Chapter 1

The Big Picture: Visual Basic's Database Features

In This Chapter

- Sampling Visual Basic's most important database features
- Connecting an application to a database in less than five minutes

Visual Basic 6 includes many tools to help you create, revise, manage, and otherwise deal with databases efficiently. This book shows you how easy it is to use those tools, and this chapter introduces the main tools. At the end of this chapter, you find a quick example. It should prove to you beyond any doubt that in choosing Visual Basic to do your database programming, you made a very wise choice, indeed. You connect an application to a database in 17 swift steps.

This chapter does *not* attempt, though, to give you all the main ideas — the terms and concepts — of database programming. Chapter 2 attempts that. So if you come across a word or two, or a concept, that's unclear in this chapter, take a look at Chapter 2 or this book's index for additional explanation and examples.

Checking Out the Database Tools in Visual Basic 6

Visual Basic 5 and earlier versions had some database facilities. But a lot was missing, too. You often had to open Access or some other database development tool to accomplish some common database programming tasks, such as designing and testing a SQL query. (Here's an example of a SQL query: `SELECT TOP 5 PERCENT * FROM tblSales`. This query means "Give me the upper five percent of sales." For a detailed description of SQL, see Chapters 18 and 19.)

Now, with VB 6, you can count on finding most everything you need right there in Visual Basic. Here are some of the highlights — the powerful database features you find in Visual Basic 6:

- **ADO (ActiveX Data Objects):** The main element of Microsoft's new *universal database access* (UDA) strategy. It's *universal* because it's eventually supposed to extract data from e-mail files, ordinary text (TXT) files, and other non-traditional data sources.
- **The ADO Data Control:** A VB control that employs the new ADO database technology. Eventually, the ADO Data Control will replace the DAO Data Control that's been on the VB Toolbox for years now.

  For extensive examples of ADO, DAO, and other Microsoft database technologies, see Chapters 14 through 16.
- **The Data View window:** A great tool when you want to manipulate the structure of a database. (The *structure* of a database is its internal organization — what categories it's divided into. It's similar to the structure of a CD collection. If you are a seriously neat person, you may arrange the CDs into various categories, somewhat analogous to the *tables* used to organize data in a database. See Chapter 2 for explanations of the major database terms.) With the Data View window, you can create and modify database diagrams, tables, views, stored procedures, or triggers. (The Data View window even permits you to drag a table and drop it into the Query Designer for instant database connection. What could be easier?)
- **The Query Designer:** A quick way to define and test SQL queries by using drop-down lists, check boxes, and other shortcuts. If you want, you can save the queries in your database, speeding up execution. Figure 1-1 shows the Query Designer, along with the Data View window.
- **The DataRepeater Control:** A container that repeats custom controls. In other words, you define a custom set of components (for example, six TextBoxes plus six Labels to display records that contain six fields). This set of components is your UserControl. Then, you place this UserControl on the DataRepeater, which enables users to scroll through a huge, ListBox-like group of these UserControls. It's one good way to display multiple records in an easy-to-navigate format, as shown in Figure 1-2.
- **The DataReport Designer:** A great new utility for those who have to create reports, this surprisingly easy-to-use tool generates reports of sufficient complexity for many tasks. In addition to showing your reports on-screen or printing them, you can even send the resulting reports to HTML files (.HTM) for display in Web pages.
- **The Data Environment Designer:** A kind of all-purpose pipeline for connecting a VB project to a database. A similar tool, the UserConnection Designer, was available in previous versions of VB. The Data Environment Designer offers everything the UserConnection Designer did, but goes beyond it. For example, the Data Environment Designer permits drag and drop: You can simply drag a table from the Data

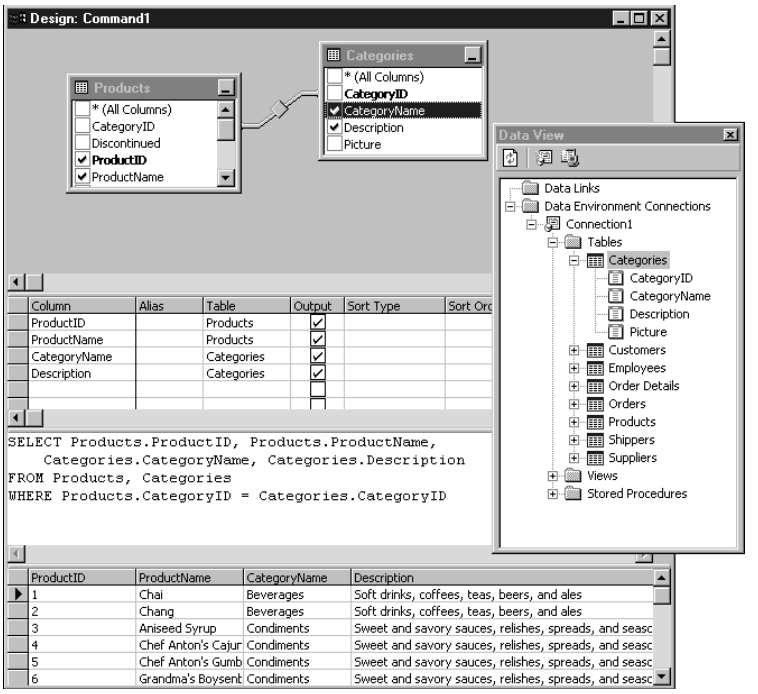

Figure 1-1: Use the Query Designer together with the Data View window to define and test SQL queries the easy way.

Environment Designer (see Chapter 4) and drop it onto a VB form. Automatically, the form fills with the right amount of Labels and TextBoxes to show the data. Each TextBox is also automatically bound to your DataEnvironment (and thus to a database).

The Data Environment Designer also permits multiple data connections. It can handle new OLE DB data sources (along with the older ODBC sources), it can be directly bound to controls, and you can manipulate it via programming. You can find lots of examples that show off this great tool in Chapter 4.

- **The ActiveX Document Migration Wizard:** A tool that makes quick, semi-automatic work of translating any of your oldie (but goodie) VB applications and utilities so you can plug them into Web pages. As we all somehow sense, the *Windows* metaphor for computing is not-so-gradually being replaced by a *browser* metaphor: Forward and Back buttons, Favorites and History lists, pages divided into frames, and all the other elements of a browser. If you're asked to add some functionality to a Web site, the ActiveX Document Migration Wizard is one of the fastest ways to get that job done (see Chapter 11).
- **The WebClass Designer (choose File➪New Project and then select IIS Application):** A designer that helps you create programming that runs on a server and can considerably beef up your Web pages. With IIS (Internet Information Server) applications, you can use classes,

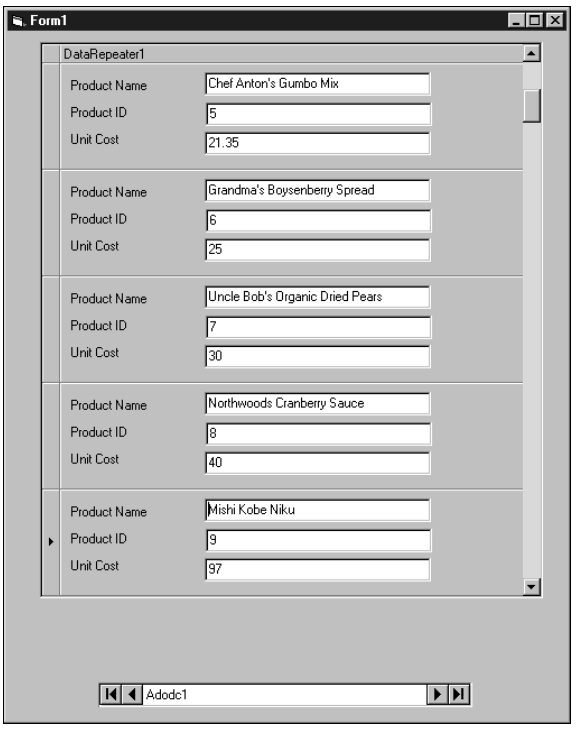

Figure 1-2: The DataRepeater offers the user a scrollable stack of multiple records.

compiled VB code, and ActiveX components. You can exploit the IIS object model, gaining extra control over the contents and behavior of Web pages you send to visitors.

The WebClass technique is superior to writing script for several reasons. For one thing, your coding is not mixed right in with the HTML, so you can keep it from prying eyes. The code is further concealed because it is compiled and it never leaves your server anyway. IIS applications also employ the ASP (Active Server Pages) object model, but VB automatically generates the necessary wrappers for you. Yet another victory of a VB Designer over the tedium of having to write all the programming by hand. See Chapter 13 for all the details about this designer.

- **The Visual Data Manager:** A tool that makes designing a new database, or manipulating an existing one, quite easy. You simply choose Add-Ins➪ Visual Data Manager, and you're a few steps away from defining the organization of your database. You can't define relationships in the Visual Data Manager or add other complexities, but you can get a good start and then later, if you decide to, add programming code or use Access to make further adjustments. (See Chapter 18 for details on relationships in database programming.)

Get Your Hands On This: Five Minutes Tops

The preceding section describes the main database-related features of Visual Basic, but this chapter wouldn't be complete without a hands-on example you can try. This book is filled with step-by-step examples showing you how to make good use of all the tools and technologies I describe in the preceding section. And the following example demonstrates how *easy* VB often makes tasks that would otherwise take days of learning and programming.

So sample this example and prove to yourself that VB is often the shortest distance between problem and solution.

In all likelihood, no faster way exists in any programming language to connect a database to an application than to use one of Visual Basic's Data Controls. To see how fast (five minutes tops) you can get data out of a database and piped into an application so the user can see it on-screen, follow these steps:

1. **Start Visual Basic by clicking its icon on your desktop.**

   Depending on how you've specified VB's startup options, you may or may not see the New Project dialog box. If you don't see this dialog box, choose File➪New Project.

2. **Double-click Standard EXE in the New Project dialog box.**

   A typical Visual Basic project template appears, with a blank Form1 as the starting point for your work.

3. **Double-click the Data Control icon in the Toolbox.**

   A Data Control appears on Form1.

   The VB Data Control is a component on the VB Toolbox that makes it a snap to open a connection to a database (without having to write any programming). Chapter 3 offers examples that show how you work with this useful component.

   If the VB Toolbox is not visible, choose View➪Toolbox.

4. **Double-click the DatabaseName property in the Properties window.**

   The DatabaseName dialog box appears, showing the VB folder. By default, Microsoft Access-style databases (they have an .MDB extension) are listed. An .MDB file is one kind of database, and it's the style you use in this example.

If the Properties window isn't visible, press F4. Also, you should see two sample databases in your VB folder (BIBLIO.MDB and NWIND.MDB). If you don't see these files, you might have the Data Control's Connect property set to some other database type. The solution is to close the dialog box, select Access for the Connect property in the Properties window, and then repeat Step 4. The Connect property defines the *type* of database you want to use the Data Control with. There are various types of databases, just as there are various types of graphics files. Each type organizes data differently, so VB needs to know which kind of database you're using.

5. **Double-click BIBLIO.MDB in the dialog box.**

   The dialog box closes, and the DatabaseName property is set to point to BIBLIO.MDB. You have connected your Data Control to the BIBLIO.MDB database. You can't just "open" a database the way you open a graphics file. It's not that simple. Between your VB application and the database stands a *database engine* (a set of tools, rules, and behaviors). In this example, you are connecting to the database through the Microsoft Jet engine, which is used by both VB and Access. For details on database connections and engines, see Chapter 5.

If you still don't see BIBLIO.MDB in the dialog box, locate it by clicking the Windows Start button and then choosing Find➪Files or Folders. If you *still* can't find it, rerun Visual Basic's setup program and agree to install all templates and samples. The BIBLIO.MDB sample database is used extensively in this book, and it's on your VB CD.

6. **Stretch the Data Control so it's about 2 inches wide and position it at the bottom of Form1, as shown in Figure 1-3.**

7. **Double-click the TextBox icon on the Toolbox.**

   VB adds a TextBox to the form.

8. **Stretch the TextBox so it's about the same size as the Data Control and position it near the top of Form1.**

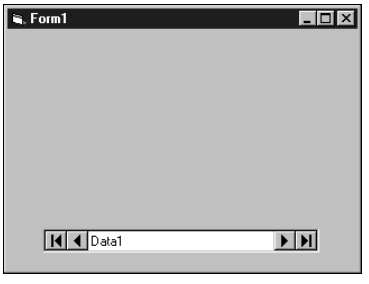

Figure 1-3: You, the happy programmer, can use a Data Control to connect your VB application to a database in no time.

9. **Click the TextBox on the form.**

   The TextBox is selected (and stretch tabs — tiny blue boxes — appear around it). The Properties window displays the TextBox properties. (If the Properties window isn't visible, press F4.)

10. **Attach the TextBox to the Data Control by double-clicking the TextBox's Datasource property in the Properties window.**

    The Datasource property changes to Data1, the name of the Data Control.

11. **Click the Data Control.**

    The Data Control's properties appear in the Properties window.

12. **Click the RecordSource property in the Properties window.**

13. **Click the drop-down arrow icon next to the RecordSource property.**

    A drop-down list appears, displaying all the tables available in the Biblio database that you're connected to, as you can see in Figure 1-4. (A *table* is the largest category in a database. For example, a database for a CD collection might be divided into these tables: Pop, Rock, Industrial, Metal, Rap, and Techno.)

14. **Click the Authors table to select it.**

    The recordset you chose now becomes the set of records that are displayed in the TextBox attached to this data control. A *recordset* is a custom list of data extracted from a database. For example, if you ask, "Let me see a list of all authors who live in Alaska," the result you get back is called a *recordset.* It's often a subset of all the data, though you can also request a recordset that includes *all* the data (every record) in a table. (See Chapter 2 for examples.)

15. **Click the TextBox to select it.**

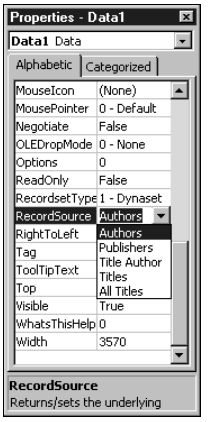

Figure 1-4: Visual Basic knows about, and shows you, all the tables in a database; all you do is click one.

16. **Keep double-clicking the TextBox's DataField property in the Properties window until *Author* appears.**

    You have now specified that the Author field (of the three available fields in the Authors table) will be the one displayed in the TextBox.

17. **Press F5.**

    Your application runs, and you see the first record in the BIBLIO.MDB database, as shown in Figure 1-5.

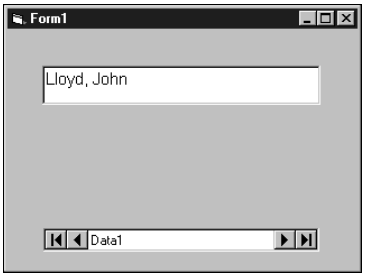

Figure 1-5: You did it! Your application displays the first record from the selected database.