Graphing crosstabs

A picture is said to be worth 1000 words, so it's not surprising that a graph or chart makes data much easier to understand. There are multiple ways to put VFP data into a graph or chart.

Tamar E. Granor, Ph.D.

In my last article, I showed how to send crosstab data to Excel, including how to build Excel's Pivot Tables programmatically. This article shows how to create graphs of the crosstab data.

There are two straightforward ways to graph this data. Excel has strong graphing capabilities and includes a tool for building what are called "Pivot Graphs." If you need graphs inside your VFP application, VFPX's FoxCharts gives you what you need.

For this article, we'll look at sales by category, including tracking by country. Listing 1 shows the query that collects the raw data, plus a couple of additional columns. Figure 1 shows partial results.

Listing 1. This query collects information about sales by product, category, year and country.

```
SELECT ProductName, ;
       CategoryName, ;
       YEAR(OrderDate) AS Year, ;
       ShipCountry, ;
       SUM(Quantity) AS NumSold ;
   FROM Orders ;
     JOIN OrderDetails OD ;
       ON Orders.OrderID = OD.OrderID ;
     JOIN Products ;
       ON OD.ProductID = Products.ProductID ;
     JOIN Categories ;
       ON Products.CategoryID = ;
          Categories.CategoryID ;
   GROUP BY 1, 2, 3, 4 ;
   INTO CURSOR csrProductsSold
```

Creating Excel's Pivot Graphs

The Pivot Table item on Excel's Insert tab includes a choice of Pivot Table or Pivot Chart. Choosing Pivot Chart lets you specify a data range and then opens the Pivot Table Wizard with an added element for the chart itself. As you work through the wizard, you create both a pivot table and a graphical representation of the data. For example, Figure 2 shows a graph of sales by category and country; you can see part of the pivot table behind the graph.

You create a pivot graph programmatically by creating a pivot table and then adding a chart. The AddChart method of the Shapes collection adds a chart to the worksheet. It accepts five parameters. The first is the chart type (from the xlChartType enumeration). The other four are for positioning the chart: left, top, width, height. Excel is smart enough to make the chart based on the pivot table you just created without your having to do anything special.

The code in Listing 2 creates the pivot table and chart. The complete code (including the query) is included in this month's downloads as PivotProductSales.prg. As with the examples in my last article, the data is exported to a CSV file and then opened in Excel. Then, a pivot cache is created containing all the data, and used to create a pivot table. Finally, AddChart is called. Chart type 51, which is the default when you work interactively, creates a vertical bar chart (which Excel calls a "column chart") with a separate bar for each data point.

Listing 2. This code creates the pivot table and pivot graph shown in Figure 2.

```
LOCAL cCSVFile
cCSVFile = FORCEPATH(FORCEEXT( ;
  "ProductSales", "CSV"), SYS(2023))

COPY TO (m.cCSVFile) TYPE csv

LOCAL oExcel AS Excel.Application, ;
      oWorkbook as Excel.Workbook, ;
      oSheet AS Excel.Worksheet
LOCAL oPC AS Excel.PivotCache, ;
      oPT AS Excel.PivotTable
LOCAL oChart AS Excel.Chart, ;
      oRange AS Excel.Range

oExcel = CREATEOBJECT("Excel.Application")
oWorkbook = oExcel.Workbooks.Open(m.cCSVFile)
oExcel.Visible = .T.

oSheet = oExcel.Sheets.Add()
oSheet.Name = "SalesByCategoryAndCountry"

oRange = oExcel.ActiveWorkbook.Worksheets( ;
  "ProductSales").UsedRange()
oPC = ;
  oExcel.ActiveWorkbook.PivotCaches.Create( ;
    1,  m.oRange, 4)&& 1 = xlDatabase
oPT = oPC.CreatePivotTable( ;
  "SalesByCategoryAndCountry!R1C1", ;
  "PivotTable1", .T., 4)
oPT.AddDataField(oPT.PivotFields("numsold"), ;
  "Units", -4157) && xlSum
```

Csrproductssold

Productname	Categoryname	Year	Shipcountry	Numsold
Alice Mutton	Meat/Poultry	1996	Belgium	40
Alice Mutton	Meat/Poultry	1996	Canada	70
Alice Mutton	Meat/Poultry	1996	France	30
Alice Mutton	Meat/Poultry	1996	Germany	15
Alice Mutton	Meat/Poultry	1996	Mexico	8
Alice Mutton	Meat/Poultry	1996	USA	71
Alice Mutton	Meat/Poultry	1997	Austria	191
Alice Mutton	Meat/Poultry	1997	Canada	50
Alice Mutton	Meat/Poultry	1997	Italy	20
Alice Mutton	Meat/Poultry	1997	Mexico	18
Alice Mutton	Meat/Poultry	1997	Spain	48
Alice Mutton	Meat/Poultry	1997	Sweden	10
Alice Mutton	Meat/Poultry	1997	UK	25
Alice Mutton	Meat/Poultry	1997	USA	165
Alice Mutton	Meat/Poultry	1998	Brazil	27
Alice Mutton	Meat/Poultry	1998	Canada	6
Alice Mutton	Meat/Poultry	1998	France	37
Alice Mutton	Meat/Poultry	1998	Mexico	10
Alice Mutton	Meat/Poultry	1998	Spain	12
Alice Mutton	Meat/Poultry	1998	USA	125
Aniseed Syrup	Condiments	1996	UK	30
Aniseed Syrup	Condiments	1997	Austria	20
Aniseed Syrup	Condiments	1997	Canada	20
Aniseed Syrup	Condiments	1997	Denmark	14
Aniseed Syrup	Condiments	1997	Germany	66
Aniseed Syrup	Condiments	1997	Venezuela	70

Figure 1. The query in Listing 1 collects data for sales of each product by year and country, and includes the product's category.

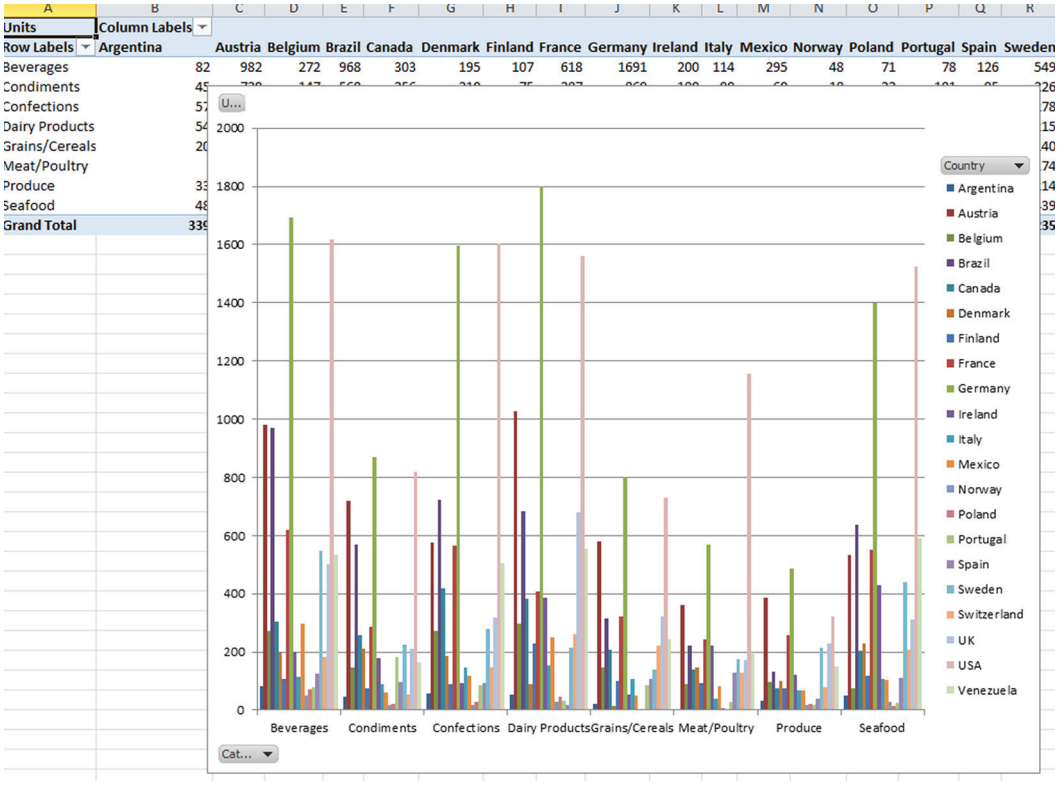

Figure 2. As you create an Excel pivot graph using the wizard, a pivot table is created as well.

```
WITH oPT.PivotFields("categoryname") ;
  AS Excel.PivotField
   .Orientation = 1 && xlRowField
   .Position = 1
   .Caption = "Category"
ENDWITH
```

```
WITH oPT.PivotFields("shipcountry") ;
  AS Excel.PivotField
   .Orientation = 2 && xlColumnField
   .Position = 1
   .Caption = "Country"
ENDWITH

oChart = oSheet.Shapes.AddChart(51, ;
  150, 50, 600, 300) && 51 = xlColumnClustered

RETURN
```

As Figure 2 shows, pivot charts offer the same sorting and filtering options as pivot tables. The Category and Country dropdowns let you sort the bars and let you filter some out based on either their labels or their values. As described in my last article, value filters work only on the grand total, not the values in the individual columns.

To add a filter, call the Add method of the PivotFilters collection for the row or column you want to filter on. For example, adding the line in Listing 3 to the end of the previous example produces the result in Figure 3. The first parameter indicates the type of filter, based on the xlPivotFilterType collection; 1 is a top count. (The collection is documented at http://tinyurl.com/ycwr2mtj.) The second parameter specifies the field to apply the filter to; here, it's the number of units sold. The third parameter is the value for the filter; here, we're asking for the top five. The result is the top five countries by total units sold. (The method has additional parameters not relevant in this case. The one you're most likely to use is a second value parameter, for cases where you're filtering for information between two values.) A version of the code setting this filter is included as PivotProductSalesFiltered.prg in this month's downloads.

Listing 3. You can filter the data in the pivot table and pivot chart using the PivotFilters collection.

```
oPT.PivotFields("Country").PivotFilters.Add( ;
  1, oPT.PivotFields("units"), 5)
```

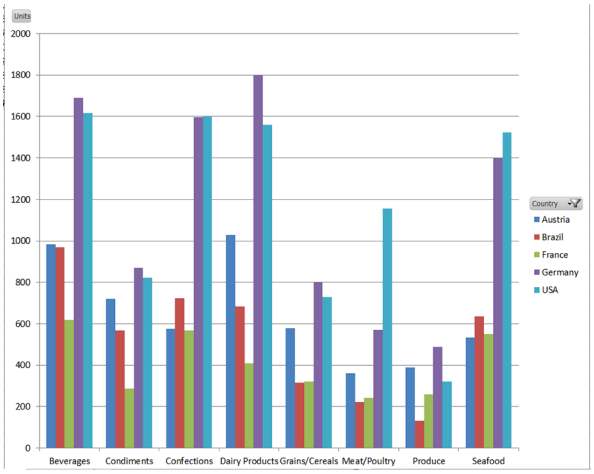

Figure 3. We can reduce the amount of data in the chart by filtering.

If you instead add a filter to choose the top three by category, as in Listing 4, you get a very different result, shown in Figure 4 (Complete code for this example is included in this month's downloads as PivotProductSalesFiltered2.prg.)

Listing 4. This line filters for the top three categories by total sales.

```
oPT.PivotFields("Category").PivotFilters.Add(;
  1, oPT.PivotFields("units"), 3)
```

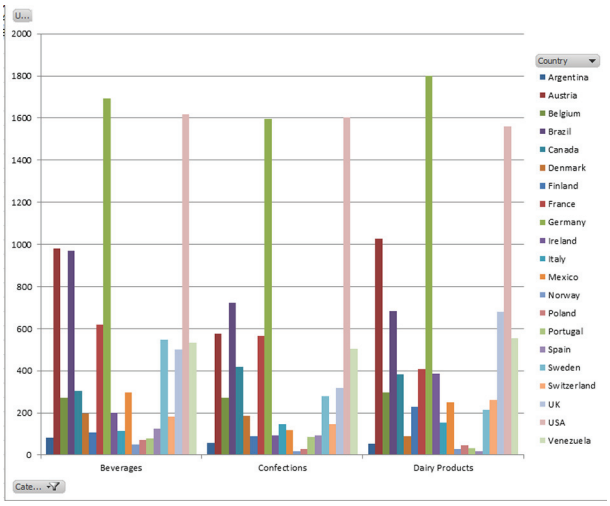

Figure 4. You can filter to show only the top-selling categories.

You can add multiple filters, resulting in reducing the data shown even more.

Creating other types of charts

Of course, you can create other types of charts and graphs. Interactively, you right-click on the chart and choose Change Chart Type. Programmatically, you simply pass a different value for the first parameter of the AddChart method.

Excel offers many types of charts. In Figure 5, some of the xlChartType enumeration is shown in the VFP Object Browser.

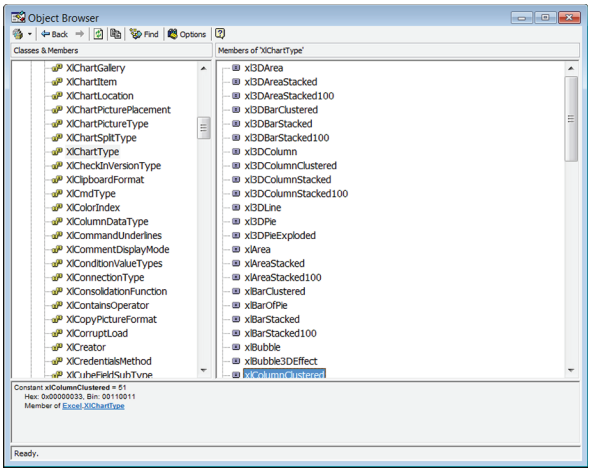

Figure 5. VFP's Object Browser lets you explore Excel's enumerations. Here, part of the list for xlChartType is shown.

The code in Listing 5 builds a pie chart showing each category's share of sales. (It assumes the data has already been collected, exported from VFP and imported into Excel. The complete program is included in this month's downloads as ProductSalesByCategoryPie.prg.) Figure 6 shows the result.

Listing 5. The first parameter of the AddChart method determines the type of chart.

```
oSheet = oExcel.Sheets.Add()
oSheet.Name = "SalesByCategoryAndCountry"

oRange = oExcel.ActiveWorkbook.Worksheets( ;
  "ProductSales").UsedRange()
oPC = ;
  oExcel.ActiveWorkbook.PivotCaches.Create( ;
  1, m.oRange, 4)
oPT = oPC.CreatePivotTable( ;
  "SalesByCategoryAndCountry!R1C1", ;
  "PivotTable1", .T., 4)
oPT.AddDataField(oPT.PivotFields("numsold"), ;
  "Units", -4157) && xlSum
WITH oPT.PivotFields("categoryname") ;
  AS Excel.PivotField
   .Orientation = 1 && xlRowField
   .Position = 1
   .Caption = "Category"
ENDWITH

oChart = oSheet.Shapes.AddChart(70, ;
  150, 50, 600, 300) && 70 = xl3DPieExploded
```

With dozens of chart types, you should be able to find one that helps your users understand their data.

Producing graphs and charts in VFP with FoxCharts

FoxCharts is one of the premiere tools available from VFPX, the community open source project for VFP. It uses GDIPlusX to provide a fairly easy way to put graphs and charts into VFP forms.

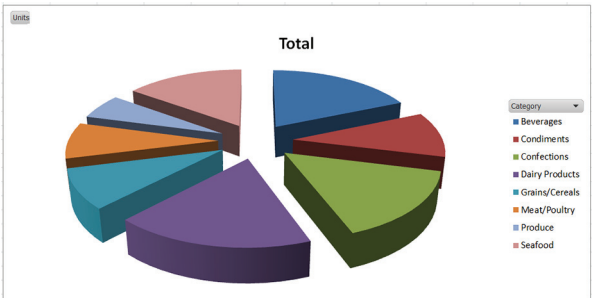

Figure 6. This chart shows the share of sales for each category.

You can install FoxCharts through Thor or directly from VFPX's new home on GitHub. The direct link is https://github.com/VFPX/FoxCharts.

While FoxCharts hasn't been covered in depth in FoxRockX, the download site includes a 43-page paper from Jim Nelson. In addition, Doug Hennig has a great paper on his website that shows you how to get started and gives you an idea as to its capabilities: (http://doughennig.com/papers/Pub/FoxCharts.pdf). So I won't go into detail on getting started. Instead, I'll focus on the kinds of charts you're likely to want to create from crosstab data.

In general, to use FoxCharts, you drop the FoxCharts class on a form and set properties and add code. (FoxCharts also comes with a tool that lets you create charts interactively. You may find that easier.) To make it easy for you to replicate my examples, I'm doing most of the work in code rather than setting properties of the chart and the form in the Property Sheet.

I've created a form class that includes the FoxCharts object and names it cntChart. I've also added a custom method called MakeChart to the form and added a call to that method in the form's Init method, as in Listing 6. The library containing the form class is included in this month's downloads as GranT060.VCX; if you use it, you'll need to point to the location where you've installed FoxCharts.

Listing 6. To encapsulate all the code that populates the chart, the form's Init method calls a custom MakeChart method.

```
This.MakeChart()
```

Creating a pie chart

We'll start by replicating the pie chart from Figure 6, though technically, it doesn't use a crosstab. The first step is collecting the necessary data. FoxCharts is a little finicky about its data and it's best if the cursor you supply contains only data you want to graph, so this form's MakeChart method starts with the query in Listing 7.

Listing 7. This query collects the data needed to create a pie chart of units sold by category.

```
SELECT CategoryName, ;
       SUM(Quantity) AS NumSold ;
   FROM Orders ;
     JOIN OrderDetails OD ;
       ON Orders.OrderID = OD.OrderID ;
     JOIN Products ;
       ON OD.ProductID = Products.ProductID ;
     JOIN Categories ;
       ON Products.CategoryID = ;
          Categories.CategoryID ;
   GROUP BY 1  ;
   INTO CURSOR csrProductsSold
```

FoxCharts lets you specify which slices should be exploded individually by providing a column in the underlying cursor. (There are other options for exploding pie charts, as well. You can indicate that a slice explodes when you click on it, or when you click on its legend, rather than specifying a fixed list.) So, the next step is to add that column to the cursor, as in Listing 8.

Listing 8. The added lDetach column lets us specify which slices of the pie should be "exploded," that is, pulled outward from the chart. Here, we say all should be.

```
SELECT *, .T. AS lDetach ;
   FROM csrProductsSold ;
   INTO CURSOR csrProductsSold
```

Once we have data, we set properties of the cntChart object (as well as setting the form's Caption). Listing 9 shows the code.

Listing 9. This code tells the FoxCharts object what type of chart to draw, what data to use, and more.

```
ThisForm.Caption = "Units sold by category"

WITH This.cntChart
   .Anchor = 15
   .ChartType = 1 && Pie
   .ColorType = 0

   .SourceAlias = "csrProductsSold"

   .ChartsCount = 1
   .Fields(1).FieldValue = "NumSold"
   .FieldLegend = "CategoryName"


   .Title.Caption = 'Units Sold'
   .Subtitle.Caption = ''
   .XAxis.Caption = 'Category'
   .YAxis.Caption = 'Units sold'

   .FieldDetachSlice = "lDetach"

   .DrawChart()

ENDWITH
```

After setting the form caption, we anchor the FoxCharts object to the form, so that resizing the form resizes the chart.

Next, we indicate that we want a pie chart (ChartType = 1) and that we should use the basic color set (ColorType = 0).

The SourceAlias property tells the chart where its data comes from, though it doesn't specify which data from the cursor to graph.

The ChartsCount property sounds like it specifies how many charts you're creating, but it actually indicates how many data series are to be specified.

The value you specify is used to set the size of the Fields collection. So, after indicating we have only one data series (which is normal for a pie chart), we indicate the data for that series (the NumSold field of the cursor) by setting the FieldValue property of Fields(1), the first element in the Fields collection. The FieldLegend property specifies the field in the data source that provides the values to appear in the legend for the chart; here, it's the CategoryName column.

Next, we specify a title for the chart. As you can see, you can have both a title and a subtitle; a subtitle seemed like overkill for this simple example.

FieldDetachSlice applies to pie charts and indicates the field in the data source that specifies whether a given slice should be exploded, so we specify the lDetach column.

Finally, we call the DrawChart method to actually create the chart. The complete code for this example is included in this month's downloads as CategorySalesPie.SCX. The resulting form is shown in Figure 7.

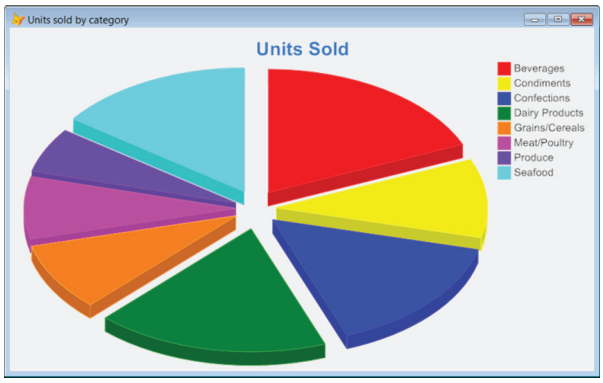

Figure 7. It takes only a little code to build this pie chart with FoxCharts.

Working with multiple data series

To graph crosstab results, we need to work with multiple data series. Each column created by a crosstab becomes a data series. The goal is to create a chart similar to Figure 2; the result is shown in Figure 8.

As in the previous example, first we need to collect the data. For this chart, we start with the query in Listing 1, and then crosstab it. Listing 10 shows the code that creates the crosstab cursor from the original query results; you need to either have FastXTab in your path or add the path in this code. Figure 9 shows partial results.

Listing 10. To create a chart showing units sold by category and country, we need to crosstab the data.

```
LOCAL oXTab AS FastXTab OF "fastxtab.prg"

oXTab = NEWOBJECT("fastxtab", "fastxtab.prg")

WITH oXTab AS FastXTab OF "fastxtab.prg"
   .cOutFile = "csrXtab"
   .cRowFIELD = "CategoryName"
   .cColField = "ShipCountry"
   .cDATAFIELD = "NumSold"
   .lCursorOnly = .T.
   .lCLOSETABLE = .T.
   .RunXtab()
ENDWITH
```

Once we have the data, as before, we set properties of the FoxCharts object. In this case, we want to set ChartCount to the number of data columns (that is, the number of countries) in the cursor. Once we do that, we need to provide information about each column. Listing 11 shows the code to specify the chart; the complete program for this chart is included in this month's downloads as CategorySalesByCountry.SCX.

Listing 11. To create the chart in Figure 8, we need to set up a data series for each country.

```
LOCAL nCountries
nCountries = FCOUNT("csrXTab") - 1

This.Caption = "Category Sales by Country"

WITH This.cntChart
   .Anchor = 15
   .ChartType = 8 && MultiBar
   .ColorType = 0

   .SourceAlias = "csrXTab"
   .FieldAxis2 = "CategoryName"
   .AxisLegend2.Rotation = -45
   .AxisLegend2.Alignment = 1 && Right

   .ChartsCount = m.nCountries

   LOCAL nSeries, nFirstDataCol
   nFirstDataCol = 2

   FOR nSeries = 1 TO m.nCountries
      WITH .Fields(m.nSeries)
         .FieldValue = FIELD(m.nSeries + ;
            m.nFirstDataCol -1 )
         .Legend = .FieldValue
      ENDWITH
   ENDFOR

   .Title.Caption = 'Units Sold'
   .Subtitle.Caption = ;
     'By Category and Country'
   .XAxis.Caption = 'Category'
   .YAxis.Caption = 'Units sold'

   .DrawChart()

ENDWITH
```

To create a bar chart with multiple data series, we set ChartType to 8. Bar charts have legends on their axes, as well as an optional side legend (like the one for the pie chart). For this chart, the Y-axis wants numeric values representing the data; we don't have to specify anything for that to happen. But the X-axis should be labelled with the category names. That's what we get by setting the FieldAxis2 property. The two lines after that angle the labels and right-justify them, so they don't overlap.

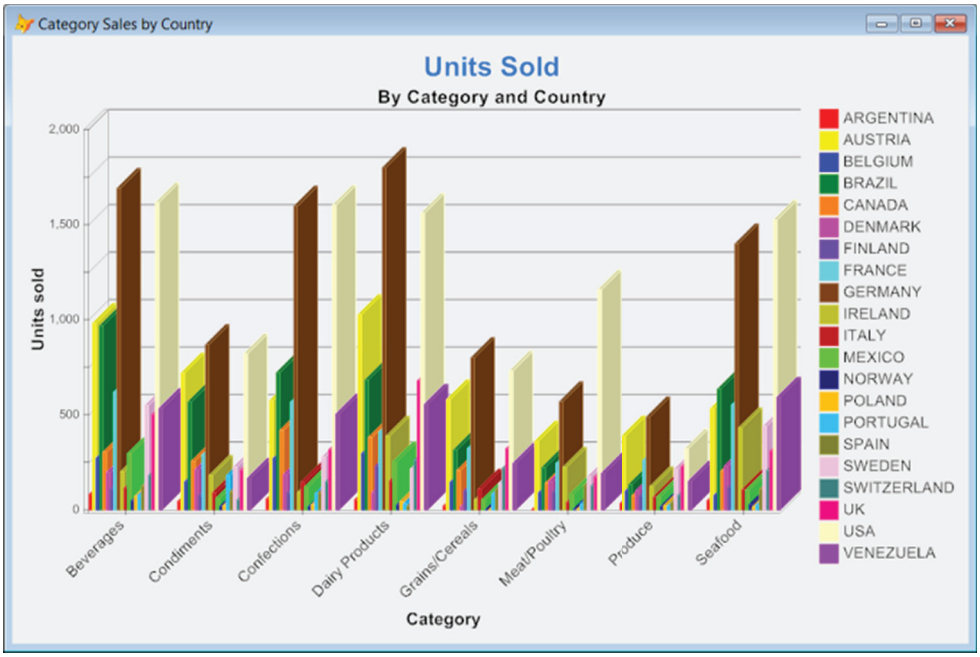

Figure 8. This is the FoxCharts version of the bar chart showing sales by country for each category.

Lots more options

Charts created with FoxCharts don't offer the ability for live filtering that Excel does, but they can include tooltips that show the value of a given item when the mouse hovers over it, as in Figure 10. Some chart types have other dynamic options, as well.

FoxCharts offers many more chart types, as well as control over colors and much more.

In addition, though we've looked at putting charts on forms, they can also be printed. See the documentation on GitHub for details.

Csrxtab

Categoryname	Argentina	Austria	Belgium	Brazil	Canada
Beverages	82	982	272	968	303
Condiments	45	720	147	568	256
Confections	57	575	270	722	418
Dairy Products	54	1027	295	683	381
Grains/Cereals	20	580	145	315	207
Meat/Poultry	0	362	89	223	141
Produce	33	388	98	133	74
Seafood	48	533	76	635	204

Figure 9. This is part of the data to be graphed. There's one column for each country.

Final Thoughts

In this series of articles, I've shown quite a few ways to deal with crosstab and pivot results, from VFP reports to various ways of getting and displayed them in Excel to charting them in VFP. I hope one of them works for you and your customers.

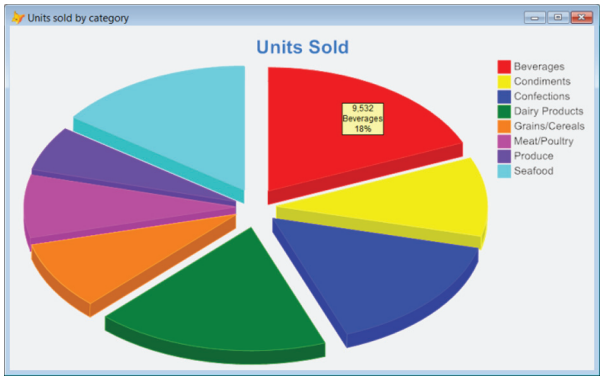

Figure 10. By default, FoxCharts shows tooltips with data values.

The key part of the code is the loop, which sets properties of the members of the Fields collection, based on the data columns in the crosstab. Setting the Legend property of the fields specifies that there should be a block legend (which, in this case, shows the colors for the countries).

Author Profile

Tamar E. Granor, Ph.D. is the owner of Tomorrow's Solutions, LLC. She has developed and enhanced numerous Visual FoxPro applications for businesses and other organizations. Tamar is author or co-author of a dozen books including the award winning Hacker's Guide to Visual FoxPro, Microsoft Office Automation with Visual FoxPro *and* Taming Visual FoxPro's SQL. *Her latest collaboration is* VFPX: Open Source Treasure for the VFP Developer, *available at www.foxrockx.com. Her other books are available from Hentzenwerke Publishing (www.hentzenwerke.com). Tamar was a Microsoft Support Most Valuable Professional from the program's inception in 1993 until 2011. She is one of the organizers of the annual Southwest Fox conference. In 2007, Tamar received the Visual FoxPro Community Lifetime Achievement Award. You can reach her at tamar@thegranors.com or through www.tomorrowssolutionsllc.com.*