

Visual Basic and Databases

3. Database Connection with the DAO Data Control

Review and Preview

- At this point, we have looked at databases and how they are structured. We have seen that the Jet database engine (part of Visual Basic) works between the database and the Visual Basic 'front-end' to manage the database. In this chapter, for the first time, we use Visual Basic to connect to a database. This connection is made with the DAO (data access object) data control. Using data bound controls, in conjunction with the DAO control, will allow us to view information in the database.

DAO or ADO – What's the Difference?

- With the introduction of Visual Basic 6, a new data control has emerged – the **ADO** (ActiveX Data Object) data control. Past versions of Visual Basic have used the **DAO** (Data Access Object) data control (also included with Visual Basic 6). So, Visual Basic 6 users (and Visual Basic 5 users, too) may be asking – what's the difference?
- Both controls do the same thing – provide a tool for working with the Jet database engine to perform database management. The controls just do it in different ways. **DAO** is the **most widespread** approach to database management and, as such, has a very large installed base of applications. **ADO** is an **emerging technology**, which will one day supplant the DAO control. Be aware, as an emerging technology, ADO still has bugs! As we work through this course, we will identify some of these bugs and figure out how to work around them.
- Why Use DAO Instead of ADO?
 - ⇒ You are modifying an existing application that uses DAO.
 - ⇒ You are developing a small desktop application.
 - ⇒ For now, the DAO control offers better database security features.
 - ⇒ Microsoft Access uses DAO, so the market will be there for a while.
- Why Use ADO Instead of DAO?
 - ⇒ You are beginning a new project, with potential for Internet deployment.
 - ⇒ ADO is easier to use than DAO.
 - ⇒ ADO is a more powerful control, allowing access to more data sources.
 - ⇒ If you don't use the Jet engine, ADO is the only way to go.
 - ⇒ ADO will eventually become the only data control, with DAO becoming obsolete.
- In this course, we will discuss the use of both controls. Obviously, only Visual Basic 6 users may use the material about the ADO data control. Visual Basic 5 users might like to look over the ADO material to become familiar with its use. In this particular chapter, we study the use of the DAO data control. The next chapter covers essentially the same material, using the ADO data control.

DAO Data Control

- The **DAO data control** is selected from the Visual Basic toolbox window. It's icon looks like this:



- The DAO data control is the primary interface between a Visual Basic application and a database. It can be used without writing any code at all! Or, it can be a central part of a complex database management system.
- The data control (or tool) can access databases created by other programs besides Visual Basic (or Microsoft Access). Some other formats supported include dBase, FoxPro, and Paradox.
- The data control can perform the following tasks:
 1. Connect to a database.
 2. Open a specified database table.
 3. Create a virtual table based on a database query.
 4. Pass database fields to other Visual Basic tools, for display or editing. Such tools are bound to the database, or **data bound controls**.
 5. Add new records, delete records, or update records.
 6. Trap any errors that may occur while accessing data.
 7. Close the database.
- As a rule, you need one data control for every database table, or virtual table, you need access to. One row of the table is accessible to each data control at any one time. This is referred to as the **current record**.

DAO Data Control Properties

- The DAO data control is connected to a database simply by setting a few properties. Important properties of this data control are:

Align	Determines where data control is displayed.
Caption	Phrase displayed on the data control.
Connect	Type of database. Default is Microsoft Access (or Jet).
DatabaseName	Returns or sets the name of the source database for the data control. Must be a fully qualified path and file name.
EditMode	Read-only at run-time. Indicates current state of editing for the current record.
Exclusive	Indicates whether the underlying database is opened for single-user or multi-user access.
ReadOnly	Indicates whether the data can be edited or not.
Recordset	A set of records defined by a data control's Connect, DatabaseName, and RecordSource properties. Run-time only.
RecordsetType	Indicates type of Recordset you want data control to create
RecordSource	Determines the table (or virtual table) the data control is attached to.
Visible	Establishes whether the data control appears on the form at run-time.

- When a DAO data control is placed on a form, it appears with the assigned caption and four arrow buttons:



The arrows are used to navigate through the table records (rows). As indicated, the buttons can be used to move to the beginning of the table, the end of the table, or from record to record. In most applications, the data control never appears on the form – its **Visible** property is almost always **False**. In this case, moving from record to record is handled programmatically, a topic discussed later in this chapter.

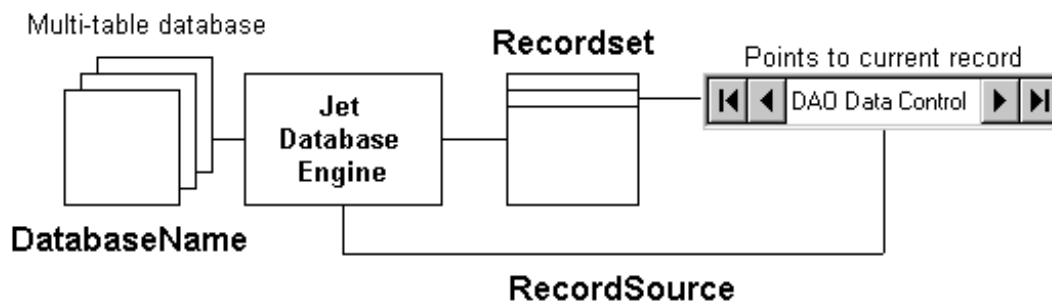
- After placing a DAO data control on a form, set the **DatabaseName** property first. Simply click on the ellipsis in the property box and choose the database. Then, set the **RecordSource** property. Click on that property's drop-down arrow and a list of valid tables will be presented. Choose the desired table (or type in a valid SQL statement - studied in Chapter 5). This establishes the **Recordset** object. By following these steps carefully, we avoid run-time errors associated with inability to find referenced data.

Recordset Object

- The **Recordset** object is an important concept. When we set the **RecordSource** property (either select a table from the database or form a virtual table via a query), the data control (using the Jet engine) retrieves the needed records and places them in the **Recordset** object for our use. We will see that this object has its own **properties** and **methods** for our use.
- There are three types of recordsets, established via the **RecordsetType** property:

Table	Representation of a native database table (not formed via a query). You can add, change, or delete records.
Dynaset	The default type, a Dynaset is formed as the result of a database query. You can add, change, or delete records from the underlying table(s). This is the most flexible Recordset type.
Snapshot	A static copy of records that cannot be updated. Used if you are just viewing or searching a database table.

- In summary, the relationship between the **data control**, its two primary properties (**DatabaseName** and **RecordSource**), and the **Recordset** object is:



Data Bound Controls

- The DAO data control allows us to easily connect to a database and form a Recordset. Yet, that control alone does not provide us with anyway to view the information in the database. To view the information, we use **data bound controls** that are special controls with properties established by database fields. A data bound control is needed for each field (column) in the Recordset (database table) you need to view. Most of the standard Visual Basic tools can be used as **data bound** controls.

- Standard data bound data controls are:

Label	Can be used to provide display-only access to a specified text data field. Caption property is data bound.
Text Box	Can be used to provide read/write access to a specified text data field. Probably, the most widely used data bound tool. Text property is data bound.
Check Box	Used to provide read/write access to a Boolean field. Value property is data bound.
Picture Box	Used to display a graphical image from a bitmap, icon, gif, jpeg, or metafile file. Provides read/write access to a image/binary data field. Picture property is data bound.
Image Box	Used to display a graphical image from a bitmap, icon, gif, jpeg, or metafile file (uses fewer resources than a picture box). Provides read/write access to a image/binary data field. Picture property is data bound.

- There are also three 'custom' data bound controls, the bound combo box, the bound list box, and the bound data grid tool. We will look at these later.

Data Bound Control Properties

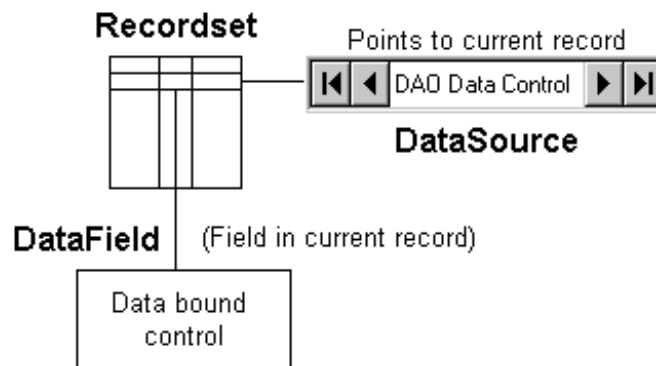
- To establish the connection of the data bound control to a database, we use a few properties:

DataChanged	Indicates whether a value displayed in a bound control has changed.
DataField	Specifies the name of a field in the table pointed to by the respective data control.
DataSource	Specifies which data control the control is bound to (indirectly specifying the database table).

- If the data in any data bound control is changed and the user moves to another record in the database, the database will **automatically** be **updated** with the new data (assuming it is not ReadOnly). Be aware of this - it is an extremely powerful feature of the data control, but also a potential source of problems.
- To make using bound controls easy, follow these steps (in order listed) in placing the controls on a form:
 1. Draw the bound control on the same form as the data control to which it will be bound.
 2. Set the **DataSource** property. Click on the drop-down arrow to list the data controls on your form. Choose one.
 3. Set the **DataField** property. Click on the drop-down arrow to list the fields associated with the selected data control records. Make your choice.
 4. Set all other properties, as needed.

Again, by following these steps in order, we avoid potential data access errors.

- The relationships between a data bound control (**DataSource** and **DataField** properties) and the DAO data control (**Recordset** property) are:



Example 3-1**Accessing the Books Database**

In this example, we begin working with the books (**BIBLIO.MDB**) database discussed in Chapter 2. This database is shipped with Visual Basic and is usually installed in the Visual Basic main directory. Using Windows Explorer, find this file. Make a copy of the database and place it in a working directory (you decide on a name – we use c:\vbdb\working) where you will build your applications. We do this to insure there is always a valid copy of BIBLIO.MDB on your computer. You will see that the power of the DAO control also opens up the possibility of doing damage to a database (we, of course, will try to minimize this possibility). So, we are just living by the adage, “Better safe, than sorry.”

1. After copying BIBLIO.MDB to your working directory, start a new application. We'll develop a form where we can look through the **Titles** table in the books database. Place a DAO data control, four label boxes, and four text boxes on the form.
2. Set the following properties for each control. For the data control and the four text boxes, make sure you set the properties in the order given.

Form1:

Name	frmTitles
BorderStyle	1-Fixed Single
Caption	Titles Database

Data1:

Name	datTitles
Caption	Titles
DatabaseName	BIBLIO.MDB (select from your working directory, don't type)
RecordSource	Titles (select, don't type)
ReadOnly	True (to protect us from ourselves, at the moment)

Label1:

Caption	Title
---------	-------

Label2:

Caption	Year Published
---------	----------------

Label3:

Caption	ISBN
---------	------

Label4:

Caption	Publisher ID
---------	--------------

Text1:

Name	txtTitle
DataSource	datTitles (select, don't type)
DataField	Title (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

Text2:

Name	txtYearPublished
DataSource	datTitles (select, don't type)
DataField	Year Published (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

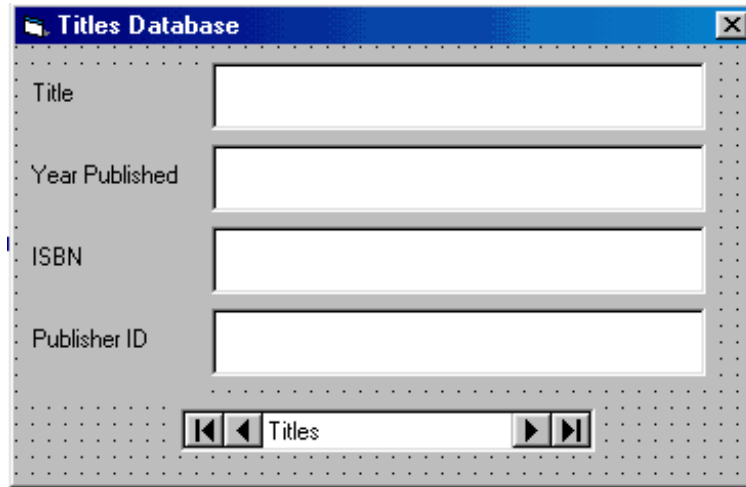
Text3:

Name	txtISBN
DataSource	datTitles (select, don't type)
DataField	ISBN (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

Text4:

Name	txtPubID
DataSource	datTitles (select, don't type)
DataField	PubID (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

When done, the form will look something like this:

The image shows a screenshot of a Visual Basic application window titled "Titles Database". The window has a blue title bar with a standard Windows close button in the top right corner. The main area of the window is light gray and contains four text input fields stacked vertically. To the left of each text field is a label: "Title", "Year Published", "ISBN", and "Publisher ID". At the bottom of the window, there is a data control component. It consists of a small rectangular box with a label "Titles" in the center. To the left of the label are two small square buttons with left-pointing arrows, and to the right are two small square buttons with right-pointing arrows, used for navigating through the data.

3. Save the application. Run the application. Cycle through the various titles using the data control. Did you notice something? You didn't have to write one line of Visual Basic code! This indicates the power behind the data control and data bound controls.

There's one last thing. If you load this example from the code accompanying the course, you will need to reset the data control's **DatabaseName** property, pointing to the directory in which you have stored the books database on your computer. In fact, you will have to do this anytime you use the examples provided with the course.

DAO Data Control Events

- Like other controls, the DAO data control has **events** that are triggered at various times during database access. In these events, we write BASIC code to perform specific needed tasks. In this chapter, we will not be using these event procedures, but we will define them to make our definition of the data control complete.
- Important DAO data control events:

Error	Triggered when a data access error occurs and Visual Basic code is not being executed.
Reposition	Triggered after data control pointer moves to a new record. Use to update information from non-data bound controls.
Validate	Event triggered when the pointer is about to move away from the current record. This event can be used to cancel an update of a record or a move to a new record.
- These events will be discussed further when we begin development of database management techniques in a later chapter.

DAO Data Control Methods

- To complete our definition of the DAO data control, we present some important methods. These methods perform certain actions on the data control:

Refresh	Requeries the database based on contents of the RecordSource property.
UpdateControls	Restores the value of bound controls to original values (if no update has been performed).
UpdateRecord	Saves the values of bound controls to the database without triggering the data control Validate event.
- Like events, DAO data control methods will be discussed further when we begin development of database management techniques in a later chapter.

DAO Data Control Recordset Properties

- The **Recordset** object of the data control has its own set of **properties**. These properties can only be accessed at **run-time**. To refer to a Recordset property, use a 'double-dot' notation. For example, if you have a data control named **datExample**, to refer to a property named **PropertyName**, the notation is:

datExample.Recordset.PropertyName

- Important data control Recordset properties are:

AbsolutePosition	Long integer that either gets or sets the position of the current record.
BOF	Returns True when the current record is positioned before any data.
Bookmark	Sets or returns a bookmark to the current record. Used as a place marker in database management tasks.
EditMode	Indicates the state of editing for the current record.
EOF	Returns True when the current record is positioned past any data.
PercentPosition	Single data type that sets or gets the position of the current record as a percentage of total records. Used for status indicators.
RecordCount	The total number of records in the Recordset.
Updatable	Read-only at run-time. If True, records in the Recordset can be modified. If False, records are read-only.

- We will look at the BOF and EOF properties in the section on Recordset Navigation. Other properties will be examined later in this course.

DAO Data Control Recordset Methods

- The data control **Recordset** also has its own set of **methods** that perform functions on the Recordset. These methods are invoked using the double-dot notation introduced for the Recordset properties. So, for a data control (**datExample**) and method (**MethodName**), you invoke the method via:

datExample.Recordset.MethodName

- Important Recordset methods are:

AddNew	Adds a new record to the Recordset. All fields are set to null and this record becomes the current record.
CancelUpdate	Used to cancel any pending updates (either with Edit or AddNew method)
Close	Closes a Recordset.
Delete	The current record is deleted from the Recordset.
Edit	Places the current record in the Recordset into edit mode.
MoveFirst	Moves the current record pointer to the first record in the Recordset.
MoveLast	Moves the current record pointer to the last record in the Recordset.
MoveNext	Moves the current record pointer to the next record in the Recordset.
MovePrevious	Moves the current record pointer to the previous record in the Recordset.
Requery	Updates the data in a Recordset object by re-executing the query on which the object is based.
Update	Saves the current contents of all data bound controls.

- We will look at the four 'Move' methods in the next section on Recordset Navigation. Other properties will be reviewed later in this course.

DAO Data Control Recordset Navigation

- We have seen that, on the form, the DAO data control has four arrows that allow the user to move to the first, next, previous, and last records in the recordset. Unfortunately, this control does not have a familiar look to a user and it may not be clear just exactly what functions the arrows perform. For this reason, we usually set the data control's **Visible** property to **False** and provide a programmatic approach to moving from record to record, or **recordset navigation**.
- Four Recordset methods replicate the capabilities of the arrow buttons on the data control: **MoveFirst**, **MoveNext**, **MovePrevious**, and **MoveLast**. For each function we need, a command button is added to the form, with a **Click** event procedure attached to the corresponding Recordset method.
- When programmatically navigating through the records, you need to be aware of the position of the current record. For example, if you are at the first record and try a **MovePrevious** method, you will move past the beginning of the file. You can use the **BOF** property to see you are at the beginning of file and disallow such a move. You need a similar check at the end of a file to disallow an invalid **MoveNext** method.

Quick Example 1 - Recordset Navigation

1. Load the project built in Example 3-1. Set the data control's **Visible** property to **False**. Add two command buttons with the following properties:

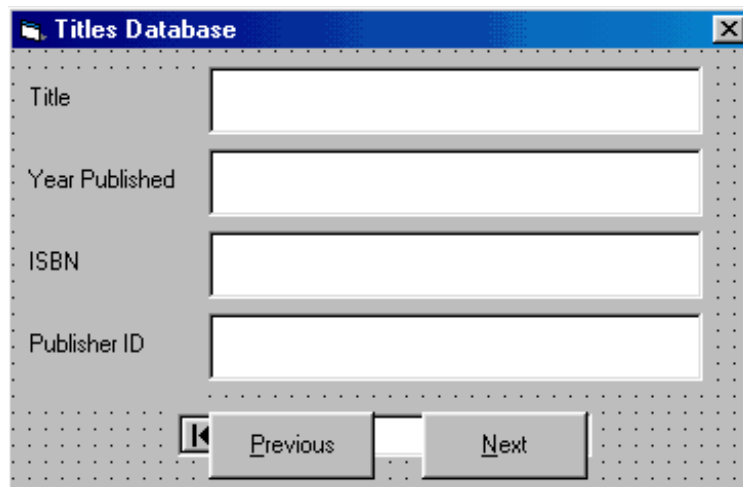
Command1:

Name	cmdPrevious
Caption	&Previous

Command2:

Name	cmdNext
Caption	&Next

The form should look like this (notice the data control is still there under the command buttons- it will only disappear at run-time).



2. Place this code in the **cmdPrevious_Click** event:

```
Private Sub cmdPrevious_Click()  
    datTitles.Recordset.MovePrevious  
    If datTitles.Recordset.BOF Then  
        datTitles.Recordset.MoveFirst  
    End If  
End Sub
```

3. Place this code in the **cmdNext_Click** event:

```
Private Sub cmdNext_Click()  
    datTitles.Recordset.MoveNext  
    If datTitles.Recordset.EOF Then  
        datTitles.Recordset.MoveLast  
    End If  
End Sub
```

4. Save and run the application. Make sure the newly added buttons work as they should. Try adding buttons to move to the first and last records. Can you write the code?

Summary

- In this chapter, we finally used Visual Basic to connect to an actual database. We learned there are two data controls available for database connection. The DAO data control (shipped with both Visual Basic 5 and Visual Basic 6) was discussed in this chapter. The same material is covered in the next chapter using the new ADO data control (Visual Basic 6 only).
- The DAO data control made connecting to a database a simple process. Only a few properties (**DatabaseName** and **RecordSource**) needed to be set at design-time to form the **Recordset** for viewing.
- Data bound controls were seen to be our 'window' into the fields representing the data control Recordset. Controls are bound by setting two properties: **DataSource** and **DataField**. Yet, even after all this work, all we can do right now is view databases, which in some applications is sufficient (think of your local library - they certainly don't want patrons changing information in their database). To build a complete database management system, we need to know SQL, the powerful language behind database queries. This is discussed following a look at the ADO data control.
- If you're feeling overwhelmed by all the material presented herein, don't worry - you'll see it many more times as you continue through this course and become a more proficient database programmer.

Exercise 3**Northwind Traders Database**

A second sample database is included with Visual Basic 5 and Visual Basic 6. It is a database (**NWIND.MDB**) used by a fictional company (**Northwind Traders**) to handle its commerce. It has eight tables. In this exercise, we repeat the tasks of Example 3-1, using one table (**Customers**) in this database. This and the next exercise give you further practice in using the DAO data control and data bound controls and allows you to study the structure of another database. In working with databases, more examples to study help make you a better database programmer.

1. Copy NWIND.MDB to your working directory and start a new application. We'll develop a form where we can look through the **Customers** table in the Northwind Traders database. Place a DAO data control, four label boxes, and four text boxes on the form.
2. Set the following properties for each control. For the data control and the four text boxes, make sure you set the properties in the order given.

Form1:

Name	frmCustomers
BorderStyle	1-Fixed Single
Caption	Customers Database

Data1:

Name	datCustomers
Caption	Customers
DatabaseName	NWIND.MDB (select from your working directory, don't type)
RecordSource	Customers (select, don't type)
ReadOnly	True (to protect us from ourselves, at the moment)

Label1:

Caption	Customer ID
---------	-------------

Label2:

Caption	Company Name
---------	--------------

Label3:

Caption	Contact Name
---------	--------------

Label4:

Caption	Contact Title
---------	---------------

Text1:

Name	txtCustomerID
DataSource	datCustomers (select, don't type)
DataField	CustomerID (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

Text2:

Name	txtCompanyName
DataSource	datCustomers (select, don't type)
DataField	CompanyName (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

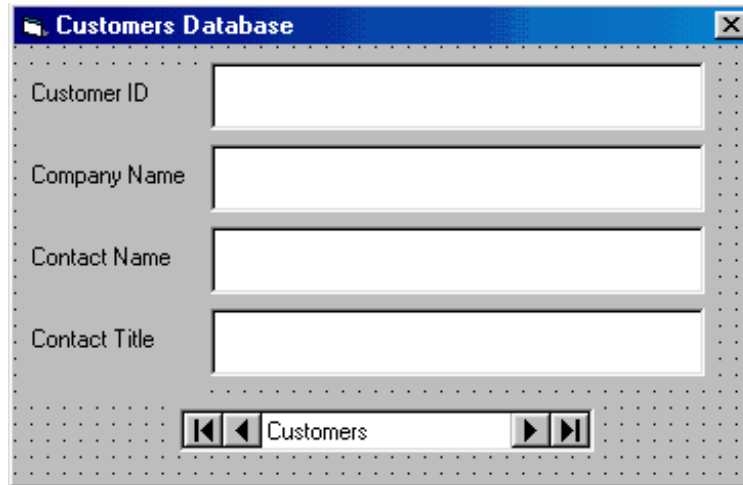
Text3:

Name	txtContactName
DataSource	datCustomers (select, don't type)
DataField	ContactName (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

Text4:

Name	txtContactID
DataSource	datCustomers (select, don't type)
DataField	ContactTitle (select, don't type)
Locked	True
MultiLine	True
Text	[Blank]

When done, the form will look something like this:



The screenshot shows a Visual Basic form titled "Customers Database". The form has a blue title bar with a close button. It contains four text boxes labeled "Customer ID", "Company Name", "Contact Name", and "Contact Title". At the bottom, there is a data control with a list box showing "Customers" and navigation buttons (back, forward, first, last).

3. Save the application. Run the application. Cycle through the various customers using the data control.

This page intentionally not left blank.