

ACCESS HACKS™

Tips & Tools for Wrangling Your Data



O'REILLY®

Ken Blattman

HACK
#61

Use Excel Functions Inside Access

Expose powerful functions available in Excel to your Access application.

Excel has many powerful built-in functions for such things as financial and statistical analysis. If you want to do the same type of analysis in Access, you can do one of the following three things: purchase an off-the-shelf code solution, write your own code for analysis, or use automation to tap into Excel's functions from inside Access. This hack shows you how to tap into Excel via automation and use spreadsheet functions, saving you time and money over the other options.

This hack involves Access working hand in hand with Excel, so you need to make sure Excel is installed on the machine on which your database will be running. This is a safe assumption in most corporate environments.

A Simple Excel Function

Excel's FV (future value) function calculates the value of an investment at some time in the future based on periodic, constant payments and on a constant interest rate. The following VBA function takes the same parameters as Excel's FV worksheet function and returns the same result as if you were using the future value function right in Excel:

```
Public Function FV(dblRate As Double, intNper As Integer, _
    dblPmt As Double, dblPv As Double, _
    intType As Integer) As Double
    Dim xl As Object
    Set xl = CreateObject("Excel.Application")
    FV = xl.WorksheetFunction.FV(dblRate, intNper, dblPmt, dblPv, intType)
    Set xl = Nothing
End Function
```

The `WorksheetFunction` property of Excel's `Application` object is key to calling Excel functions from code, whether in Access or even directly in Excel's VBA environment. With this property, nearly every Excel worksheet function is available to build into a solution.

Figure 7-11 shows a form that takes input from a user and calls the FV function from the Calculate Future Value button.

Clicking the Calculate Future Value button executes the following code:

```
Private Sub cmdFV_Click()
    Dim dblFV As Double
    dblFV = FV(txtRate / 12, txtNper, txtPmt, dblPv, frmType)
    MsgBox "FV = " & dblFV, vbInformation, "Future Value"
End Sub
```

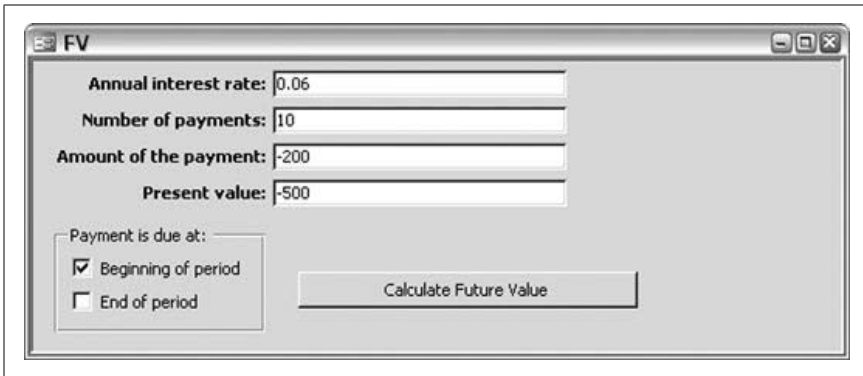


Figure 7-11. Calling the FV function from a form

The `cmdFV_Click` event calls the FV function and displays the message box shown in Figure 7-12. You can modify the code to write the solution back to a table or to display it elsewhere on the form object as needed.

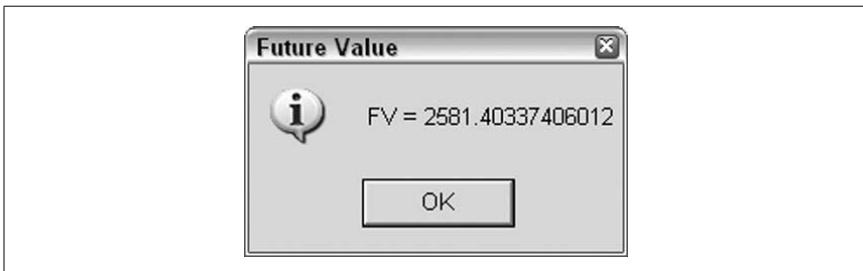


Figure 7-12. Message box displayed from the `cmdFV_Click` event

An Excel Function with an Array Parameter

The example of calculating a future value required five parameters to be passed into Excel, and with the magic of automation, we got the result back. However, what would happen if one of those parameters were an array, as many are in Excel?

If an Excel function requires an array or table array, you can pass it an array or a multidimensional array created in Access and get back the needed result. Let's look at the code you'd use to call Excel's percentile worksheet function, which returns the *k*th percentile of values that you specify from a given array of values:

```
Public Function Percentile(strTbl As String, strFld As String, k As Double)
    As Double
    Dim rst As ADODB.Recordset
    Dim dblData() As Double
```

Use Excel Functions Inside Access

```
Dim xl As Object
Dim x As Integer
Set xl = CreateObject("Excel.Application")
Set rst = New ADODB.Recordset
rst.Open "Select * from " & strTbl, CurrentProject.Connection,
adOpenStatic
ReDim dblData(rst.RecordCount - 1)
For x = 0 To (rst.RecordCount - 1)
    dblData(x) = rst(strFld)
    rst.MoveNext
Next x
Percentile = xl.WorksheetFunction.Percentile(dblData, k)
rst.Close
Set rst = Nothing
Set xl = Nothing
End Function
```

With this function, we pass the table name and field name to be read into the Access array, which in return is passed into Excel's percentile function along with the kth percentile value that we are looking for in the array of values. It's worth noting that you can pass the function a query name instead of a table, depending on the application's requirements.

Figure 7-13 shows a form that displays a subform that is bound to the tblData table and displaying the SampleData field in datasheet mode.

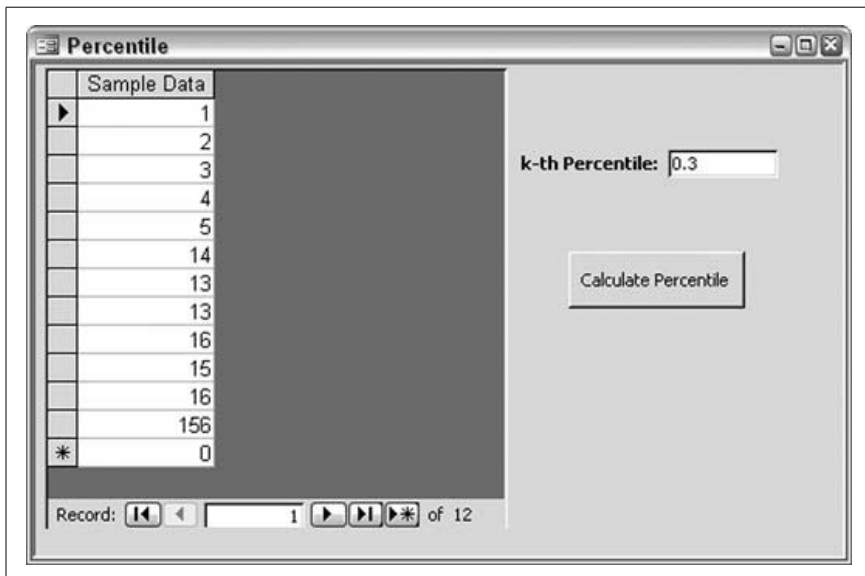


Figure 7-13. Calling the percentile function from a form

This sample calculates the 30th percentile from the list 1, 2, 3, 4, 5, 14, 13, 13, 16, 15, 16, 156 when the user clicks the Calculate Percentile button. Clicking the Calculate Percentile button executes the following code:

```
Private Sub cmdPercentile_Click()  
    Dim dblPercentile As Double  
    dblPercentile = Percentile("tblData", "SampleData", txtK)  
    MsgBox "Percentile = " & dblPercentile, vbInformation, "Percentile"  
End Sub
```

This code produces the message box in Figure 7-14.

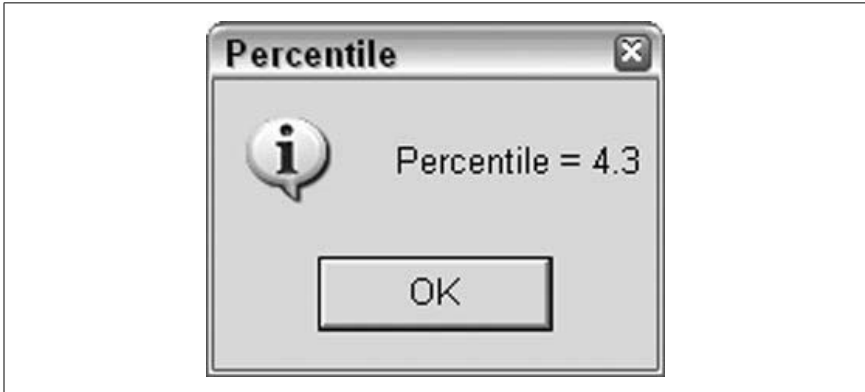


Figure 7-14. The message box displayed from the `cmdPercentile_Click` event

As noted previously with the `FV` function, you can write this return value back to a table or display it on the form. You can also call the `FV` function or `Percentile` function from a query or use it on a report.

Other Excel Spreadsheet Functions

You can call more than 100 functions using the `WorksheetFunction` method of the Excel object via automation. Keep in mind that some are redundant with built-in Access functions, such as Excel's `ISNUMBER` and Access's `ISNUMERIC`, and others, such as `ISERR` and `ISNA`, aren't of much use unless you are doing some other advanced spreadsheet automation.

You also have to consider whether the overhead of automation is acceptable in your application. It might not be as efficient as a well-written custom function. However, it can be a huge timesaver if you don't have time to write your own custom functions such as the `Percentile` function.

—Steve Huff