Cool Tricks for the Sophisticated Developer

Christopher Weber

95 97 2000 2002 DOWNLOAD

This month, Chris Weber addresses questions from real-life, Access training seminar attendees about report totals, splash screens, trapping keystrokes, managing control widths, when there's more in a memo than meets the eye, triple state check boxes, tables for one, and using the new Dirty event. The answers aren't always obvious, especially to the new generation of Access users.

I have a report that lists order totals by employee with a running total alongside. However, I only want to show the running total field alongside the very last order total. I know I can just put the total in a footer below the list, but the requirement is to show it alongside the last entry to re-create the same look we had in a previous system. I've tried hiding the control by setting its visible property, but I can't seem to figure out when the last entry will format so that I can unhide it. Is there a property that will let me do this?

As I understand your question, you currently have the data shown in Figure 1, but you'd like it to print as shown in Figure 2. The fundamental problem here is either finding out when the last order total will format, or possibly knowing ahead of time what the total will be so that when the running total control equals that value it can be made visible. In either case, a domain function comes to the rescue.

One way of finding which item is the last one is to use a domain function. I'm particularly fond of domain functions as they're fast and easy to use, require a single line of code, and don't require saving a separate query object in the database container. Regardless of which domain function you use, they all have the same general syntax:

```
functionname(fieldname, table or query name, _
  [optional wherecondition])
```

The wherecondition is the same as a where clause in a query without the word "WHERE," just as you might use in an OpenForm method.

For my first solution to knowing when the last order total will print, you'll need to know the print sequence of the employee's order totals. For example, if you have the totals listed by date, then you can use the DMax function in an invisible control in a header

Figure 1. How the data looks.

Emp SS	Last Name	First Name	Order Date	Sale Total	
111-11-1111	Davolio	Nancy			
			17-Jul-1996	$1,614.88	**$1,614.88**
			01-Aug-1996	$1,376.00	**$2,990.88**
			04-May-1998	$446.85	**$190,367.38**
			05-May-1998	$484.50	**$190,851.88**
			06-May-1998	$1,255.72	**$192,107.60**

Figure 2. How the data *should* look.

Emp SS	Last Name	First Name	Order Date	Sale Total	
111-11-1111	Davolio	Nancy			
			17-Jul-1996	$1,614.88	
			01-Aug-1996	$1,376.00	
			04-May-1998	$446.85	
			05-May-1998	$484.50	
			06-May-1998	$1,255.72	**$192,107.60**

for the employee. The control will contain the last totals date, and you can use it to determine when you're displaying the last total. Assuming the employee Social Security number is being used as the identifier for the employee and is a field in the report's RecordSource, the control source for your hidden control (which I've called txtMaxOrdDate) would be:

```
=DMax("OrderDate", "Orders", "EmpSS='" & _
     [EmpSS] & "'")
```

If you're limiting the display to a date range, then the date criteria will also have to be coupled into the DMax()'s wherecondition with an And operator (the query will probably run much faster if the OrderDate field is indexed, especially if the OrderDate and EmpSS are part of the same index). I should warn you that this solution will only work if each order date for an employee is unique. In most businesses, this is probably not the case. But, as we'll see, there's another approach.

When the total for the last date is being printed, you can set the visibility of your running total field to true and have it display. This is accomplished by a single line of code in the Detail section's OnPrint event:

```
Private Sub Detail_Print( _
        Cancel As Integer, PrintCount As Integer)
  txtRunningTotal.Visible = _
        (txtOrdDate = txtMaxOrdDate)
End Sub
```

If you don't have a unique piece of information that will tell you the printing sequence, you can use my second solution, which depends on determining ahead of time what the total will be. For this solution, I use the DSum() function to determine the total of all the orders for the employee. I then display the running sum control when its value is equal to the total calculated by my DSum function. As in my first solution, I place a hidden control (called txtSumTotal) in the employee header and add up the employee's order totals:

```
=DSum("SaleTotal", "qrptEmployeeSalesTotalNoFooter", _
  "EmpSS='" & [EmpSS] & "'")
```

Note in this example that instead of summing the values in the Orders table directly, I'm summing the Total field from the totals query that's the basis for the report. This solution is computationally more intensive, but the result is the same. Again, the visibility of the running total control in the detail section is set in the Detail section's OnPrint event:

```
Private Sub Detail_Print(Cancel As Integer, _
  PrintCount As Integer)
txtRunningTotal.Visible = _
  (txtRunningTotal = txtSumTotal)
End Sub
```

There's a warning associated with this solution also. You'll have to make sure that you've formatted both the control for the domain function in the employee header and the running sum control in the Detail to use the same number of decimal places. Since Access can do a text comparison between the two controls, if the controls aren't formatted identically, then Access may not recognize that the domain function's summation and the running total are essentially equal.

The first solution using the maximum OrderDate can be found in rptEmployeeSalesTotalNoFooter_A in the Source Code file for this month's "Access Answers" column (available at www.smartaccessnewsletter.com). The summation solution has been implemented in rptEmployeeSalesTotalNoFooter_B.

How can I suppress the appearance of the Access splash screen prior to the opening of my database's splash screen? I don't mean not to give Microsoft credit for its platform, but can the splash be hidden?

Well, this is an oldie but goodie. I haven't had anyone ask me this since Access 2.0, when machines were typically slower and that splash screen seemed to hang there forever. But I tested the old solution, and it still works. Just create a 1x1 pixel, gray bitmap image in Windows Paintbrush and save it with the same name as your database front end, in the same directory as your front end. For example, for the Northwind.mdb, your file would be named Northwind.bmp and stored in the same directory. The next time you load your app, if you've got a magnifying glass you might catch the gray pixel in the center of your screen, but your users won't catch it.

Be aware, however, that your application won't load any faster. The Access splash screen isn't only an advertisement but also feedback to us that Access is indeed loading resources from its various libraries. If you're already using a form within Access for a splash screen, just take a screenshot of it (Alt-Print Screen), paste it into Paintbrush, crop as necessary, and save it as a BMP with the appropriate name. Then drop it into your application's directory and have Access display it before your first form. It's always good user interface design to give your users feedback that something good is happening.

I have a combo box that has just two values, Yes and No, and is bound to a Yes/No field in my table. It works just fine, but my users insist that the box should let them Auto Expand whether they're in insert or overwrite mode. So, if they click in the box that says "No" and type a "y," the box should begin the overwrite and Auto Expand to say "Yes." The problem is that the result ends up being "yesNo," "Nyo,"

"Nyeso," and the like, and Access complains that the item isn't in the list. How can I fix this?

I always tell my students, "Before you write any code, let the product do what it already knows how to do." In other words, use the properties and methods of the objects you're working with to solve typical problems. Chances are there's already a property or method designed to accomplish your goal. In this case, however, that philosophy will steer you down the wrong path.

The Auto Expand property of the combo box was designed to save users time by automatically matching items in the list with whatever they type at the start of the control's text. So you might be tempted to set the SelStart property of the control to 0 whenever the user entered the control. But they could still be changing the current value and type "Nyes," "Yno," and so on.

If your combo box were always in overwrite mode, their requirement might seem easy to fulfill. But there's no way to be assured what state the combo box has when your user enters the control. Users may have clicked your combo box with the mouse to place the cursor at a specific letter or have set a different "Behavior entering field" option under the Keyboard section of the Tools | Options menu. So now you'd be tempted to use the SelLength property to force the user into overwrite mode:

```
Private Sub cboYesNo_Enter()
  cboYesNo.SelLength = 3
End Sub
```

Unfortunately, that code doesn't work when you click into the control with the mouse, unless you set a break point and step through it. It's as if the Access event model is battling between the code and the mouse click events. And, even if you did force the control into overwrite mode, the user might change his or her mind mid-stride and switch from "Yes" to "No" and vice versa. In this instance, it's time to go past the Auto Expand property and begin trapping keystrokes.

Because users will always begin typing their choice with either "Y" or "N," I can trap for just these two letters and suppress any others. Each time the user presses "Y," the following code sets the value of the combo box to True (-1) and, conversely, for "N" to False (0). I let some keys fall through, but most keystrokes set the KeyCode parameter to 0, effectively discarding the entry:

```
Private Sub cboYesNo_KeyDown(KeyCode As Integer, _
  Shift As Integer)
  Select Case KeyCode
    Case vbKeyY
      cboYesNo = -1
    Case vbKeyN
      cboYesNo = 0
    Case vbKeyTab, vbKeyReturn, vbKeyUp, _
              vbKeyDown, vbKeyF4, vbKeyEscape
      'default behavior
    Case Else
      KeyCode = 0
  End Select
End Sub
```

The keystrokes whose KeyCodes I don't suppress include the Tab, Return, Up and Down arrows (all for navigation), F4 (to drop the list open), and Escape keys (to undo changes).

The Access Form Wizards always make the controls on my forms either too wide or too narrow for the data. Is there any way to control this? I always have to adjust the widths manually, and, even then, I rarely get it right.

My first reaction is that you probably *don't* want the Wizard to do this. To know exactly how wide to make a control for a field, the Form Wizard would have to read through all of your data (potentially millions of records) and find the widest value, and then calculate the control's width using that many characters in the font for the style that you've chosen. There are several reasons why you wouldn't like this. First of all, the Wizard would have to get the lengths for all of the controls on the form. That's potentially a separate query to be run for each field in your underlying record source. Second, there's no information stored with a font to tell Access what the average character width of a letter is. Third, users of the Wizard often change the font style anyway, so all that work would be moot.

There is a way, however, for you to adjust the control width correctly, provided you have a sufficient sample of data to draw a sample from—if, for example, you wanted to know the optimum width for the Company Name from the Customers table. First, you create a query that returns the longest name in the database:

```
SELECT TOP 1 Customers.CompanyName,
  Len([CompanyName]) AS Length
FROM Customers
ORDER BY Len([CompanyName]) DESC;
```

For the Northwind database, this gives me "FISSA Fabrica Inter. Salchichas S.A." at 36 characters. The next step is to open your form in design view and set the Company Name text box's Control Source to ="FISSA Fabrica Inter. Salchichas S.A." Then set the control's font to the one you'll be using. In Figure 3 (on page 19), I've selected Verdana 12 pt., bold.

Now, you can run the form and see whether you need to make adjustments. I always leave space for a couple more characters just in case my data spreads even wider over time. The result is shown in Figure 4.

You've now set your control length correctly. You'll have to re-open your form in design view and set the text box's control source back to the Company Name field in your underlying query. Repeat the process for any other controls that are giving you trouble, and you're on your way. With screen space at a premium, this method could save you from a lot of post-delivery complaints.

Our data has numerous memo fields, but I have screen space restrictions (800x600) that don't allow me to set my text boxes wide enough to always show all the data. Even though I've set the memo's text box's Scroll Bars property to Vertical, my users can't tell whether there's more text in the control unless they tab into it and it gets the focus. Is there a way to force the scroll bars to always be visible if there's more text than meets the eye?

Unfortunately, the Scroll Bars property doesn't have any sub-properties that allow you to adjust its behavior. The scroll bars only appear when the memo control has the focus. When the last line of text in a memo is the end of a sentence or when users have formatted their memos by using the Ctrl-Enter keys to insert line breaks in the memo, it's impossible to tell whether all the data in the control is being displayed. Users shouldn't have to read the memo contents to determine whether there might be more to follow.

Lots of developers will add instructions to the memo control's label such as "Shift-F2 to zoom," which helps, but it still doesn't attack the root of the problem: How is the user to know when there's more text in the memo that's out of sight?

My solution is to display a "More…" hot link when the text in the memo exceeds a specified length (more than should be visible in the control), as shown in Figure 5. If the text fits, the "More…" option doesn't show (see Figure 6, on page 20).

If the user clicks on the "More…" link, Access opens the Zoom Box displaying more text than the form can accommodate (see Figure 7, on page 20).

To implement this solution, you need to have a signal that toggles the "More…" hot link on and off for each record. You could write code to test the length of

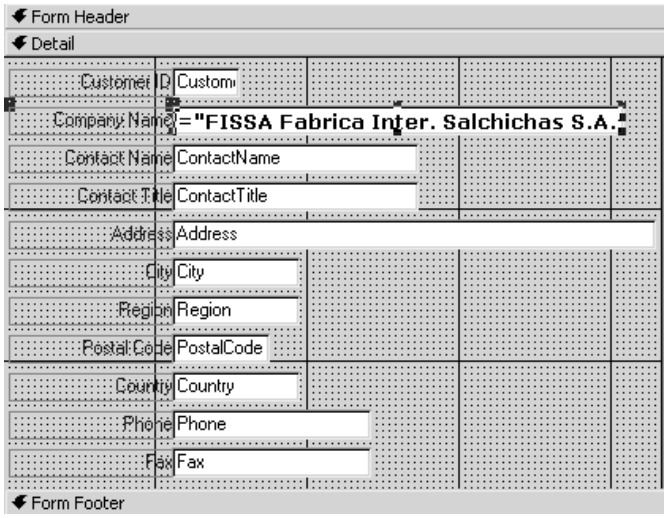

Figure 3. Setting the control width with sample data.

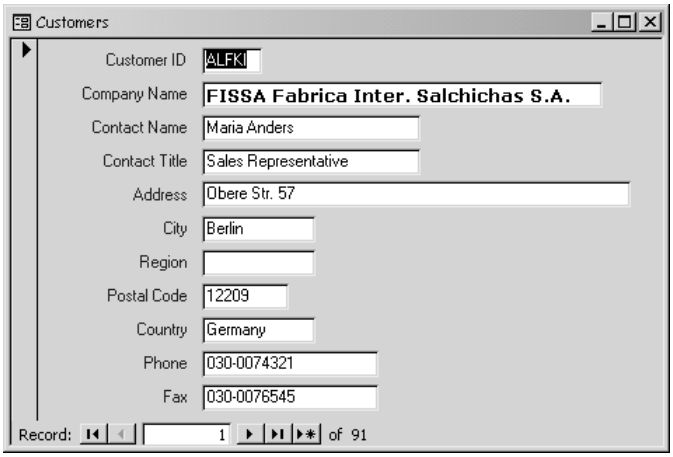

Figure 4. The final result.

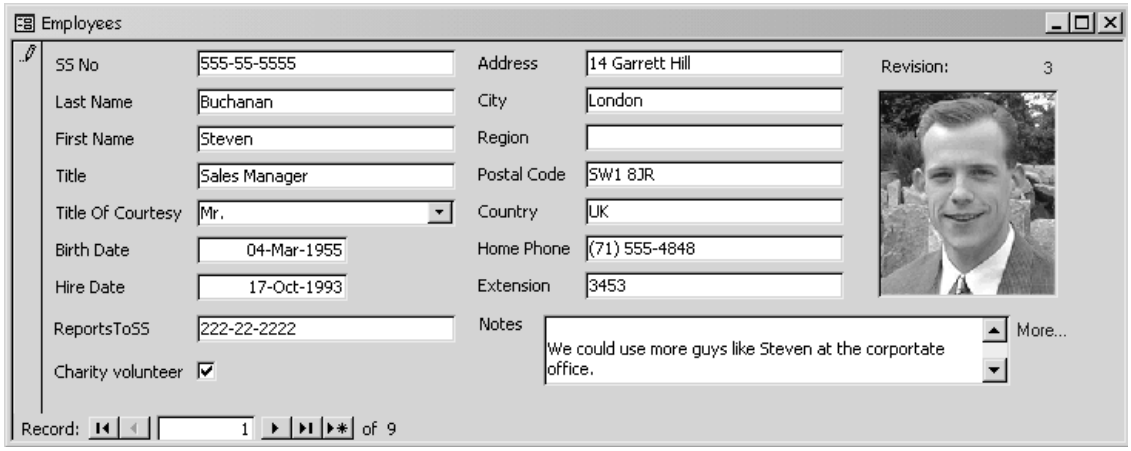

Figure 5. The "More…" option in view.

the memo for each record as you move into it, but then the user might edit the memo, and you'd have to write code to accommodate the edits as well. Instead, I've opted to use Access's ability to recalculate in the underlying query. The record source for the form has an extra field called MoreNotes:

```
SELECT Employees.*,
  IIf(Len([Notes])>105 Or InStr([Notes],Chr(13)),
      "More...",Null) AS MoreNotes
FROM Employees;
```

The MoreNotes field returns "More…" if the length of the Notes field is greater than 105 characters or if the field has a line feed character (the user has pressed Ctrl-Enter to insert carriage returns). Otherwise, the field returns Null. I determined the magic number 105 by filling my memo control with lowercase w's and counting how many it took to fill the control. This is just an approximation, but it seems to work, as the w's accommodate the widest lowercase character in the font that I'm using. Sometimes the "More…" appears even though the control's contents are all visible, but it's a small tradeoff for the convenience to my users.

Now, on the form I've created a TextBox called txtMoreNotes to the right of my memo field. Its control source is the MoreNotes field returned from my query (see Figure 8, on page 21).

Notice that the control is disabled and locked. I don't want it getting the focus in either selection or insert mode, even though the user couldn't edit it (calculated fields can't be changed). What you don't see in the design view picture is a transparent button, cmdMoreNotes, that sits directly on top of txtMoreNotes. The button's OnClick event calls a generic function MoreClick(), passing my trigger control and the actual memo control. The OnClick property of the button looks like this:

```
=More_Click([txtMoreNotes], [txtNotes])
```

The More_Click function receives the name of the control whose control source is the MoreNotes field from the underlying query, and the name of the control that contains the memo that should be zoomed in on:

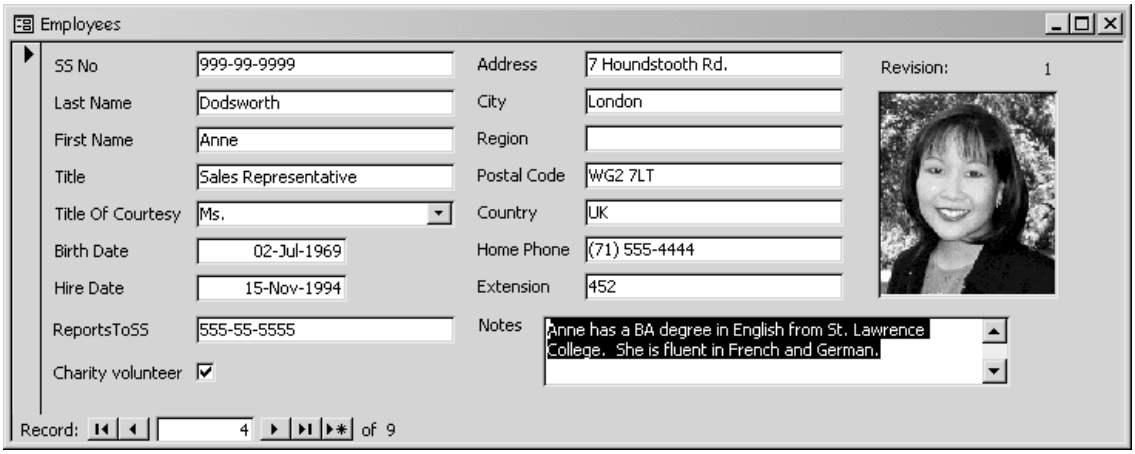

Figure 6. "The More…" option invisible.

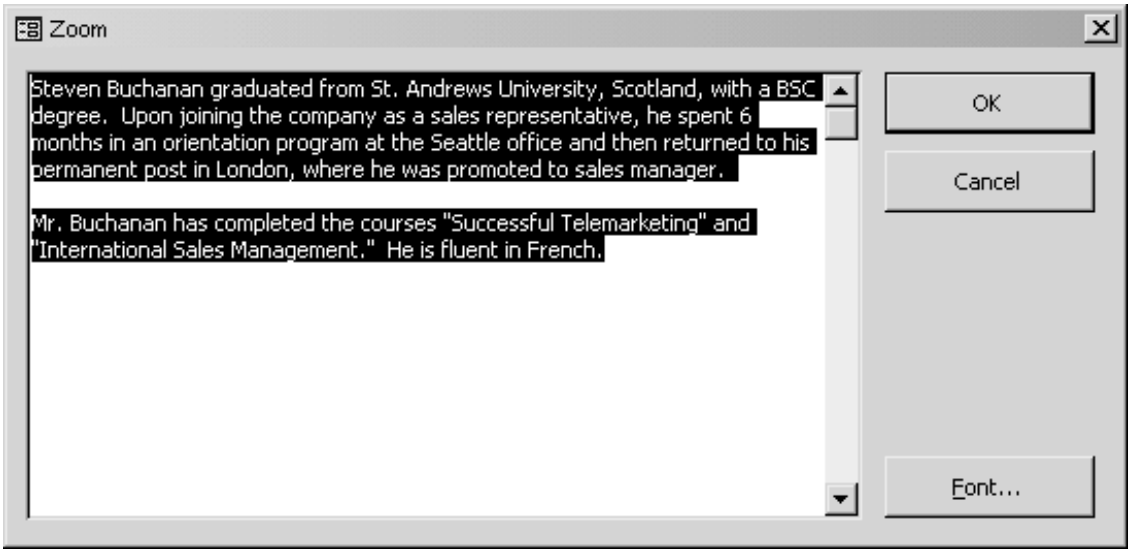

Figure 7. The Zoom Box with the full content of the Memo field.

```
Function More_Click(ctlMore As Control, _
  ctlToShow As Control)
'generic Zoom box call for More... controls
  If Not IsNull(ctlMore) Then
    ctlToShow.SetFocus
    SendKeys "+{F2}"
  End If

End Function
```

If the MoreNotes control isn't Null (that is, it's displaying "More..."), then the memo control is given the focus, and the SendKeys action opens the Zoom Box. The sample database includes frmEmployeesWithMemoMore, which demonstrates this solution.

We have a table that tracks information whose records are constantly updated. We need to track how often the record is updated by assigning a sequential revision number to the record whenever it's edited. The revision number needs to show on the form when the users begin the revision, so that they can write it down on the corresponding paperwork. I've been able to retrieve and assign the next number in the form's AfterUpdate event, but the event is often fired when the user navigates to another record and it's too late to jot it down. I've tried controlling the record movement, but there always seems to be a loophole they get through. Any ideas?

First of all, be sure that your Revision field has its default value property set to 1. This way, Revision will always have a value when the user creates a new record. Because the requirement is that the Revision needs to be available when the users begin their edits, I think that AfterUpdate is too late, not to mention the loopholes in navigation that you've run into. Taking a step back, I think you'll need to trap the moment the user begins editing.

If you're using Access 97, this is a complicated problem. You can try to trap keystrokes in the form's KeyDown event and determine during each stroke if the form is Dirty. If it is, you can toggle a Revision flag and set the revision number. frmRevisionIn97 in the sample database uses this technique. The form's Current event resets the Revision flag for subsequent edits to other records:

```
Option Compare Database
Option Explicit

Dim fRevisionSet As Boolean

Private Sub Form_Current()
  fRevisionSet = False
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, _
  Shift As Integer)

  If Not Me.NewRecord Then
    If Me.Dirty Then
      If Not fRevisionSet Then
        Me.Revision = Me.Revision + 1
        fRevisionSet = True
      End If
    End If
  End If

End Sub
```

Unfortunately, this isn't foolproof, as the user might begin an edit, undo the edit, and begin editing again. In this scenario, the Revision gets incremented, undone, and subsequently left with the old value should the user begin editing again. The Current event only resets the Revision flag after a navigation to another record. There's no Undo event to reset the flag.

In Access 2000, the answer is simple and seems to be foolproof. This solution is implemented in frmEmployeesWithMemoMore in the sample database. The new Dirty event comes to the rescue. Whenever the user dirties the record, the Revision gets incremented. If the user undoes the edits, the increment is lost:

```
Private Sub Form_Dirty(Cancel As Integer)
  If Not Me.NewRecord Then
    Me.Revision = Me.Revision + 1
  End If
End Sub
```

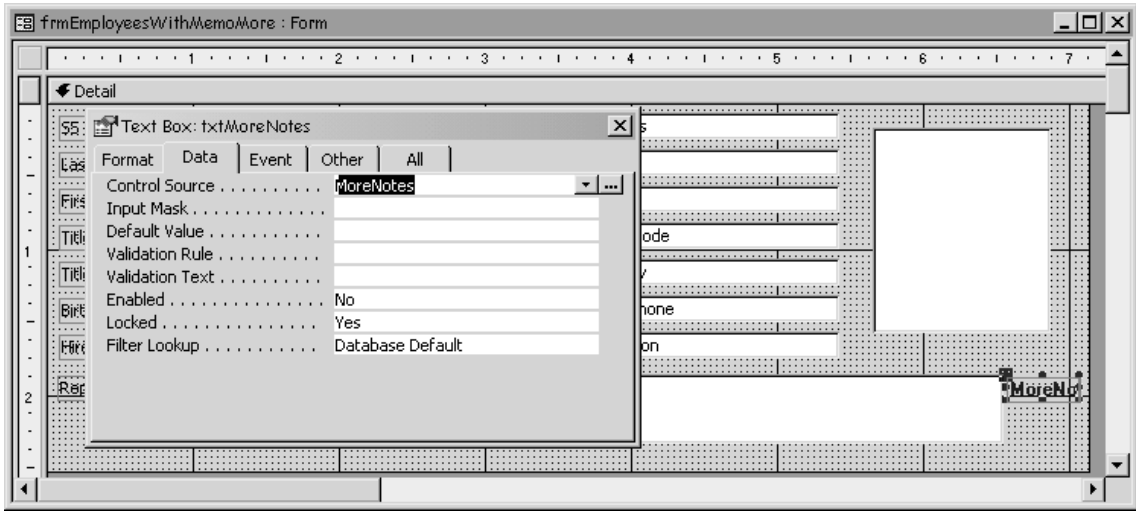

Figure 8. Setting up the trigger field.

Again, I test for the new record since it already has its default Revision value of 1. Otherwise, I increment. The only scenario where I can see this solution failing is if the user makes edits from the revision sheet, saves the record, and then makes more edits. At that point the revision number will be incremented twice. But then, what do you consider a revision? Perhaps they've actually made two revisions at this point and should record both numbers.

Lastly, watch out for the Zoom Box. Opening it for any field and closing it by clicking the OK button constitutes an edit even if no change has been made. The only workaround that I can think of is to override the default Zoom Box with one of your own.

I have a survey form with numerous Yes/No questions tied to Yes/No fields in the underlying table. I've placed check boxes on the form and set their Triple State property to Yes. The value should be Yes (true) when checked, No (false) when unchecked, but neither (gray) if the person hasn't answered the question. Every time I look at the results, the answers are either Yes or No even though some people didn't answer the question, or the question didn't apply. Why can't I track folks who don't respond?

The problem lies in your choice of data type in the underlying table. Yes/No fields have just two possible values, True or False. They default to False unless you specifically set the default value property of the field to True. However, in your survey, you're actually tracking three possible values—Yes, No, and non-responsive (unknown). Thus, you need a data type that can be left Null, which means "unknown."

Change your field types to Integer and leave their default value properties blank. Just like any other number, Access will assume the value is Null unless you fill it in. Your triple state check boxes will assign the value -1 when checked, 0 when unchecked, and Null when gray. The integer field will accept the possible entries, and you can count selections in your results just as if it were a Yes/No field. All the zeros will register as false, any record with a non-zero number in it will register as true (-1), and Nulls represent those records where the question had no response. frmEmployeesWithMemoMore in the accompanying database uses this technique for its Charity Volunteer field.

I need a table that can contain only one record of system information that gets retrieved at various points in my database's code. Is there a way to force Access to allow only one record?

This is another case where built-in properties won't solve the problem. An understanding of table design principles, however, will. Just provide a primary key field that allows only one value. I use a Byte field with a validation rule of "=1," as shown in Figure 9.

Give your one record an ID of 1, and you're all set. No other records will be able to be saved. The example pictured can be found in the accompanying database.

In closing…

I'd like to close this month's column with a tip given to me by one of my Access training seminar attendees: If you want to copy a table or query into Word (or Excel), you can do it without opening the object in Access. Just highlight the object in the database container, copy, switch to Word, and paste!

How long have I been working with Access and didn't know this? There's always something new to learn. ▲

DOWNLOAD SA0202AA.ZIP at www.smartaccessnewsletter.com

Christopher Weber travels throughout the U.S. teaching Access development and programming seminars for The DSW Group in Atlanta. 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. www.access-training.com

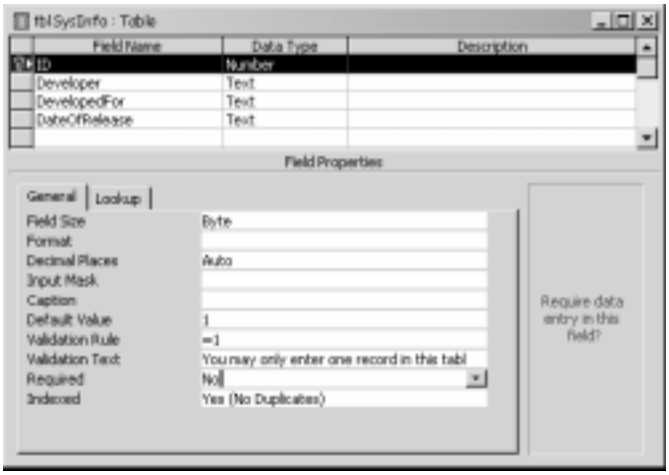

Figure 9. Defining a table to contain a single record.

Know a clever shortcut? Have an idea for an article for *Smart Access*?
See the back page for the contact information where you can send your ideas.