Mine Your Data With DynamiCube

Doug Hennig

Providing your users with the ability to analyze their data by breaking it down an unlimited number of ways sounds like a "pie in the sky" goal. However, using the DynamiCube ActiveX control, you can easily add this feature to any VFP application in less than an hour.

In the July 1997 issue of FoxTalk, I presented a technique and code for using a grid to drill down into data. The idea was to allow the user to see different levels of data breakdown. For example, at the top level, they could see sales by employee. They could then drill down into an employee to see sales for that employee broken down by product, and drill down into a product to see sales for that employee and product broken down by country, and so on. It took me quite a while to figure out how to do this, and I wasn't completely satisfied with the behavior (due to the way VFP handles grid partitions). This month, we're going to look at another way of doing this using a commercially available ActiveX control.

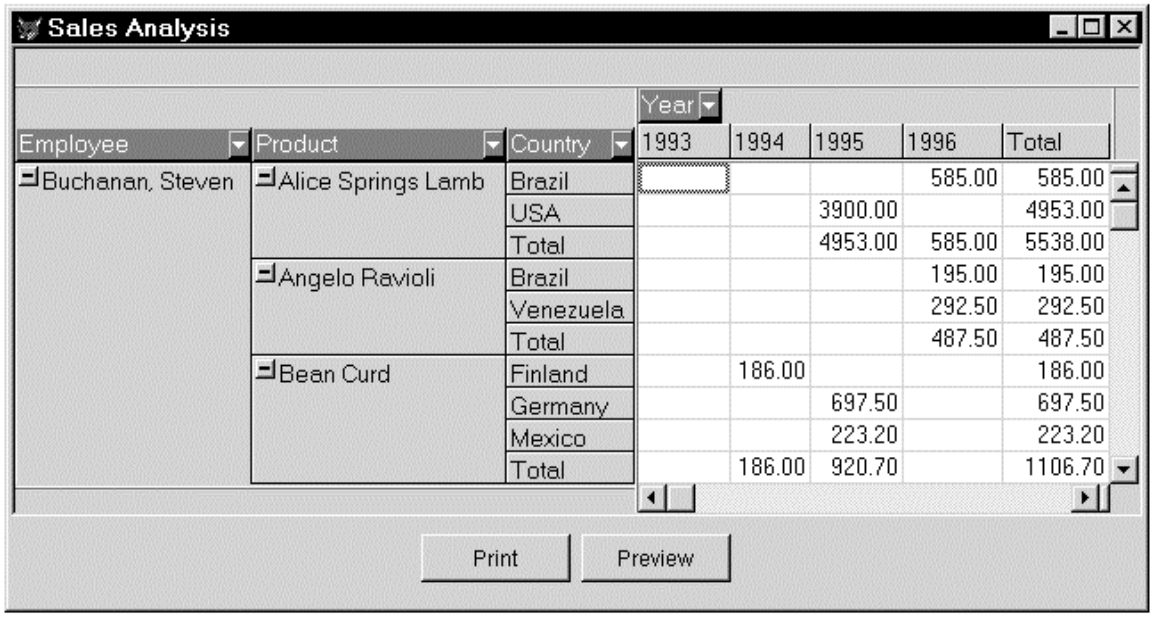
DynamiCube

Data Dynamics has created an ActiveX control called DynamiCube. This "dynamic data mining" control is similar to Excel pivot tables but with a lot more flexibility and capabilities. It can be used in applications created with languages supporting ActiveX controls, such as VFP, Visual Basic, or Delphi, and can be used on Web sites to provide interactive data reporting.

Data Dynamics calls DynamiCube "an open, interactive OLAP front-end, implemented as an Internet-enabled lightweight ActiveX control". DynamiCube provides extremely fast calculation, summarization, and presentation of multi-dimensional data. It can be either bound directly to data or used unbound by manually loading data. Because of its interactive nature, users can rearrange the fields in the control and have DynamiCube instantly recalculate and display the results.

Figure 1 shows a VFP form called SALECUBE that I created with the DynamiCube control. This form, which we'll look at in detail in a moment, displays sales information from the TESTDATA database that comes with VFP.

Figure 1. VFP form showing the DynamiCube control in action.



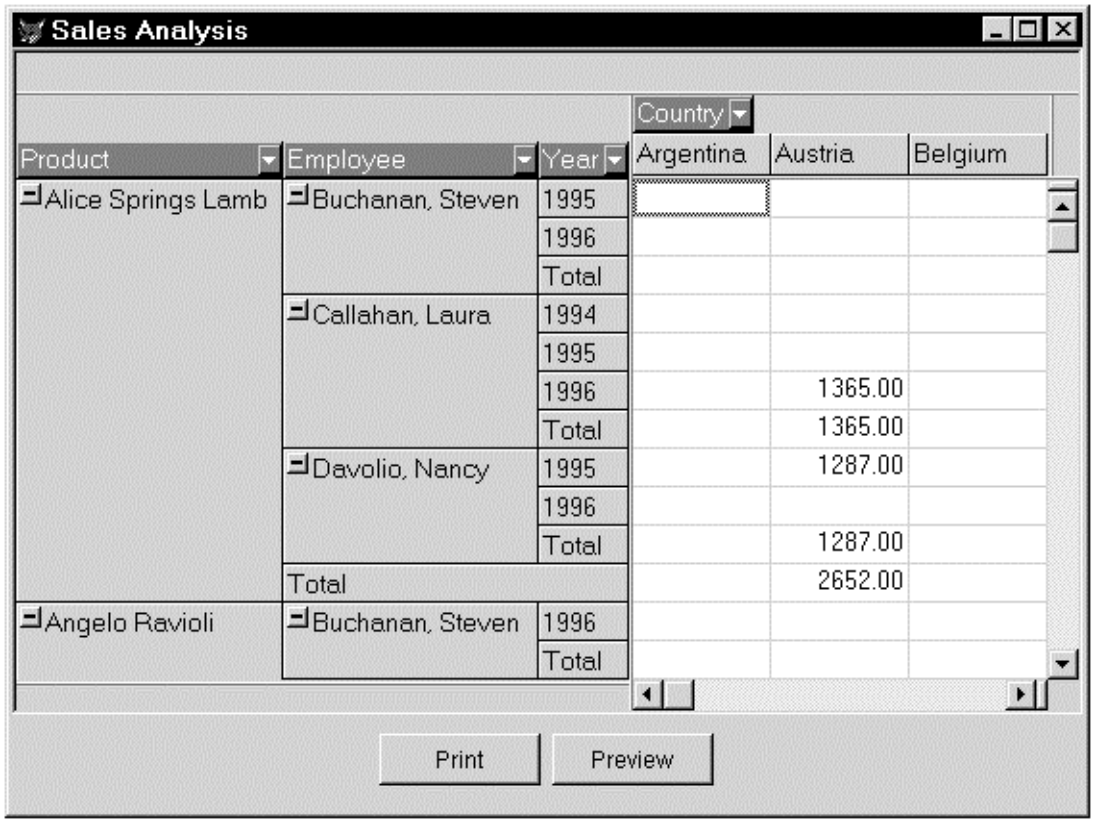
This form shows sales broken down by employee, product, country, and year. Totals for each of these breakdowns are automatically calculated and displayed. For example, under the set of countries for a particular employee and product is a "Total" entry showing the total sales for the specific employee and

product. The last "years" column shows the total sales for the specific employee, product, and country. Scrolling to the bottom of the control shows column totals for each year.

Run this form so you can play with the DynamiCube control to get an idea of its features (note: before you run or modify the form, you'll need to install the control on your system; see the next section in this article for details). Here are some of the things a user can do with this control:

- Resize columns as desired.
- Expand or collapse breakdown fields. For example, click on the "-" in front of a product name to collapse it and notice all country details disappear and the sales grid portion of the control displays the rolled up sales value by year for the product. Similarly, collapse an employee name and notice breakdowns by product disappear so only rolled up sales values by year are shown for the employee.
- Filter on any field. Column headings are combo boxes; dropping the combo box down displays all the values in the column as check boxes. Unchecking a value filters that value out of the list, automatically adjusting totals appropriately.
- Print or preview the data. Click on either the Print or Preview buttons. DynamiCube has several properties allowing you to control the look of the printed output, such as headers and footers, margins, fonts, orientation, color, etc.
- Now the coolest feature: breakdown the sales values differently than originally presented by simply dragging a column to a new location. For example, drag the employee column heading until it appears after the country column. Sales values are instantly recalculated so they are now broken down by product, then country, then employee. Drag country up beside year, and year down beside employee. The form will now appear similar to that in Figure 2.

Figure 2. Sales broken down differently after rearranging the DynamiCube.



Using the DynamiCube Control

The DynamiCube control is used as any other ActiveX control. First, install it on your system. You can either order it ($499 per developer license) or obtain a 30-day trial version from the Data Dynamics Web site (www.datadynamics.com). Running the supplied EXE installs the OCX, help, and other support files in a directory of your choosing (a refreshing change from ActiveX controls which insist on installing in \WINDOWS\SYSTEM) and registers it in the Windows Registry. Next, make it available to VFP by choosing the Controls page in the VFP Tools Options dialog, selecting ActiveX controls, and then checking the checkbox for the DCube class in the list of installed controls. The DynamiCube control will then be available in the ActiveX controls toolbar when you're working in the Class or Form Designer.

The help file (DCUBE.HLP) that accompanies the control is indispensable for working with the control, providing details on properties, events, and methods (PEMs) of the objects contained within the control. The DynamiCube control consists of three objects: the control itself, the Fields collection, and the DataItems collection. Each has its own set of PEMs you'll work with. For example, you'll set up the fields that appear in the control using the Add method of the Fields collection.

Here are details on how the SALECUBE form was created. First, the form was created from the VFP Form class, and the following properties set in the Property Sheet:

- AutoCenter: .T.
- Caption: Sales Analysis
- MinHeight: 200
- MinWidth: 540
- Name: frmSales

Normally, I'd populate the DataEnvironment with the tables used as the source of data, but to prevent pathing problems when you try out this form (for example, if you run the form on a different drive than where the VFP TESTDATA database is located), I just manually opened them in the Load method of the form. Here's the code:

```
set talk off
set deleted on

* Open the tables we need.

open database (home() + 'SAMPLES\DATA\TESTDATA')
use CUSTOMER in 0
use EMPLOYEE in 0
use ORDERS   in 0
use ORDITEMS in 0
use PRODUCTS in 0
```

Next, I dragged the DynamiCube control from the ActiveX control toolbar to the form and sized it as desired. Although I didn't, you could set some properties of the control as desired by right-clicking on it and choosing "DCube Class Properties" from the menu. Some obvious ones you may want to change are fonts, color, and printer settings.

The Init method of the form sets some properties and creates some data fields for the DynamiCube control. The DCConnectType property indicates how the control is connected to the data source. DynamiCube can use Remote Data Object (RDO), Data Access Object (DAO), or ODBC to directly connect to the data source. For simplicity and performance, I chose to not have the control bound directly to data but to instead manually load the data, so I set DCConnectType to 99. The HeaderCaption property is the text that will be printed on the top of each page. The Add method of the Fields collection adds a field to the control. The first parameter of this method is the name of the field and the second is the caption for the column heading. The third parameter is the "orientation" of the field; a field can be a row (2), a column (1), a data field (3), a page, or hidden. In this case, we're using Employee, Product, and Country as rows, Year as a column, and Sales as the data values in the grid. Because loading the data can take a while and we don't want the user to watch the control flash as it's built, we'll temporarily move it off-screen.

```
with This.oCube
  .DCConnectType = 99
  .HeaderCaption = This.Caption
```

```
  .Fields.Add('Employee', 'Employee', 2)
  .Fields.Add('Product',  'Product',  2)
  .Fields.Add('Country',  'Country',  2)
  .Fields.Add('Year',     'Year',     1)
  .Fields.Add('Sales',    'Sales',    3)
endwith

* Move the form off-screen so the user doesn't see it
* until the DynamiCube has loaded all its data.

This.Left = -(This.Width + 10)
```

The FetchData event of the control fires automatically when the DCConnectType property is set to 99 and the control needs data to display. In this event, we'll use a SQL SELECT statement to create a cursor of individual sales, including the employee name, product name, country, and year. We'll then process this cursor, using the AddRow method of the DynamiCube control to add a set of data for all fields at once. Notice we haven't summarized or grouped the data; the control will do that for us automatically, so we just provide it with the raw values.

```
local lnLength, ;
  laData[7]

* Create a cursor containing the data we want to display.

lnLength = len(EMPLOYEE.LAST_NAME + ;
  EMPLOYEE.FIRST_NAME) + 2
select PRODUCTS.ENG_NAME as PRODUCT, ;
    year(ORDERS.ORDER_DATE) as YEAR, ;
    ORDITEMS.UNIT_PRICE * ORDITEMS.QUANTITY as PRICE, ;
    CUSTOMER.COUNTRY, ;
    padr(trim(EMPLOYEE.LAST_NAME) + ', ' + ;
      EMPLOYEE.FIRST_NAME, lnLength) as EMPLOYEE ;
  from PRODUCTS, ;
    ORDERS, ;
    ORDITEMS, ;
    CUSTOMER, ;
    EMPLOYEE ;
  where PRODUCTS.PRODUCT_ID = ORDITEMS.PRODUCT_ID and ;
    ORDITEMS.ORDER_ID = ORDERS.ORDER_ID and ;
    ORDERS.CUST_ID = CUSTOMER.CUST_ID and ;
    ORDERS.EMP_ID = EMPLOYEE.EMP_ID ;
  into cursor SALES

* Populate the DynamiCube with this data.

scan
  laData[1] = trim(EMPLOYEE)
  laData[2] = trim(PRODUCT)
  laData[3] = trim(COUNTRY)
  laData[4] = YEAR
  laData[5] = PRICE
  This.AddRow(@laData)
endscan

* Close the cursor.

use
```

The QueryComplete event fires when the control has finished summarizing all its data. All we'll do here is re-center the form to move it back on-screen.

```
Thisform.AutoCenter = .T.
```

All that's left is a few bells and whistles. The Click events of the Print and Preview buttons call the Print and PrintPreview methods of the control. The Resize event of the form resizes the DynamiCube control and repositions the buttons.

```
local lnWidth, ;
  lnGap, ;
  lnLeft
with This
  .LockScreen      = .T.
  .oCube.Width     = .Width
  .oCube.Height    = .Height - 50
  .cmdPrint.Top    = .Height - 40
  .cmdPreview.Top  = .Height - 40
  lnWidth          = .cmdPreview.Left + ;
    .cmdPreview.Width - .cmdPrint.Left
  lnGap            = .cmdPreview.Left - ;
    .cmdPrint.Left - .cmdPrint.Width
  lnLeft           = int((.Width - lnWidth)/2)
  .cmdPrint.Left   = lnLeft
  .cmdPreview.Left = lnLeft + .cmdPrint.Width + lnGap
  .LockScreen      = .F.
endwith
```

Summary

The DynamiCube ActiveX control can provide data mining features that would take you days, weeks, or even months to write in native VFP code. Providing your users with the ability to analyze their data in different ways without constantly asking you to create new reports is something both you and they will get used to very quickly.

Doug Hennig is a partner with Stonefield Systems Group Inc. in Regina, Saskatchewan, Canada. He is the author of Stonefield's add-on tools for FoxPro developers, including Stonefield Database Toolkit for Visual FoxPro and Stonefield Data Dictionary for FoxPro 2.x. He is also the author of "The Visual FoxPro Data Dictionary" in Pinnacle Publishing's "The Pros Talk Visual FoxPro" series. Doug has spoken at user groups and regional conferences all over North America, and spoke at the 1997 Microsoft FoxPro Developers Conference. He is a Microsoft Most Valuable Professional (MVP). CompuServe 75156,2326 or dhennig@stonefield.com.