Access Data in a Flash

VB4's new bound controls make database access quick and painless.

by Chris Barlow

One of the neatest things about Visual Basic is the easy database access. I guarantee that sooner or later you'll need to write a program that reads or updates a database—many programmers spend the majority of their time writing database access code. This kind of programming used to be time consuming, partly because you had to learn a custom API for each type of database. If you are just getting started with Visual Basic, I think you'll be amazed at how simple database access can be.

Earlier versions of Visual Basic let you easily bind a text-box or label control to a database record set. But Visual Basic 4.0 has added three data-aware controls that make it even easier to write powerful data-entry forms: a list box, a combo box, and a grid control. You'll fill these with data by binding them to the VB4 Data control, which accesses your database.

VB4's new Data control is powerful: you can use it to access any ODBC or Jet database including Microsoft Access, dBase, Paradox, SQL Server, and Oracle. The Data control is robust: programmers were cautious about using earlier versions because of poor control over error conditions, but the new version is much better. The Data control is simple: just set the DatabaseName and Recordset properties and you're ready to bind some controls.

Let's review how to bind a simple text-box control to the Data control. When binding a control to a Data control, choose the field from the Data control's record set that you want to display. With most controls, simply setting the DataSource and DataField properties causes the control to display the data from that field. Scrolling through the record set updates the bound control to display the current field information for that record. If you change the data in the bound control, the new data for that field is written back to the database record.

Start with a simple example that uses BIBLIO.MDB, the sample Microsoft Access database that ships with Visual Basic. Start a Visual Basic 4.0 project and add a Data control to the form. Right-click on the Data control and select the Properties menu item. Then set the DatabaseName property to point to the BIBLIO.MDB database. Now move down and click on the button in the RecordSource property to see a list of the tables in this database. Select the Titles table. The Data control is now set to display the Titles table in the Biblio database.

Now add three text-box controls to the form and select all three by clicking on the form and dragging the selection box around the text-box controls. Press F4 to view the Properties window and set the DataSource property to the Data1 Data control. Because you selected all three text-box controls, you have simultaneously set this property for all three controls. These controls are now "bound" to this Data control. Now click on one of the text-box controls and set the DataField property to the "Title" field. Set the other text-box controls to point to the "Year Published" and "Description" fields.

Now run your program. As you press the navigation buttons on the Data control, you'll see the field information appear in the text-box controls. Try changing the data in one of the text-box controls and scroll forward one record then back again—you'll see that your change has been written to the database (see Figure 1). Could anything be easier?

CUSTOMIZE YOUR RECORD SET

With the Data control, you don't have to display an entire table in a database. You can specify any SQL statement as the RecordSource property. For example, right now your program displays the Titles table in the sequence of the primary key. But how do you view the titles in alphabetical order? Simply create a SQL statement with an "Order by" clause.

Add this code in the Form_Load event:

```
Private Sub Form_Load()
Data1.RecordSource = "select * from _
   Titles order by Title"
End Sub
```

Run your program and scroll the Data control. The titles are now displayed in alphabetical order.

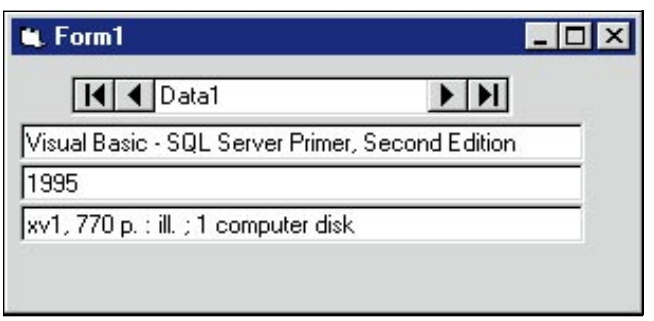

FIGURE 1 ***Simple Bound Controls.*** *Notice how easy it is to create a "live" form that allows a user to view and edit information from a database.*

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications including the ObjectBank and the ObjectJob Systems, where he and Ken Henderson hold a software patent related to decentralized distributed asynchronous object-oriented systems. Chris holds degrees from Harvard Business School and Dartmouth College where he worked with Drs. Kemeny and Kurtz on the BASIC language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.

Now try something more difficult using the Data control, such as displaying information from multiple tables. Create a more complex SQL statement that joins the Titles and Publishers table using the PubID field and displays two fields from the Titles table and one field from the Publishers table.

Build the SQL statement in three separate lines. The first line specifies the three fields to select from the tables. The second line shows that this query is a join of the Titles and Publishers table, and the third line indicates that the join is accomplished by linking the PubID field in each table. Notice that the DataField property of the three text-box controls is also set in this procedure:

```
Private Sub Form_Load()
Dim SQL$
SQL = "select Title, [Year Published], Name "
SQL = SQL & "from Titles, Publishers "
SQL = SQL & "where Titles.PubID = Publishers.PubID"
Data1.RecordSource = SQL
Text1.DataField = "Title"
Text2.DataField = "Year Published"
Text3.DataField = "Name"
End Sub
```

Run your program and scroll the Data control. You'll see the fields from both tables.

Now that you know how to bind a text-box control to a database record set, you can build a data-entry form for the Titles table. You could draw a text-box control for each field in the table and bind it to the field, but it's easier to use the Data Form Designer add-in that comes with Visual Basic.

The Data Form Designer is cool. You can specify a table or SQL statement from a database and choose the fields that you want on your form. The Data Form Designer creates the form and even adds buttons and code to add, update, and delete records from the record set. Start a new project, then select the Add-In Manager menu item from the Add-Ins menu and check the Data Form Designer add-in (see Figure 2).

Click on the Data Form Designer menu item on the Add-Ins menu, name the new form DBCtrl2, and open the Biblio database. Select the Titles table as the RecordSource, add all the fields to the form, and click the "Build the form" button (see Figure 3). The Data Form Designer creates a new form named frmDBCtrl2. Now remove the default Form1 from your project.

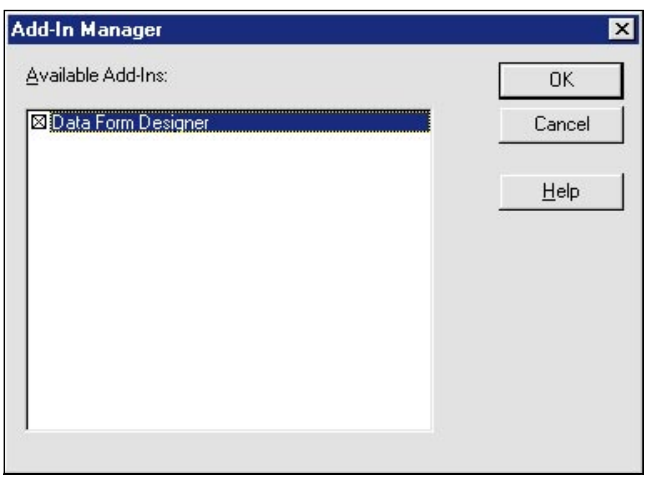

FIGURE 2 ***Manage Your Add-Ins.*** *The Data Form Designer is one of the neat new add-ins to Visual Basic 4.0. Look at the Visual Basic Samples\DataWiz directory for the source code.*

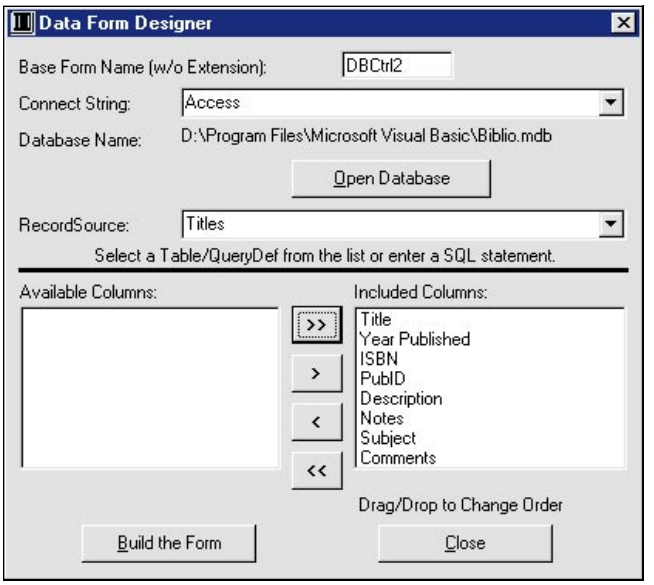

FIGURE 3 ***Creating a Data Form.*** *The Data Form Designer makes it simple to create a base form that you can easily customize.*

Select Options under the Tools menu, and use the Project tab to set the new form as your startup form and run your project. You have a working data-entry form for the Titles table without having to write any code (see Figure 4)!

THE NEW CONTROLS ARE BOUND TO BE GOOD

The only problem with this form is that because it displays the fields as they appear in the table, it displays the PubID rather than the publisher's name. This makes it difficult to use this form to add a new title because you need to know the PubID for the publisher. To let the user choose the publisher by name, you'll need to use the new DBList and DBCombo bound controls. To add these controls to your toolbox, select the Custom Controls menu item from the Tools menu and check the Microsoft Data Bound List Controls item.

The DBCombo control is similar to the regular ComboBox control in that you can bind the text to a field in the database record set. However, you can also bind the list portion of the DBCombo control to a different data source. In this example, you can bind the list portion of the DBCombo control to the Publishers table while binding the text portion to the Titles table.

Begin by widening the form slightly and adding another Data control to the form. Because your user won't need to use this Data control, set the Visible property to False or move it to the far right of the expanded form. Point the Data control to the same Biblio database and set the RecordSource to the Publishers table. Then add a DBCombo control next to the PubID field on the form. Set the DataSource to Data1 and the DataField to PubID, as you would with any control. Now the text portion of this control is bound to the Titles table.

Next set the RowSource property to the Data2 Data control and set the ListField property to the Name field. This is the field that will be displayed when the user clicks on the combo button to drop-down the list. Finally, set the BoundColumn property to the PubID field to indicate which field will be copied from the Data2 Data control's record set to the Data1 Data control's record set.

Now run your project and scroll the Data1 Data control. Note that the DBCombo control displays the name of the publisher

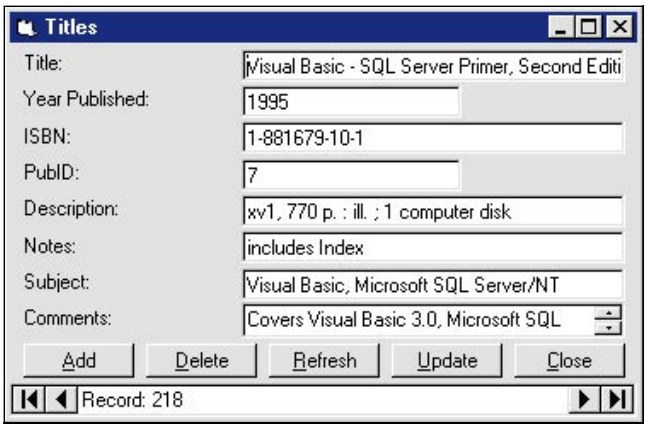

FIGURE 4 ***Like Magic!*** *This is the form created by the Data Form Designer. Notice how the Data Form Designer adds code to display the record number in the caption of the Data control.*

for each record. You can delete the text-box control for PubID and position the DBCombo control in its place. Don't forget to resize the form to hide the second Data control.

Often you want to present the fields from many database records in a spreadsheet or grid format so the user can scroll across the fields and up and down the records while making changes to any field within any record. A common use of the grid would be to display the subform data in a main form. Let's use the data-bound grid control to build an Author form that displays a grid of all Titles for that Author.

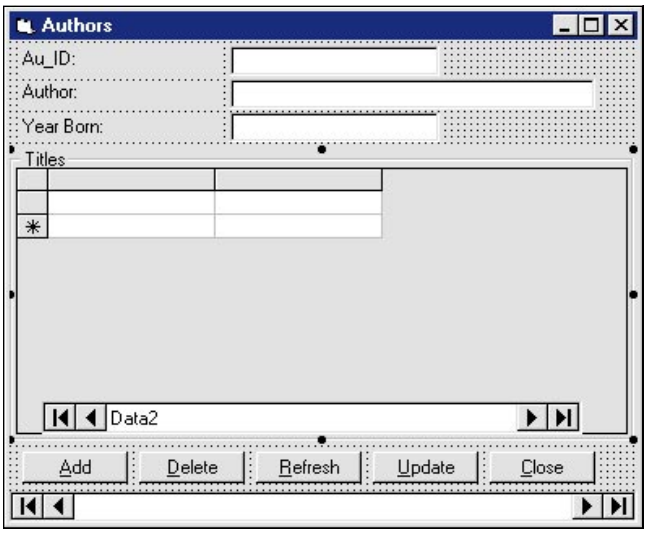

FIGURE 5 ***A Form Within a Form.*** *The bound grid control makes it easy to create sub forms within a main form, as you would in Microsoft Access.*

First, make sure the DBGrid control is in your toolbox. If it's not, add it by selecting the Custom Controls menu item from the Tools menu and checking the Apex Data Bound Grid item.

Start a new project and use the Data Form Designer to create an Authors form named DBCtrl3. Expand the form's height and move the button controls near the bottom of the form. Insert a Frame control between the fields and the buttons and add a second Data control and a DBGrid control (see Figure 5).

To link the Data2 Data control to the Author being displayed, you need to set its RecordSource property whenever the Data1 Data control is repositioned. Fortunately, the Data control triggers a Reposition event whenever it moves to a different record.

In order to allow multiple authors for a single title, the Biblio database has been designed with a Title Author table that links Titles to Authors by the ISBN number of the book. You'll need to join the Titles and the Title Author tables on this ISBN field to select the proper records from the Titles table.

In the Data1_Reposition event, add this code to create a SQL statement that selects all the Titles for this Author. You can get the current Au_ID by referencing the appropriate text-box control or you can get it directly from the Fields collection of the Data1 record set:

```
SQL = "Select * from Titles, [Title Author] "
SQL = SQL & " where Titles.ISBN = [Title Author].ISBN "
SQL = SQL & " and [Title Author].Au_ID = "
SQL = SQL & Data1.Recordset.Fields("Au_ID")
Data2.RecordSource = SQL
Data2.Refresh
```

After you add this code and run the project, notice that you can edit both the Author information on the main form and the Title information in the DBGrid. You can scroll the DBGrid control horizontally and see all the fields in the joined tables, including the linking ISBN fields.

You can see how easy Visual Basic makes it to create your own data-entry forms. The code discussed in this column, contained in a file called GS0796.ZIP, is available on FTP's online sites as well as on the VBCD. (For details, see "How To Reach Us" in Letters to the Editor.) ×