Getting Past Doh – More Than One Quote VBA

Phil Graham, Bradford Shearbridge

At the Access Lunchtime User Group (www. accessusergroups.org/lunch/) meeting on March 28, 2017, I asked those in attendance about their experience in searching text or string variables that contained an apostrophe or single quote. I was using the customer table from the Northwind database and found a problem when searching for Martin O'Donnell.

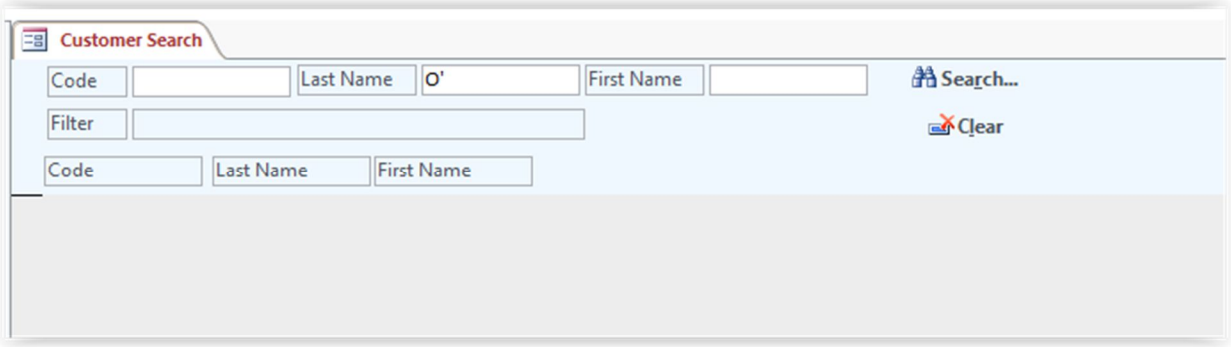

Fig. 1 Search results using O'

My search strategy was based on dynamically building a filter statement using LIKE and a wild card character *. Please note that these are the basic code outlines and do not show error handling and all comments associated with modules and procedures.

```
Private Sub cmdSearch_Click()
    Dim varWHERE As Variant

    varWHERE = Null

    varWHERE = "[CustCode] LIKE '" & Me.txtContactCode & "*'"

    varWHERE = (varWHERE + " AND ") & "[LastName] LIKE '" & Me.txtLastName & "*'"

    varWHERE = (varWHERE + " AND ") & "[FirstName] LIKE '" & Me.txtFirstName & "*'"

    If IsNull(varWHERE) Then
        MsgBox "You must enter at least one search criteria.", vbInformation, gstrAppTitle
        Exit Sub
    End If

    DoCmd.ApplyFilter , varWHERE

End Sub
```

Fig. 2 VBA code for searching CustCode, LastName and FirstName fields

I found VBA syntax for quotes can be challenging and the recommendation for single quotes is to use the Replace() function and utilize two single quotes. Thinking that this was a candidate for code reuse, I decided to write a function instead of inviting future typing errors from using too many quotes in a statement.

```
Public Function fReplaceEmbedQuote(strText As String)
' replace an embedded single quote with two single quotes
    fReplaceEmbedQuote = Replace(strText, "'", "''")
End Function
```

Fig. 3 User function for text with embedded single quote

I updated the search code and found that I was still not able to locate the record for Martin O'Donnell.

```
Private Sub cmdSearch_Click()
    Dim varWHERE As Variant

    varWHERE = Null

    varWHERE = "[CustCode] LIKE '" & Me.txtContactCode & "*'"

    varWHERE = (varWHERE + " AND ") & "[LastName] LIKE '" & fReplaceEmbedQuote(Me.txtLastName) & "*'"

    varWHERE = (varWHERE + " AND ") & "[FirstName] LIKE '" & Me.txtFirstName & "*'"

    If IsNull(varWHERE) Then
        MsgBox "You must enter at least one search criteria.", vbInformation, gstrAppTitle
        Exit Sub
    End If

    DoCmd.ApplyFilter , varWHERE

End Sub
```

Fig. 4 Revised VBA code for searching CustCode, LastName and FirstName fields

Cindy Krist, SMB Partners LLC (www.smbpartners.com), volunteered to review my code and see if she could offer any insight. She shared a form that converts SQL to VBA syntax.

Fig. 5 SQL to VBA form (Courtesy of Cindy Krist, SMB Partners LLC)

After checking the filter statement created by the Search procedure, it was noted that the quote mark I was using in my search criteria was a single quote (ASCII 39 – ') while O'Donnell contains another type known as a right single quote (ASCII 146 – ’). This explained why no records were returned in my search results!

There are three possible characters that might be used from the ASCII character set 0 – 255. These are ASCII 39 Single Quote, ASCII 146 Right single quote and ASCII 180 Acute accent. Keyboards in Germany, the Netherlands and some other EU nations can have the ’ character located near the Backspace key.

Cindy Krist also provided a solution that uses double quotation marks to handle an embedded single quote. This would eliminate the need to use the Replace() function

varWHERE = (varWHERE + " AND ") & "[LastName] LIKE """ & Me.txtLastName & "*"""

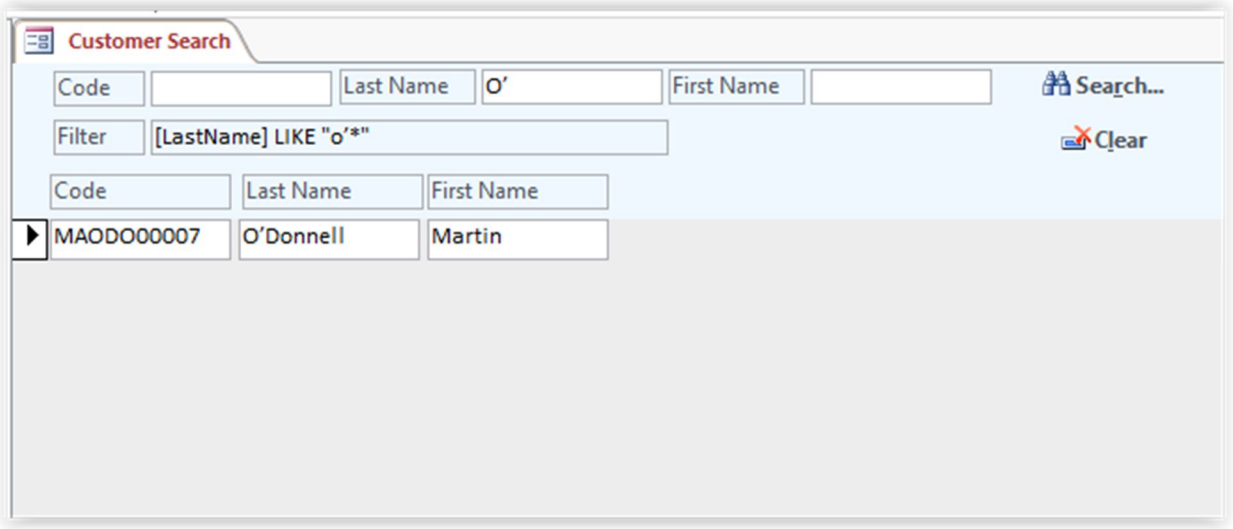

Fig. 6 Search result using right single quote

I added some records so that I could have the different types of quote marks for testing. My next task will be to develop a procedure that addresses how to return any type of quote mark or acute accent in the standard ASCII character set available in Access irrespective of the type entered in the search criteria. Any suggestions on accomplishing this will be welcomed.

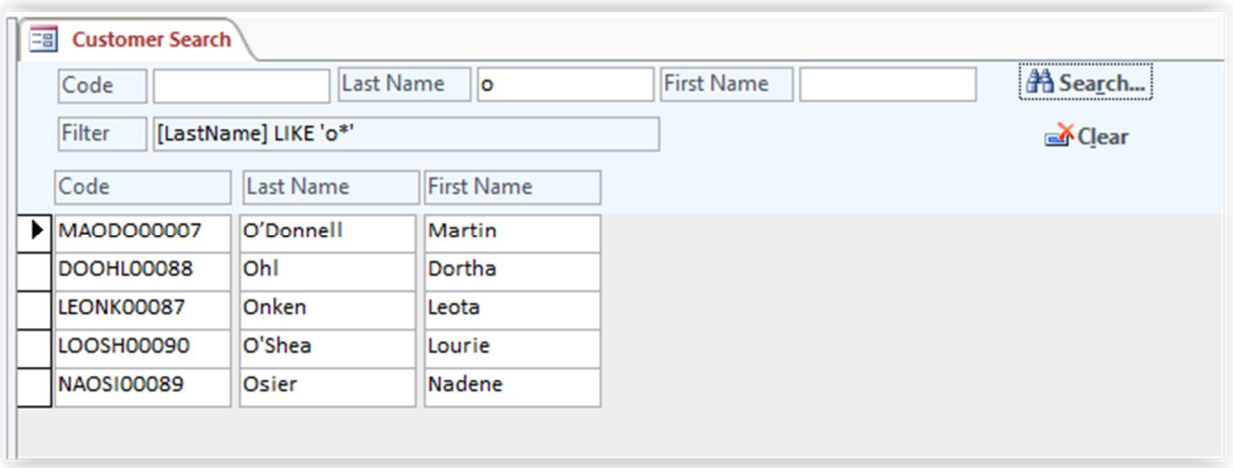

Fig. 7 Examples of different single quote marks

Learning about the different types of quotation marks resulted in my having to update the procedure used to generate a customer code. I use a combination of the first two letters in the FirstName, first three letters in the LastName and a formatted string of the CustomerID fields.

My approach tests for the presence of the different possible quote marks as well as the acute accent mark (Fig. 8). I then remove any of these characters and return three characters from the name with another function (Fig. 9). The revised function that returns the formatted code can be found in Figure 10.

As well as proving the adage that old dogs do learn new tricks, this confirms that we can all discover different solutions to solving a problem. Enjoy your coding experience!

```
Private quoteChr39 As String                ' US Keyboard single quote ' Chr(39)
Private quoteChr146  As String              ' Right single quote ’ Chr(146)
Private quoteChr180 As String               ' Accute accent mark ´ Chr(180)
Private arrQuoteMark As Variant
Public intQuoteLocation As Integer
Public intQuoteMarkAsc As Integer
Public Function fQuoteInText(strText As String) As Boolean
    Dim i As Integer

    quoteChr39 = Chr(39)
    quoteChr146 = Chr(146)
    quoteChr180 = Chr(180)
    arrQuoteMark = Array(quoteChr39, quoteChr146, quoteChr180)

    fQuoteInText = False

    For i = LBound(arrQuoteMark) To UBound(arrQuoteMark)
        If InStr(fReplaceEmbedQuote(strText), arrQuoteMark(i)) <> 0 Then
            fQuoteInText = True
            intQuoteLocation = InStr(fReplaceEmbedQuote(strText), arrQuoteMark(i))
            intQuoteMarkAsc = Asc(Mid(strText, intQuoteLocation, 1))
        End If
    Next i
End Function
```

Fig. 8 Test for quote and accent mark

```
Public Function fLastNameNoChr39(strText As String) As String
    Dim rhsC As String
    Dim lhsC As String

    If fQuoteInText(strText) Then
        Select Case intQuoteLocation
            Case 4
            lhsC = Mid(strText, 1, intQuoteLocation - 1)
            Case 3
            lhsC = Mid(strText, 1, intQuoteLocation - 1)
            rhsC = Mid(strText, intQuoteLocation + 1, 1)
            Case 2
            lhsC = Left(strText, intQuoteLocation - 1)
            rhsC = Mid(strText, intQuoteLocation + 1, 2)
            Case 1
            rhsC = Mid(strText, intQuoteLocation + 1, 3)
            Case Else
            lhsC = Left(strText, 3)
        End Select
    End If
    fLastNameNoChr39 = Trim(lhsC & rhsC)
End Function
```

Fig. 9 Function to remove single quotes from LastName field

```
Public Function fMakeContactCode(ContactID As Long) As String
    Dim strContactID As String

On Error GoTo fMakeContactCode_Err

    If Not IsNull(ContactID) Then
        strContactID = CStr(ContactID)
        strContactID = String(5 - Len(strContactID), "0") & strContactID

        If Not fQuoteInText(fContactLastName(ContactID)) Then
            fMakeContactCode = UCase(Left(fContactFirstName(ContactID), 2) & Left(fContactLastName(ContactID), 3)) & strContactID
        Else
            fMakeContactCode = UCase(Left(fContactFirstName(ContactID), 2) & fLastNameNoChr39(fContactLastName(ContactID))) & strContactID
        End If
    Else
        Exit Function
    End If

fMakeContactCode_Exit:
    'Shut Down Statements here
Exit Function

fMakeContactCode_Err:

    MsgBox "Error " & Err.Number & " (" & Err.Description & ") in procedure fMakeContactCode of Module modContact"
Resume fMakeContactCode_Exit

End Function
```

Fig. 10 Function to generate customer code