

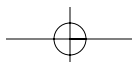
CHAPTER

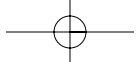
ONE

1

Automating a Database without Programming

- Using the Database Wizard
- Automating operations with the Command Button Wizard
- Navigating with hyperlinks
- Creating a navigation control center with the Switchboard Manager
- Controlling the user interface
- Creating custom menus and toolbars





A *database* is a collection of records and files. To create a database, you need a system that will help you to store, retrieve, and sort your data, as well as analyze and convert it into useful information. If the database is large or complex, you'll probably want to use a commercial computer database application such as Microsoft Access.

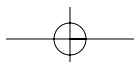
Access has a terrific set of tools and wizards to help you create a database, including *tables* to store data, *queries* to retrieve and manipulate data, *forms* to enter and view data, *data access pages* to view and work with data from the Internet or an intranet, and *reports* to print information. But if you stop at this point, you'll have taken advantage of only a fraction of the power that Access offers; you'll have used only five of the seven database container objects, having left *macros* and *modules* untouched.

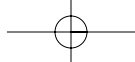
Without macros and modules, a database is *interactive*. In an interactive database, the user initiates each individual action the computer carries out by choosing a menu command or by clicking a toolbar button. The user is the one who supplies the connections between the forms and reports in the database. In order to perform tasks, the user needs to know which menu commands to use and which sequence to use them in, as well as how the forms and reports are related. In an interactive database, the user has complete control. A knowledgeable user has the power to use the interactive database in productive ways. A less sophisticated user has the power to corrupt the data and damage the database by selecting the wrong command at the wrong time.

In this book, you'll learn how to transform your interactive database into an *automated database application*. A well-designed, fully automated database application can be used by any user. The user doesn't need to know the sequence of steps for a task or the Access commands. The user needs only to click a single button to execute a complicated task.

When you create a fully automated application, you create a custom user interface. The *user interface* is what users see on the screen and how they use the keyboard and mouse to communicate with the computer. In the custom application's user interface, the user clicks command buttons to move between tasks, perform data-entry operations, find records, and print reports. The custom user interface is where the user lives in your database application. From the user's perspective, the custom user interface *is* your database application.

When creating the new interface, you should supply the tools to open forms, perform data entry, locate specific records or groups of records, import data, archive





old records, and print reports. You should also provide a choice of paths for navigating through your database, making sure that users always know where they are and how to backtrack along the path.

Access provides a set of wizards and helpers to assist you with some of the automation. This chapter introduces you to the Database Wizard for creating the first draft of a complete application, the Command Button Wizard and Combo Box Wizard for creating automated command buttons and combo boxes, and the Switchboard Manager for creating road maps to the forms and reports in the application.

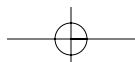
This chapter also shows you how to use hyperlink techniques from Internet technology to navigate between database objects. You'll learn how to use hyperlinks to navigate directly from a form in your application to any document in your computer's file system or in any other computer that is connected to your computer using the Internet's TCP/IP network.

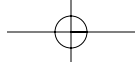
At each stage of user interface construction, your goals are to build in ease of use, intuitive understanding, and protection of the application. This chapter shows you how to create custom menus and toolbars so your application provides only the tools and commands that a user needs. You'll learn how to protect your application with a password and how to set startup conditions so a user who survives the password test is greeted by your application's startup form and its custom menus and toolbars. This chapter ends with a preview of VBA (Visual Basic for Applications), the powerful programming language used in Access, and gives you a glimpse of the additional power you'll have when you learn to use it.

Using the Database Wizard

The Database Wizard can help you to create database applications for a number of different business and personal scenarios. Some of these are as follows:

Asset Tracking	Ledger
Contact Management	Order Entry
Event Management	Resource Scheduling
Expenses	Service Call Management
Inventory Control	Time and Billing





Once you identify the scenario that is closest to the application you want to create, the wizard, in the usual wizard style, displays a series of screens telling you about the application and soliciting your input. After collecting your choices, the wizard uses the template you selected to create and customize the necessary tables, queries, forms, reports, data access pages, and modules.

NOTE

Data access pages are a new feature in Access 2000 that allow users to extend database applications to the corporate intranet by creating data-bound HTML pages quickly and easily. This helps users share information faster and more efficiently.

The Database Wizard is able to create both simple and complex databases. Depending on the scenario you choose, the wizard may create several groups of tables. When there are pairs of tables in a many-to-many relationship, the wizard automatically resolves the relationship into a pair of one-to-many relationships by creating a relationship table. The wizard creates simple data-entry forms for each table and may even create a form/subform combination to display a one-to-many relationship. The wizard creates summary reports appropriate to the scenario you choose.

Creating Navigation Paths with Switchboards

After creating the individual data-entry forms and summary reports, the wizard automatically creates forms called *switchboards*, which provide navigation paths between groups of forms and reports. The wizard creates a Main Switchboard to serve as the control center for the application. The Main Switchboard has command buttons for each of the basic database tasks. Clicking a button on the Main Switchboard takes you to a form that you use to perform a database task, such as entering data into one of the tables. Clicking a button on the Main Switchboard may also take you to another switchboard with buttons that take you to other forms, reports, or other switchboards. Figure 1.1 illustrates switchboard navigation paths.

The buttons react when you click them because the wizard has created an individual set of instructions for each button. The wizard writes instructions and stores them in one of two places:

- In *standard modules* that are listed as separate objects in the Modules pane of the Database window
- In *form modules* and *report modules* that are built into the forms and reports (as part of the form or report definition), stored as part of the form listed in the Forms pane or the report listed in the Reports pane of the Database window

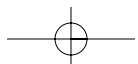
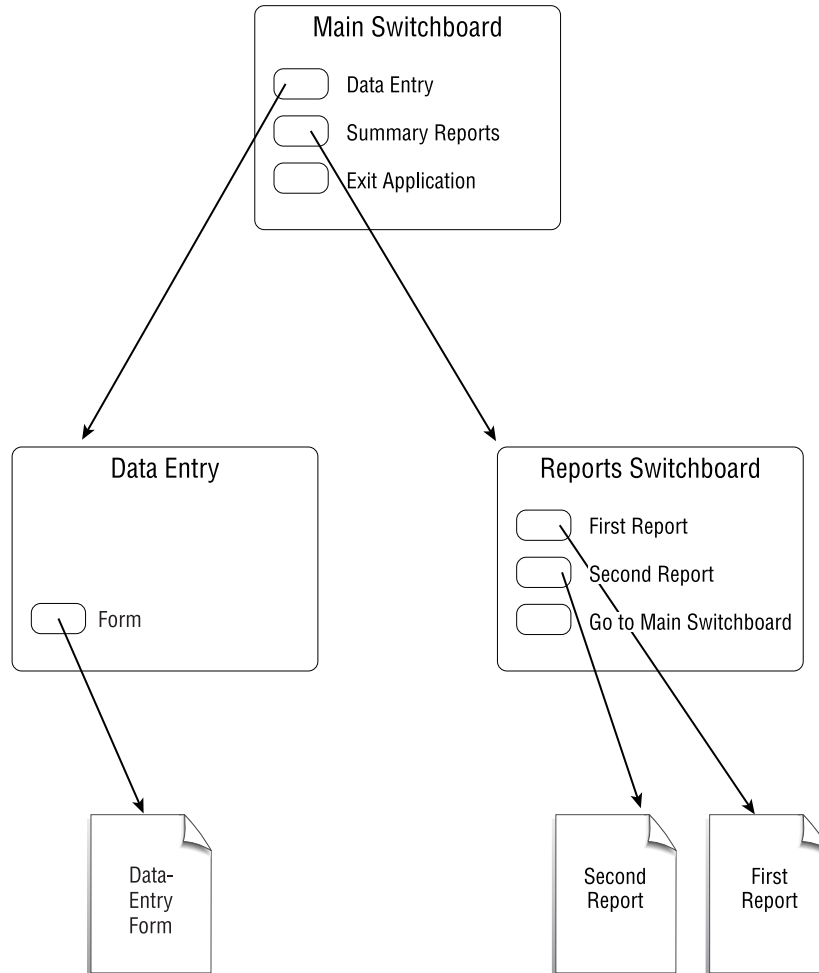


FIGURE 1.1:
The switchboard connection

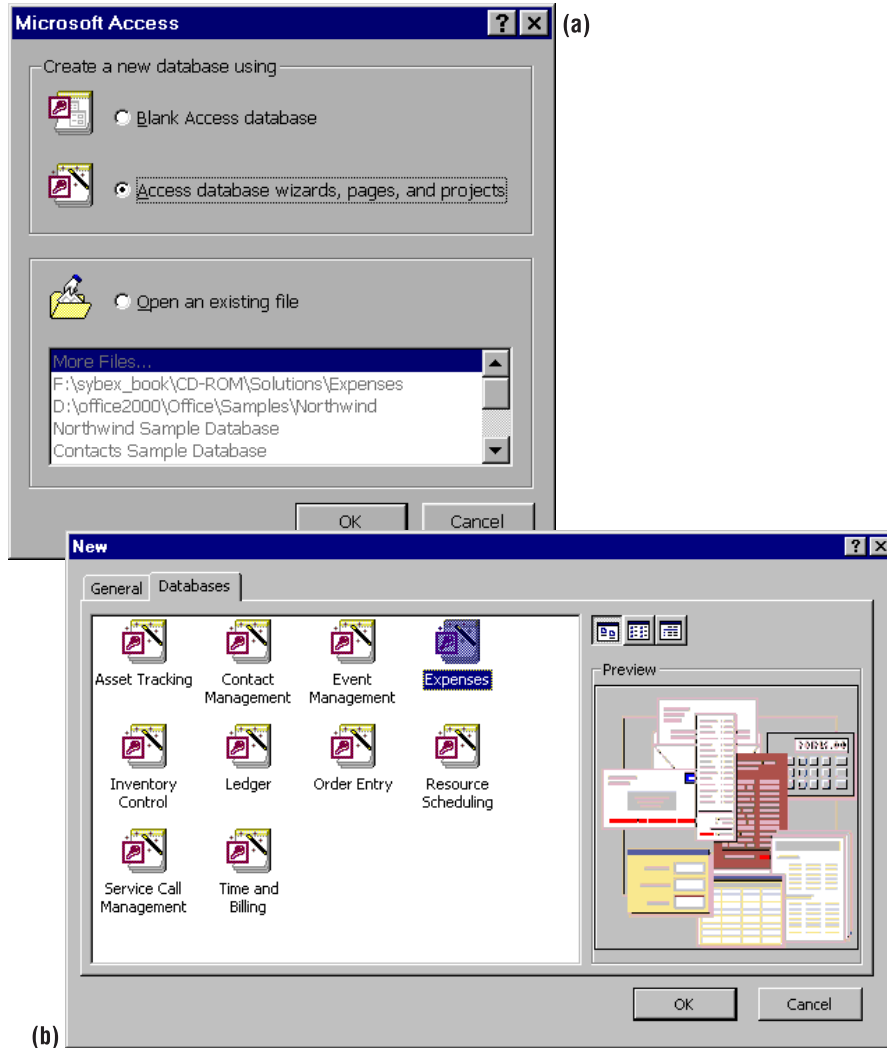


To observe the Database Wizard at work, we'll create an application for tracking employee expenses.

1. Start up Access 2000 and click the Access Database Wizards, Pages, and Projects radio button (see Figure 1.2a). Click OK, then click the Databases tab in the New dialog and choose Expenses as the template to use to create your new database (see Figure 1.2b).

FIGURE 1.2:

Summon the Database Wizard in the opening dialog (a) and select a template for the new database (b).



2. In the next dialog, enter **Expenses** as the name and save the database to the VBAHandbook folder (see Figure 1.3). (If you haven't created this folder, see the Introduction for instructions on setting it up.) Click the Create button to start the Database Wizard. The wizard's first screen explains the kinds of information the database will manage (see Figure 1.4).

FIGURE 1.3:

Name and save your database.

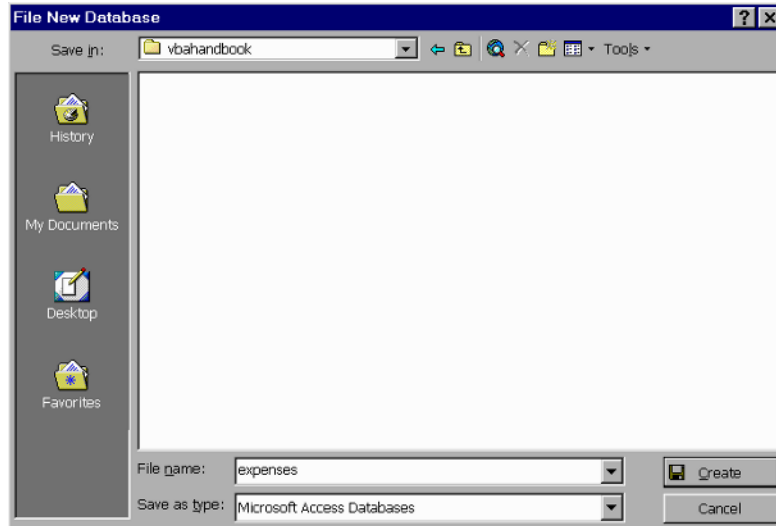
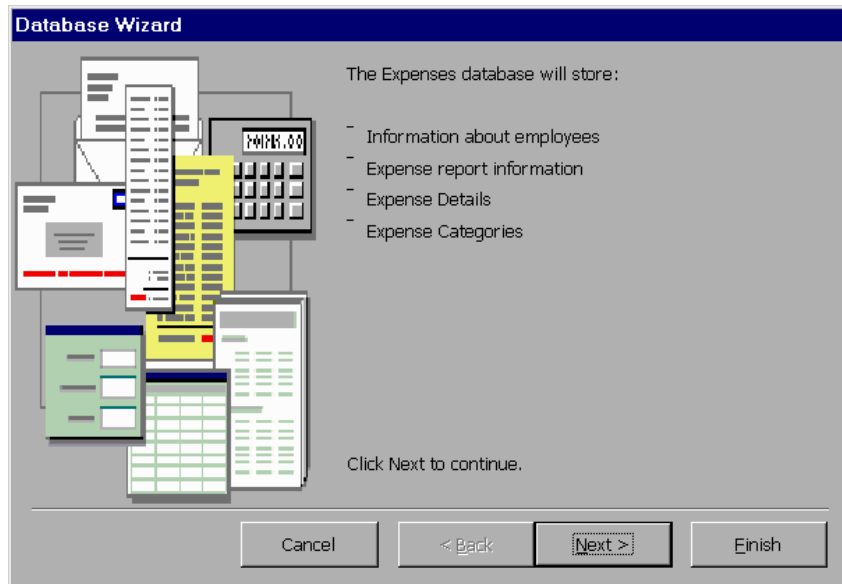


FIGURE 1.4:

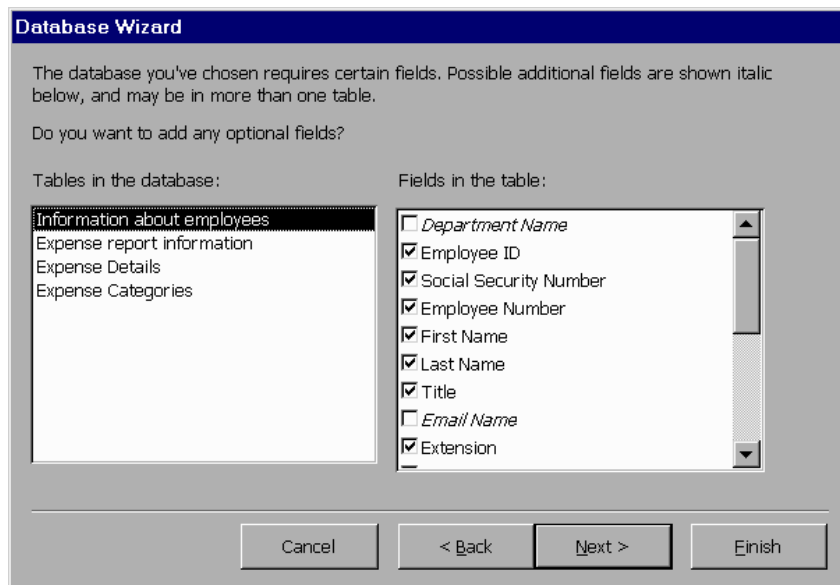
The Database Wizard explains the kinds of information the database will manage.



- The next screen gives you the opportunity to make minor changes in the database (see Figure 1.5). The list box on the left displays the tables to be created. When you click on a table, the list box on the right changes to display the fields for the selected table. You can't add new tables or delete tables from the list, but you can add the fields shown in italics. For each table, check the fields you want to add.

FIGURE 1.5:

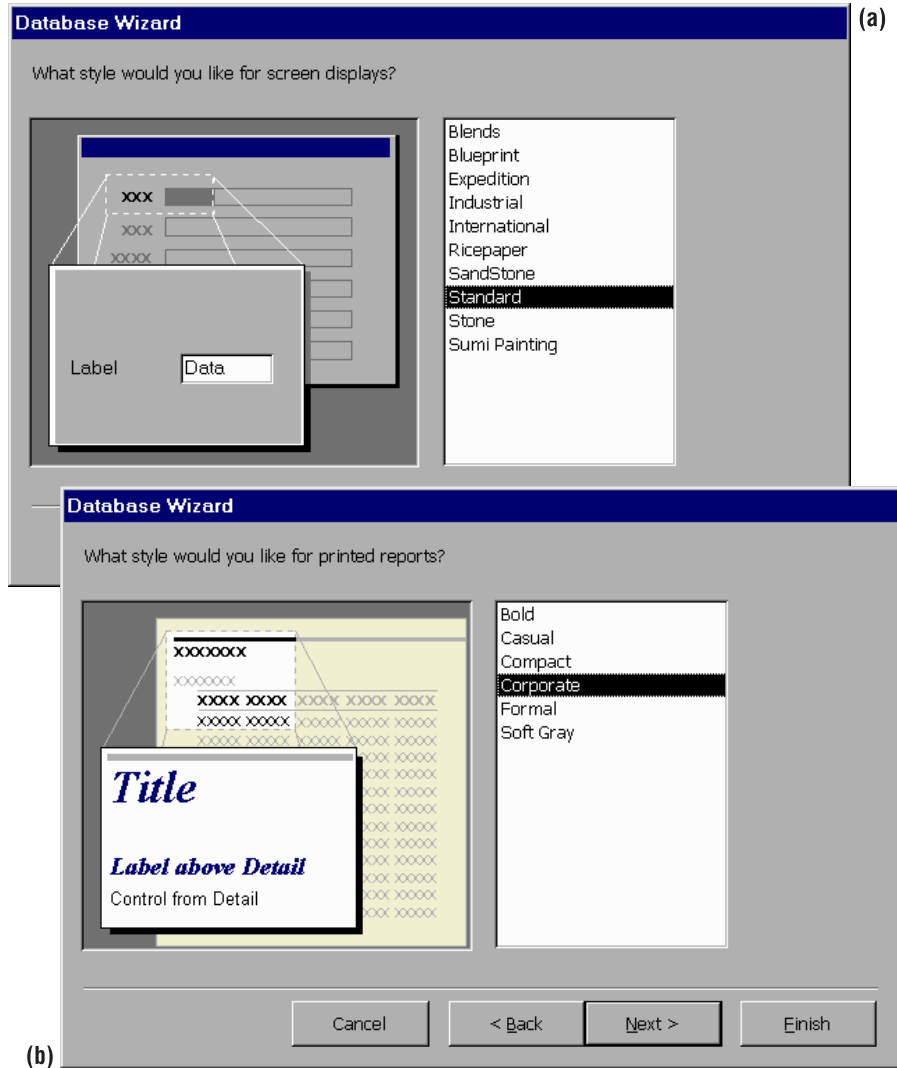
You can choose to add optional fields to some of the tables and populate the database with sample data.



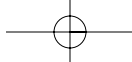
- Specify styles for the forms (see Figure 1.6a) and reports (see Figure 1.6b) in the next two screens.
- You can use the next screen to enter a title for the database and include a bitmap picture (see Figure 1.7a). If you add a picture, it will appear on reports that the wizard creates. In the final screen (see Figure 1.7b), you can select to start the database immediately after it is created and to display help. Clicking the Finish button puts the wizard to work. While the wizard toils, a dialog displays one progress meter showing the overall progress and another progress meter showing the progress in creating a specific object.

FIGURE 1.6:

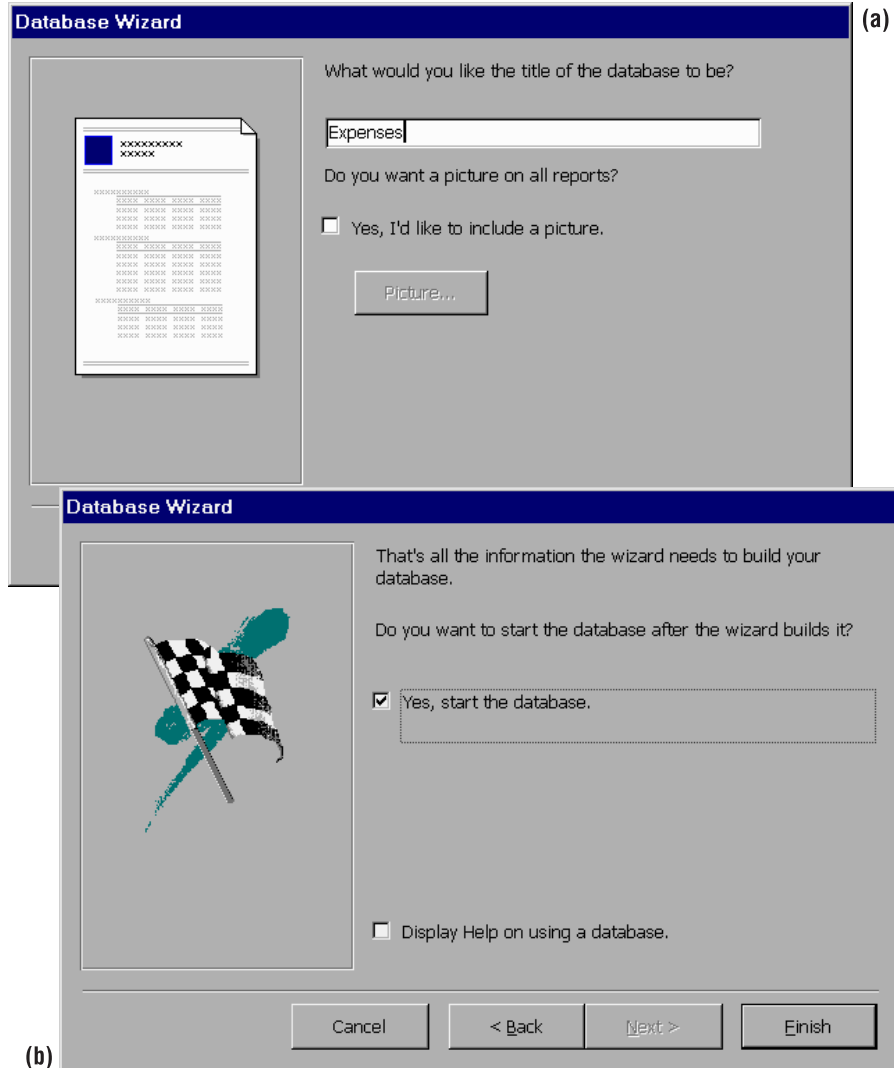
You can specify styles for forms (a) and reports (b).

**WARNING**

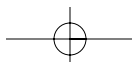
If you have not yet configured a printer on your computer, Access will generate an error indicating that it cannot properly create the database. In this instance, just select Start > Settings > Printers and double-click Add Printer. This will launch the Add Printer Wizard.

**FIGURE 1.7:**

Enter a new title and maybe a bitmap picture (a). After you click Finish in the final screen (b), the wizard starts to work.



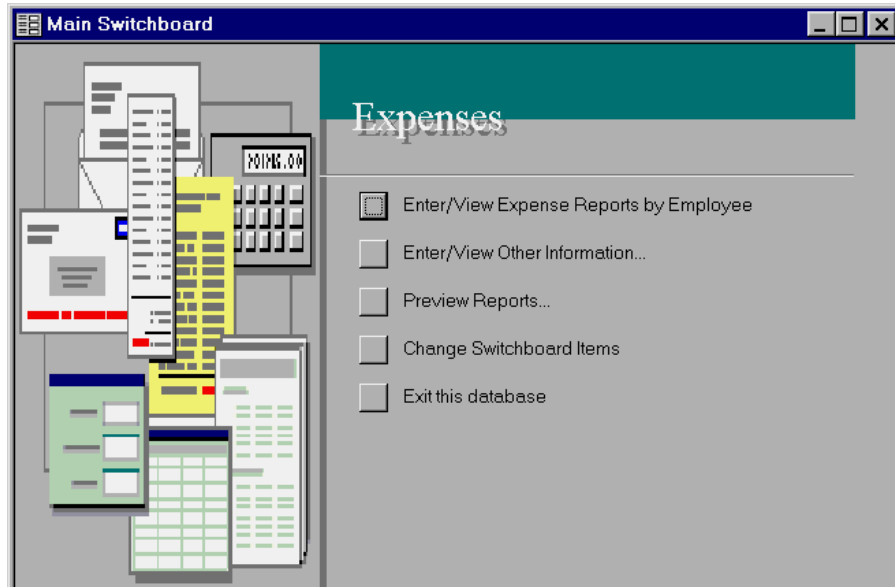
The wizard creates the tables and relationships, a form named Switchboard, and the forms and reports. The last text that flashes above the lower progress meter states that the wizard is setting database properties. When the job is



finished, the Main Switchboard is displayed (see Figure 1.8), and the Database window is minimized.

FIGURE 1.8:

The Main Switchboard is the central dispatch for the database application.



Exploring the Application

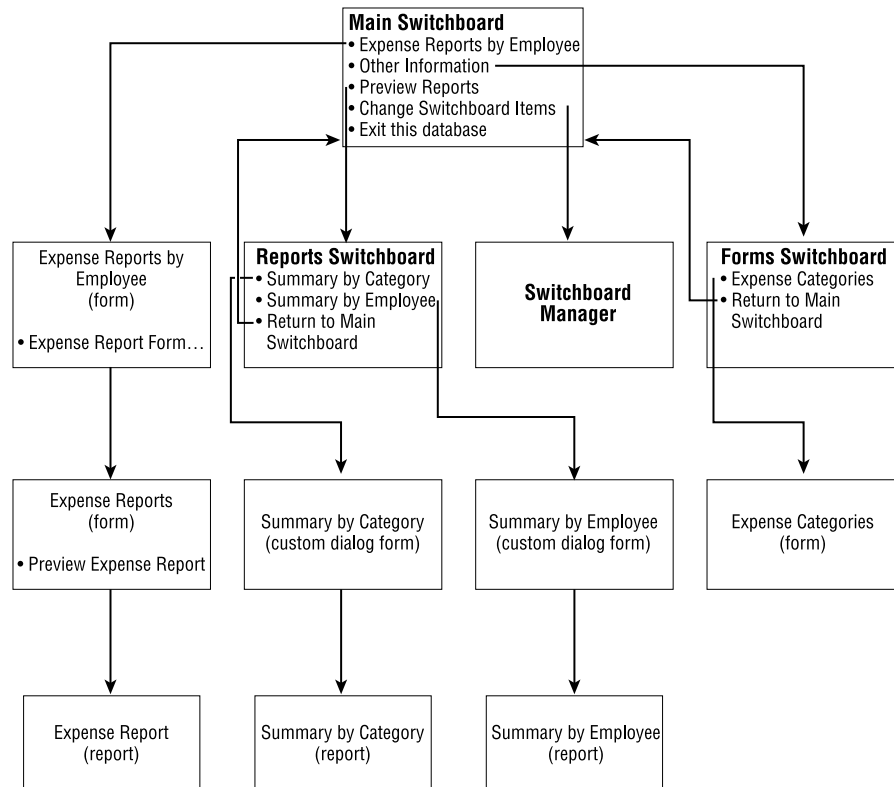
The Main Switchboard is the first step in navigating through the application. (You'll learn how to use the Main Switchboard as a startup form later in this chapter.) The Main Switchboard gives an immediate sense of the main tasks that the application manages. The command buttons direct you to forms for carrying out database tasks or to other switchboards that may branch to still more forms and switchboards. The last two buttons appear on every Main Switchboard that the Database Wizard creates. We'll explore the Change Switchboard Items button later in the chapter. Clicking the Exit this database button closes the database without exiting Access.

Take a few minutes to travel to other forms by clicking buttons. The buttons on the switchboards and forms provide navigational paths through the application to the various tasks. The overall organization emerges as shown in the task flow

diagram in Figure 1.9. The directional arrows in the task flow diagram indicate whether a form has a command button that takes you back to a previous form.

FIGURE 1.9:

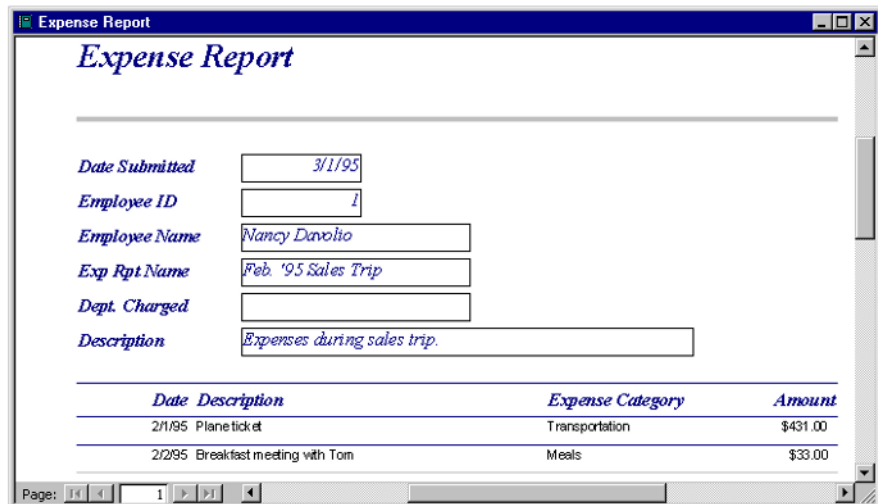
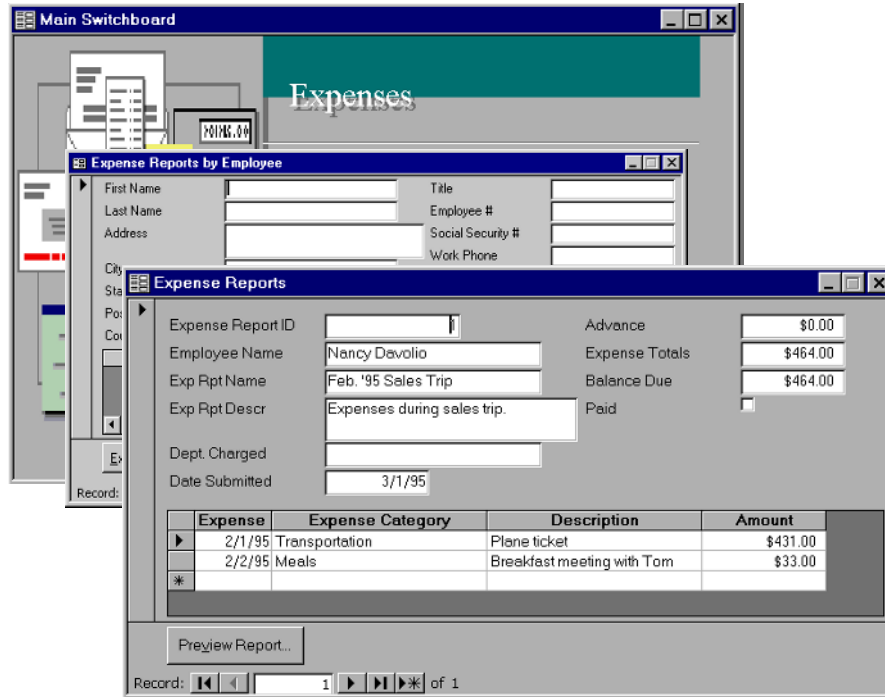
The task flow diagram for the Expenses application displays the underlying sequence, or flow, of the tasks.



The first button on the Main Switchboard takes you to the Expense Reports by Employee form, which has a button that opens the Expense Reports form. Clicking the Preview Report button on the Expense Reports form opens a preview of the Expense Report report. Clicking this sequence of buttons takes you along a one-way path from the Main Switchboard to the Expense Report report (see Figure 1.10). The path is one way because there are no command buttons to take you back to the Main Switchboard. (Of course, you can use a form's default Close button in its upper-right corner or choose the Close command from the File menu to close the form and return to the previous form.) Later, we'll make the navigation back to the previous form easier by adding command buttons to the forms.

FIGURE 1.10:

The one-way path from the Main Switchboard to the Expense Reports form to the Expense Report report

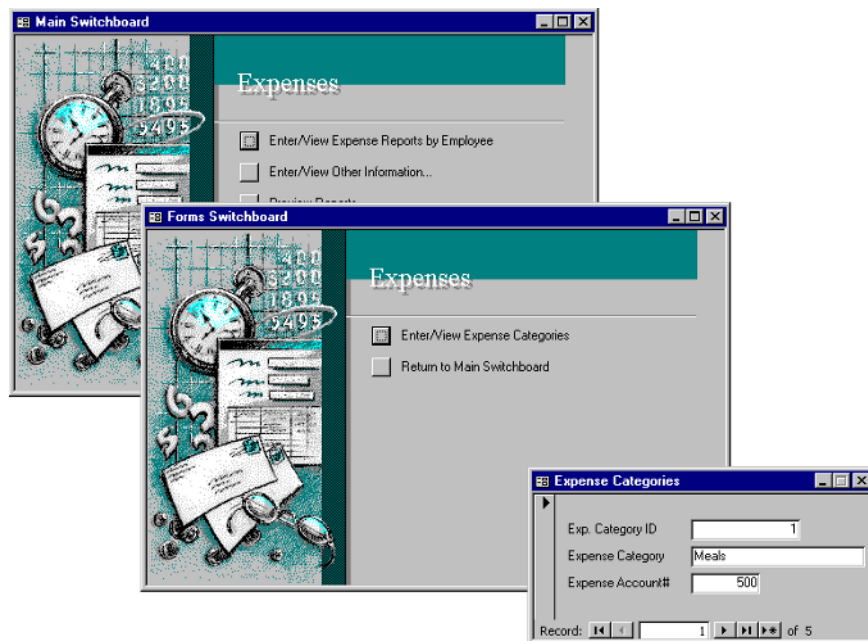


Notice that when you select an employee in the Expense Reports by Employee form and click the button on the form, the Expense Reports form opens with information for the same employee; that is, the opened form is *synchronized* with the form that opened it. If you click into the first form, move to a different employee, and then click back into the second form, you'll see that the two forms remain synchronized. Similarly, when you click the Preview Report button on the Expense Reports form, the report that opens is synchronized with the form that opened it.

The second button on the Main Switchboard takes you to another switchboard called the Forms Switchboard, where you have the choice of displaying the Expense Categories data-entry form or returning to the Main Switchboard (see Figure 1.11).

FIGURE 1.11:

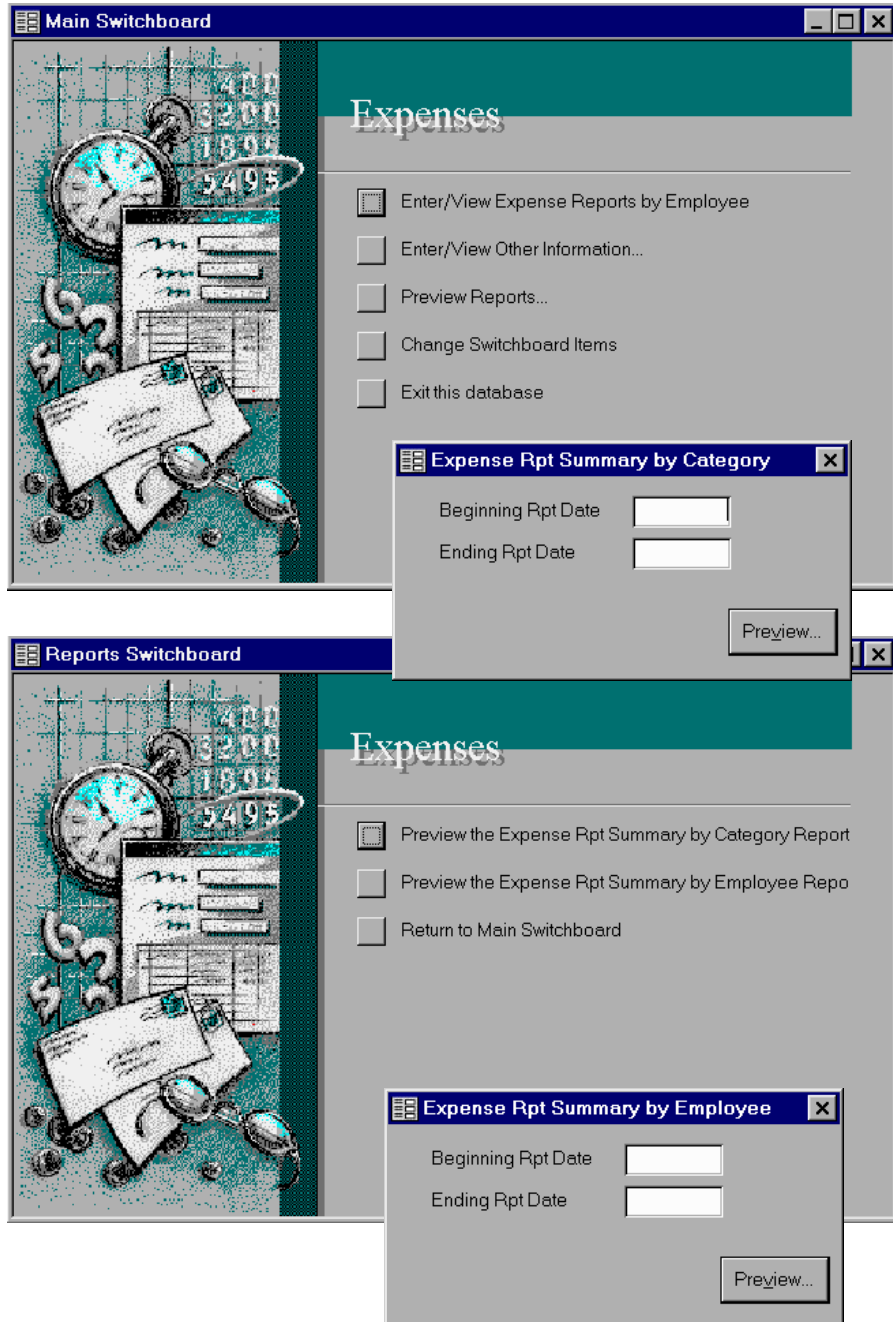
The path from the Main Switchboard to the Expense Categories form

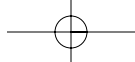


The third button branches to a third switchboard, called the Reports Switchboard, where you can click either of two summary reports or return to the Main Switchboard. Clicking either of the summary report buttons opens a custom dialog form, where you enter the beginning and ending dates for a report (see Figure 1.12). Clicking the Preview button after entering dates takes you to a preview of a summary report for the specified interval.

FIGURE 1.12:

The path from the Main Switchboard to the custom dialog boxes for collecting date input for the summary reports





The Database Wizard has done more than simply provide navigational paths between forms and reports. The wizard also has built a custom dialog form for collecting input. In this example, the date interval is the criterion for a parameter query that selects the appropriate records for the summary report (a parameter query that gets its information from a form is using a technique called *Query By Form*, which you'll learn about in Chapter 13).

Examining the Wizard's Work

Let's look behind the scenes to explore how the wizard accomplishes some of its tasks. The Database Wizard uses a number of elementary and advanced techniques, some of which you'll be learning to use in your database applications.

Restore the Database window (by maximizing the Expenses window) and note the following:

- There is only one form called Switchboard, yet we have seen three switchboards in the Expenses example: the Main Switchboard, the Forms Switchboard, and the Reports Switchboard. If you open the Switchboard form in Design view, you see a form with eight command buttons and eight blank labels (see Figure 1.13). The Database Wizard uses this form for all of the switchboards. Each switchboard is created on-the-fly as a different version of the same form when you click a command button. Clicking the second button on the Main Switchboard converts the form into the Forms Switchboard, and clicking the third button converts the form into the Reports Switchboard. The wizard has created instructions for converting the form, including changing the caption, displaying the correct number of buttons and labels, and empowering the command buttons displayed by each version of the form to carry out their specific tasks.

TIP

To open an object in Design view, select the object in the Objects pane of the Database window by clicking it, then click the Design button.

- In addition to the four data tables, there is a Switchboard Items table. One of the fields in this table, the ItemText field, holds the labels for the buttons on the various switchboards. The other fields store information for creating the switchboards and making the buttons work. Note that this table is the record source for the Switchboard form (record sources will be discussed in Chapter 6).

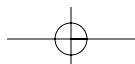
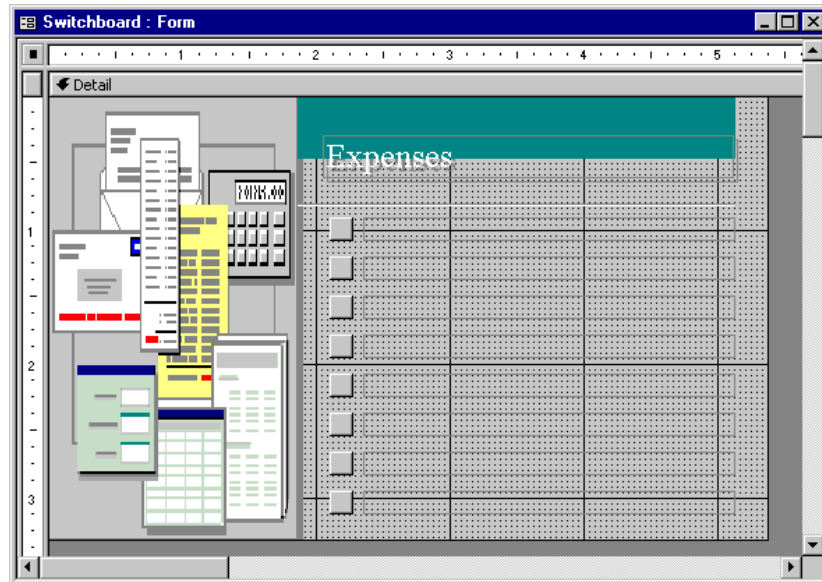


FIGURE 1.13:

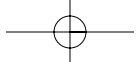
Design view of the Switchboard form



- Although the wizard has used the Query By Form technique to select records based on your input in a dialog form, there are no queries listed in the Queries pane. There are, in fact, no queries stored as *saved queries* in the Expenses application. If you've studied the way the Form Wizard and Report Wizard create their objects, you know that these wizards use SQL statements instead of saved queries as the record sources for the forms and reports. Most of the applications that the Database Wizard can create have no stored queries and use only SQL statements directly for record and row sources.
- There is a single Report Date Range form that the wizard uses for both of the custom dialog forms. The wizard has created instructions to change the caption depending on the button you click in the Reports Switchboard.
- There is a single standard module named Global Code listed in the Modules pane of the Database window. There are no macros listed in the Macros pane.

Exploring the Standard Module

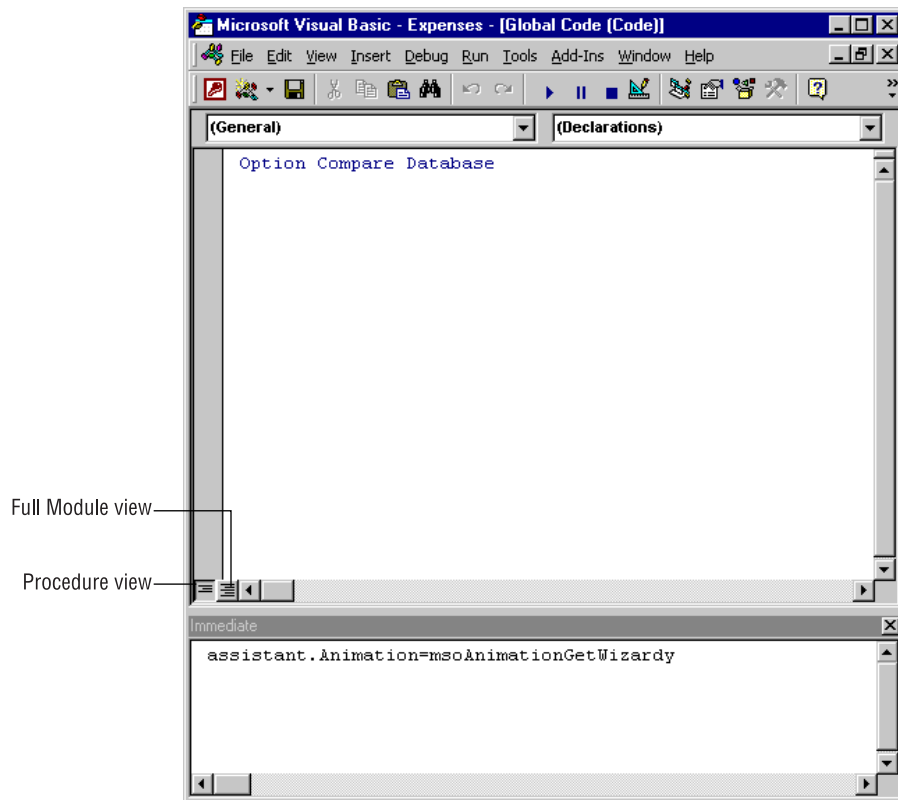
Let's explore the standard module. Double-click the Global Code module in the Modules pane of the Database window to display the Module window. What is displayed in the window depends on the Module window option settings on



your computer. Most likely, the window opens to the module's first pane (see Figure 1.14), called the Declarations section. In this pane, you store directions to Access and "declare" the names of constants, variables, and certain functions that you intend to use in the module. The Declarations section stores a statement for option settings. If you see additional text, then you are displaying Full Module view; click the button in the extreme lower-left corner of the window to display Procedure view. You use subsequent module panes to store the sets of instructions.

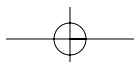
FIGURE 1.14:

The first pane of the Global Code module is the Declarations section.



NOTE

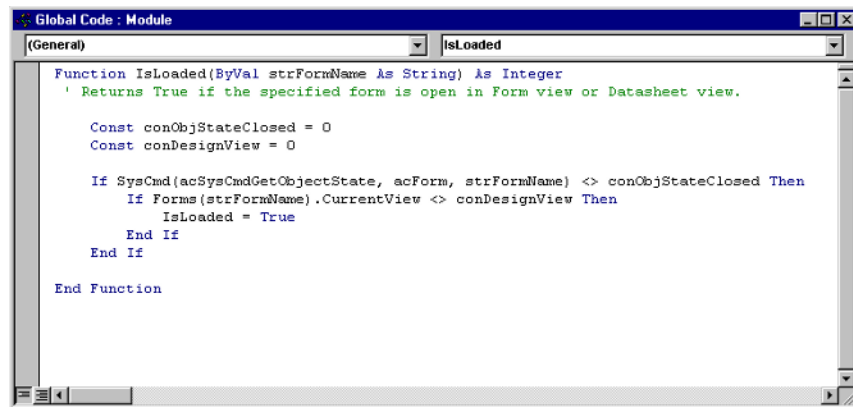
You can change the default view of the Module window by choosing **Tools > Options > Editor tab** and checking or clearing the **Default To Full Module View** check box. For this chapter, clear the check box to display the declarations and the procedures in separate panes.



A module stores instructions written in the VBA programming language. In VBA, you write sets of instructions in units called *procedures*. To see the procedures stored in the Global Code module, click the down arrow of the combo box on the right. The combo box list indicates the module has only one procedure, called `IsLoaded` (see Figure 1.15).

FIGURE 1.15:

The `IsLoaded` procedure



```

Global Code - Module
(General) | IsLoaded
Function IsLoaded(ByVal strFormName As String) As Integer
' Returns True if the specified form is open in Form view or Datasheet view.

Const conObjStateClosed = 0
Const conDesignView = 0

If SysCmd(acSysCmdGetObjectState, acForm, strFormName) <> conObjStateClosed Then
    If Forms(strFormName).CurrentView <> conDesignView Then
        IsLoaded = True
    End If
End If

End Function
  
```

Even though we haven't talked about the VBA language, it is obvious that this procedure isn't responsible for opening forms or reports, not to mention synchronizing them. As you'll learn later in the book, the Expenses application uses the `IsLoaded` procedure to determine whether a specified form is open.

Exploring the Procedure for a Command Button

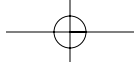


We know there must be additional procedures for the command buttons, so we'll explore a button. Open the Expense Reports by Employee form and switch to Design view. Then select the Expense Report Form button and, if necessary, open its property sheet by clicking the Properties button in the toolbar.

An object's property sheet lists all of the properties you can set at design time. In addition, most objects have properties that aren't listed in the property sheet. (You'll learn about the unlisted properties in later chapters.) The full set of an object's properties describes the object at a particular moment and is called the object's *state*.

NOTE

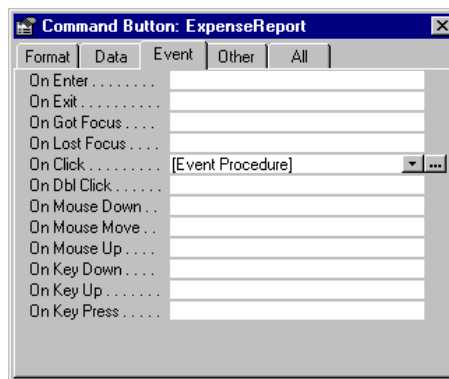
You are in design mode, called *design time*, whenever the active window is the Design view of one of the Database window objects.



Whenever you change a property, you change the object's state. For example, when you click a command button, you change its state from unclicked to clicked. In Access, many of an object's changes in state are given special treatment; these particular changes in state are called *events*. When you click a command button, it recognizes the Click event. For each of its events, the object has a corresponding *event property* listed in the Event tab of the object's property sheet. Figure 1.16 shows the 12 event properties for the command button.

FIGURE 1.16:

The event properties for a command button



In most cases, the name of the event property is the word “On” followed by the event's name. For example, the OnClick event property corresponds to the Click event. The name of an event property suggests the action that causes the object's state to change. For example, OnMouseDown suggests that when you press a mouse button while the pointer is over the button, the command button recognizes an event called MouseDown. While you can make similar deductions about other event properties and be correct most of the time, it's best to invest some time using online help to learn the precise definition of each event property. Click Help > What's This?, position the pointer over a property, and click. This opens Microsoft Access Help to the event you selected (see Figure 1.17). You'll learn more about specific events and the order of events in Chapter 2.

The reason that events are important is that events are programming opportunities. You can create a list of instructions—that is, a *program*—and tell Access to execute the program when an object recognizes one of its events. This programming technique is called *event-driven programming*.

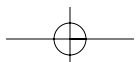


FIGURE 1.17:

Online help for the Mouse-Down and MouseUp events

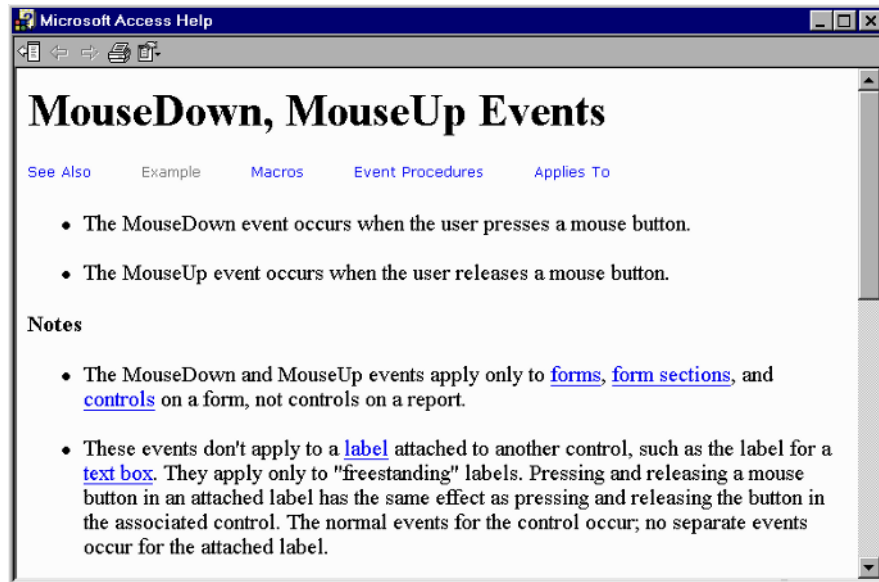
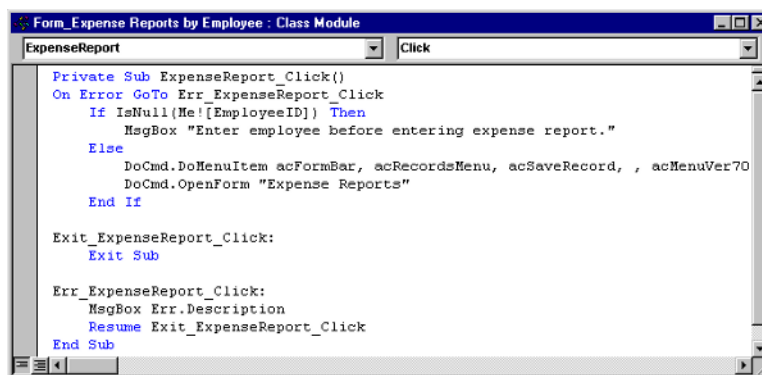
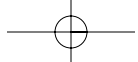


Figure 1.16 shows [Event Procedure] as the setting for the OnClick property. This means that a VBA procedure has been created and assigned to the button's Click event. Access runs the procedure when you click the button. A VBA procedure that runs when an object recognizes an event is an *event procedure*. When you click the Build button at the right of an event property, the Module window opens showing the event procedure (see Figure 1.18).

FIGURE 1.18:

The event procedure for the OnClick property of the command button





You'll learn how to create VBA event procedures later in this book. For now, note that the Module window's title bar displays the caption `Form_Expense Reports by Employee`, which means we are looking at the form module for the Expense Reports by Employee form. Access automatically names a form module for a form using the word "Form" and the form's name, separated with an underscore; the general pattern, or syntax, is `Form_formname`. Similarly, Access automatically names a report module using the syntax `Report_reportname`.

The Database Wizard creates event procedures for each of the command buttons. Some procedures simply open other forms or reports; other procedures also synchronize the form or report to display the matching record.

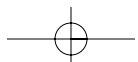
The Database Wizard provides a well-designed first draft for the application. Someone who knows Access can navigate through its tasks easily. However, for the Expenses application to be described as *fully automated*, it must be further developed so that someone who doesn't know Access can use it without detailed instructions. Let's explore some of the other Access helpers you can call on to add features that make the application easier to use.

Using the Command Button Wizard

The Command Button Wizard automatically creates command buttons and VBA event procedures for the buttons' Click events. This powerful wizard can create procedures for many common database operations. Table 1.1 shows the 33 actions you can automate with the Command Button Wizard.

TABLE 1.1: Operations Automated with the Command Button Wizard

Category	Task
Record Navigation	Go to First, Last, Next, Previous Record, Find Record, Find Next
Record Operations	Add New Record, Delete Record, Duplicate Record, Print Record, Save, Undo Record
Form Operations	Apply Form Filter, Edit Form Filter, Close Form, Open Form, Open Page, Print a Form, Print Current Form, Refresh Form Data
Report Operations	Print Report, Preview Report, Mail Report, Send Report to File
Application	Run Notepad, Run MS Word, Run MS Excel, Quit Application, Run Application
Miscellaneous	Print Table, Run Macro, Run Query, Run AutoDialer



The task flow diagram shown earlier in Figure 1.9 shows that the Database Wizard has created a one-way path from the Main Switchboard to the Expense Reports by Employee form. To make a two-way path between the form and the Main Switchboard, we'll create a command button to close the Expense Reports by Employee form and return to the Main Switchboard.



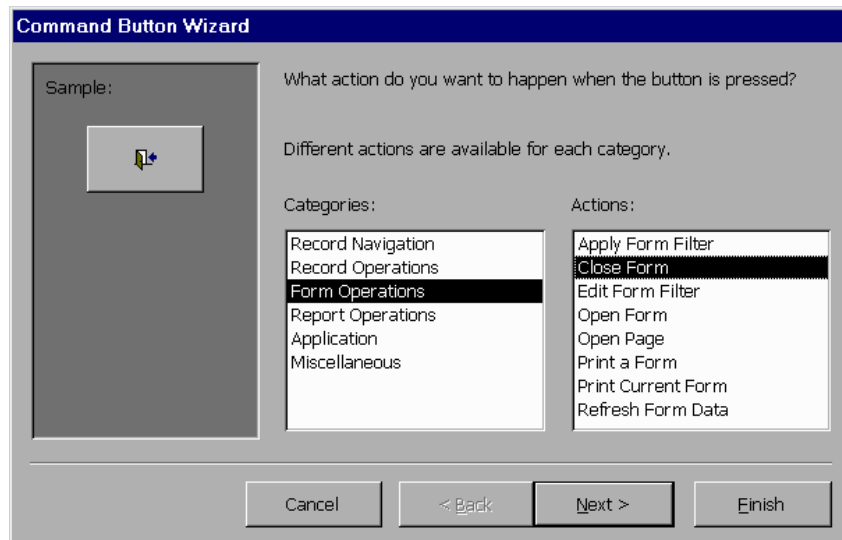
1. Open the Expense Reports by Employee form in Design view. If the Control Wizards toolbox button is not pressed, click the button now to activate the Control Wizards.



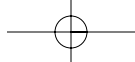
2. Click the Command Button tool and then click in the form next to the Expense Report Form button. The wizard's first screen shows a list of six categories of tasks and a list of the actions for each category.
3. Select the Form Operations category and the Close Form action (see Figure 1.19). Click the Next button.

FIGURE 1.19:

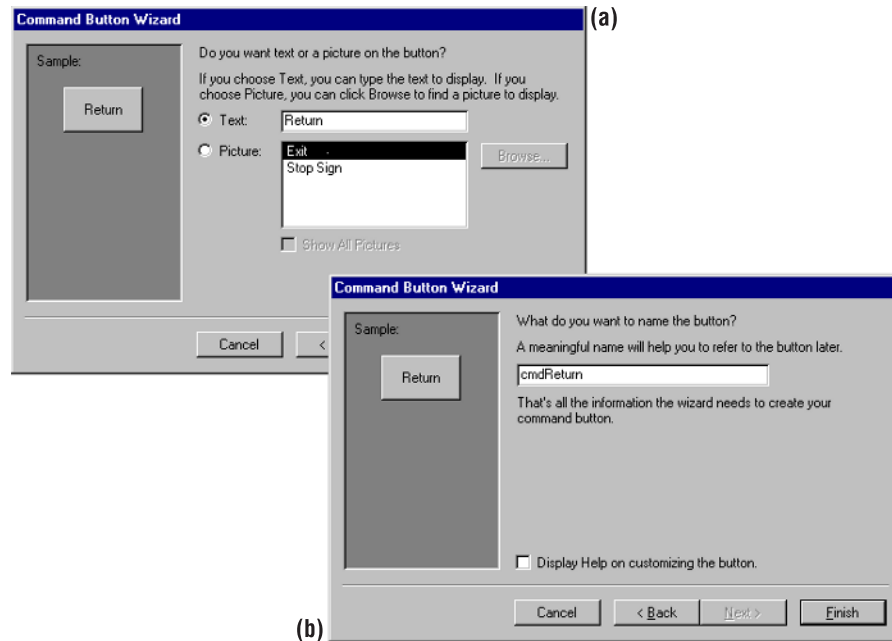
The Command Button Wizard can automate common database operations.



4. In the next screen, you design the button's appearance. For a consistent look, enter **Return** as a text caption instead of using a picture (see Figure 1.20a). Click the Next button.
5. The final screen gives you the opportunity to name the button. Enter **cmdReturn** (see Figure 1.20b).

**FIGURE 1.20:**

You can design the button's appearance (a) and give the button a name (b).



6. Click the Finish button in the last wizard screen. The wizard creates the command button and attaches a VBA procedure to carry out the action you specified. Let's review the procedure.
7. Make sure the Expense Reports by Employee form is open in Design view. With the newly created Return button selected, click the Properties button. Click in the OnClick event property and click the Build button to the right of the property box. The event procedure appears in the Module window (see Figure 1.21). The combo box on the left displays the button's name, and the combo box on the right displays the name of the event. In later chapters, you'll learn about the purpose of each line in the procedure; for now, just note that the DoCmd.Close line is the instruction that closes the form.
8. Save the form, switch back to Form view, and click the new Return button. The form closes, and you are returned to the Main Switchboard.

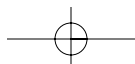
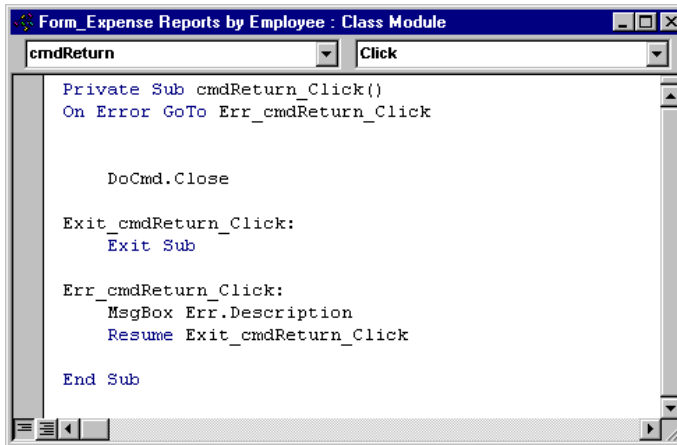


FIGURE 1.21:

The event procedure to close a form



```
Private Sub cmdReturn_Click()  
On Error GoTo Err_cmdReturn_Click  
  
DoCmd.Close  
  
Exit_cmdReturn_Click:  
Exit Sub  
  
Err_cmdReturn_Click:  
MsgBox Err.Description  
Resume Exit_cmdReturn_Click  
  
End Sub
```

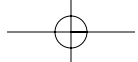
NOTE

Choose object names with care. One reason for selecting the name carefully is that it's very inconvenient to change an object's name later. A more important reason is that this is your opportunity to make your work as an application developer much easier by using meaningful names. For example, by choosing the name `cmdReturn` instead of accepting the default (such as `Command7`), we document that the object is a command button by using the *cmd* prefix (called a *tag* in naming-standard vocabulary), and we document that clicking the button returns to the previous form by including the word *Return* as the descriptive part (called the *base name*). Chapter 2 discusses naming standards.

Using Hyperlinks to Navigate

Hyperlinks were a new navigation feature in Access 97, and they have been enhanced in Access 2000. A *hyperlink* is a piece of text, an image, a toolbar button, a menu command, or a command button that you click to jump to another location. In Access, you can use hyperlinks to open and jump to the following:

- Another object in your database
- Any available file stored in your computer's file system (on your computer or on another computer in your local area network)

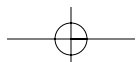


- A specified sublocation within any Microsoft Office file, including a bookmark in a Word document, a range in an Excel spreadsheet, a slide in a PowerPoint presentation, or an object in another Access database
- Any available file in your company's private intranet or in the public Internet—if the file is an HTML document, you can jump to a specified sublocation within the document
- A blank e-mail form pre-addressed to the e-mail address you specify.

In order for a hyperlink to work, you must use the correct format to identify the location you want to jump to. For example, to identify a file on an intranet or on the Internet, you use an Internet address, called a *Uniform Resource Locator* (URL). For example, `http://www.microsoft.com/` identifies the home page for the Microsoft Web site on the World Wide Web; `ftp://ftp.microsoft.com` identifies the Microsoft ftp site. To identify a file in your computer's file system, you use a standard format for specifying the path to the file called the *universal naming convention* (UNC) path. For example, `C:\Program Files\Microsoft Office\Access\Samples\Cajun.htm` identifies the HTML document you may have installed as part of Access, and `C:\Program Files\Microsoft Office\Access\Samples\Northwind.mdb` identifies the Northwind sample database for Access. The Internet or path address of the file is called the *hyperlink address*. You can specify the location within the Microsoft Office file or an HTML document, called the *hyperlink subaddress*, using the syntax shown in Table 1.2.

TABLE 1.2: Hyperlink Subaddress Syntax

Type of File	Syntax for a Location within the File
Microsoft Access	The name of a Database window object. If there are several objects with the same name, Access looks up objects in the following order: forms, reports, queries, tables, data access pages, macros, modules, views, schemas, stored procedures, SQL linked tables, linked tables, and triggers. You can also enter <i>object type object name</i> ; for example, to specify a report named Suppliers, use the syntax Report Suppliers.
Microsoft Word	The name of a bookmark. You must define the bookmark in Word before you can jump to it.
Microsoft Excel	The name of a range. Use the syntax <i>sheet!range</i> . For example, to specify the target as the L8 cell in the worksheet named Source, use the syntax Source!L8.
Microsoft PowerPoint	The number of a slide. For example, to specify the tenth slide, use the syntax 10.
HTML document	The Name tag.



When the target of the hyperlink is another object in the current Access database, you don't need to specify the hyperlink address; you specify only the hyperlink subaddress. For example, if you are working in the Expenses database and want to create a hyperlink with the Expense Categories form as the target, you need only specify the subaddress as Expense Categories or as Form Expense Categories.

If the target of the hyperlink is another object in the Access database you are currently working with, clicking the hyperlink in Access opens the object. A table or query opens in Datasheet view, a form opens in Form view, a report opens in Print Preview, a data access page opens in Page view, and macros and modules open in Design view. If the target is an object in another Access database, the current Access window minimizes, and a second instance of Access opens and displays the target object. If the target is a file in another Microsoft Office application, the current Access window minimizes, and the application opens and displays the target. If the target is a file or document on the intranet or Internet, the Access window minimizes, and the browser opens to display the document (your TCP/IP connection must be open when you click the hyperlink).

Access 2000 provides three ways to use hyperlinks for navigation:

- You can store hyperlink addresses in tables and display the hyperlinks in a datasheet or in a form control bound to the hyperlink field.
- You can create a hyperlink in an unbound control on a form using a label, a command button, or an image.
- You can convert a menu command into a hyperlink.

Storing Hyperlinks as Data in a Table

Access 2000 provides a Hyperlink data type for storing hyperlink addresses. For example, in an order-entry database application, you can store the Internet addresses for your suppliers, just as you store their other contact information, and display the addresses in a suppliers form. When you move the mouse pointer over the text box that displays the hyperlink address, the pointer icon changes to a pointing finger and the status bar displays the hyperlink address. Clicking the hyperlink takes you to the hyperlink target.

The Hyperlink data type lets you store information in three parts: the first part, *displaytext*, is the text that you want to display in the field; the second part, *address*, is the hyperlink address; and the third part, *subaddress*, is the hyperlink subaddress. The first part is optional. The parts are separated by the pound sign, as in *displaytext#address#subaddress#*. If you enter display text in the Hyperlink field, Access

shows only the *displaytext* in the cell and does not display the rest of the address; if you omit *displaytext*, Access shows only the hyperlink address.

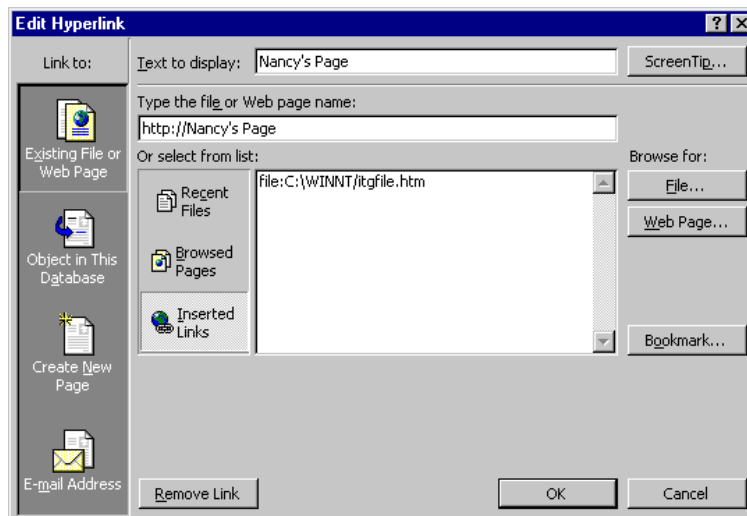
The easiest way to enter a hyperlink target is to use the Insert Hyperlink toolbar button. As an example, we'll add a hyperlink field to the Employees table. This field will store the URL for each employee's personal home page.

1. Open the Employees table in Design view. Click in a blank FieldName cell and enter **HomePage**. Choose Hyperlink from the Datatype combo box.
2. Save the table and switch to Datasheet view.
3. Click the HomePage cell for the first employee and type the text that you want to display. For example, type **Nancy's Page** and then click the Insert Hyperlink button in the toolbar. You can specify a file to link to or a Web page name from a list of recently accessed files, recently accessed Web pages, or existing inserted links (see Figure 1.22). By default, if you entered display text, the file or Web page name in the text box below appears as the protocol `http://` followed by the display text. You'll need to clear the text box before entering the hyperlink address.



FIGURE 1.22:

Using the Edit Hyperlink dialog to enter an address and subaddress



4. Delete the text in the Type the File or Web Page Name text box and type in `http://www.microsoft.com/employees/davolio.htm` as the fictitious URL for the home page for the first employee. Click OK. Access displays the hyperlink text in the standard way: The text is shown underlined and blue.

5. Click another record to save the entry and reset the pointer. When you move the pointer back over the cell, the pointer icon changes to a hand with a pointing finger. If we had entered a real URL, clicking the cell would open the browser and display the document.
6. To edit the hyperlink, right-click the cell and choose Hyperlink from the shortcut menu. Click in the Display Text box in the fly-out menu. Select the text (*Nancy's page*), press Delete to delete the display text, and then press Enter. The cell now shows the hyperlink address. Note that you can change the address or subaddress of the hyperlink by choosing Edit Hyperlink from the fly-out menu to display the Edit Hyperlink dialog.
7. Open the Expense Reports by Employee form in Design view. Display the Field list (by choosing View > Field List) and drag the HomePage field to the form just below the Work Phone text box. Switch to Form view. The hyperlink for the employee's home page is displayed in the HomePage text box (see Figure 1.23).

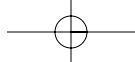
FIGURE 1.23:

Displaying a bound hyperlink in a form

Exp Rpt Name	Date Sub	Advance	Total Expense:	Amount Due
Feb. '95 Sales Trip	3/1/95	\$0.00	\$464.00	\$464.00

Using a Hyperlink as an Unbound Form Control

When you don't want the hyperlink to change with each record, you can use a label, a command button, or an image to create a hyperlink on a form. For example, when you want to use a hyperlink to navigate from a form to another form or to a report, you create the hyperlink directly on the form. To create an unbound hyperlink, place



a label, a command button, or an image on the form. You can use the Caption property of the label or the command button to describe the hyperlink. To create the hyperlink, set the control's HyperlinkAddress and HyperlinkSubAddress properties. To create a hyperlink to another object in the current Access database, leave the HyperlinkAddress property blank.

To explore unbound hyperlinks, we'll create three types:

- A hyperlink for a label that opens and displays another object in the same database
- A clickable image hyperlink that opens and displays an object in another Access database
- A command button hyperlink that opens a browser and displays a Web page on the World Wide Web

Creating a Label Hyperlink

To create a hyperlink as a label control, follow these steps:

1. Create a new unbound form named **frmHyperlinks**. Open the form's properties by clicking the square block in the upper-left corner of the form window, then clicking the Properties button. Set the Caption property to **Hyperlinks**.

NOTE

A new form must be saved in order for it to be named.

2. Place a label control on the form and type **Categories** directly into the label. Click in the HyperlinkSubAddress property in the label's property sheet. Click the Build button at the right of the property box to display the Insert Hyperlink dialog.
3. Click the Object in This Database button. The Insert Hyperlink dialog displays the objects in the current database listed in tree format by type (see Figure 1.24).
4. Choose the Expense Categories form and click OK. Access sets the HyperlinkSubAddress property to Form Expense Categories, creates the hyperlink, and displays the label caption in blue, underlined text.
5. Save the form, switch to Form view, and click the label. The Expense Categories form opens.

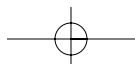
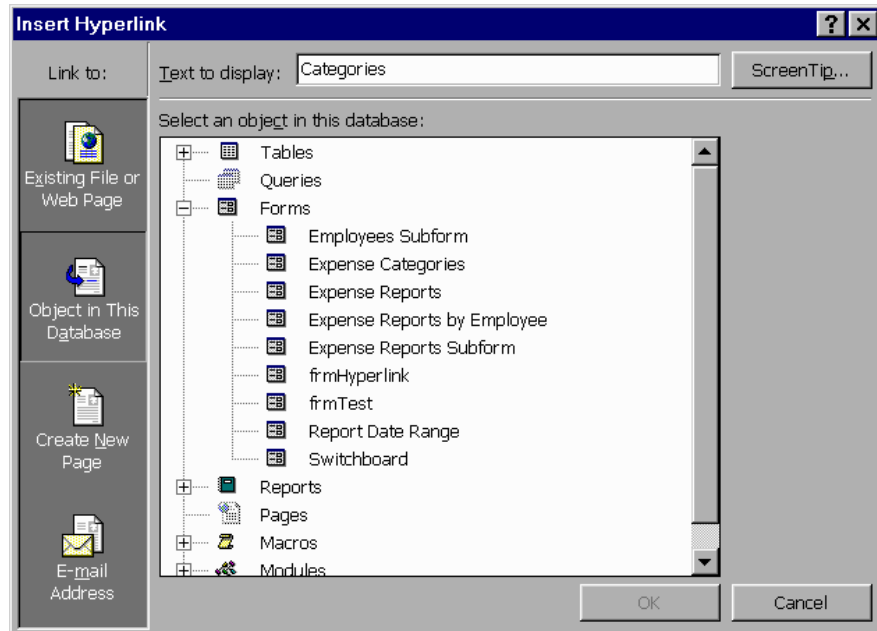


FIGURE 1.24:

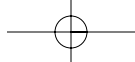
Use the Insert Hyperlink dialog to choose the database object.



Creating a Clickable Image

Here are the steps to create a clickable image:

1. Open the Hyperlinks form in Design view and choose **Insert > Picture**. Select an image in the Insert Picture dialog. We'll use the dove.wmf file in the Microsoft Office\Clipart\Popular folder. To resize the image, set the SizeMode property to Stretch, click a corner of the image-selection rectangle, and drag to the desired size. We'll use this image to create a hyperlink to a form in another Access database.
2. With the image selected, click the Properties button. Click the Build button next to the Hyperlink subaddress property to display the Insert Hyperlink dialog. Click the Object in This Database button to see the list of objects in the database (see Figure 1.24). Select the Expense Categories form and click OK. Access sets the HyperlinkSubAddress property to Form Expense Categories.
3. Save the form, switch to Form view, and click the image. The current form minimizes, and the Expense Categories form is displayed.



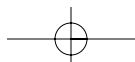
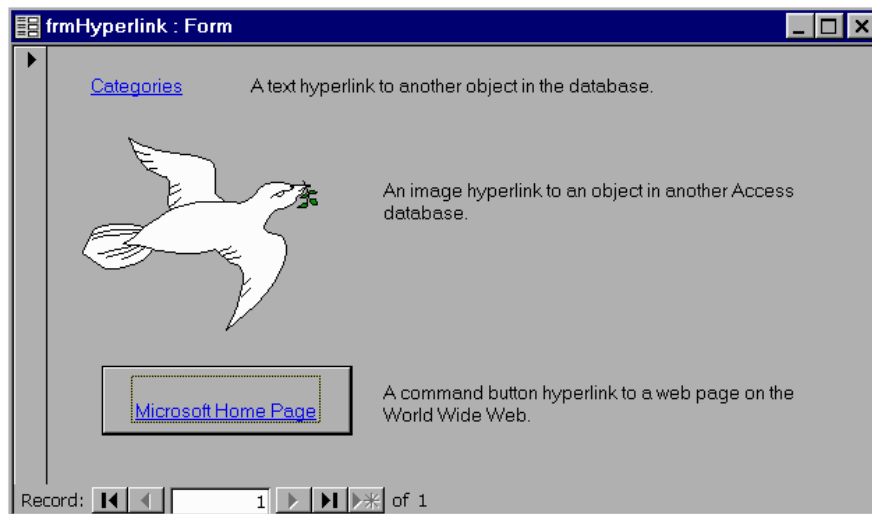
Creating a Command Button Hyperlink

The following steps create a command button as a hyperlink:

1. Switch to Design view. If necessary, click the Control Wizards button to deselect the wizard (the Control Wizards are activated when the button is pressed in), then click the Command Button tool and draw a command button on the form. We'll use this button to create a hyperlink to a Web page on the World Wide Web.
2. Set the Caption property to **Microsoft Home Page**.
3. Click the HyperlinkAddress property in the command button's property sheet and enter the URL to the Microsoft home page: **http://www.microsoft.com/**. When you press Enter, Access creates the hyperlink, changes the button's font color to blue, and underlines the text to indicate a hyperlink.
4. Save the form and switch to Form view (see Figure 1.25). Click the command button. If your Internet connection is open and you have Internet Explorer installed, a new instance of the Internet Explorer starts and the browser locates and displays the Microsoft home page.

FIGURE 1.25:

The Hyperlinks form with hyperlinks for a label, an image, and a command button



Changing the Hyperlink Control Type

After creating a hyperlink control as one of the three types, you can change the control type for the hyperlink. In Form Design view, select the control and choose **Format > Change To**. The fly-out menu displays commands for the other two control types as active commands. Click the type you want to change to.

If you change from a label or command button to an image, the **Picture** property of the image control displays the word “none,” because you haven’t selected an image. To insert an image, click the **Build** button to the right of the **Picture** property and choose a bitmap image to display.

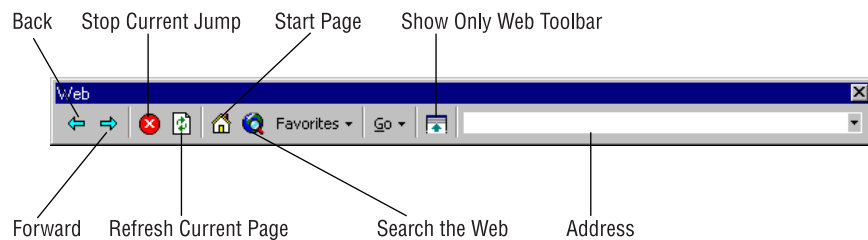
If you change from an image to a label or command button, Access sets the **Caption** property of the changed control to the value of the **HyperlinkAddress** property. If the **HyperlinkAddress** property is blank, it changes the **Caption** property to the value of the **HyperlinkSubAddress** property.

Using the Web Toolbar

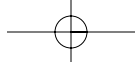
When you click hyperlinks to jump to other objects, Access 2000 maintains an internal history list of the hyperlink targets you’ve visited. Access provides a **Web toolbar** for navigation (see Figure 1.26). You use the **Back** button to return to the previous hyperlink target and the **Forward** button to go to the next hyperlink target on the history list (the **Forward** button can only take you to a target that you have already visited). If you change your mind after activating a hyperlink, you can click the **Stop** button on the Web toolbar to stop following the link.

FIGURE 1.26:

Use the Web toolbar to navigate between hyperlinks.



Using images, labels, and command buttons as hyperlinks is an easy way to open and display another object. When you create the hyperlink by setting the **HyperlinkAddress** and **HyperlinkSubAddress** properties, you are limited to



opening an object and moving to a location within the object; you can't specify additional actions. However, when you click the control to activate the hyperlink, the control recognizes the Click event. If you want to take any other action, such as hiding the form that contains the hyperlink or synchronizing an opened form or report, you can write a program and request that Access run the program when you click the control. We'll discuss programming Click events later in the book.

Using the Combo Box Wizard

The Expense Reports by Employee form in the Expenses database displays a record for each employee. Our example includes only three employees as sample data. With so few employees, finding the record for a specific employee is a simple matter of browsing the records using the navigation buttons at the bottom of the form. With more employees, you need a more efficient way to find a specific employee. The Combo Box Wizard provides the solution. You can use the Combo Box Wizard to create a lookup combo box. With a lookup combo box, you select a value from the list, and Access automatically locates and displays the corresponding record.

In our example, we'll use the wizard to create a combo box that lists each employee's name in alphabetical order and a procedure that runs when you select a name. The procedure finds and displays the record for the employee that you selected.



1. Open the Expense Reports by Employee form in Design view. If necessary, click the Control Wizards button to activate the wizards. Click the Combo Box tool in the toolbox and then click in the form next to the Return button. The first screen asks for the source of the values you want in the combo list.
2. Choose the third option to create a lookup combo box (see Figure 1.27).
3. The next screen displays a list of the fields in the form's underlying record source (see Figure 1.28). We'll display only the employees' names in the list, but we'll also select the EmployeeID field because it is the primary key for the Employee table. Select the EmployeeID, LastName, and FirstName fields. Then click Next. The wizard creates a VBA procedure that uses the EmployeeID key to find the unique record corresponding to the employee's name you choose.

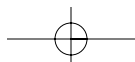


FIGURE 1.27:

You can use the Combo Box Wizard to create a lookup combo box.

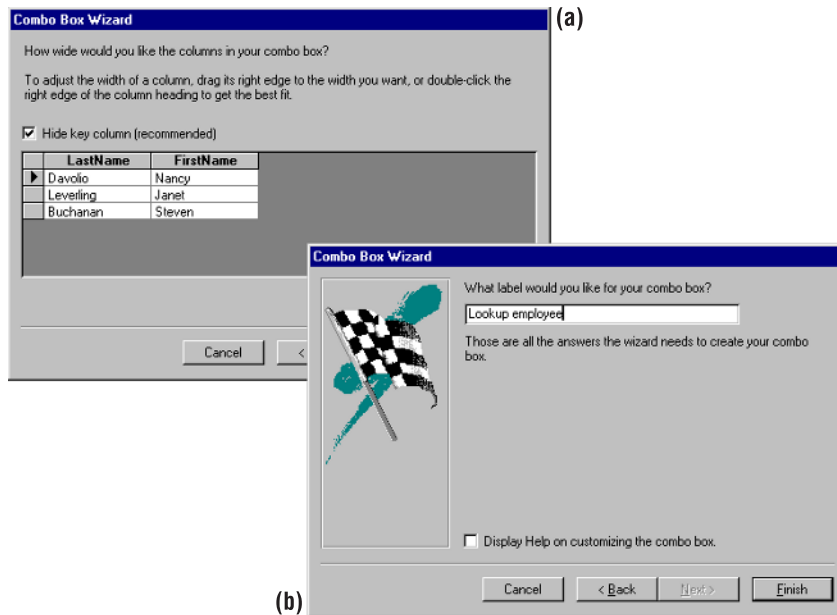
FIGURE 1.28:

The Available Fields are the fields in the form's underlying record source.

4. You can modify the list's appearance in the next screen (see Figure 1.29a). Be sure to leave the Hide key column check box checked. Click the Next button. The final screen gives the opportunity to customize the label (see Figure 1.29b). Type **Lookup employee** in the text box.

FIGURE 1.29:

Modify the combo box list (a) and customize the label (b).



5. Click the Finish button. The wizard creates the lookup combo box and attaches the VBA procedure to find the record.

Note that the property sheet indicates that an event procedure now exists for the AfterUpdate event (see Figure 1.30a). The property sheet also shows that the wizard has assigned a default name using the syntax `Combomn`, where `nn` is a sequentially assigned number controlled by Access 2000, which doesn't help at all in identifying the purpose of this combo box. (It would have been more helpful if the Combo Box Wizard had allowed you to name the control `cboEmployee` instead.) We'll explain how the BeforeUpdate and AfterUpdate events work in Chapter 2. For now, note that the combo box recognizes the AfterUpdate event the instant after you select an employee's name. Click the Build button to the right of the AfterUpdate property box to view the VBA procedure (see Figure 1.30b). This procedure *synchronizes* the form to the value displayed in the combo box by finding and displaying the record that matches the combo box.

To test the wizard's work, save the form, switch back to Form view, and select an employee's name from the list (see Figure 1.31). The form automatically displays the synchronized record.

FIGURE 1.30:

The Combo Box Wizard creates an event procedure for the AfterUpdate event (a). The event procedure (b) synchronizes the form's recordset to the combo box value.

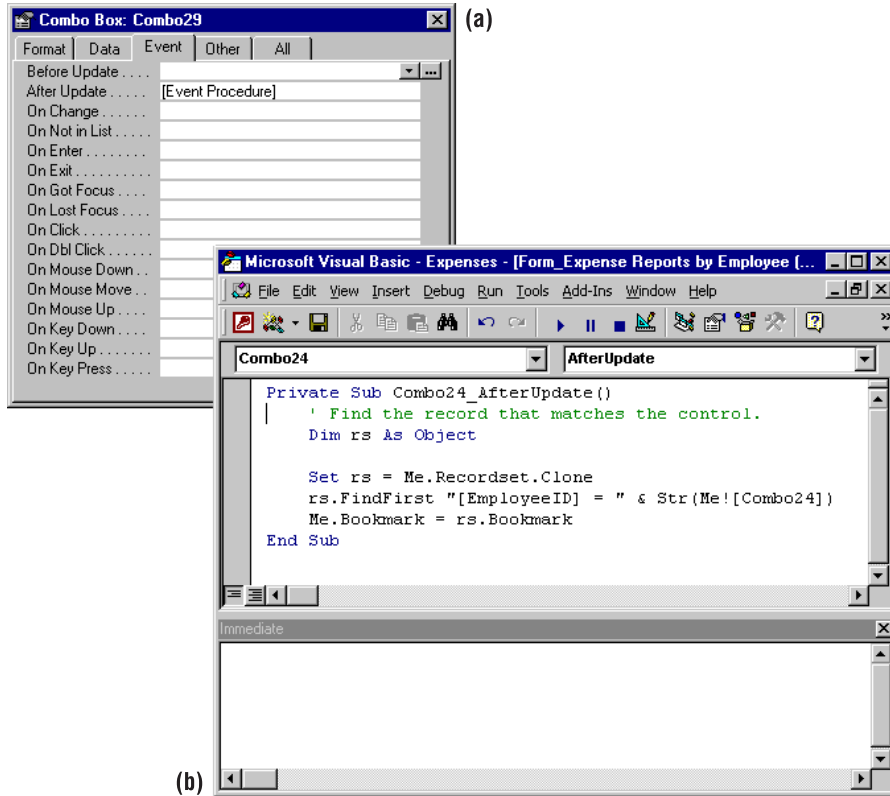
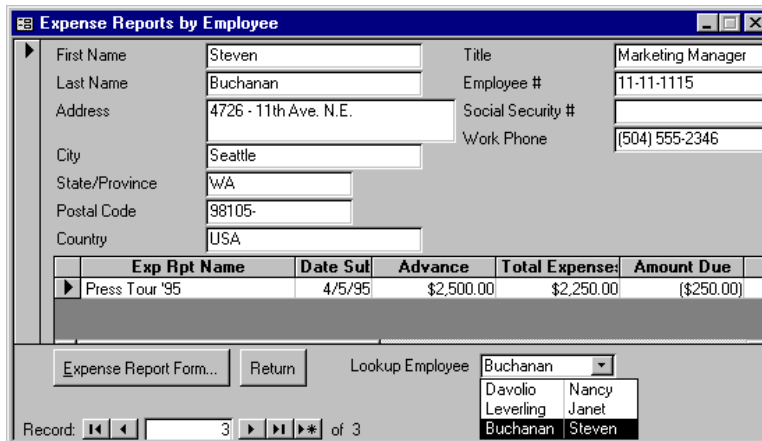
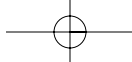


FIGURE 1.31:

The customized Expense Reports by Employee form provides a return path and a lookup.





Using the Switchboard Manager

After the Database Wizard creates a basic application, you can customize it by adding forms and reports. When you add a new form or report, you can provide a path to it by adding a command button that runs a program or by adding a hyperlink to the appropriate form or to one of the switchboards. To add a button to a switchboard, click the Change Switchboard Items button on the Main Switchboard. Your click summons another Access helper, the Switchboard Manager.

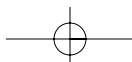
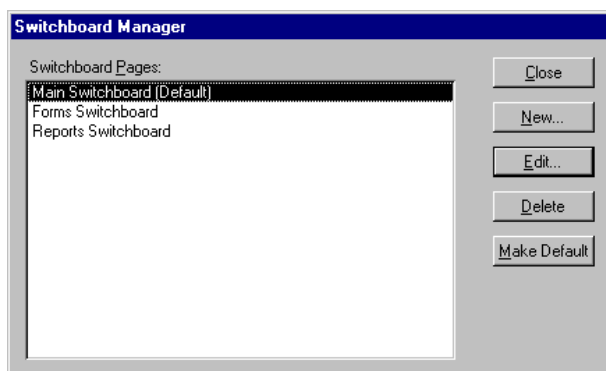
NOTE

In Access, you can't place buttons on tables or queries, so tables and queries are not usually displayed in a custom user interface. Instead, you build the new user interface entirely out of forms and reports. The form, data access pages, and report objects have been designed to respond to a user clicking a button, selecting a value from a list, or entering a value in a text box.

The purpose of the Switchboard Manager is to help you modify the Switchboard form created by the Database Wizard. You also can use the Switchboard Manager to create a new Switchboard form if you started from scratch rather than using the Database Wizard to create the database. The Switchboard Manager shown in Figure 1.32 lists the three switchboard pages in the Expenses application. The Main Switchboard is shown as the default; the default page is the one that is displayed when you first open the Switchboard form.

FIGURE 1.32:

The Switchboard Manager

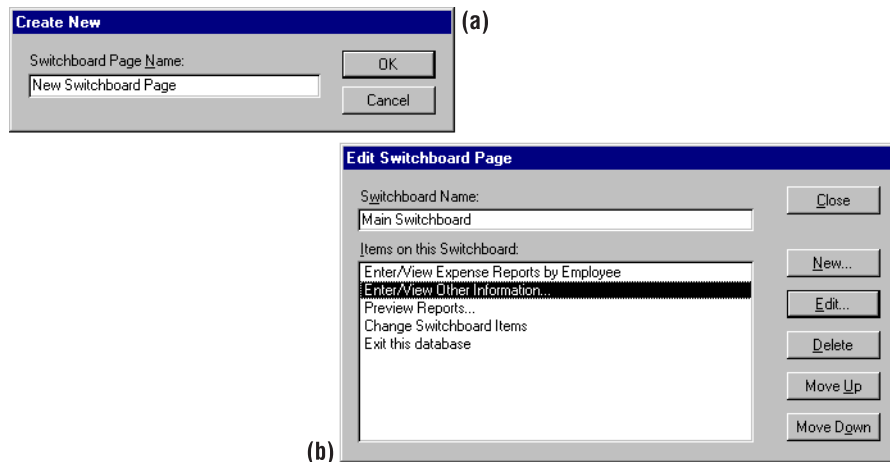


The Switchboard Manager buttons work as follows:

- The New button creates a new switchboard page. Click the New button to name the new switchboard page (see Figure 1.33a).
- The Edit button modifies the buttons on the selected switchboard page. Click the Edit button to display the Edit Switchboard Page dialog (see Figure 1.33b). For the selected page, this dialog displays the labels for the buttons on the page. You can add a new button, change the label of an existing button, delete a button, and move a button up or down in the list.
- The Delete button deletes the selected switchboard page.
- The Make Default button changes the default to the selected switchboard page.

FIGURE 1.33:

Add a new switchboard page (a) or edit an existing page (b).



As an example of using the Switchboard Manager, we'll change the label for the second button on the Main Switchboard to the slightly more informative label of Enter/View Expense Information.

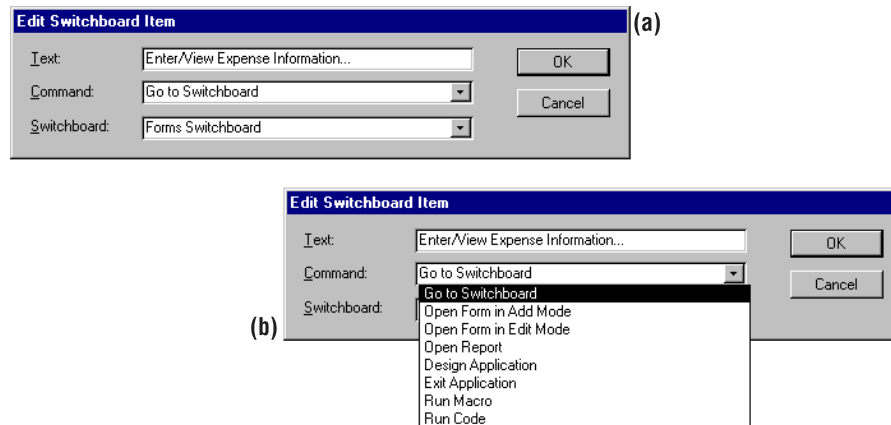
1. With the Main Switchboard selected in the Switchboard Manager dialog, click the Edit button and select the second item in the list. Click the Edit button on the Edit Switchboard Page dialog to display the Edit Switchboard Item dialog (see Figure 1.34a).
2. Change the label's text in the first box to **Enter/View Expense Information**. You can change the button's action by selecting any of the eight actions in the

Command combo list (see Figure 1.34b). After you select an action, the third box changes to display appropriate choices. In our case, the command Go to Switchboard takes you to another switchboard page, and the third combo box displays the name of the switchboard page (Forms Switchboard).

3. Click OK to close the Edit Switchboard Item dialog, click Close to close the Edit Switchboard Page dialog, and click Close to close the Switchboard Manager. The Main Switchboard displays the changed label for the second button.

FIGURE 1.34:

The Switchboard Manager lets you change the button's label (a) and its action (b).



Controlling the User Interface

After you use the Control Wizards and the Switchboard Manager, your customized application should be easier for you to use, but you still don't want to turn it over to a novice. At this stage, you have designed navigation paths but have not restricted the user to those paths. The complete built-in command environment is exposed, because all of the menu commands, shortcut menu commands, toolbar buttons, and keyboard shortcuts that are built into Access are available for switching to Design view and changing the design of the application. The Database window is also displayed, though minimized. The novice user has immediate access to the objects that make up your application.

Before turning the application over to a novice, you should protect the application by hiding any commands that could be used to change the application's design and by hiding the Database window. Access 2000 provides tools for controlling the

user interface and protecting your application. In this section, you learn about three of the tools that don't require programming:

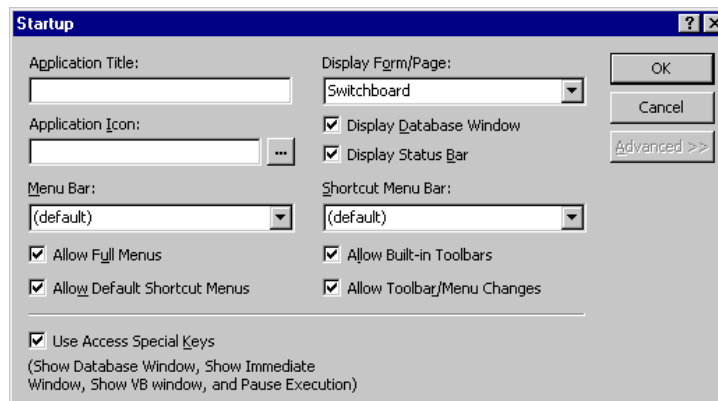
- You can control many features of the Access environment by setting startup properties; for example, you can specify whether or not the Database window is displayed when you start up the application, and you can disable the keyboard shortcuts that display the Database window (the F11 or Alt+F1 key combinations). However, without programming, you cannot prevent users from pressing the Shift key to bypass your startup property settings when opening a database.
- You can prevent unauthorized entry into a database by defining a password. Simple password security restricts entry into your database but offers no protection after a user who knows the password has opened the database.
- You can replace the built-in menu bars, menus, and toolbars with custom versions that include only the commands required for using your application. Without programming, you can include built-in commands and commands for opening database objects. However, you must use macro programming in order to create custom commands for running a set of instructions and in order to customize the keyboard.

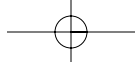
Setting Startup Properties

Startup properties dictate how a database application looks when it starts up. Choose Tools > Startup to display the Startup dialog and click the Advanced button to display the options (see Figure 1.35).

FIGURE 1.35:

You can set most of the startup properties for a database using the Startup dialog.





You use the Startup dialog to set the properties shown in Table 1.3. The application title and icon settings take effect as soon as you close the Startup dialog. The other settings take effect the next time you open the database.

TABLE 1.3: Startup Properties in the Startup Dialog

Startup Property	Description
Application Title	Specifies the text that appears in the application's title bar. When this property isn't set, the words "Microsoft Access" appear by default.
Application Icon	Specifies the icon that appears in the application's title bar. When this property isn't set, the Access key icon appears by default.
Display Form/Page	Specifies the name of the form or page to display on startup. When you create an application with the Database Wizard, this property is set to the Switchboard form by default.
Display Database Window	Specifies whether or not the Database window is displayed when you start up the application.
Display Status Bar	Specifies whether or not the status bar is displayed when you start up the application. Normally, you want to keep the default to display the status bar because you use it to display on-screen help.
Menu Bar	Displays a custom menu bar that replaces the built-in menu bar in all windows of your application except where you have created a custom menu bar for a specific form or report.
Shortcut Menu Bar	Displays a custom shortcut menu bar that replaces the built-in shortcut menu bar in all windows of your application except where you have created a custom shortcut menu bar for a specific form, report, or form control.
Allow Full Menus	Specifies whether Access should display the full set of built-in menus and menu commands or only the reduced set of built-in menus. The reduced set doesn't include commands that allow you to modify the application.
Allow Default Shortcut Menus	Specifies whether Access should display the built-in shortcut menus.
Allow Built-In Toolbars	Specifies whether Access should display the built-in toolbars. You can clear the check box to hide all of the built-in toolbars but still display custom toolbars.

Continued on next page

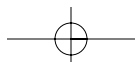


TABLE 1.3 CONTINUED: Startup Properties in the Startup Dialog

Startup Property	Description
Allow Toolbar/Menu Changes	Specifies whether Access should allow users to change toolbars and menu bars. If you clear the check box, you disable the right mouse button, the Close button on the toolbars, and the Toolbars command on the View menu. You can still move, size, and dock the toolbars and menu bar and hide or unhide toolbars.
Use Access Special Keys	Specifies whether Access should enable the special built-in key combinations: F11 or Alt+F11 (displays, unhides, and restores the Database window), Ctrl+G (displays the Immediate window), Alt+F11 (displays the VB Editor), Ctrl+F11 (toggles between the built-in menu bar and the custom menu bar), and Ctrl+Break (in a database or a project, suspends execution of a VBA procedure; in a project, also stops Access from retrieving records from the server).

The simplest way to control the commands available to the user is to set startup properties as follows:

- Hide the Database window and prevent the user from displaying the window after starting the database.
- Hide all toolbars and shortcut menus.
- Replace the built-in menus with an alternate reduced set of built-in menus.

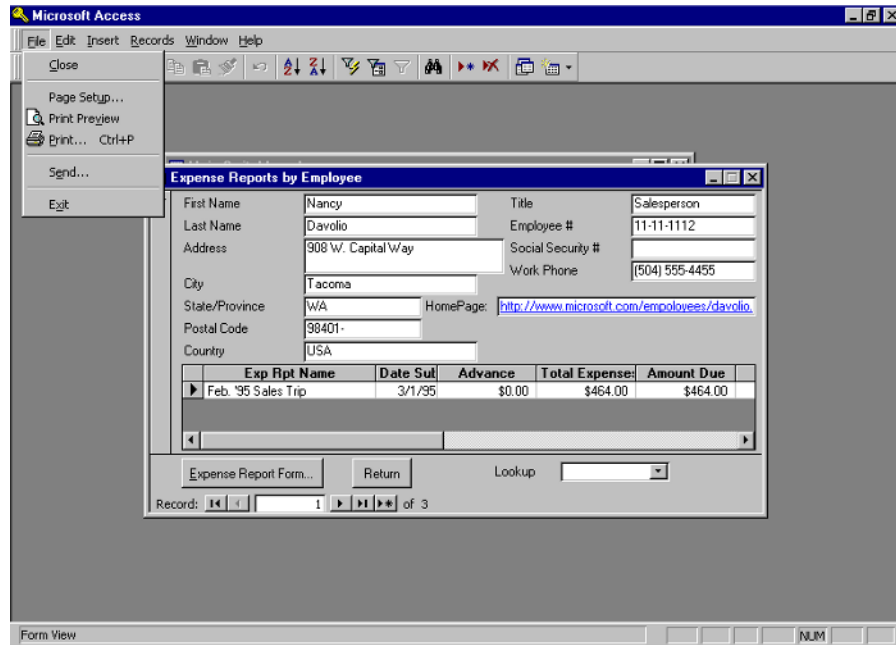
The reduced set includes only the commands that are useful when a user is working with a custom database application in views other than Design view and eliminates the menu commands for changing the design of a database object as well as several other menu commands. Figure 1.36 shows the reduced menu bar for Form view. This menu bar doesn't include the View menu or Tools menu and eliminates some menu commands on the remaining menus.

We'll use the Startup dialog to customize the title bar and protect the application.

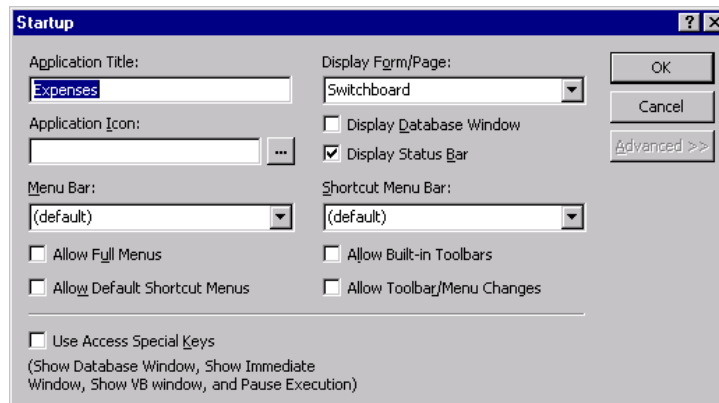
1. Choose Tools > Startup and click the Advanced button.
2. Set the Application Title property to **Expenses**.
3. Clear all of the check boxes except the one to display the status bar (see Figure 1.37).

FIGURE 1.36:

The reduced Form view built-in menu bar

**FIGURE 1.37:**

Using the Startup settings to hide the Database window and restrict the available menu, toolbar, and keyboard commands.



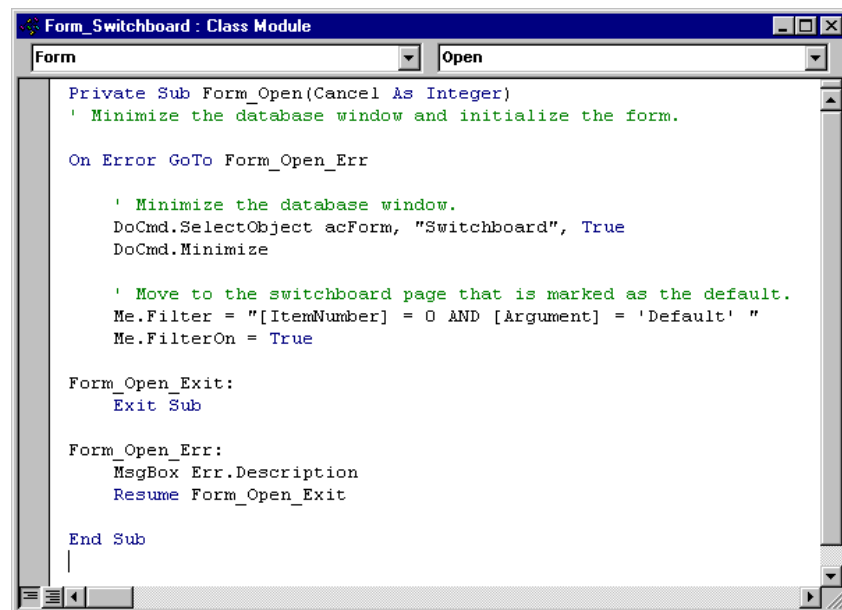
4. Click OK. Notice that the application title bar changes to Expenses immediately. In building the Expenses database, the Database Wizard creates a procedure that runs when the Switchboard form first opens and includes statements that redisplay and minimize the Database window. The procedure runs after

Access starts the database according to the startup settings and, therefore, overrides the startup setting to hide the Database window. In the next few steps, we'll modify the procedure to eliminate the statements so that the Database window remains hidden.

5. Open the Switchboard form in Design view. Open the property sheet by selecting View > Properties. (You could also open the property sheet by double-clicking the square control to the left of the ruler in the design grid.) Click the OnOpen property and then click the Build button to the right of the property box. The Module window displays the event procedure that runs when the form opens (see Figure 1.38). The two lines that follow the comment `Minimize the database window` are the instructions that display and minimize the Database window.
6. Click to the left of the line `Minimize the database window`, drag to select the three lines, and then press Delete.
7. Save the changes and close the Switchboard form.
8. Click the Close box in the Database window. Reopen the database by choosing Expenses in the list of recently opened databases in the File menu. The Expenses application opens with the reduced menus, no toolbars, and no shortcut menus.

FIGURE 1.38:

The instructions to display and minimize the Database window



```

Form_Switchboard : Class Module
Form
Open
Private Sub Form_Open(Cancel As Integer)
' Minimize the database window and initialize the form.

On Error GoTo Form_Open_Err

' Minimize the database window.
DoCmd.SelectObject acForm, "Switchboard", True
DoCmd.Minimize

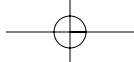
' Move to the switchboard page that is marked as the default.
Me.Filter = "[ItemNumber] = 0 AND [Argument] = 'Default' "
Me.FilterOn = True

Form_Open_Exit:
Exit Sub

Form_Open_Err:
MsgBox Err.Description
Resume Form_Open_Exit

End Sub

```



With these settings, you won't be able to display the Database window by pressing F11 or Alt+F1 after you start the database. Note, however, that you can still bypass the startup property settings by pressing the Shift key when you first open the database. (In Chapter 14, you'll learn how to disable the Shift key bypass as well.)

Protecting Your Application with a Password

You can prevent unauthorized people from opening your application by defining a password. When you protect a database with a password, users must enter the password before they can import its objects into another database or compact the database.

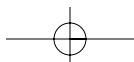
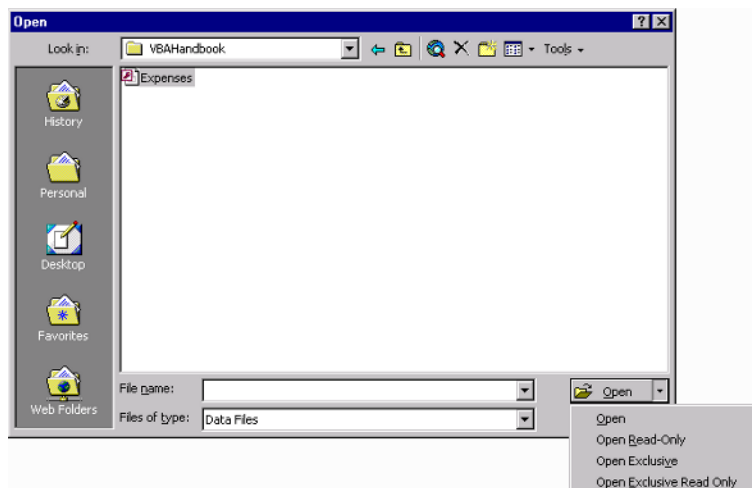
Before you can set a password, you need to establish exclusive access to the database. A database must be closed before you can specify exclusive mode. When you open a database in exclusive mode, you prevent anyone else from opening the database.

We'll define a password for the Expenses database.

1. Close the database by clicking the Exit this database button in the Main Switchboard.
2. Choose File > Open, select Expenses, and click Open Exclusive in the Open drop-down list while pressing the Shift key (see Figure 1.39).

FIGURE 1.39:

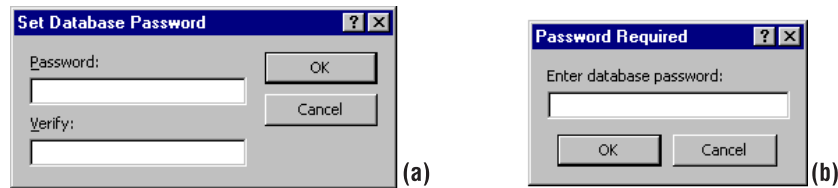
Open a database in exclusive mode before setting a password.



3. Choose Tools > Security > Set Database Password. Use the Set Database Password dialog to set a password (see Figure 1.40a).
4. In the Password text box, type **expenses**. In Access, passwords are case-sensitive. Type **expenses** again in the Verify text box, then click OK. In the Database window, click the Close box.
5. Choose Expenses from the list of recently opened databases in the File menu. Enter **expenses** in the Password Required dialog (see Figure 1.40b). The database opens with the reduced set of menus.

FIGURE 1.40:

Use the Set Database Password dialog (a) to set a password and the Password Required dialog (b) to gain entry to the password-protected database.



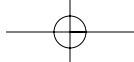
6. For the next section we need access to the full menus, so we'll restart the database. Click the Exit this database button. Press the Shift key and choose Expenses from the list of recently opened databases in the File menu. Enter the password and press the Shift key while clicking OK. The database opens with the full set of menus.

WARNING

Make sure you record your password somewhere. If you forget or lose your password, you can't open the database.

Although adding a password to a database is an easy way to restrict entry to the database, password protection is very limited:

- Anyone with a disk editor or similar utility program can read your data without opening the database. You can prevent this by encrypting the database. To encrypt a database, close the database and choose Tools > Security > Encrypt/Decrypt Database. Select the database in the Encrypt/Decrypt Database dialog and click OK. You will be prompted to supply a name for the encrypted database before the actual encryption occurs.
- To remove password protection, you simply choose Tools > Security > Unset Database Password. Enter the password in the Unset Database Password



dialog and click OK. This means that anyone who knows the password and has access to the Unset Database Password command can change or clear the password.

TIP

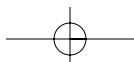
A better way to protect your database is to use the user-level security that Access provides. With user-level security, you can define users and groups of users and specify different levels of access to data and the database objects. For more information about user-level security, see *Access 2000 Developer's Handbook* by Paul Litwin, Ken Getz, and Mike Gilbert (Sybex, 1999).

Creating Custom Menus and Toolbars

We've used the Startup dialog to display a reduced set of menu bars and menu commands and to hide all toolbars and shortcut menus. Although we have achieved some degree of protection for our database, we still haven't made the command environment as helpful as it could be. Menu bars hide their commands; the new user must search through the drop-down menus to learn what commands are available and then remember where the commands are. In addition, the reduced menu bars may not provide the set of commands the user wants. Toolbars are a better way to display the available commands. With toolbar buttons visible at all times and ScreenTips displayed as the user browses through the buttons, your application is easier to learn and to use. Shortcut menus are also helpful; most users are familiar with using the right mouse button to display a shortcut menu.

In Access 2000, you can edit built-in menu bars, shortcut menus, and toolbars, as well as create custom menu bars and toolbars using the Customize dialog. Access 2000 calls these three objects *command bars* and provides common ways to customize them. For example, you can design menu bars that display menus with command buttons and combo boxes, and toolbars that display menu names with drop-down menus. You have enormous flexibility in how you design menu bars, shortcut menus, and toolbars in Access 2000, but you should probably stick to the standard Windows designs to avoid confusing your users.

To display the Customize dialog, right-click in the toolbar and choose Customize from the shortcut menu. The Toolbars tab lists both the built-in toolbars and new toolbars that you create (see Figure 1.41a). A check in front of an item means that the item is currently displayed. You can display any toolbar by clicking its check box. In addition to the toolbars, the list includes Menu Bar as a checked item that



represents the menu bar that is currently displayed. Since the menu bar for every view has a View menu with a Toolbars command, you can display the menu bar you want to work with before opening the Customize dialog.

NOTE

Before you can customize the toolbar, you may need to ensure that this option is set in the Startup menu. To do this, select Tools > Startup and make sure there is a check in the Allow Toolbar/Menu Changes check box.

The Commands tab displays command categories in the list box on the left (see Figure 1.41b). When you select a category, the list box on the right changes to display the commands in the selected category. The Categories list also includes categories for each of the types of database objects except modules and data access pages. If you select one of these categories, the list box on the right displays all of the objects of that type. For example, selecting the ActiveX category, which represents all of the ActiveX controls currently installed and registered on your computer, displays the list of your ActiveX controls in the list box on the right. When you select the New Menu item in the Categories list, the Commands list displays the New Menu command that you use to create a new menu. The Options tab lets you specify additional command-bar options (see Figure 1.41c). For example, you can choose whether to display ScreenTips on toolbars.

The list in the Toolbars tab of the Customize dialog includes the Shortcut Menus item, which represents the shortcut menus. Click the Shortcut Menus item to display a menu bar with menu items for each of the main categories of shortcut menus (see Figure 1.42a). You can click on a menu item to display a drop-down menu that lists all of the shortcut menus available in that category. Click the Form category to display the shortcut menus available when a form is the active window (see Figure 1.42b). Note that there are shortcut menus for each of the form views. Finally, you can select Form View Record from the list to view the shortcut menu for that view (see Figure 1.42c).

Customizing a Built-In Command Bar

To customize a built-in command bar, you must display the command bar when the Customize dialog is open. As explained earlier, you can display any toolbar or any shortcut menu by checking the item in the Toolbars list, but you can only customize the menu bar that is displayed before you open the Customize dialog.

FIGURE 1.41:

Use the Customize dialog to modify existing menu bars, shortcut menus, and toolbars and to create new ones. The dialog has tabs to list the toolbars (a), display most of the built-in commands (b), and display options (c).

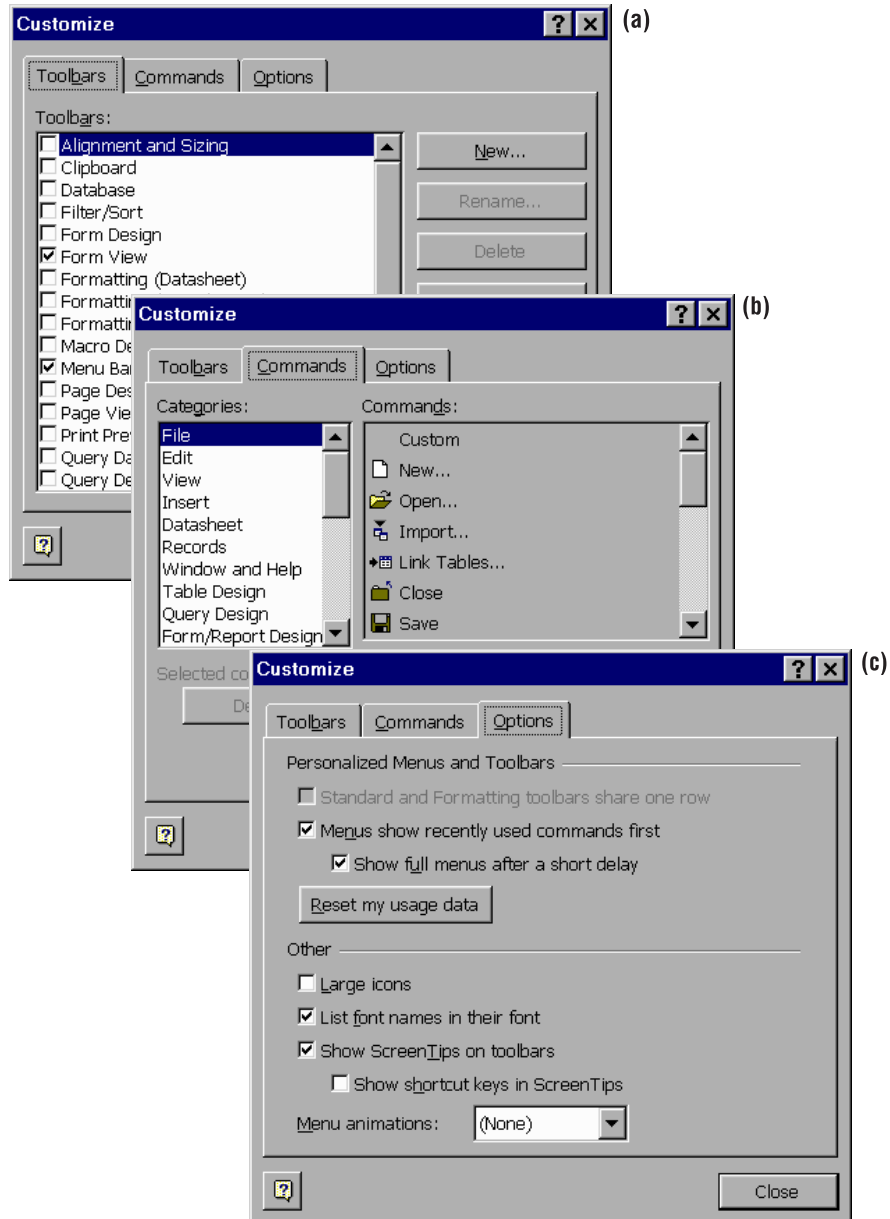
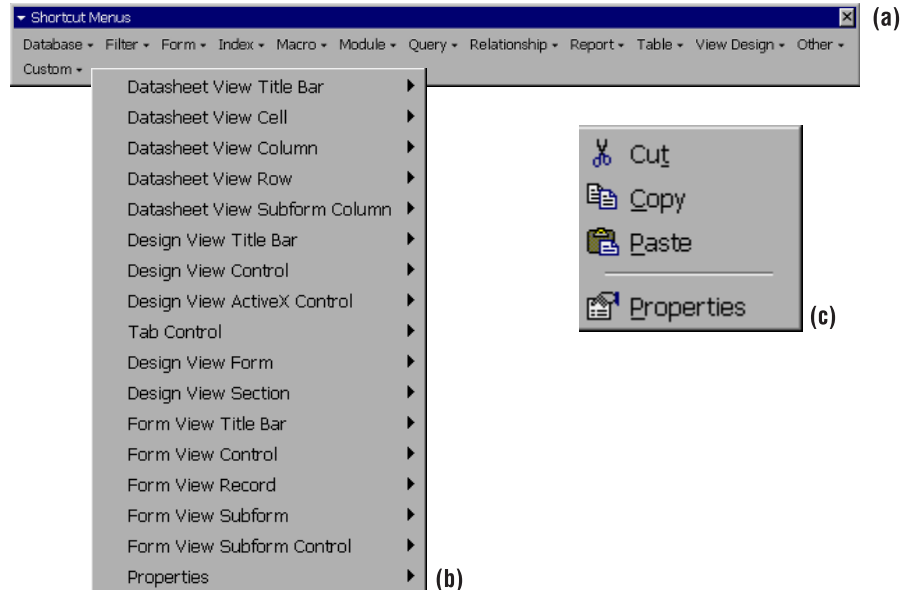


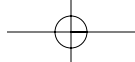
FIGURE 1.42:

The menu bar representing the shortcut menus (a), the shortcut menus when a form is active (b), and the Form Record View shortcut menu (c)



You can customize the displayed built-in command bar as follows:

- To remove a command from any command bar, click the command or command button and drag it off the command bar and into the work area.
- To move a command to a new location on a command bar or to move a command from one command bar to another command bar, click the command you want to move and drag it to its new location.
- To copy a command from one displayed command bar to another displayed command bar, hold down the Ctrl key as you drag the command to its new location.
- To add a built-in command to a command bar, you can copy the command from another command bar or select the command in the Commands list and drag the command to its new location on a command bar.
- To add a command to open a table, query, form, or report, choose the type in the Categories list, select the object in the Commands list, and drag the object to its new location on a command bar. To add a command that runs a program you have created as a macro, choose All Macros from the Categories list, select the macro you want to run, and drag the macro to its new location on a command bar.



- To undo the changes to a command bar, select the command bar in the Toolbars list in the Customize dialog and click the Reset button. Access displays a message asking you to confirm the reset.

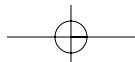
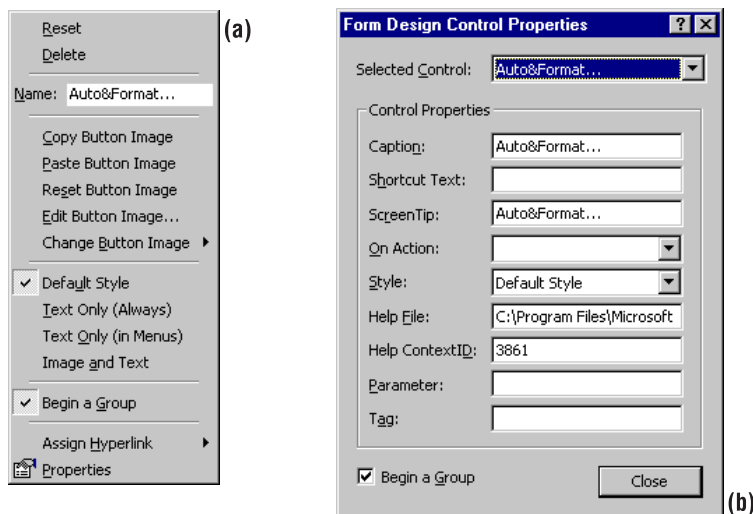
To modify a toolbar button's or menu's name, appearance, or properties, right-click the item to display the shortcut menu (see Figure 1.43a). You can make selections for the following customizations:

- Undo changes to the command, delete the command, or change its name.
- Copy, paste, or modify the button image.
- Specify the command as text, as an image, as both text and image, or as the default style.
- Add a horizontal bar to a menu or a vertical bar to a toolbar to separate commands into groups.
- Assign a hyperlink.

To display the properties for the command, click the Properties command to display the Control Properties dialog for the selected item. For example, Figure 1.43b shows the dialog for the AutoFormat command in the Format window. To run a VBA procedure that you have created as a function procedure (you'll learn about the different kinds of procedures in later chapters), you enter the name of the function procedure in the On Action box using the syntax `=functionname()`.

FIGURE 1.43:

You can modify a command or menu name. Right-click the command with the Customize dialog open to display the shortcut menu (a). Click the Properties command to edit the properties (b).



Creating and Displaying Custom Menu Bars

You can create a new custom menu bar that includes built-in menu commands, commands to open any Database window object, and your own custom commands. You can display a custom menu bar in two ways:

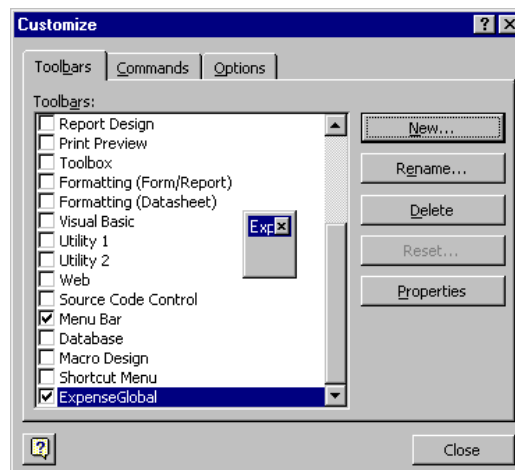
- Create an individual custom menu bar and attach it to a specific form or report. The menu bar is displayed whenever the form or report is the active object.
- Create a global menu bar that is displayed in all windows except for those forms and reports that have their own individual custom menu bars.

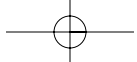
As an example, we'll create and display a new global menu bar. There are several ways to build the menu bar and menus. We'll look at a few techniques in these steps.

1. Choose View > Toolbars > Customize or right-click the menu bar and select Customize from the shortcut menu.
2. Click the New button and type **ExpenseGlobal** in the New Toolbar dialog as the name of the new global menu bar. Click OK. Access creates a small blank command bar and adds the ExpenseGlobal item to the list of toolbars (see Figure 1.44).

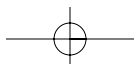
FIGURE 1.44:

Creating a new global menu bar





3. Click the Properties button to display the Toolbar Properties dialog. You use this dialog to specify whether the command bar is a menu bar, a toolbar, or a shortcut menu (choose Popup for a shortcut menu). Choose Menu Bar in the Type combo box and click Close.
4. Hold down the Ctrl key and select the File menu name in the menu bar you are displaying, drag to the blank command bar, and release. When you drag a menu name, you also drag the pull-down menu.
5. Click the File menu on the new menu bar to display the pull-down menu. One by one, select commands and drag them off the menu; leave only the Close, Save As, Page Setup, Print Preview, Print, and Exit commands.
6. Hold down the Ctrl key and select the Edit menu name in the displayed menu bar, drag to the right of the File menu in the blank command bar, and release. Click the Edit menu on the new menu bar to display the pull-down menu. One by one, select commands and drag them off the menu; leave only the Undo, Cut, Copy, and Paste commands.
7. Select New Menu from the Categories list in the Commands tab of the Customize dialog. Click the New Menu command in the Commands list and drag it to the right of the Edit menu on the new menu bar. Right-click in the menu name (New Menu by default) and change the name to **Forms**. Click the new menu name to display the small, blank pull-down menu.
8. Choose the All Forms category in the Commands tab of the Customize dialog, select Expense Reports by Employee, and drag it to the pull-down menu for the Forms menu. Select Expense Categories and drag it to the pull-down menu.
9. Click the ExpenseGlobal toolbar and drag it toward the top of the window to dock the new toolbar below the other toolbars you may be displaying.
10. Select the Toolbars tab and click the Properties button to display the Toolbar Properties dialog. You can use this dialog to allow or prevent menu bar changes. Choose ExpenseGlobal from the combo list and clear all of the check boxes to prevent changing or moving. Click Close, and the Customize dialog appears. Click Close again. The global menu bar is finished.
11. You can display the global menu bar by setting one of the startup properties. Choose Tools > Startup command in the Tools menu. Click the down arrow on the Menu Bar combo box, select the ExpenseGlobal menu bar, and click OK. The new menu bar will be displayed the next time you start up the application.



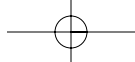
You use the same technique to create an individual menu bar for a form or a report. The only difference between an individual menu bar and a global menu bar is in how you arrange for the menu bar to be displayed. While you specify a global menu bar as a startup property, you specify an individual menu bar for a form or report by setting the form's or report's `MenuBar` property to the name of the menu bar. When you display a form or report that has its own custom menu bar, Access displays the custom menu bar instead of the custom global menu bar (or the default built-in menu bar if your application doesn't have a global menu bar).

Creating and Displaying a Custom Shortcut Menu

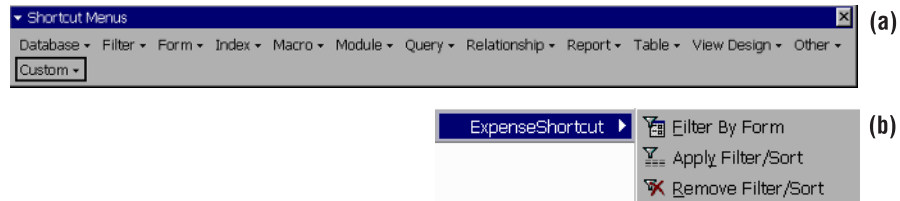
You can create custom shortcut menus for forms, reports, and controls on forms. You can create an individual shortcut menu for each form or report and a global shortcut menu that is displayed whenever the active form or report doesn't have an individual shortcut menu.

As an example, we'll create a simple custom shortcut menu bar that we'll use as a global shortcut menu.

1. Right-click the menu bar and select `Customize` from the shortcut menu.
2. Click the `New` button and name the new command bar **ExpenseShortcut**. Click the `Properties` button in the `Customize` dialog, select `Popup` from the `Type` list box, click `OK`, and click the `Close` button. The small command bar disappears. To build a shortcut menu, you must display the `Shortcut Menus` menu bar.
3. Click in the `Shortcut Menus` item in the `Toolbars` list. Click the small black arrow to the right of the `Custom` menu at the right end of the `Shortcut Menu` menu bar to display a drop-down list containing the `ExpenseShortcut` menu (see Figure 1.45a). Click the small black arrow to display a small, blank fly-out menu where you can build the new shortcut menu. We'll create the menu by dragging copies of the filter commands from the `Form View` shortcut menu.
4. Click the `Form` menu on the `Shortcut Menu` menu bar, click the `Form View Subform` item to display the shortcut menu. Hold down the `Ctrl` key and then, one by one, select the `Filter By Form`, `Apply Filter/Sort`, and `Remove Filter/Sort` commands and drag them to the `ExpenseShortcut` custom menu (see Figure 1.45b). (You drag to the small black arrows to display the menus.)
5. Click the `Close` button to close the `Shortcut Menu` menu bar and the `Customize` dialog.

**FIGURE 1.45:**

Creating a custom
shortcut menu



6. Choose Tools > Startup, select ExpenseShortcut from the Shortcut Menu Bar combo box, and click OK. The next time you start up the application, the custom shortcut menu is displayed when you right-click a form.

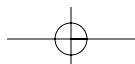
You use the same technique to create an individual shortcut menu for a form, form control, or a report. The only difference between an individual shortcut menu and a global shortcut menu is in how you arrange for the shortcut menu to be displayed. You specify a global shortcut menu as a startup property. You specify an individual shortcut menu for a form, form control, or report by setting the form's, control's, or report's ShortcutMenuBar property to the name of the shortcut menu bar. When you display a form or report that has its own custom shortcut menu, Access displays the custom shortcut menu instead of the custom global shortcut menu (or the default built-in menu if your application doesn't have a global shortcut menu).

Creating and Displaying a Custom Toolbar

Although you can display only one menu bar and one shortcut menu at a time, you can display any number of built-in and custom toolbars. One simple strategy for providing custom toolbars for your application is to hide all of the built-in toolbars (by clearing the Allow Built-in Toolbars check box in the Startup dialog), display a custom global toolbar, and display custom individual toolbars for forms and reports. You can carry out this simple strategy without programming.

As an example, we'll create a simple global toolbar.

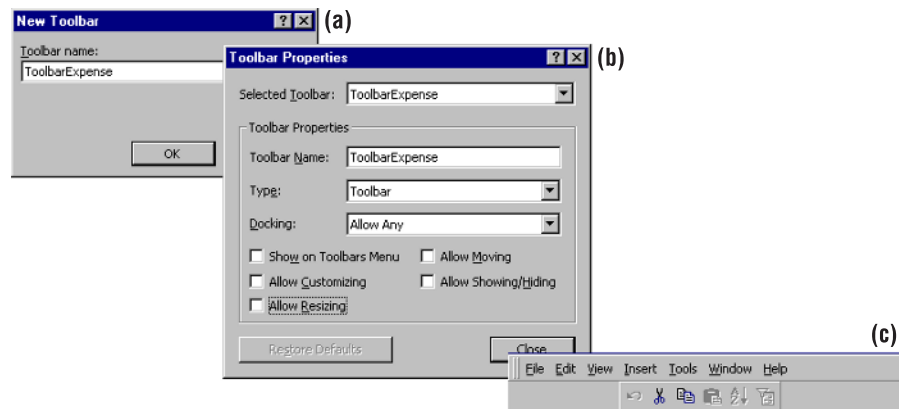
1. Right-click the menu bar and select Customize from the shortcut menu.
2. Click the New button, name the new toolbar **ToolbarExpense** (see Figure 1.46a), and click OK. Access adds the new toolbar to the list in the Toolbars tab.
3. From the Edit category in the Commands tab of the Customize dialog, select and drag the Undo, Cut, Copy, and Paste commands. From the Records category, drag the Sort Ascending and Filter by Form commands.



4. Click the title bar of the new toolbar and drag and dock it below the menu bar.
5. Click the Properties button on the Toolbars tab of the Customize dialog to display the Toolbar Properties dialog (see Figure 1.46b). You use this dialog to allow or prevent menu bar changes. Choose `ToolbarExpense` from the combo list and clear the check boxes to prevent customizing, resizing, or moving the toolbar.
6. Make sure that `Menu Bar` and `ToolbarExpense` are the only checked items in the Toolbars list in the Customize dialog. Click the Close button. Figure 1.46c shows the built-in Database window menu bar and the new toolbar.
7. Click the Close button in the Database window.

FIGURE 1.46:

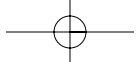
The New Toolbar dialog for naming a custom toolbar (a) and the Toolbar Properties dialog for setting the properties (b) of the new toolbar (c)



Reviewing Our Automated Access Application

This chapter has introduced you to many of the tools that you can use to automate a database, to restrict the menu and toolbar commands, and to protect your application without programming. We've explored the tools by creating the Expenses application. Let's look at the Expenses application and review its current state.

1. Choose the File menu and, from the list of most-recently opened databases at the bottom of the File menu, choose Expenses to open the database. Access displays the Password Required dialog.



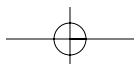
2. Type **expenses** in the text box and click OK. You must use all lowercase letters, because the password is case-sensitive (Access won't accept mixed-case versions, such as Expenses). The Database window is hidden, and the global menu bar and custom toolbar are displayed.
3. Click Enter/View Expense Reports by Employee. Right-click in the form to display the custom global shortcut menu.
4. Click the Return button and then click the Exit this database button on the Main Switchboard.

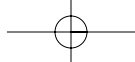
Because the Expenses database has some protection from design changes, you can comfortably turn it over to someone who doesn't know Access. Keep in mind, though, that these protection measures can still be disabled by pressing Shift when entering the password.

You can continue to use the techniques described in this chapter to improve the application. For example, by using the Command Button Wizard, you can make it easy to navigate on two-way paths to all of the application's forms and reports. You can also use the Command Button Wizard to add the other automated tasks in Table 1.1. You can add additional lookup combo boxes using the Combo Box Wizard. You can add forms and reports, and you can use the Switchboard Manager to include the new objects on switchboards. You can create additional custom menu bars, shortcut menus, and toolbars to make your application easier to use. Working with only the Access helpers and the various property dialogs such as the Startup dialog and the Customize dialog, you have the power to create fully automated database applications without going any further in this book. Do you need to read any further? Yes. Read the next section to find out why.

Beyond Wizards and Helpers

The Database Wizard is very powerful, but it can create only databases based on the few templates that come with Access. If your application does not fall into one of those categories, you'll need to build the database from scratch and then use the other wizards and helpers to automate the database. And although it is possible to create a fully automated application with the Access wizards, the applications you can create are still limited in the kinds of operations they can carry out.





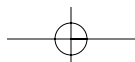
For example, an application created with the wizards limits data validation to the kinds of data-validation rules you can enter in table field properties and form control properties. This means that you can use only a single validation rule to validate a record, even though in some cases you might want to carry out a sequence of validation tests. Also, if the data entered doesn't satisfy the validation rule, you can display only a single message, even though you might want to display different messages depending on the value that was entered.

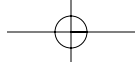
The validation tests are performed according to default timing rules. For example, Access tests the validation rules for a control on a form when you try to tab out of the control and tests the validation rules for a record, such as uniqueness of the primary key value (entity integrity), when you try to save the record. Often, you'll want to change the timing of the validation tests. For example, you may want to test for uniqueness when you tab out of the primary key control instead of waiting until the other data for the record has been entered.

Another example of a database operation you can't set up with the wizards is *transaction processing*. A *transaction* is a set of operations that you handle as a single unit; either you carry out all of the operations in the set or you don't carry out any of them. If you begin the transaction and one operation fails, you roll back the previous operations by returning the data to the state it was in before you started the transaction. A good candidate for transaction processing is the archive process, in which you append records to historical tables and then delete them from the current data tables. Either both operations should occur or neither should occur. Access doesn't provide a transaction processing helper.

Additionally, you may want your Access application to be a component in a complex "mega application" that includes Excel spreadsheets and Word documents and requires Access, Excel, and Word to communicate automatically with each other according to your specific instructions. Or you might want to take advantage of the hundreds of useful procedures in the procedure library that the Windows operating system makes available or the procedures stored in the libraries of other applications (see Chapter 15 for more information about procedure libraries). You can't use the helpers for communication outside Access.

The rest of this book takes you beyond the helpers. You'll create your own programs to automate a database from scratch using the Access VBA programming language. In VBA programming, you create procedures stored either in standard modules listed in the Modules pane of the Database window or in form or report modules that are built into forms and reports. Access 2000 also offers an independent class module.





The part of Microsoft Access that you interact with to create and display the database objects is called the *Access Application*, and the part that manages the data in a database is called the *database engine*. In an Access database, the database engine is called *Jet*. In an Access project, the database engine is known as the Microsoft Database Engine, or MSDE. The two parts of Access communicate with each other using *data-access languages*.

Data-access languages are not stand-alone programming languages; instead, they are special-purpose languages that you use along with a programming language to specify the data you want to retrieve from or add to the tables.

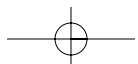
You've been using one of the data-access languages from the beginning of your work with Access, perhaps without being aware of it. Structured Query Language, or SQL (sometimes pronounced "sequel"), is the language that Access uses every time you create a query in Design view. SQL is also the language that the Form, Report, Combo Box, and List Box Wizards often use to specify record sources and row sources. VBA programming uses SQL for working with tables and queries.

Another data-access language is called Data Access Objects, or DAO. The DAO language is used only in VBA programming. You use DAO to write instructions for creating and manipulating the objects that are managed by the Jet database engine. These objects include the tables and queries in your database, but also the objects that Jet uses to restrict access to the data and manage the database. There are two basic roles for DAO:

- To give an alternative method for specifying data (although in most cases the SQL method has better performance)
- To provide the ability to create and modify database objects as part of VBA procedures

In addition to DAO, you will learn about ADO, or ActiveX Data Objects. ADO acts as an interpreter between the Application object and the MSDE.

Most of the time, you create all of the tables and queries in an application using their Design windows. However, there are times when you prefer to create tables and queries programmatically. For example, you use DAO when you write a VBA procedure that creates a new temporary table as part of an automated data-import process.



Summary

This chapter has introduced you to the tools that you can use to automate a database without doing any programming yourself. The tools rely on setting properties in dialogs and property sheets and on using the wizards and helpers to write programs for you. The wizards demonstrate the concept of writing a program as either a macro or a VBA procedure and then arranging for Access to run the program automatically when the user takes an action. The programming environment in Access is event driven. Certain user actions cause controls, forms, and reports to recognize changes called events. Access runs programs when events occur.

Following is a summary of the helpers we've explored:

- The Database Wizard creates partially automated applications, complete with switchboards for navigation, command buttons to open and synchronize forms and reports, and custom dialog boxes that collect user input before selecting records for summary reports.
- The Command Button Wizard creates command buttons with scripts for automating a wide variety of simple database tasks.
- The Combo Box Wizard creates lookup combo boxes that can automatically look up and display a record that matches the value in the combo box.
- The Switchboard Manager creates and modifies a Switchboard form for providing navigation paths to forms, reports, and other switchboards.
- The Startup dialog lets you control how your application starts up and lets you protect your application's design.
- The Customize dialog lets you create custom menu bars, shortcut menus, and toolbars.

You are ready to use programming to provide the automatic connections between the objects in your database—not only the familiar Database window objects, but also additional objects that will be new to you. The next chapter starts you thinking about how objects are designed in Access and about how you can control them with your programs.