Flexible Normalization and Denormalization: Case 2

Helen Feddema

95 97 2000 2002 2003 DOWNLOAD

Helen Feddema approaches the same problem as Garry, but this time manages her data to provide the users with the output that met their needs.

A reader asked me how he could convert a table with more than 100 questionnaire fields to a more manageable format, with the fields converted to records in a table to make it easier to tabulate the data. Effectively, this is the reverse of Garry's problem where he converted multiple records into one: I'm converting a single record into multiple records. To make matters more interesting, my reader wanted to be able to save each survey's results in a separate table, which forces me to re-create the table and query with each processing run.

In the Helen.mdb file that comes with this article, you'll find the tblSurvey table (part of which is shown in Figure 1) that has the raw data from the questionnaires. It has 44 fields (cut down from the original table, which had more than 100 fields). There's a Text field, ID, which is the key field, and the other fields are either Boolean or Text, with the Text fields taking a numeric value from 1 to 5.

To switch the fields to records, I first created a table (with the prefix zstbl to indicate that it's a system table) with just three fields: SurveyID, a Long Integer field indexed Yes (Duplicates OK), Question and Answer (both text fields). This table is copied to create a results table that's filled from code.

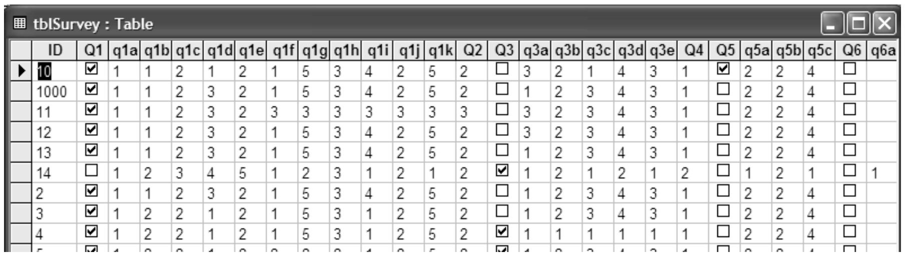
The CreateResultsTable function fills a results table with records containing field names and values from the original tblSurvey and creates a totals query based on it (qtotAnswers) that totals the number of Yes, No, and 1 through 5 answers for each question. For convenience, the function can be run from the mcrCreateResultsTable macro, or (for consistency with Garry's database) the frmCreateResultsTable form. This query is the record source for a simple report, which is shown in Figure 2.

I begin by defining the variables that I'll be needing:

```
Public Function CreateResultsTable()

On Error Resume Next

   Dim fld As DAO.Field
   Dim flds As DAO.Fields
   Dim rstSource As DAO.Recordset
   Dim rstTarget As DAO.Recordset
   Dim strPrompt As String
   Dim strResultsTable As String
   Dim strSourceTable As String
   Dim strTableTemplate As String
   Dim strTitle As String
   Dim strReport As String
   Dim strQuery As String
   Dim strSQL As String
   Dim lngCount As Long
   Dim strCurrentDate As String
   Dim intResult As Integer
   Dim rpt As Access.Report
```

The next step is to delete the existing table for this survey (if it exists) and re-create it, ready to accept the new data (the surveys were generated each day so I used the current date to name each version of the table):

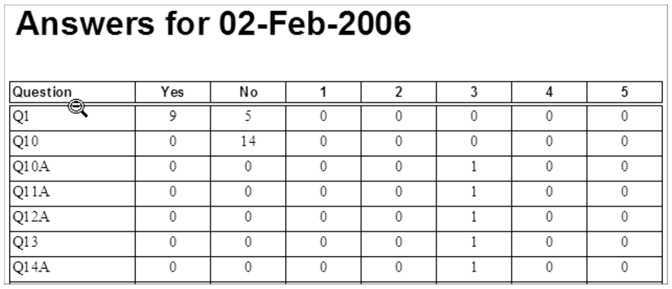
```
   strTableTemplate = "zstblSurveyResults"
   strCurrentDate = Format(Date, "dd-mmm-yyyy")
   strResultsTable = "tblSurveyResults_" & _
     strCurrentDate
   DoCmd.DeleteObject objecttype:=acTable, _
    objectname:=strResultsTable

On Error GoTo ErrorHandler

   'Make copy of table template
   DoCmd.CopyObject newname:=strResultsTable, _
```



tblSurvey : Table

ID	Q1	q1a	q1b	q1c	q1d	q1e	q1f	q1g	q1h	q1i	q1j	q1k	Q2	Q3	q3a	q3b	q3c	q3d	q3e	Q4	Q5	q5a	q5b	q5c	Q6	q6a
10	☑	1	1	2	1	2	1	5	3	4	2	5	2	☐	3	2	1	4	3	1	☑	2	2	4	☐	
1000	☑	1	1	2	3	2	1	5	3	4	2	5	2	☐	1	2	3	4	3	1	☐	2	2	4	☐	
11	☑	1	1	2	3	2	3	3	3	3	3	3	3	☐	3	2	3	4	3	1	☐	2	2	4	☐	
12	☑	1	1	2	3	2	1	5	3	4	2	5	2	☐	3	2	3	4	3	1	☐	2	2	4	☐	
13	☑	1	1	2	3	2	1	5	3	4	2	5	2	☐	1	2	3	4	3	1	☐	2	2	4	☐	
14	☐	1	2	3	4	5	1	2	3	1	2	1	2	☑	1	2	1	2	1	2	☐	1	2	1	☐	1
2	☑	1	1	2	3	2	1	5	3	4	2	5	2	☐	1	2	3	4	3	1	☐	2	2	4	☐	
3	☑	1	2	2	1	2	1	5	3	1	2	5	2	☐	1	2	3	4	3	1	☐	2	2	4	☐	
4	☑	1	2	2	1	2	1	5	3	1	2	5	2	☑	1	1	1	1	1	1	☐	2	2	4	☐	

Figure 1. The table with raw survey data in numerous fields.



Answers for 02-Feb-2006

Question	Yes	No	1	2	3	4	5
Q1	9	5	0	0	0	0	0
Q10	0	14	0	0	0	0	0
Q10A	0	0	0	0	1	0	0
Q11A	0	0	0	0	1	0	0
Q12A	0	0	0	0	1	0	0
Q13	0	0	0	0	1	0	0
Q14A	0	0	0	0	1	0	0

Figure 2. A report based on a totals query giving the number of each answer for each question.

```
   sourceobjecttype:=acTable, _
   sourceobjectname:=strTableTemplate
```

Now that the table is created, I can fill it with data. As in Garry's example, I read each record and then process each field creating a record as I go:

```
strSourceTable = "tblSurvey"
Set dbs = CurrentDb
Set rstSource = dbs.OpenRecordset(strSourceTable)
Set rstTarget = dbs.OpenRecordset(strResultsTable)
Do While Not rstSource.EOF
   Set flds = rstSource.Fields
   For Each fld In flds
      rstTarget.AddNew
      If fld.Name <> "ID" Then
         rstTarget![SurveyID] = rstSource![ID]
         rstTarget![Question] = fld.Name
         If fld.Type = dbBoolean Then
            rstTarget![Answer] = _
             IIf(fld.Value = True, "Yes", "No")
         Else
            rstTarget![Answer] = fld.Value
         End If
         rstTarget.Update
      End If
   Next fld
   rstSource.MoveNext
Loop
rstSource.Close
```

Reporting on the data

However, the table was only part of the solution. I also needed to create a query that would total the results and serve as the basis for a report. I first deleted any existing copy of the query and then generated a new one:

```
'Delete old totals query
strQuery = "qtotAnswers"
DoCmd.DeleteObject objecttype:=acQuery, _
  objectname:=strQuery

Set dbs = CurrentDb
strSQL = "SELECT [" & strResultsTable _
 & "].[Question], " _
 & "Sum(IIf([Answer]='Yes',1,0)) AS YesAnswer, " _
 & "Sum(IIf([Answer]='No',1,0)) AS NoAnswer, " _
 & "Sum(IIf([Answer]='1',1,0)) AS 1Answer, " _
 & "Sum(IIf([Answer]='2',1,0)) AS 2Answer, " _
 & "Sum(IIf([Answer]='3',1,0)) AS 3Answer, " _
 & "Sum(IIf([Answer]='4',1,0)) AS 4Answer, " _
 & "Sum(IIf([Answer]='5',1,0)) AS 5Answer " _
 & "FROM  [" & strResultsTable & _
"] GROUP BY [" & strResultsTable & "].[Question];"
Debug.Print "SQL for " & strQuery & ": " & strSQL
lngCount = CreateAndTestQuery(strQuery, strSQL)
Debug.Print "No. of records: " & lngCount
If lngCount = 0 Then
   strPrompt = "No records found; canceling"
   strTitle = "Canceling"
   MsgBox strPrompt, vbOKOnly, strTitle
   GoTo ErrorHandlerExit
End If
```

The final step is to update the report with the current date (stored in the Tag property for display in the txtTitle textbox) and ask the user if they want to view it:

```
strReport = "rptAnswers"
DoCmd.OpenReport reportname:=strReport, _
 view:=acViewDesign, windowmode:=acHidden
Set rpt = Reports(strReport)
rpt.Tag = strCurrentDate

strTitle = "Finished"
strPrompt = strResultsTable & _
" results table created; open report?"
intResult = MsgBox(strPrompt, vbYesNo, strTitle)
If intResult = vbYes Then
   DoCmd.OpenReport reportname:=strReport, _
    view:=acViewPreview
Else
   DoCmd.Close objecttype:=acReport, _
            objectname:=strReport
End If
```

Finally, exit the function, with an error handler to take care of any errors:

```
ErrorHandlerExit:
   Exit Function

ErrorHandler:
   MsgBox "Error No: " & Err.Number & _
       "; Description: " & Err.Description
   Resume ErrorHandlerExit

End Function
```

The CreateAndTestQuery function listed below is handy for creating (and re-creating, as needed) a query in code. I use it to re-create the totals query qtotAnswers, based on the newly created results table (see Figure 2).

```
Public Function CreateAndTestQuery( _
  strTestQuery As String, strTestSQL As String)
  As Long

On Error Resume Next

   Set dbs = CurrentDb
   dbs.QueryDefs.Delete strTestQuery

On Error GoTo ErrorHandler

   Set qdf = dbs.CreateQueryDef(strTestQuery, _
    strTestSQL)

   'Test whether there are any records
   Set rst = dbs.OpenRecordset(strTestQuery)
   With rst
      .MoveFirst
      .MoveLast
      CreateAndTestQuery = .RecordCount
   End With

ErrorHandlerExit:
   Exit Function

ErrorHandler:
   If Err.Number = 3021 Then
      CreateAndTestQuery = 0
      Resume ErrorHandlerExit
   Else
      MsgBox "Error No: " & Err.Number & _
        "; Description: " & Err.Description
      Resume ErrorHandlerExit
   End If

End Function
```

You can find a sample database with all of the code in the accompanying download file. In addition to the Microsoft DAO 3.6 Object Library, my sample database also uses the Scripting Runtime Library. ▲

DOWNLOAD **604FEDDEMA.ZIP at www.pinnaclepublishing.com**

Helen Feddema is an independent developer and writer on Access and Office topics who lives in the middle of New York state. Her latest book is *Expert One-on-One Microsoft Access Application Development* (Wiley, ISBN 0764559044).