List Boxes, Access Bugs, and More

Chris Weber

In this jumbo-sized edition of Access Answers, Chris Weber has something for everyone. You'll find out about a subtle Access bug, some more obvious bugs, and how to avoid using edit masks (and why you'd want to).

I have a list box on a form that displays all of my reports for my users so that they don't need to have direct access to the database container. My list box uses a value list generated from the documents list in the reports container. However, as the list of reports has grown, my list box has stopped working. Access tells me that the string for the value list's row source is too long. Is there any way around this limitation?

I generally avoid using the forms and reports containers to generate these lists for this very reason. The listbox and combobox controls in Access have a row source size limitation of 2,048 characters. Instead of looping through the reports documents collection, you can query the MSysObjects system table in Access to get your list using the value –32764 for the Type. To build a list of forms use the value –32768:

```
'for reports
SELECT Name FROM MSysObjects WHERE Type = -32764

'for forms
SELECT Name FROM MSysObjects WHERE Type = -32768
```

The Type values for each kind of object can be gleaned from MSysObjects by comparing the object names in your database to the numbers. In the sample database for this month's column (available in the Download file at www.smartaccessnewsletter.com), cboSelectFromMSysObjects on the form frmColumnSizeToFit uses Type –32768 along with a Like clause to fill the list with the database's subforms. Incidentally, if you're filling the list for an add-in, you can use the IN clause along with the name of the calling database to make JET look in your current database:

```
"SELECT Name FROM MSysObjects IN '" & _
  CurrentDb.Name & "' WHERE Type = -32768"
```

Unfortunately, you can't get at other properties of the document object through the MSysObjects table. For example, if you want to display only nonhidden reports, or reports that have descriptions, you'll still have to resort to using the Reports collection.

We've recently had a couple of clients whose forms opened more slowly day after day even though tests on our network showed no signs of degradation using the same back end. A dataset that took five seconds to load across our network began to take upwards of four minutes at the client site. We spent hours tracing down possible network problems and repeating tests. Reinstalling the front end stopped the problem temporarily, but after a few days it began again. What are we doing wrong?

You aren't doing anything wrong. I suspect you've taken advantage of the Name AutoCorrect option under the Tools | Options | General tab or, more accurately, it has taken advantage of you. Name AutoCorrect allows Access 2000 to auto-magically cascade field name changes to your queries, forms, and reports. This is a great feature if you're evolving a data structure. However, checking these options can cause extremely poor performance in your Access 2000 database over time.

We ran into the same problem until we noticed that the prior release to the client didn't have AutoCorrect features enabled. After disabling the options, the performance problem was immediately resolved. Microsoft only acknowledges the problem if security is enabled and if the Access 2000 Service Pack 1 is NOT installed. This is not true. In our case, the client had the service pack installed and wasn't using Access security. Just turning AutoCorrect off solved the problem.

I've noticed that I can get a better compact for my database by copying all of my objects to new database. However, I can't seem to find a way to copy my toolbars. Any ideas?

You know, I used to do the same thing for simpler databases. Unfortunately, this doesn't handle your toolbars as you noted, as well as other niceties such as startup properties. In Access 97, it's still easy to copy toolbars by linking to the MSysToolbars table in the previous database and copying the records to the new database (The MSysToolbars is visible after selecting Tools

| Options | System Objects). I know of no way of easily copying toolbars in versions 2000 and 2002. However, there's a much easier way to get the same level of compaction.

What's happening when you move the objects to the new mdb file, is that you avoid copying old, unused code that the VBA editor fails to discard. This dead code causes the undue bloat in your mdb file. The way to get rid of this code is to occasionally open your database in a session of Access that's been started with the /decompile switch. Using the Windows Start | Run selection, type in the path to the MSACCESS.EXE file for the version you're using, in quotes, followed by the switch:

```
"C:\Program Files\MSOffice97\Office\MSACCESS.EXE"
           /decompile
```

Then open your database. The status bar will tell you that Access is "Converting database to currently installed version of VBA." This will be followed by a message box confirming the decompile (see Figure 1).

Now, close Access, restart it normally without the /decompile switch, and reopen your database. Open any module, and select Compile and (in Access 97) select Save All Modules, or (in Access 2000 and later) select Compile projectname. Now compact as usual. You'll see a significant improvement in the size reduction, especially in larger databases.

I've created a subform without a recordsource that dynamically gets its recordsource from VBA code in the parent form (based on user choices): subWhatever.Form .Recordsouce="qryBlahBlahBlah." The form worked just fine for a while, months in fact, but now I keep getting a runtime error stating that the .Form property of the subform doesn't exist. The error didn't occur until after a compile some time after I'd created totally unrelated code in another form's class module.

I've seen this behavior in both Access 97 and 2000. This bug is really freaky, as the debug window readily recognizes the subform's property, but the VBA engine can't resolve it while running the code. The first two times this happened to me, I was able to get around it by starting Access with the /decompile switch, decompiling my application, and then continuing work. Somehow, the recompilation ordered the p-code differently, and all was well—for a little while, anyway. The third time, the engine settled on a different line, and decompiling had no effect whatsoever. I was ready to tear my hair out. I found a workaround (two frustrating hours later) by supplying the subform with a recordsouce in design view that returns no records:

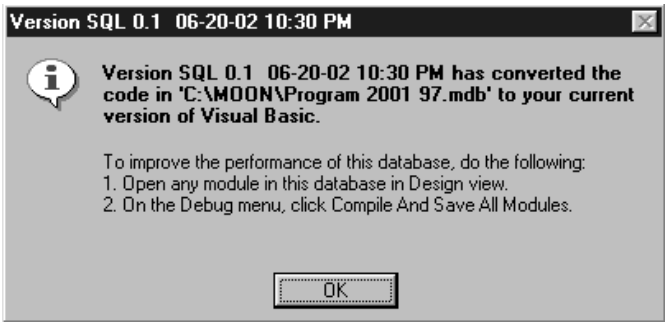

Figure 1. This message indicates that your unused code is being discarded.

```
SELECT id FROM table WHERE true=false
```

Suddenly, the VBA engine liked my subform again and would change the recordsource for the .Form without complaint.

Why a subform should have to have a recordsource for the VBA compiler to recognize it as a form, I don't know. There's no consistency to the problem. I'm not sure if Microsoft is aware of this problem as it's intermittent at best, but there you have it.

Is there an option that can be set to cause Access not to show user-defined lookup lists in JOIN fields in table datasheets? There are times when I'd really like to see the underlying child key values to verify the results of my recordset code.

The ability to set up field lookups in Access's table designer is a great timesaver downstream. Having these fields default to combobox controls while designing forms can save hours of work in a large project. Unfortunately, later on when you're developing reports, the fields continue to show up as lookups, which really doesn't make sense on reports where the user can't edit the data. And, when it comes to testing data bugs, those lookups in table and query datasheet views can hide the key values you really need to see.

I've created a function to get rid of the lookups after the bulk of my design work is complete. It uses HasProperty(), a generic routine for testing whether an object has a specific property in its Properties collection.

```
Sub DropThoseCombos()
On Error Resume Next
Dim tdf As TableDef, fld As Field

For Each tdf In CurrentDb.TableDefs
  For Each fld In tdf.Fields
    If fld.Properties("displaycontrol") = 111 Then
      If HasProperty(fld, "RowSource") Then
        Debug.Print fld.SourceTable & " : " & fld.Name
        fld.Properties("displaycontrol") = 109
        Debug.Print fld.Properties("displaycontrol")
      End If
    End If
  Next fld
Next tdf

End Sub
```

```
Function HasProperty(pobj As Object, _
                     pstrName As String)
On Error Resume Next
Dim strProperty As String

strProperty = pobj.Properties(pstrName).Name
HasProperty = (Err = 0)

End Function
```

You can use the routine to just print fields that are defaulted to combo boxes before you change them back to text boxes—just comment out the line that sets the DisplayControl property to 109. Both of the functions listed previously are included in this month's source code: Just run DropThoseCombos()from the debug window. Now your key values can be used to reveal those bugs.

I have a form that contains a subform used for ad hoc querying. The main form allows the user to select from hundreds of tables in our database and see the results in predefined views created as forms shown in the subform. The subform is loaded on the fly depending on the result of the choice that the user has made. I want to use a datasheet subform because my users need to freeze columns, hide columns, as so forth. When the subform opens, many of the descriptive column names of the fields are cut off in the datasheet. I want to be sure the column width is never less than the descriptive field name (at least to begin with).

Wow! You've certainly covered the difficult parts of this. You've created views and corresponding form controls for your users to view the data. Very neat! In fact, the information you needed to do this is the same information you need to set the datasheet's column widths.

At first I was going to recommend that you determine the width of each column from the field's name in the SELECT clause. If you used a fixed-width font, you could then resize each column relative to the font's character width. But that's a lot of work, and there's a far easier way.

A datasheet's ColumnWidth property supports a reserved value of -2 that makes the column "size to fit." With that information, you can use the subform's underlying query, or a temporary query based on the SQL statement in the subform's recordsource, to retrieve the SELECT fields returned and make each column "size to fit."

Here's the code to the put in the AfterUpdate event of the combo box used to load the subform, but you could run the same code in the SourceObject's Load event:

```
Private Sub cboSelectFromMSysObjects_AfterUpdate()
Const SIZE_TO_FIT = -2
Dim fld As Field, qry As QueryDef

Me!subDataSheetSizeToFit.SourceObject = _
                        cboSelectFromMSysObjects
Set qry = CurrentDb.CreateQueryDef("", _
      subDataSheetSizeToFit.Form.RecordSource)
For Each fld In qry.Fields
  Me!subDataSheetSizeToFit.Form!(fld.Name). _
                  ColumnWidth = SIZE_TO_FIT
Next fld

End Sub
```

The results can be seen in **Figure 2**.

The sample form, frmColumnsSizeToFit is included in this month's Answers column database. You'll note that the "size to fit" only allows memo fields to expand to the subform container's width.

There's a drawback to this method—you can't include OLE Object fields in your forms. This isn't really a problem; your users couldn't view the OLE data in a datasheet anyway. But, if you've used the form wizards to create your subforms, they always include all fields from the underlying tables, regardless of what fields you've chosen to show. As a general rule, always redefine the recordsource of a form created with a wizard to retrieve only those fields you've selected to show or need for linking or underlying code.

I've inherited a database that needs many enhancements and error-handling procedures. My users are compiling a set of changes for me, but often their requirements are cryptic because they describe what's on the screen or use business terms for the data controls that need modifications. Being new to the project, I can't directly translate their descriptions into Access object names. Is there any place that Access maintains a list of the forms, reports, and control names and captions so that I can make the translation?

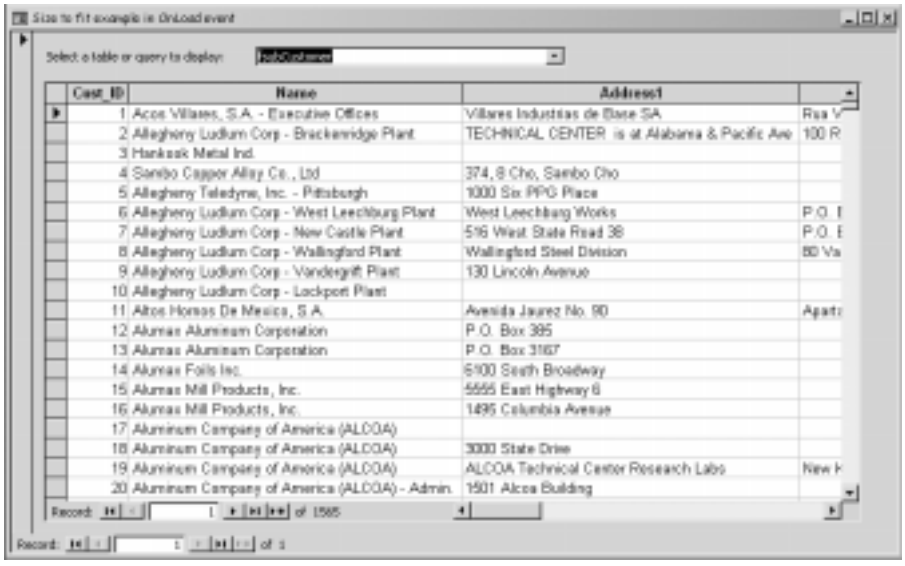

Figure 2. The results of using "size to fit" on column headings.

All of us who've entered a project late or inherited someone else's work have encountered this same problem. I used to get on the phone and ask my users to walk me through the interface to find out what form or report they were referring to, put the object into design view, and then determine the names for my change control document. Nonetheless, there are some situations when this won't work—for example, if a form opens modally and you can't get at its design view through the toolbar or menu.

I have two simple techniques I use to sidestep this problem. The first is to always have a hot key in my AutoKeys macro that uses the RunCommand action with the Design View argument. This way, even if a form is opened as a dialog, I can still punch in my hot key combination and make it switch to design view.

My second technique also uses a hot key combination. This time, however, it runs a function that returns the name of the active form, active report, and active control on the screen. Here's the routine that I attach to the hot key:

```
Function GetActiveObjectNames()
On Error Resume Next
Dim strForm As String, strReport As String, _
                       strControl As String

strForm = Screen.ActiveForm.Name
strReport = Screen.ActiveReport.Name
strControl = Screen.ActiveControl.Name
MsgBox strForm & vbCrLf & strReport & vbCrLf & strControl, _
      vbInformation, "Current object name(s):"

End Function
```

The use of On Error Resume Next is important here. There will never be both an active form and active report at the same time, so the assignment for one or the other will leave its variable as a zero-length String. What's more, if you use the Screen reference directly in the message box statement, the entire statement will fail. Using String variables, the failed assignment just inserts a blank line in the message box. And there's never an active control on a report, so that error will also be plowed under.

Now, when my users attempt to describe a form, report, or control to me, I have them press my hot key combination (Ctrl-Shift-0), which calls GetActiveObjectNames(). The routine then displays the name of the object and the control that has the focus. Believe it or not, users immediately stop questioning the object naming conventions as soon as they see how much faster I can react to their requests—we all save valuable time. GetActiveObjectNames() is in this month's source code along with the AutoKeys macro that calls it.

I have a form that looks just like our previous paper system for entering work orders. It's accessible from a subform that lists all the work orders for a selected client. When the user clicks the new button on the work orders subform, my form opens in add mode as a dialog. Upon return, the list is requeried to show any records added. This works fine. Now, my users want to print out the new work orders as they finish entering them. I thought this would be simple, but every time they click the print button to put the form into preview mode, it's blank. If they close the form, the new entry is in the list, so clearly the new work order wasn't lost and prints out just fine if reopened. What am I doing wrong?

It appears that this is just a "strange behavior" that Access exhibits when putting data entry mode dialogs into print preview. Because this is a generic problem, I'm not going to recreate your form here. There is, however a solution to the problem.

Instead of just issuing the Runcommand method of the DoCmd object with the acCmdPrintPreview constant, you need to first take your form out of data entry mode by setting the form's DataEntry property to false. However, this causes the form's recordset to move off your new record and preview the first default record in the set. Furthermore, if you haven't saved all the required fields, this will generate a validation error.

So, first save the record by setting the form's Dirty property to false. Any data errors will generate an error message through Jet or your error handler. Next, capture the primary key field's value. Now, set the DataEntry property to false and use the Filter property of the form to move back to your record. Now, when you call DoCmd.RunCommand acCmdPrintPreview, your new record shows up, thanks to this code:

```
Private Sub cmdPreview_Click()

Dim lngWrkOrdr As Long

  If Me.DataEntry Then
    Me.Dirty = False
    lngWrkOrdr = Me.WrkOrdr
    Me.DataEntry = False
    Me.Filter = "WrkOrdr=" & lngWrkOrdr
    Me.FilterOn = True
  End If
  DoCmd.RunCommand acCmdPrintPreview
  DoCmd.RunCommand acCmdPreviewFourPages

End Sub
```

There's one drawback to this solution, however, and I believe you're probably already experiencing it. When you print-preview the dialog form, the form falls out of dialog mode. If you move the form aside and look at your subform, you'll notice that the requery that's supposed to occur after the dialog closes has already fired. If you aren't expecting the user to add more records or edit the new work order after printing, all is well. Otherwise, the subform list may be out of sync when the user closes the work order form. I don't know any way of determining when the preview is over to let you either close the form or put it back into data entry mode.

My opinion is that Access needs OnPreview, OnPrintClose events for forms in order for their print and print preview modes to be truly useful, just like the existing Filter and Apply Filter events. Alternately, you could capture the primary key value as I did here, set the dialog's Visible property to false to hide it, show the new work order in a report designed to look just like the form, and then pop up the dialog again after the report closes. No matter what route you follow, as soon as the dialog becomes invisible, your calling form will requery.

Why does Access's database container keep changing the created and modified dates of my forms and reports every time I compact? I compact often because I'm making extensive changes to the database during daily development. But the date and time of the newly modified forms and reports reads the same as all my previous work—the date and time of the compact. This makes my latest work impossible to find by modified or created date. Is there a property I can use to change this behavior?

This is certainly annoying behavior. I go through the same frustration myself working with Access 2000. If you're working in Access 97, this isn't the case (and you get a Help file that functions properly!). Worse, I don't think you can change the behavior of the Access 2000 Forms and Reports collections. However, there's a kludge/workaround that you can use. Put the forms and reports that you're actively working on into a "group." The default "Favorites" does the job just fine. Now, when you compact your database, the dates and times in the forms and reports tabs will have the date of the latest compact. But the shortcuts in your group will have the actual dates and times they were last modified prior to being put into the group.

However, the group shortcuts will *not* pick up the modified date for subsequent edits. You'll have to delete the shortcuts from the group and drag the objects back in from the forms and reports containers prior to your next compact. Hopefully, you're not working on too many things at once.

I've created numerous queries prior to a major redesign of our database. Many of my queries would still work with just minor changes to the SQL statement. For example, statements where I used the old "Order Details" table could easily be changed to the new "tblOrderDetails" using a text editor. If I could open the query to its SQL statement directly, I could copy the statement to Word and use Find-and-Replace to quickly reflect the changes in the new schema. The problem is, when I open the old queries, the QBE interface opens by default and doesn't recognize the "Order Details" as a table. All the fields I've projected from the table now have "ExprN" tagged on as aliases. It really makes editing the SQL difficult. Is there any way to open a query in SQL view directly?

I don't know of any way to open directly to the SQL editing window. However, there's an easy but not so obvious way to get to your query's SQL statement, but it's not by opening the editor. Instead, use the Immediate window (press Ctrl-G) to get the SQL property of your query in the database's querydefs container:

```
?CurrentDb.QueryDefs("qryYourQuery").SQL
```

Now, copy the SQL statement from the Immediate window into Word and make your Find-and-Replace edits. Finally, open the query in design view, switch to SQL view, and paste in the changes. Switching back to the QBE will quickly let you know if your edits were successful.

I've inherited a database that's denormalized and uses repeating groups of text fields to hold memo information. I want to take the contents of the fields and append it all into a common memo field in the table and then delete the repeating groups. During the append operation, I'm inserting Chr(13) between the fields so that I have a line break between the entries. But now, when I open my table in datasheet view, there are little box symbols at the end of each line but no line break. How can I concatenate these fields together with line breaks but not see the character marks?

This is a fairly common problem that requires an uncommon solution. Using your current method with a calculated field in the query produces the results shown:

```
SELECT [Notes1] & Chr(13) & [Notes2] & Chr(13) &
         [Notes3] AS Notes FROM Employees;
```

The result is certainly not what you're looking for. Notice that each block of memo is displaying a square box where the Chr(13) was inserted (see **Figure 3**). The trick here isn't to use the Chr() function directly within your query but, rather, to pass your fields to a VBA function that returns the data formatted with carriage returns. My routine ConcatFieldsWithLineBreaks () will do the trick:

```
Function ConcatFieldsWithLineBreaks( _
   intCrLfs As Integer, _
   ParamArray Fields() As Variant) As Variant
Dim i As Integer
Dim varMemo  As Variant
Dim strCrLfs As String

  varMemo = Null
  For i = 1 To intCrLfs
    strCrLfs = strCrLfs & vbCrLf
  Next i
  For i = 0 To UBound(Fields())
    varMemo = (varMemo + strCrLfs) & Fields(i)
  Next i
```

```
  ConcatFieldsWithLineBreaks = varMemo

End Function
```

There are several interesting things to point out in this code. First of all, the parameters that you pass in include the number of carriage return/line feeds you want between each field (intCrLfs), and an unspecified number of fields that are received in a parameter array, Fields(). VBA automatically loads the array with as many values as you pass in, which makes the function flexible for all situations. Next, note that the "memo" that you're creating is a variant, varMemo, which gets initialized as a NULL. This variant is used so that the function can accept NULL field values in the parameter array argument. A String variable would never accommodate NULLs in your fields.

As the code loops through each value in the array, varMemo is updated with its previous value plus (+) a vbCrLf and then concatenated to the current value in the array. The first time the assignment is made, the default NULL value cascades through the (+) concatenation to vbCrLf resulting in a NULL value, which is then concatenated (&) to the field value. If the first field value is NULL, varMemo remains NULL, and the next pass through the loop drops the vbCrLf as the NULL cascades again. If the field value isn't NULL, varMemo equals it, and the next pass through the loop appends a vbCrLf before concatenating the next field in the array.

Using the function is simple. Specify the number of vbCrLf's you want between field values, and pass the fields you want to concatenate together as a comma-separated list. The query is simpler to build than manually inserting Chr(13) function calls, and the result is exactly what you're looking for (as shown in **Figure 4** and **Figure 5**, on page 22).

You can see a sample query using qryConcatFieldsWithLineBreaks() in the database accompanying this month's Answers column.

I have a small dialog that my users can open to view pictures associated with my real estate entries. I'd like my users to be able click and drag the image to a larger size, as they feel necessary. I can't get the form that contains the image to expand the image frame contained within it as it's resized. Any ideas?

As a matter of fact, yes. I'll bet you're trying to set the width and height of the image frame to the form's width and the detail section's height during the form's Resize event. This drove me nuts the first time I tried it until I finally discovered the InsideWidth and InsideHeight properties of the form. These hold the values you need for your image frame dimensions. Use this code in the form's resize event, and all should be well:

```
Private Sub Form_Resize()
  fraImage.Height = Me.InsideHeight
  fraImage.Width = Me.InsideWidth
End Sub
```

By the way, don't forget to set the image frame's Size Mode property to either Stretch or Zoom so that the image can expand as needed. Also, if your images are complex photographs, you may want to have the form use the image's color palette for better display. A snippet of code in the form's Current event takes care of this nicely, with On Error Resume Next to accommodate records with NULL values:

```
Private Sub Form_Current()
On Error Resume Next
  Me.PaintPalette = fraImage.ObjectPalette
End Sub
```

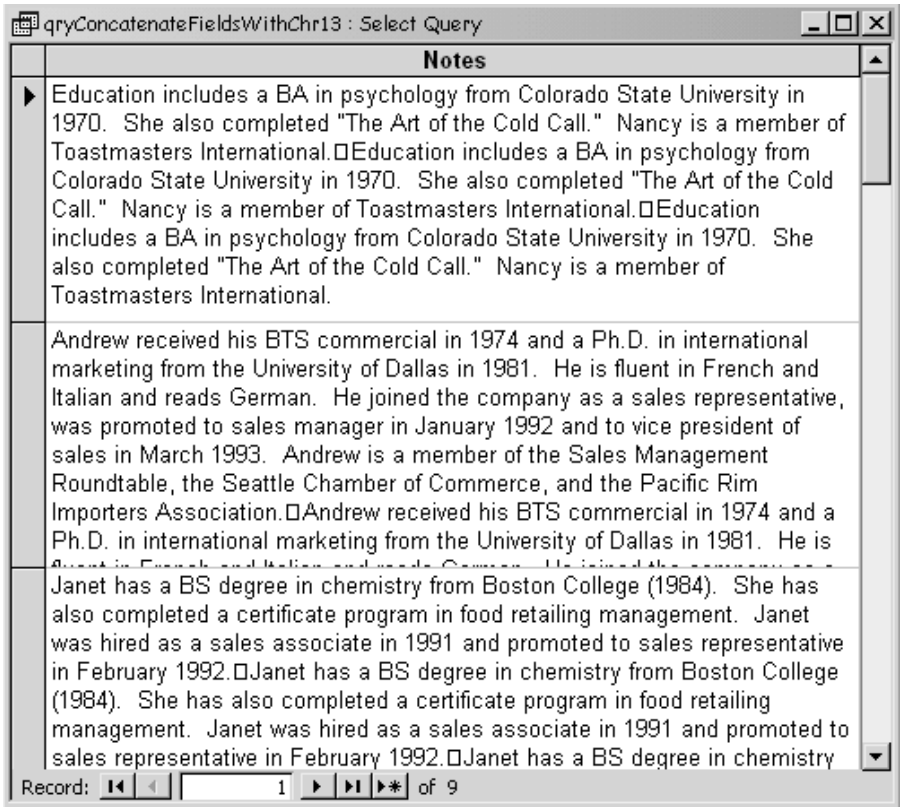

Figure 3. Text with unprintable characters instead of intended line feeds.

You can see these techniques at work in frmEmployeePhotoSizeToFit in this month's download (see **Figure 6**, on page 23).

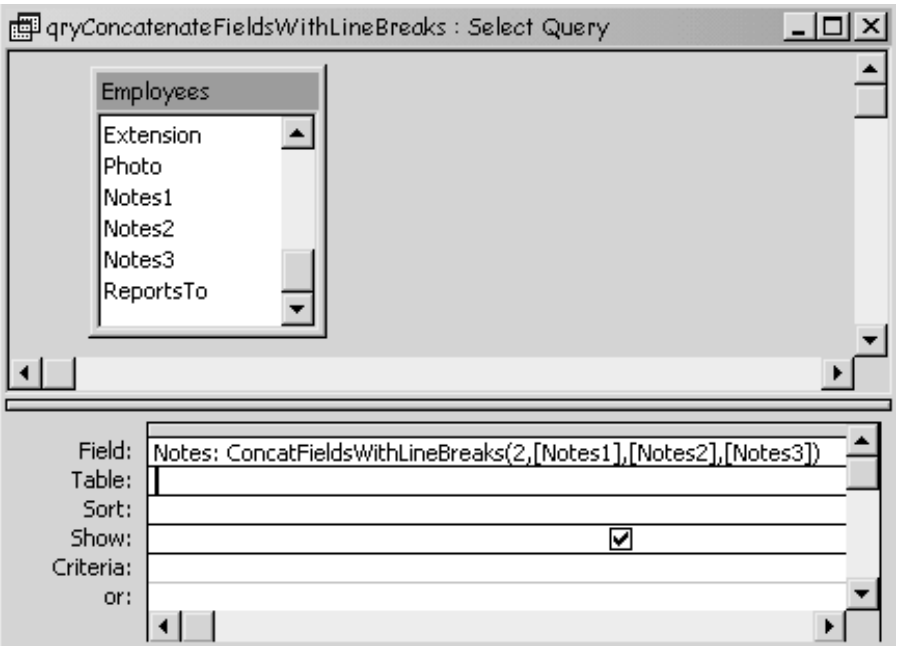

Figure 4. A query using the ConcatFieldsWithLineBreaks function.

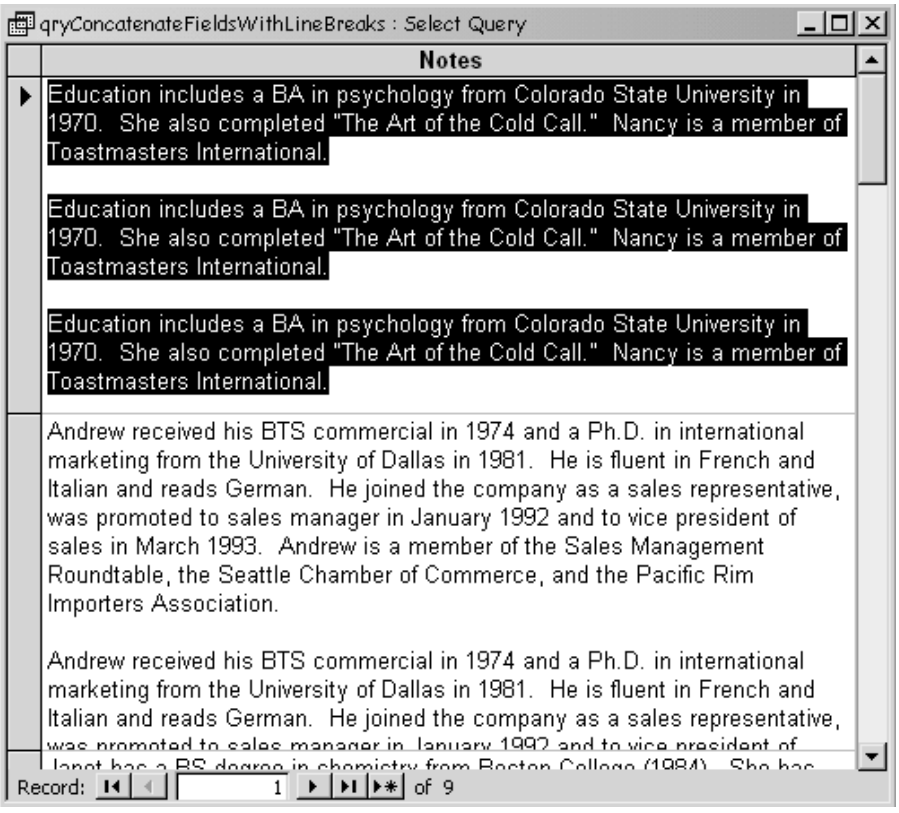

Figure 5. The correct line-broken output.

I've been using input masks on my date fields throughout my database to "help" my users easily enter date information. But then they began complaining that they couldn't paste dates into masked controls. Now, they've learned about Access's Ctrl-; shortcut for piping in the current date and are complaining to me that the mask doesn't allow them to use it. Is there a way for the mask to allow the pasting of data and use of the built-in keyboard shortcut?

Every time the topic of input masks comes up in my Access training seminars, I feel like Dennis Miller about to go off on a rant. First of all, as you pointed out, they don't accommodate shortcut keys and pasting of data. What's more, they also interfere with the format property, can cause inconsistencies in your data if they're used in one spot but not another (for example, phone fields), and even for the simple case of dates, they don't provide an input mask text property that you can display to your users when they violate the mask's pattern. I've seen very few implementations where they haven't eventually backfired resulting in some rework of the application. All this work just to save users from typing two slashes...

The last time this topic came to a head in one of my classes, we sidetracked and created key-trapping code that could be used for date fields in lieu of the input mask. We decided that users should only be able to type numbers and slashes and that a format property setting of "mm/dd/yyyy" would suffice to validate the input.

Well, if you've ever entertained this exercise, it quickly balloons far beyond your original expectations. You immediately realize that numbers and slashes alone just won't do. What about common editing strokes like Delete and Backspace, navigation strokes like Tab, Shift-Tab, and Enter, positioning strokes including F2 and the Left and Right arrow keys? And we shouldn't forget Ctrl-; and Ctrl-X,

Ctrl-C and Ctrl-V. Finally, when you begin testing, you realize you also have to accommodate both central and numeric keypads.

We saw it through, however, and created a KeyDown event that covered the bases. We then extracted the code and made it generic enough to be called from the KeyDown event for any date field. On the way, we found that there weren't VBA KeyCode constants for the forward slash (/) next to the right shift key or for the semicolon key, so we created our own by Debug.Printing the KeyCode value as we pressed them. Otherwise, the VBA constants came to the rescue and made our code very readable:

```
Sub DateMaskKeyDown(KeyCode As Integer, _
      Shift As Integer)
Const SEMICOLON = 186, FORWARD_SLASH = 191

  If Shift And acCtrlMask Then
    Select Case KeyCode
      Case vbKeyC, vbKeyV, vbKeyX, SEMICOLON
        'allow copy, paste, cut and today's date
      Case Else
        KeyCode = 0
    End Select
  ElseIf Shift And acShiftMask Then
    Select Case KeyCode
      Case vbKeyTab
        'allow shift tab
  Else
    Select Case KeyCode
      Case vbKey0 To vbKey9, vbKeyNumpad0 To _
           vbKeyNumpad9, vbKeyDivide, FORWARD_SLASH, _
           vbKeyF2, vbKeyBack, vbKeyDelete, _
           vbKeyTab, vbKeyReturn, vbKeyRight, _
           vbKeyLeft, vbKeyHome, vbKeyEnd
        'do nothing
      Case Else
        KeyCode = 0
    End Select
  End If

End Sub
```

DateMaskKeyDown () is called from the KeyDown event of a date field passing in the KeyCode and the Shift parameters Access provides.

```
Private Sub txtOrdDate_KeyDown(KeyCode As Integer, _
                               Shift As Integer)
  Call DateMaskKeyDown(KeyCode, Shift)
End Sub
```

Take this routine, or leave it; use it, or forget about it. I think it works better than the mask, but that's just my opinion. Whatever you do, just say NO to those input masks. I don't want to go off on a rant… ▲

DOWNLOAD **SA0902AA.ZIP at www.smartaccessnewsletter.com**

Christopher Weber travels throughout the country teaching Access Development and Programming seminars for The DSW Group in Atlanta, GA. He's been an Access developer since its first release, enjoys working with clients, and heads the DSW Group's Access development and training team. cweber@thedswgroup.com, www.access-training.com.

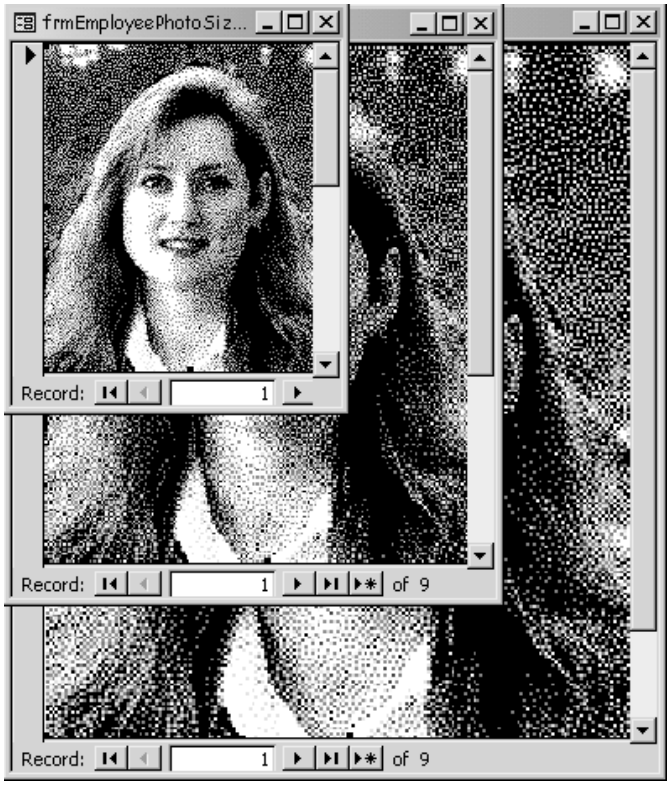

Figure 6. A form with graphics resized to fit.