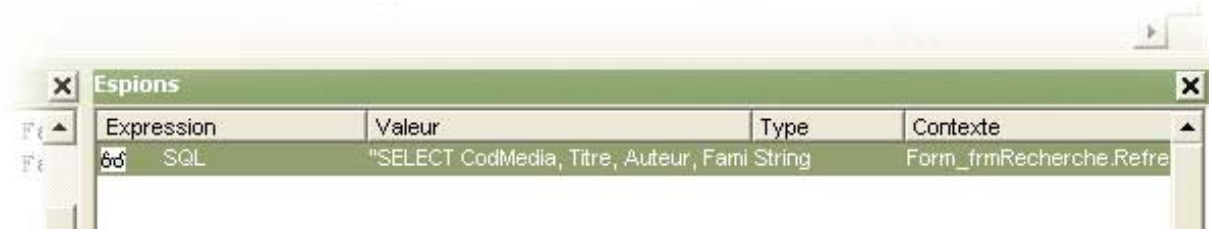
une ligne dans le volet "Espions" apparaît.

A l'exécution, examinons les valeurs que prend notre espion durant le mode pas à pas.

sur la première ligne (notre point d'arrêt) :



Puis dans le mode pas à pas :



Nous pouvons ainsi suivre l'évolution en lecture directe de l'évaluation d'une expression.

NB : Les valeurs prises par les espions ne peuvent excéder 255 caractères.

V - Débogage par une requête de test

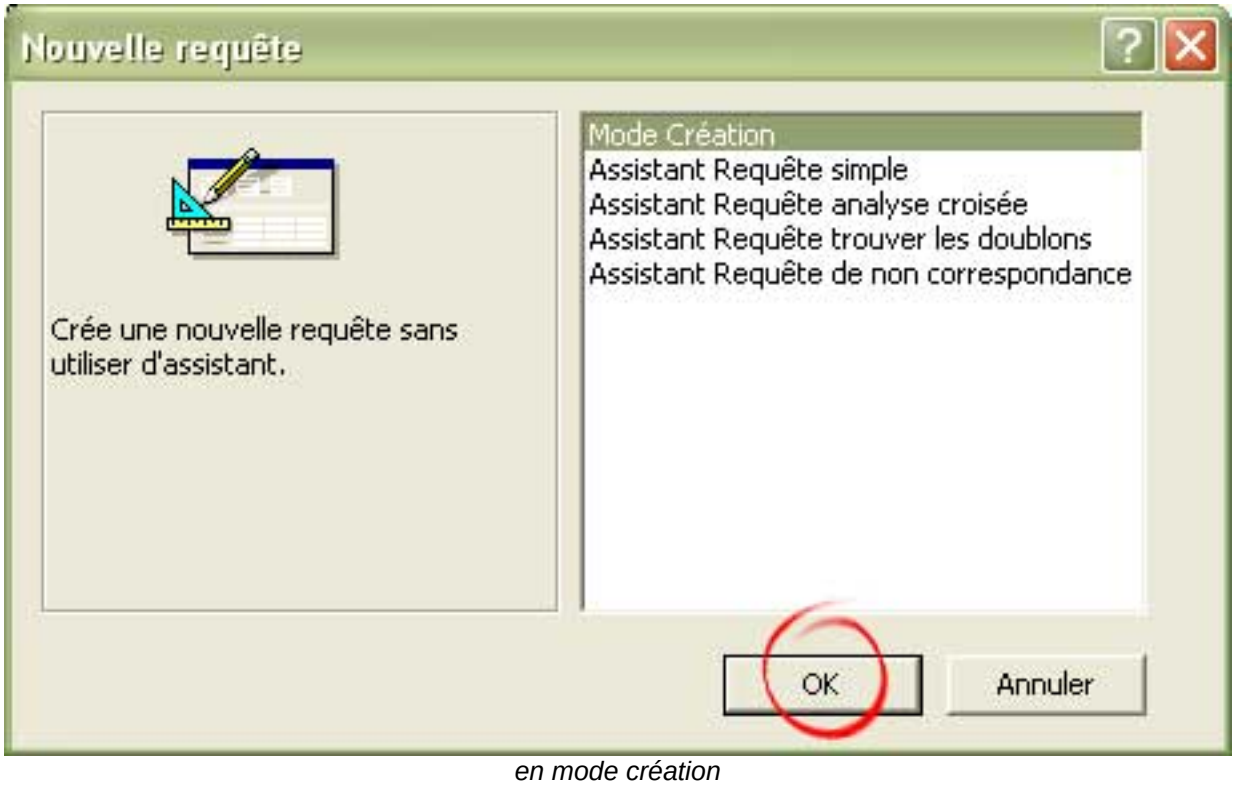
Revenons maintenant à notre cas principal, c'est à dire le débogage d'une requête écrite par le code VBA.

V-A - Par copier-coller

C'est une méthode "manuelle".

Nous allons chercher le SQL qui a été affiché dans la fenêtre Exécution, nous le copions et nous allons le coller dans une requête vierge.

Nous créons une nouvelle requête.

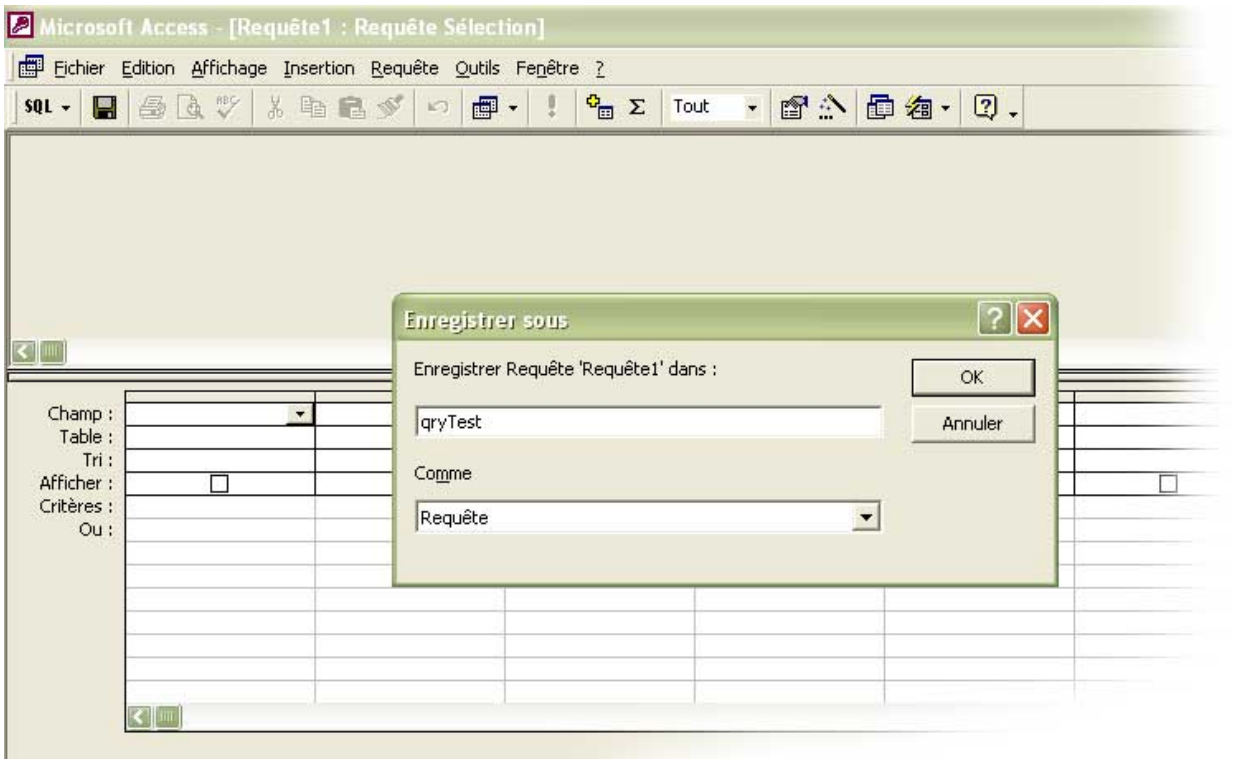


en mode création

Pour que la requête soit vierge, nous n'ajoutons aucune table.

cliquons sur Fermer pour n'ajouter aucune table

Enregistrons notre requête : **qryTest**



Nous passons en mode SQL.

Et nous n'avons plus qu'à coller le SQL.

Nous reviendrons sur l'analyse plus tard.

V-B - Par le code

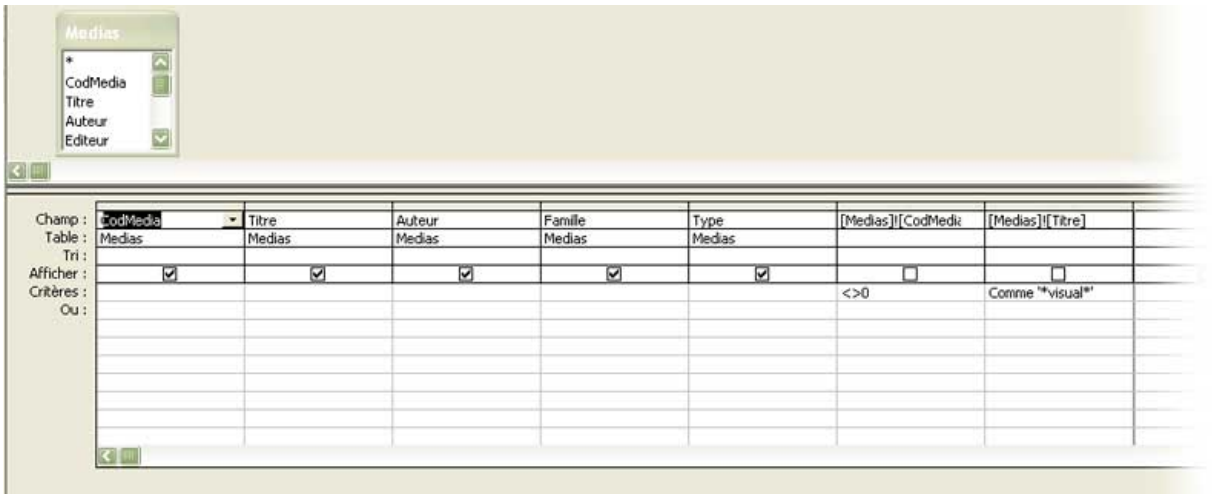
Nous insérons les lignes suivantes :

```
SQLWhere = Trim(Right(SQL, Len(SQL) - InSt

SQL = SQL & ";"

Debug.Print SQL
CurrentDb.QueryDefs("qryTest").SQL = SQL
DoCmd.OpenQuery "qryTest", acViewDesign
```

La première ligne permet d'attribuer dynamiquement à une requête notre chaîne SQL.

La seconde ligne ouvre en mode modification la requête de test.

Ce qui fait qu'à l'exécution de la sub, la requête de test va s'ouvrir en mode création en affichage QBE.

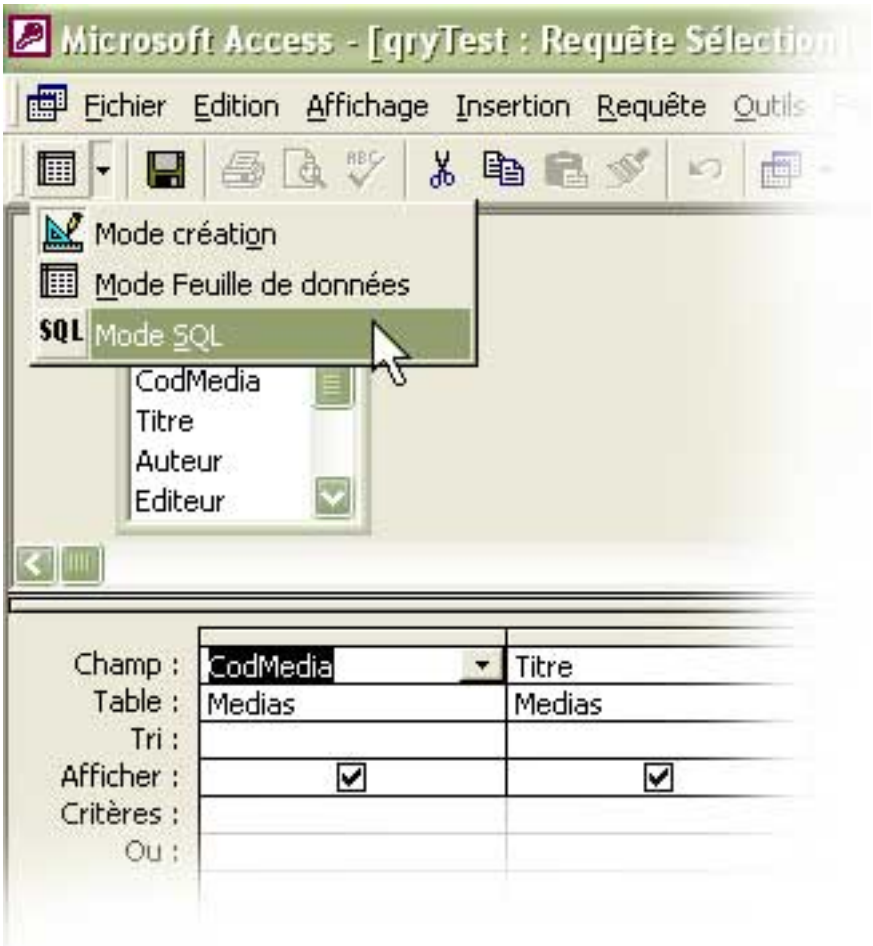


Ce mode d'affichage est un des plus visuels, surtout lorsque nous ne sommes pas experts en SQL.

V-C - Résolution du bogue

Dans ce cas précis, nous identifions l'erreur : le critère ="*Visual*" n'est pas valide, il faut utiliser Comme (Opérateur Like en SQL).

Après correction, nous repassons en mode d'affichage SQL :



Il nous suffira de nous inspirer du SQL valide pour corriger notre procédure VBA.

VI - Frequently Abominable Queries (FAQ)

VI-A - Confusion Chaine SQL et Variable

```
If Not Me.chkFamille Then
    SQL = SQL & "And Medias!Famille = 'Me.cmbRechFamille' "
End If
```

Le Debug.Print vous montrera que votre requête va rechercher les Familles dont la valeur est la chaîne "Me.cmbRechFamille".

Vous devez sortir Me.cmbRechFamille de la chaîne pour qu'Access évalue la valeur, et c'est cette valeur qui sera dans votre chaîne SQL.

```
If Not Me.chkFamille Then
    SQL = SQL & "And Medias!Famille = '" & Me.cmbRechFamille & "' "
End If
```

VI-B - Variable vide

Cette fois votre code est irréprochable, mais le Debug.Print révèle ceci :

```
.... WHERE Medias!Famille = '' ....
```

Votre variable est Null, les causes possibles peuvent être :

. Contrôle situé sur un formulaire non accessible / non ouvert / dont le chemin d'accès est erronné

. Erreur de libellé sur le nom

. Erreur de typage de la variable

Assurez vous que la variable prend bien une valeur, si votre application comporte la possibilité d'un Null, prévoyez le grâce à la fonction Nz().

VI-C - Type de données

Votre code semble exact, mais hélas il n'est pas approprié avec les normes de l'implémentation SQL d'Access.

Pour rappel :

```
' Gestion d'une Date : format US entre ##
WHERE [Table]![MonChampDate] = #mm/dd/yyyy#
' Gestion d'un chaîne : entre quotes ' '
WHERE [Table]![MonChampChaine] = 'Quelquechose'
' Gestion d'un numérique : ne rien mettre
WHERE [Table]![MonChampNumerique] = 2005
```

VI-D - Espaces manquants ou superflus

Le Debug.Print va vous permettre d'identifier à l'oeil nu ce genre d'erreur.

```
SQL = SQL & "WHERE [MaTable]![MonChamp] = '" & maVariable & "'"
SQL = SQL & "AND [MaTable]![MonChamp] = '" & maVariable & "'"
```

ce code d'apparence correcte va donner un SQL de ce type

```
WHERE [MaTable]![Monchamp] = 'Toto'AND ...
```

Le remède :

```
SQL = SQL & "WHERE [MaTable]![MonChamp] = '" & maVariable & "' "
SQL = SQL & "AND [MaTable]![MonChamp] = '" & maVariable & "' "
```

Ensuite le cas d'un espace superflu à l'intérieur d'un critère, suite à une faute d'inattention :

```
SQL = SQL & "WHERE [MaTable]![MonChamp] = ' " & maVariable & " ' "
SQL = SQL & "AND [MaTable]![MonChamp] = '" & maVariable & "' "
```

va donner :

```
WHERE [MaTable]![Monchamp] = ' Toto ' AND ...
```

VI-E - Présence de Simple Quote (') ou Double Quote (") dans la variable

Nous avons vu que l'implémentation SQL d'Access requiert la mise entre quotes ' ' d'une chaîne en critère.

Ce codage est cependant mis en échec quand la valeur de la variable contient un quote.

Exemple :

```
SQL = SQL & "WHERE [MaTable]![MonChamp] = '" & maVariable & "' "
```

Si maVariable vaut : "l'écureuil"

```
WHERE [MaTable]![Monchamp] = 'L'écureuil' AND ...
```

Ce qui vous vaut une belle erreur.

Le remède est de doubler ce Quote.

Pour les versions 2000 et ultérieures, il suffit d'utiliser la fonction **Replace()**.

```
SQL = SQL & "WHERE [MaTable]![MonChamp] = '" & Replace(maVariable, "'", "''") & "' "
```

Pour les versions 97 et antérieures, il faut faire de même en utilisant une fonction **Replace()** de substitution (disponible dans la FAQ)

> ***Vous rencontrerez le même type de problème si vous délimitez vos chaînes par des double quotes (") et si la variable contient un double quote.***
>
> ***La solution est là aussi de doubler ce caractère, toujours avec la fonction Replace()***

VI-F - Autres

Cette liste des FAQ n'est pas exhaustive, n'hésitez pas à me contacter pour que j'ajoute d'autres types d'erreurs fréquentes.

VII - Conclusion

Cet article vous a, je l'espère, donné quelques pistes pour mieux écrire vos requêtes via VBA et dans le cas où les problèmes subsistent les déboguer.

J'adresse un grand merci à toute l'équipe de rédaction d'Access pour leurs relectures et leurs pertinentes suggestions d'améliorations.

Un remerciement tout particulier pour Argyronet qui a superbement magnifié les copies d'écran.