

24

The Finishing Touches

You're in the home stretch. You have almost all the forms and reports required for the tasks you want to implement in your application, but you need some additional forms to make it easier to navigate to different tasks. To add a professional touch, you should design a custom menu bar and a custom toolbar for most forms (and perhaps for some reports) and custom shortcut menus for forms, form controls, and reports. You should take advantage of built-in tools to check the efficiency of your design, and you should make sure that none of your forms allow design changes when in Form view. Finally, you need to set the startup properties of your database to let Microsoft Access know how to get your application rolling, and you need to perform a final compile of your Visual Basic code to achieve maximum performance.

Creating Custom Menus and Toolbars

When your application is running, the user probably won't want or need some of the Access design features. However, you might want to provide some additional toolbar buttons on your form toolbar and menu bar so that the user has direct access to commands such as Save Record and Find Next. For example, open the LawTrack Contacts sample database (*Contacts.mdb*), open the *frmContactsPlain* form in Form view (which uses the standard Form View toolbar), right-click the toolbar, and select Custom Form Toolbar. You can see some useful differences between the two toolbars, as shown in Figure 24-1. Buttons the user won't need (such as Form View, Save, Print, Print Preview, Format Painter, and New Object) aren't available on the custom toolbar. (None of the forms in the LawTrack Contacts database are designed to be printed.) However, the

Custom Form Toolbar does have Close and Save Record buttons added at the left end, and the New Record and Delete Record buttons are clustered with the new Save Record button. In the LawTrack Contacts application, all forms (except *frmContactsPlain* and a few other example forms) have their Toolbar property set to use the custom toolbar.

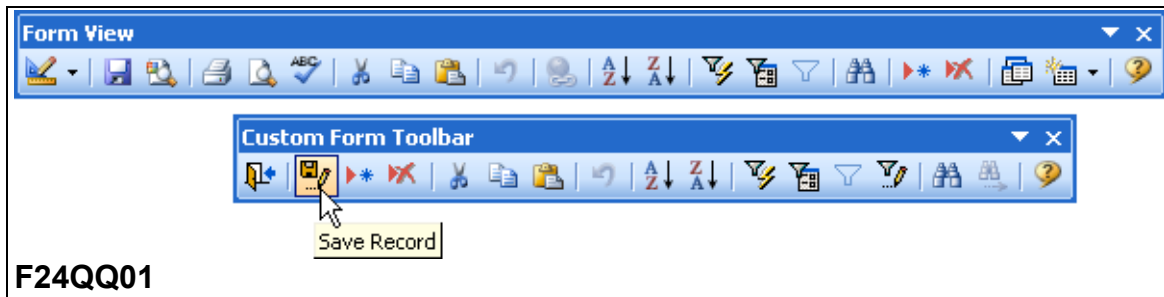


Figure 24-1.

Comparing the standard Form View toolbar (top) and the Custom Form Toolbar (bottom) from the LawTrack Contacts sample database.

The same is true of the built-in menu bars and shortcut menus. For example, you don't want your users to be able to select Design view from any of the built-in menus. For most forms, you also don't want the user to be able to switch to PivotTable or PivotChart view. The following sections show you how to build custom menus and toolbars.

Understanding the Customize Toolbar Facility

To create a new toolbar or modify an existing one, you must enter toolbar customize mode. Open the Customize dialog box by choosing Toolbars from the View menu and then choosing Customize from the submenu. (You can also right-click on any open toolbar to open the toolbar shortcut menu and then choose Customize from that menu.) The Customize dialog box with the Toolbars tab selected is shown in Figure 24-2.

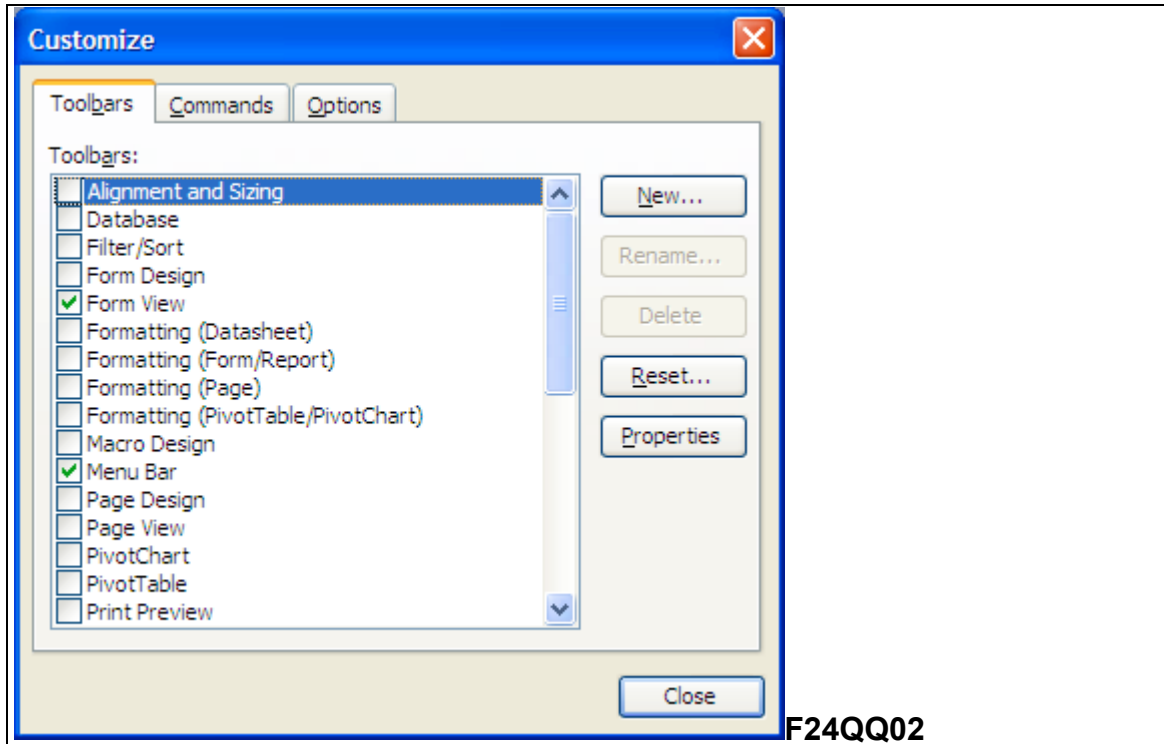


Figure 24-2.
The Toolbars tab of the Customize dialog box.

On the left side of the Customize dialog box, you can see the names of all the built-in toolbars in Access. A check mark next to a toolbar name in this list indicates that the toolbar is currently open. You can open any of the toolbars by selecting the check box next to the toolbar name.

The term toolbar refers generically to toolbars, menu bars, and shortcut menus. As you'll learn later, you can set the Type property of a custom toolbar to define whether it appears as a toolbar, menu bar, or shortcut menu.

Near the end of the list, you'll see three special built-in toolbars that Access displays only if you select them in this dialog box: Shortcut Menus, Utility 1, and Utility 2. The two utility toolbars are empty, so you can add buttons of your choice to them to create custom toolbars that are available in all the databases you open on your machine.

The Shortcut Menus toolbar is a “placeholder” for all shortcut menus, including custom ones that you have defined for your application. You’ll learn how to create a custom shortcut menu later in this chapter. In the LawTrack Contacts sample database, you can see a Custom Form Menu Bar, a Custom Form Toolbar, a Custom Print Menu, and a Custom Report Toolbar already defined at the bottom of the list.

If you open one of the built-in toolbars in a context in which the toolbar would not normally be open, the toolbar remains open until you close it. For example, if you open the Customize dialog box while the focus is on the Database window and then open the Form Design toolbar, the toolbar remains open no matter what you are doing in Access. Likewise, if you close a toolbar in a context in which that toolbar is normally open (for example, if you close the Formatting toolbar in a Form window in Design view), that toolbar remains closed until you open it again within the usual context or from the Customize dialog box.

***[new_gear]*Inside Out**

Setting options that affect all Office applications

In the Customize dialog box, you can click the Options tab to see options you can set that affect all Microsoft Office applications. By default, none of the Office applications show you all the commands available when you first click a menu. Instead, Office keeps track of which menu commands you use most frequently (known as adaptive menus) and shows you those first. When you leave the **Show full menus after a short delay** option set (the default), Office applications reveal the hidden menu items if you leave the menu open for a second or two. If you clear the option to reveal all commands after a short delay, you can still click the arrow at the bottom of the menu command list to show all commands. If you leave the shortened menu feature enabled,

you can select **Reset menu and toolbar usage data** if you think the menus aren't showing your most frequently used commands when you first open them.

You can select the **Always show full menus** option to disable these features. As an application developer, I like to see all the menu commands whenever I open a menu, so this is always the first option I reset after installing a new version of Microsoft Office.

Also on the Options tab are check boxes that you can select to display large icons in buttons on toolbars, to list font names in their font, to display ScreenTips, and to display shortcut keys in ScreenTips. If you're working on a large monitor at a high resolution (1024×768 or 1280×1024), you might find the larger toolbar icons easier to see. The buttons with large icons are approximately 50 percent wider and taller than the standard buttons. There's also an option on this tab to animate your menus in various ways when you open them—interesting effects if you have a fast graphics card on your PC.

Any new toolbar that you define is available only in the database that you had open at the time you created the toolbar. If you want to define a custom toolbar that is available in all the databases that you work with on your computer, you must modify one of the built-in toolbars. You can also use the two blank toolbars—Utility 1 and Utility 2—to create a custom set of toolbar buttons that are available in any database on your machine. One drawback to these two toolbars is that you cannot give them custom names. Also, any changes that you make to these toolbars on your machine won't move with your database to another machine. Any change you make to any built-in toolbar (including Utility 1 and Utility 2) is effective only for your computer. If you have made changes to one of the built-in toolbars or menu bars, you can select it in the Customize

dialog box and click the Reset button to return the toolbar to its default. Access prompts you to confirm this action so that you don't inadvertently erase any custom changes you've made.

Creating a Custom Form Toolbar

You can create a custom toolbar for your forms to remove some of the built-in features and add other commands that your user might find useful. You can also create a custom toolbar for reports, and the design process for either is the same. The following sections show you how to create a custom form toolbar.

Defining a New Toolbar

Click the New button on the Toolbars tab in the Customize dialog box to begin defining a new toolbar, and Access prompts you for a name for your new toolbar. If you want to follow the example in this section to create a new toolbar in the LawTrack Contacts sample database, provide a name like **Sample Custom Toolbar** in the New Toolbar dialog box and click OK. You'll see the name appear at the bottom of the Toolbars list, and an empty toolbar in the form of a small, gray window will open in the Access workspace. Select your new toolbar in the Toolbars list, and click the Properties button to open the dialog box shown in Figure 24-3.

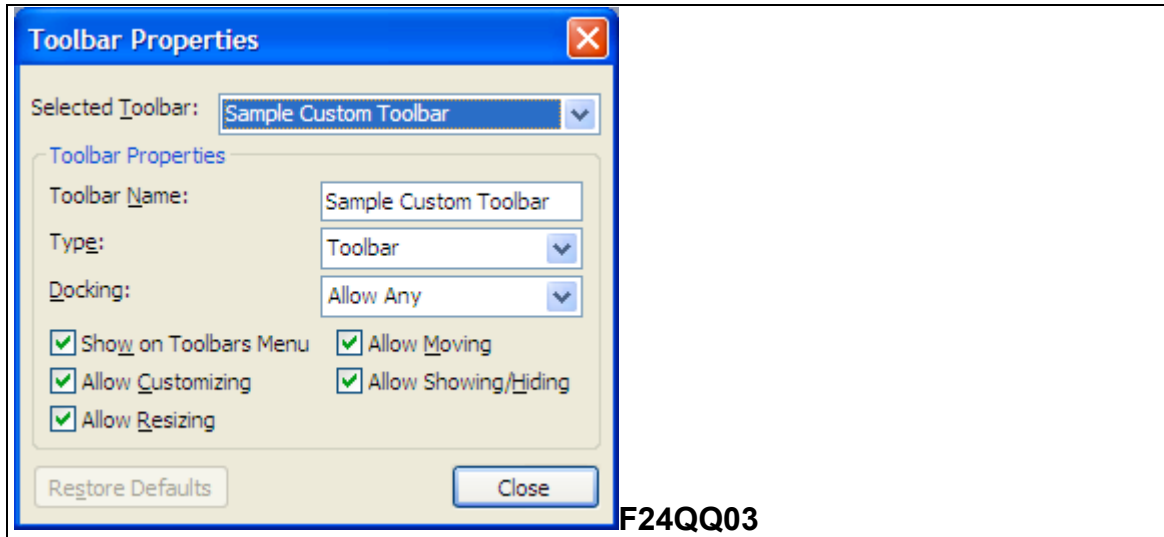


Figure 24-3.
The Toolbar Properties dialog box.

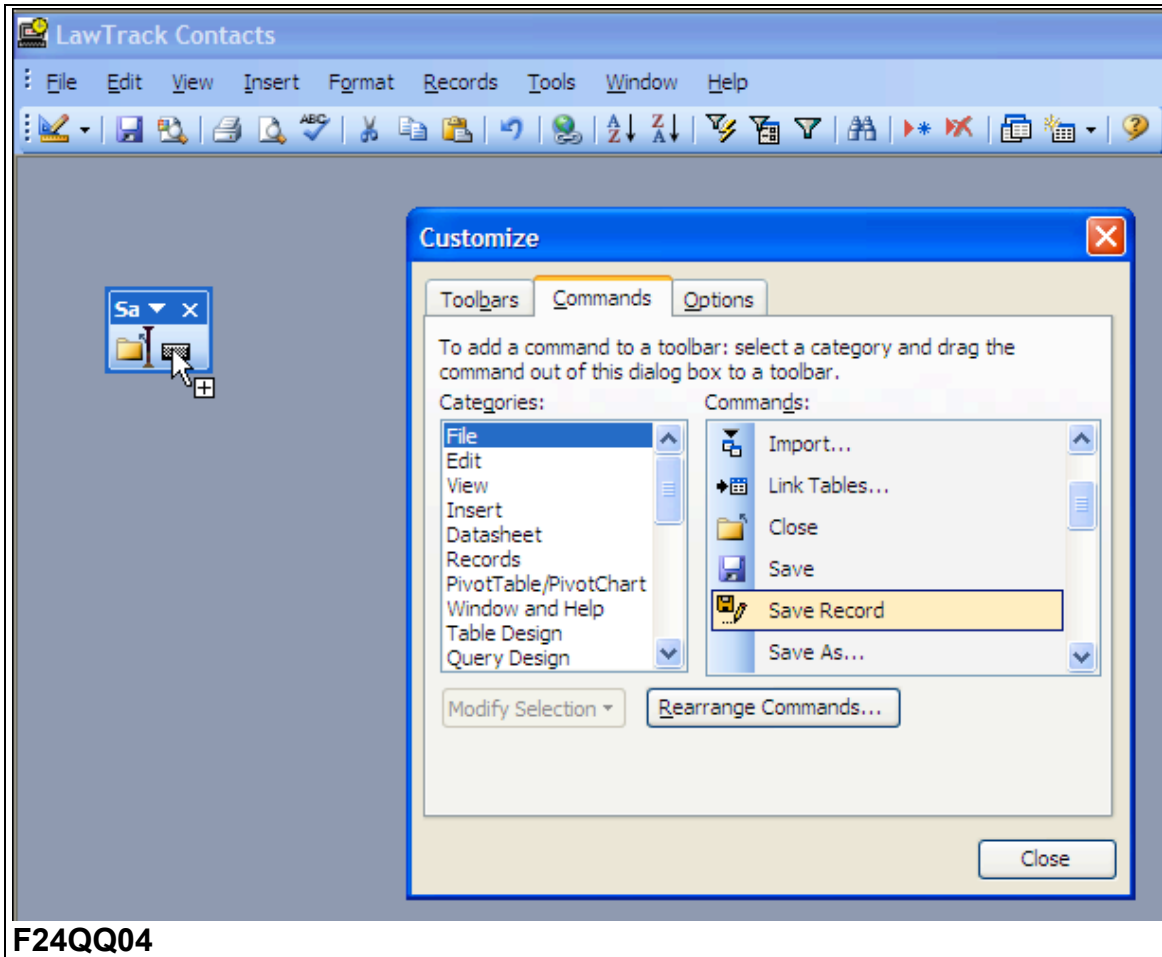
If you want to build a custom toolbar to use with one or more of your forms, it's a good idea to first open one of the target forms in Form view because this makes the built-in form menu and toolbar available. As you'll see below, it's often easiest to copy the commands or toolbar buttons you want from built-in menus or toolbars rather than build them from scratch. In all of the examples in this section, I have the frmContactsPlain form in the LawTrack Contacts database open in Form view.

In the Toolbar Properties dialog box, you can select any toolbar from the Selected Toolbar drop-down list. If you select a custom toolbar (not a built-in one), you can rename it and set its type—Menu Bar to define a custom menu, Toolbar to define a custom toolbar, and Popup to define a custom shortcut menu. Because we're building a toolbar, be sure Type is set to Toolbar. If you want to, you can position your toolbar (you'll have to click Close in the Toolbar Properties dialog box, position the toolbar, and then choose Properties again in the Customize dialog box) and then restrict where the user can move it. For example, you can dock the toolbar at the bottom of your screen and

then set Docking to either Can't Change or No Vertical. You can also dock the toolbar on either side and choose No Horizontal.

For custom toolbars, you can select **Show on Toolbars Menu** to make the toolbar available in the list of toolbars displayed from the View menu's Toolbars command or when the user right-clicks any toolbar or menu bar. You can't change this option for built-in toolbars. You can choose from additional options that determine whether this toolbar can be customized, resized, moved, or hidden. Finally, for built-in toolbars, you can click the Restore Defaults button to undo any changes you made. (This button appears dimmed in Figure 24-3 because a custom toolbar is selected.) Close this dialog box to go to the next step.

Click the Commands tab in the Customize dialog box to display the list of available commands, shown in Figure 24-4. The custom toolbar in the LawTrack Contacts sample database starts with a Close button and a Save Record button at the left end of the toolbar. You can find the Close command partway down the Commands list in the File category. As you can see, the Save Record command is located two down from Close. When you find the command you want in the Commands list, drag it onto your new toolbar. Notice that the icon for the custom Close command doesn't match the one on the Custom Form Toolbar in the LawTrack Contacts application. You'll learn how to change the icon later.



F24QQ04

Figure 24-4.

The Customize dialog box and command category buttons; the custom toolbar under construction is open at the left.

The next several buttons for the LawTrack Contacts custom toolbar are all available on the built-in Form View toolbar. You can copy them from this toolbar rather than hunt for them in the Commands list. You can drag buttons from one toolbar to another, but when you do that, you're *moving* the button. In other words, you're deleting the button from the built-in toolbar and adding it to your custom toolbar. To copy a button from an existing toolbar, press the Ctrl key and hold it down while you drag the button. When you're copying a button, Access displays a small plus sign in a white box

next to your mouse cursor, as shown in Figure 24-4. A large I-beam appears on the receiving toolbar to indicate where you're dropping the new command.

If you didn't open a form before starting to build your toolbar, you can also force open the built-in Form View toolbar by selecting it in the Toolbars list on the Toolbars tab of the Customize dialog box. Unless you're building the toolbar while a form is open in Form view, some buttons appear dimmed (disabled) and some buttons are not visible. As soon as you open a form in Form view, the appropriate buttons become available. If you force the Form View toolbar open, remember to close it before exiting the customize facility or it will stay open all the time.

If you want to duplicate the Custom Form Toolbar in the LawTrack Contacts database, copy the following buttons from the built-in Form View toolbar or the Commands tab of the Customize dialog box in this order (starting to the right of the Close and Save Record buttons): New Record, Delete Record, Cut, Copy, Paste, Undo, Sort Ascending, Sort Descending, Filter By Selection, Filter By Form, Apply/Remove Filter (labeled Toggle Filter in the Records category), Advanced Filter/Sort, Find, Find Next, and Help. (You won't see these buttons on the Form View toolbar unless you also have a form open.) You can find the Advanced Filter/Sort command in the Records category and the Find Next command in the Edit category.

Customizing Your New Toolbar

After you build a toolbar, you can rearrange the buttons and add dividing lines between them. You can also change the button image, the label in the ScreenTip, and the button style. Finally, and perhaps most importantly, you can define a custom macro or function that you want Access to execute when the user clicks the toolbar button. If

you're not still in toolbar customize mode, choose Toolbars from the View menu and then choose Customize from the submenu to open the Customize dialog box, shown in Figure 24-2. (Access opens the dialog box and displays the last tab you selected, so you might see the Commands or Options tab instead.)

When you open the Customize dialog box, all toolbar buttons become editable.

You can

- Remove any button from any open toolbar (including any built-in toolbar) by clicking the button and dragging it off its toolbar.
- Move any button by clicking it and dragging it to a new location on its toolbar or another toolbar.
- Copy a button from one toolbar to another by holding down the Ctrl key while you drag.
- Add a button by dragging it from the Commands list to any toolbar.
- Right-click a button to open a shortcut list of properties that you can modify to change the look of the toolbar button.
- Right-click a button and choose Properties from the shortcut list to view all the properties for a button and define a custom action for a toolbar button.
- Click a button and then click Modify Selection on the Commands tab of the Customize dialog box to open the properties for the selected button.
- Click the Rearrange Commands button in the Customize dialog box to select any menu or toolbar and add a new command, delete a command, or move commands.

Creating Button Images

To make your custom toolbar look just like the Custom Form Toolbar in the LawTrack Contacts database, you first need to change the Close button to show a

different image. If there's another button that has an image you want to use, you can right-click that button to open the shortcut menu and choose Copy Button Image to place that button's bitmap on the Clipboard. Move to the button you want to change, right-click, and choose Paste Button Image. (See Figure 24-5.)

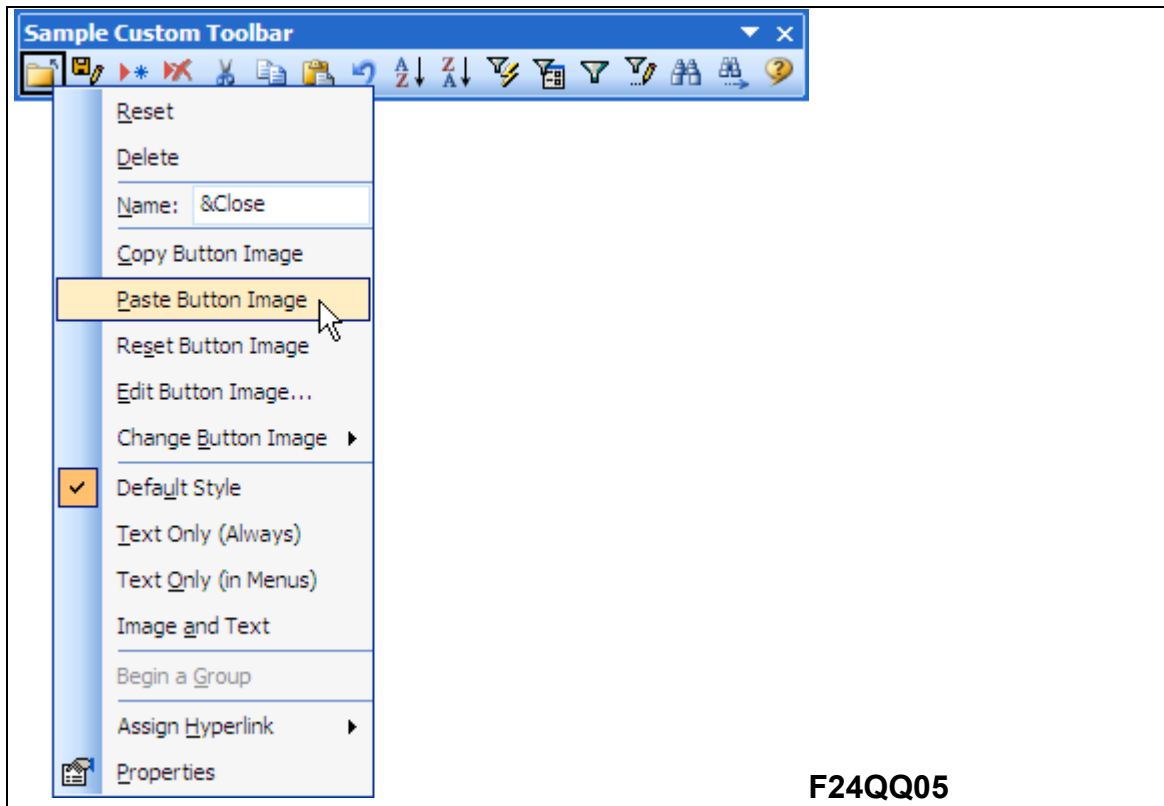


Figure 24-5.
Pasting a custom button image onto a toolbar button.

If you're good at visualizing button images by setting individual pixel colors on a 16-by-16-pixel square, choose Edit Button Image from the shortcut menu to open a simple design window for the bitmap image. Finally, if you have a 16-by-16-pixel bitmap or icon file that you want to use as a button image, open that file in an image-editing program (for example, Microsoft Paint on the Windows Accessories menu of the Start button), copy the bitmap to the Clipboard, and then choose Paste Button Image from the

shortcut menu. In this case, I used an arrow pointing to an open door. You can find this image on the companion CD, saved as [CloseArrow.bmp](#).

Arranging Buttons

To make buttons easier to use, it's often useful to cluster buttons that perform similar functions by adding a dividing line between those clusters. To create a dividing line to the left of any button, right-click the button and turn on the Begin A Group property near the bottom of the shortcut menu. (Access makes this property unavailable for the first button or menu on a toolbar.) On the Custom Form Toolbar in the LawTrack Contacts application, you can find a dividing line before the Save Record, Cut, Undo, Sort Ascending, Filter By Selection, Find, and Help buttons.

Assigning Custom Actions

Last, but not least, you can define a custom macro or function that you want Access to run instead of the built-in action whenever the user clicks the button. In the LawTrack Contacts application, every form that edits data has a Save command button (cmdSave), and code in the Click event of this button calls a save routine specific to the form. The cmdSave_Click event in each form is public so that it can be called as a method of the form. In the modUtility module, you can find a public function called SaveRecord that finds the currently active form (if any) and then calls that form's cmdSave_Click event. The code is as follows:

```
Function SaveRecord() As Variant
'-----
' Called from a custom menu or toolbar Save Record
' attempts to find the public cmdSave_Click
' procedure in the currently active form and
```

```
' execute the save by calling that form's procedure.  
' If there is no active form the function exits.  
' Created By: JLV 06/09/03  
' Last Revised: JLV 06/09/03  
' -----  
' Place to save the currently active form  
Dim frm As Form  
    ' Skip any error  
    On Error Resume Next  
    ' Attempt to get the active form  
    Set frm = Screen.ActiveForm  
    ' If error, bail  
    If Err <> 0 Then Exit Function  
    ' Now try to call the form's save procedure  
    frm.cmdSave_Click  
    ' Done  
    Err.Clear  
End Function
```

The Save Record button (and the Save Record command on the custom form menu) calls this function so that the custom record save code for each form executes instead of the standard Access command. The code uses the ActiveForm property of the Screen object to find the form that currently has the focus, and then it calls the cmdSave_Click method of the form. So, instead of simply executing the standard Access save record, this causes a custom save procedure for the form to execute instead—which allows you to code additional tests specific to each form before finally saving the record.

To define a custom action for a toolbar button, right-click that button to open the shortcut menu and then choose Properties. You'll see the Control Properties window, as shown in Figure 24-6. (Access uses the name of the toolbar as the first part of the window name.)

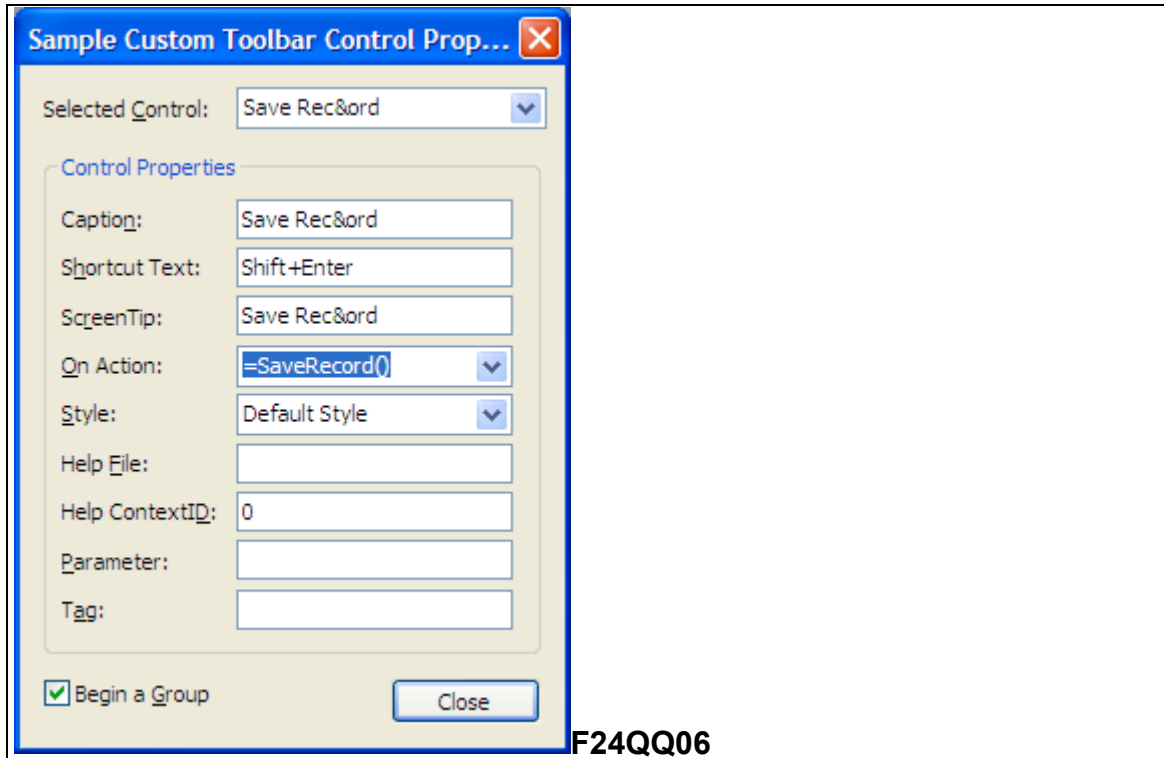


Figure 24-6.

The Control Properties window, where you can define a custom action for the control.

In the Control Properties window, you can choose any control on the toolbar from the Selected Control list at the top of the window. As you can see, you can redefine the caption for controls that display text and change the ScreenTip text if you have ScreenTips enabled. When the command is on a menu, adding an ampersand (&) before one of the letters in the Caption box establishes the access key for that menu item (the letter appears underlined); however, commands on toolbars do not have an access key. When you copy a command from the Commands list, Access includes both a caption and a ScreenTip in case you place the command on a menu bar. The Shortcut Text property also applies only to controls you define on menus.

You can change the style of the control—**Default Style** (an icon only on a toolbar or an icon and text on a menu), **Text Only (Always)**, **Text Only (in Menus)**, or **Image**

and Text. In addition, you can define a help file and help context for the command. (If you've copied a built-in command as shown in Figure 24-6, you probably want to leave the original help pointers set for the built-in help topic.) The Parameter and Tag properties are advanced settings for programmers who build their own command bars and commands using Visual Basic.

The critical setting in this window is the On Action property. You can set this property to a macro name or specify a public function by entering an equals sign (=) followed by the name of the function. This works exactly like the event properties in forms and controls. In this case, you want to call the SaveRecord function, and the function requires no parameters, as shown in Figure 24-6.

After you finish building your toolbar, you can use it with any form by opening that form in Design view and setting the Toolbar property to point to this custom toolbar. You can also open any report in Design view and set the Toolbar property to specify the toolbar you want displayed when the report is opened in Print Preview. I like to design my application with one custom toolbar for all forms and another for all reports. However, you can define a different custom toolbar for each form and report if you like.

Creating a Custom Form Menu Bar

After you build a custom form (or report) toolbar, it would be helpful (and consistent) to replace the built-in menu bar with a custom menu bar. You can then set the Menu Bar property of your forms (or reports) to point to the custom menu bar. For example, the Custom Form Menu Bar in the LawTrack Contacts sample database, as shown in Figure 24-7, has a limited set of commands that matches the functionality of the

custom toolbar described in the previous section. When you open the File menu, you can see that the commands a user shouldn't need, such as New Database, aren't available on the custom menu.

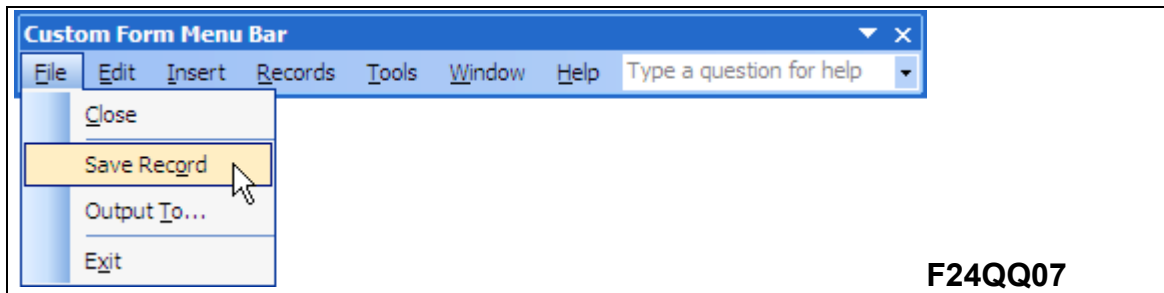


Figure 24-7.

The custom File menu on the Custom Form Menu Bar eliminates unneeded commands, such as Open Database, New Database, Save, Export, and Database Properties.

If you or your user chooses to leave adaptive menus turned on (by entering customize mode and leaving **Always show full menus** turned off on the Options tab), this won't affect the design of your custom menus. You can still design full menus for your application. With this option turned on, Access shows only the most recently used options first on your custom menus.

Prior to Microsoft Access 97 (version 8), the only way you could define a custom menu bar was to construct a complex series of macros. You needed one macro consisting of AddMenu actions to define the basic menu bar. You also needed an additional macro for each drop-down menu or submenu in that menu bar. You might have used as many as a dozen macros just to define one custom menu bar.

Access 2003 still supports AddMenu macros for backward compatibility, but it's much easier and more flexible to define a menu using the customize facility. For several

of the standard menus, such as Edit, Insert, Records, and Window, you can copy the built-in menu “as is.”

One of the advantages of copying built-in menus is that you automatically get any additional commands that appear on these menus in particular situations. For example, if you copy the built-in Edit menu, when the focus is on an OLE object, you will see OLE object editing commands at the bottom of the Edit menu on your custom menu bar. If you were to build these menus on your own, you would first have to place the focus on a control that activates these commands before opening the Customize dialog box and then copy the commands from the built-in Access menus. (No, you can’t find these commands in the Edit category on the Commands tab.) If you build a custom menu on your own and copy selected focus-sensitive commands from the built-in menu, you still get the desired effect of having the commands appear only when the focus is on an appropriate object. As another example, if you copy the built-in Window menu, you get the built-in list of open windows at the bottom of your custom Window menu.

To start building a custom menu, be sure you have the frmContactsPlain form open in Form view. (This reveals all the built-in commands on the built-in menu bar.) Right-click any menu bar or toolbar, and select Customize from the shortcut menu. Click the New button on the Toolbars tab, name the menu bar **Sample Custom Menu Bar** in the New Toolbar dialog box, and click OK. Click the Properties button, and in the Toolbar Properties dialog box, set the Type to Menu Bar. Close the dialog box to go to the next step.

Next, click the Commands tab in the Customize dialog box, and scroll to the bottom of the Categories list. The custom menu bar in the LawTrack Contacts database (Custom Form Menu Bar) uses the original Edit, Insert, Records, and Window menus, but it has customized File, Tools, and Help menus, and it doesn't include the View or Format menus at all. To make your new menu match, select the New Menu category, which provides one special entry that allows you to start a new menu on the menu bar. Drag the New Menu command from the Commands list to your menu bar three times, once for each of the custom menus you need to build. Right-click each New Menu item, and change its name to reflect the menu name you want, as shown in Figure 24-8. Adding an ampersand (&) before one of the letters in the menu name establishes the access key for that menu item, much like using an ampersand in the Caption property of a form label or command button.

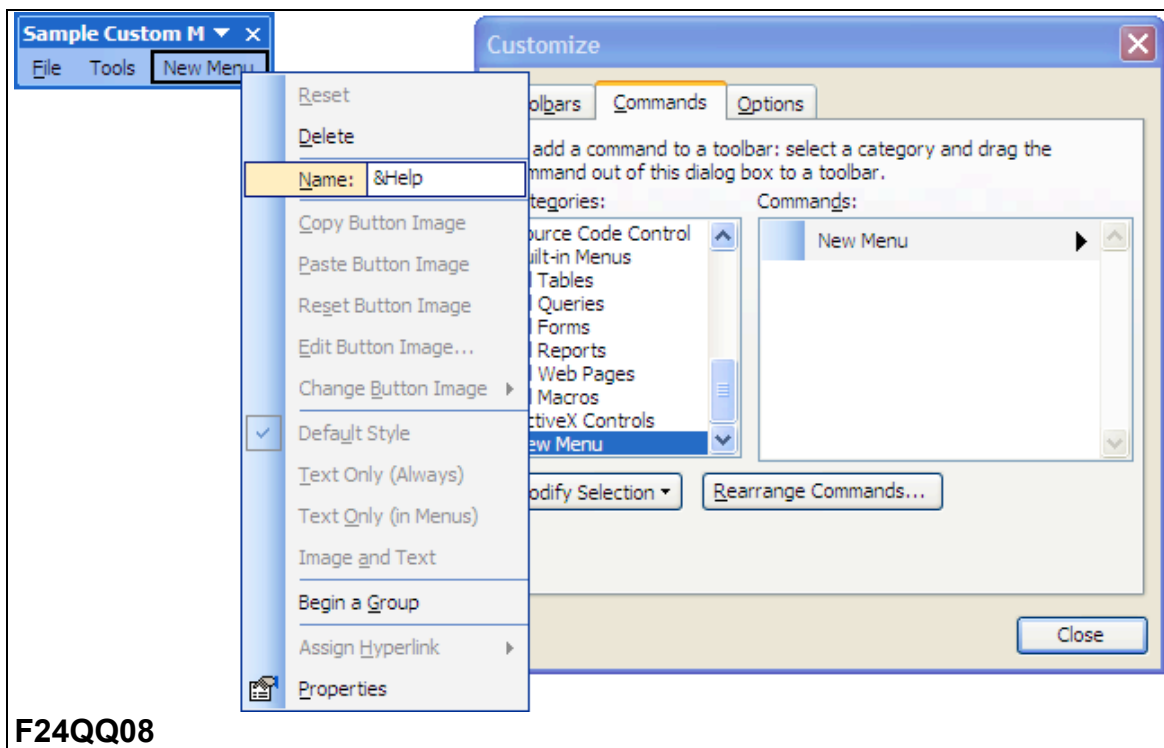


Figure 24-8.

Using the New Menu command to set up custom menus on your custom menu bar.

You don't want to customize the built-in Edit, Insert, Records, or Window menus in this exercise. You can copy them either from the built-in menu bar named Menu Bar (probably docked right above the built-in toolbar) or from the Built-In Menus category on the Commands tab of the Customize dialog box. Be sure you place the menus in the same order as the standard Access menu bar (File, Edit, Insert, Records, Tools, Window, Help) so that your application is consistent with what a user is familiar with seeing in other applications. If you copy the menus from the built-in menu bar, be sure to hold down the Ctrl key! If you don't hold down the Ctrl key, you will remove them from the built-in menu bar and place them only on your custom menu bar.

When you copy a built-in menu (either from an available menu bar or from the Built-In Menus category), Access does *not* make an independent copy of the original. So, if you copy a menu and then change some of its properties (for example, you delete one of the commands from the menu), you're also affecting the built-in menu. This is why you must use the New Menu command to build the custom File, View, Tools, and Help menus in this example. If you were to copy the originals and delete unwanted commands from the copies, you'd be deleting them from the built-in menus as well.

Now you can add the commands you want to your three custom menus. You can copy commands either from the Commands list or from another menu bar. Remember to hold down the Ctrl key when you copy a command from another menu bar so that you *copy* the command, not *move* it. When you drag a command to a menu bar and hover over one of the menu commands, Access drops down the current list of commands. You can then move the cursor down the list. Access shows you a horizontal bar on the drop-

down menu where it will place your command if you release the mouse button at that point. It takes a bit of hand-eye coordination, but it's not too difficult. You can always grab a command and move it if you drop it in the wrong place. Figure 24-9 shows adding the Office Links command to the Tools menu just below the Spelling command.

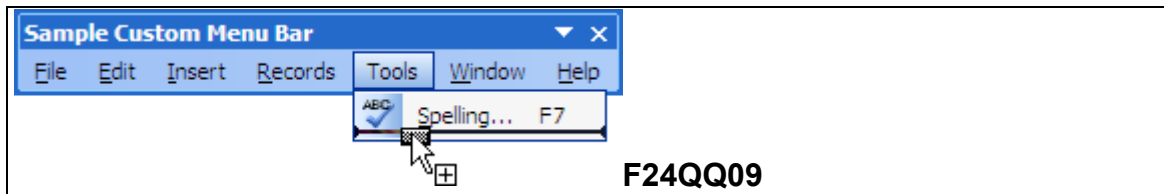
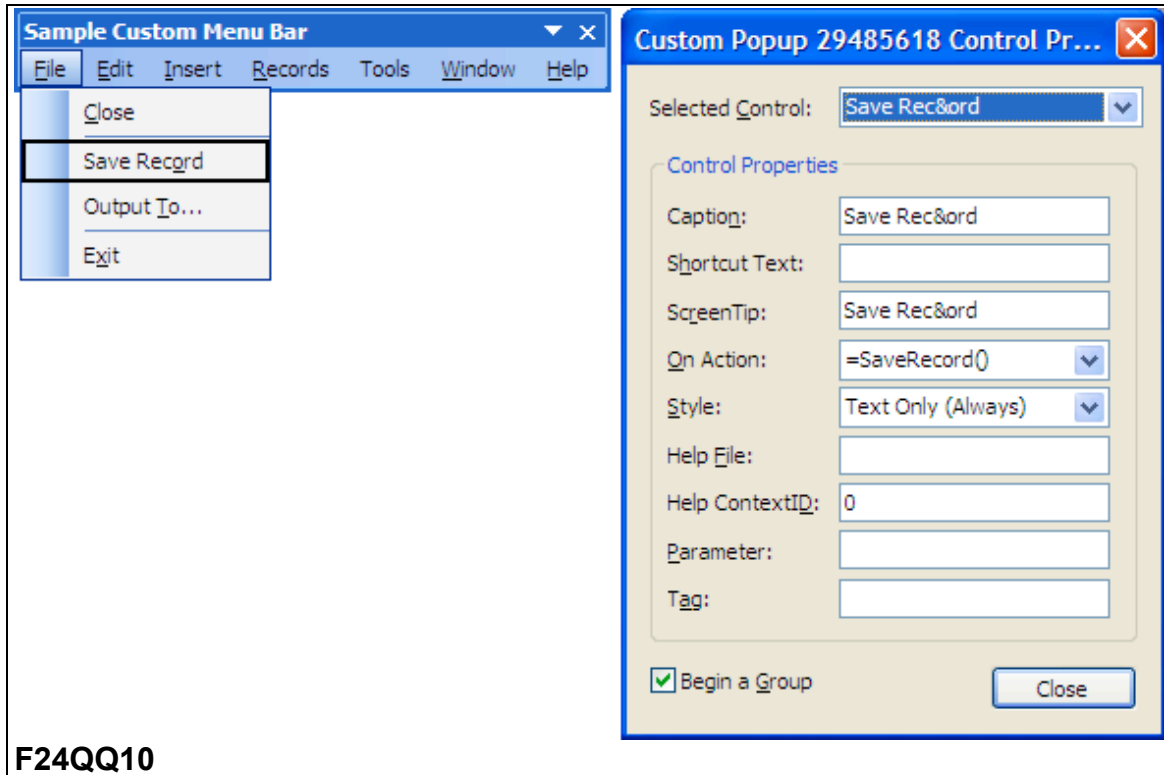


Figure 24-9.
Adding the Office Links command to a custom Tools menu.

When you copy a command from one menu to another, Access resets the command's Begin A Group property to off. You might want to turn this property back on for commands that normally begin a group, such as the AutoCorrect Options command on the Tools menu that you'll find on the Custom Form Menu Bar in the sample databases.

To create the custom menu bar in the LawTrack Contacts sample database, you'll need to place the Close, Save Record, Output To, and Exit commands on the File menu and the Spelling, Office Links, and AutoCorrect Options commands on the Tools menu. To further customize the Save Record command, right-click it and choose Properties from the bottom of the shortcut menu to display the Control Properties dialog box, as shown in Figure 24-10. Set the On Action property to point to the SaveRecord function so that the menu command works just like the custom toolbar button you created earlier.



F24QQ10

Figure 24-10.

Customizing the Save Record command on your custom menu bar.

On the custom Help menu on the Custom Form Menu Bar, you can find commands for Microsoft Access Help and About LawTrack Contacts (which opens the frmAbout form). To make your new menu match, copy the Microsoft Access Help command from the built-in menu bar or from the **Window and Help** category on the Commands tab of the Customize dialog box. Next, go to the All Forms category on the Commands tab of the Customize dialog box, and drag the frmAbout form onto the drop-down portion of your Help menu. Open the properties of the frmAbout command, change its Caption to **About LawTrack Contacts...**, and change its Style to **Text Only (Always)** to remove the form icon.

When you're finished, you can place the name of your custom menu bar in the Menu Bar property of each form that you want to open with your custom menu bar

displayed instead of the built-in menu bar. As with custom toolbars, you can design different custom menu bars for each form or report if you like. Open a form in Form view, and you can see both the custom menu bar and custom toolbar, as shown in Figure 24-11.

The screenshot shows the 'LawTrack Contacts' form in Microsoft Access 2003. The form has a custom menu bar with the following items: File, Edit, Insert, Records, Tools, Window, and Help. Below the menu bar is a custom toolbar with buttons for Close, Cancel, Save, Add New, Search..., Edit All, and Print. The form itself is titled 'LawTrack Contacts' and displays contact information for François Ajenstat. The form is divided into several sections: Contact Info, Notes, Companies, Events, and Products. The Contact Info section includes fields for Contact Type (Trainer), Title (Mr.), Last Name (Ajenstat), First Name (François), Commission (0.00%), and a photo. There are also sections for Work Contact Info and Home/Personal Contact Info, each with address, phone, fax, and email fields. The status bar at the bottom shows 'Record: 1 of 19' and 'FLTR'.

Figure 24-11.

A form that has both a custom menu bar and a custom toolbar.

Creating Custom Shortcut Menus

Don't forget that your users can right-click on the forms and reports in your application to open a shortcut menu. Forms have a ShortcutMenu property that you can set to No (false) to disable all shortcut menus. But Reports don't have this property, so unless you define a custom ShortcutMenuBar for reports, users will see the built-in menu.

Figure 24-12 shows you the three built-in shortcut menus—for a form, a control on a form, and a report. You can see that they provide some options that you probably don't want your users to be able to choose—such as switching to Design view or opening the Properties window for a form control.

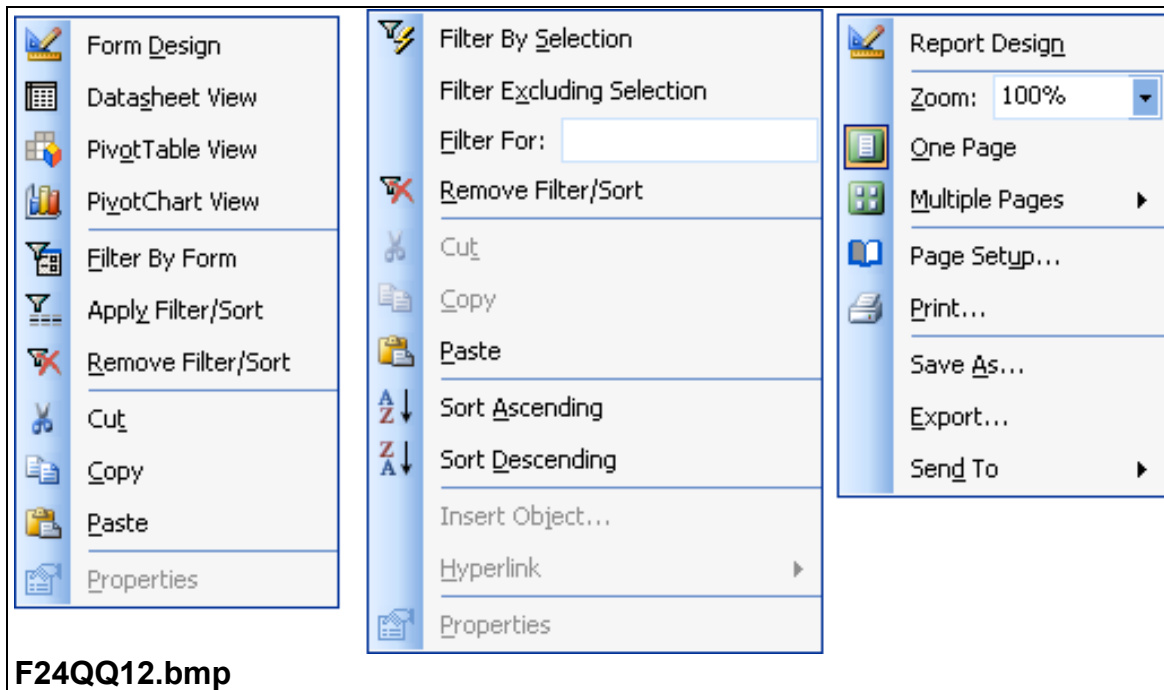


Figure 24-12.

The built-in form, form control, and report shortcut menus.

If you specify a custom shortcut menu for a form, that custom menu also becomes the shortcut menu for all the controls on the form. You might want to allow the user access to some of the shortcut menu features for controls, such as the filter or sort options. So if you define a shortcut menu for your forms, you should also create one for the controls on your forms.

Building a shortcut menu is very similar to building a custom menu bar—with a twist. Access anchors all shortcut menus on the Shortcut Menus toolbar, and it provides a special Custom entry at the right end of the toolbar where you can create your custom

shortcut menus, as shown in Figure 24-13. You can open the Shortcut Menus toolbar only when you're in toolbar customize mode.

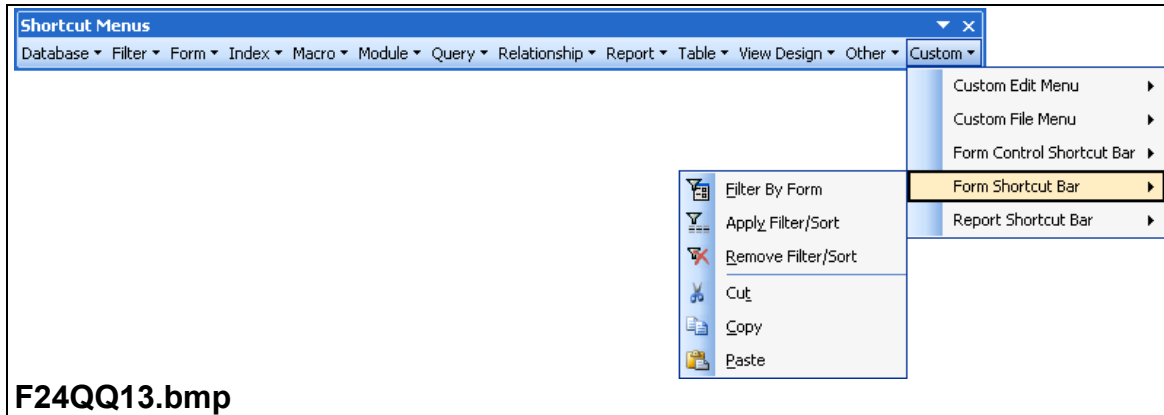


Figure 24-13.

The custom shortcut menu anchored on the Shortcut Menus toolbar.

To begin defining a custom shortcut menu, enter customize mode by right-clicking any toolbar and choosing Customize from the shortcut menu or by choosing Customize from the Tools menu. Begin a new toolbar by choosing the Toolbars tab in the Customize dialog box and then clicking the New button. Give your new toolbar a name in the New Toolbar dialog box and click OK. Immediately click Properties on the Toolbars tab and change the Type to Popup. Your new toolbar disappears, and Access shows you the warning dialog box shown in Figure 24-14 that informs you that you can find your new toolbar on the Custom list of the special Shortcut Menus toolbar. Click OK to close the dialog box and then click Close in the Toolbar Properties window.

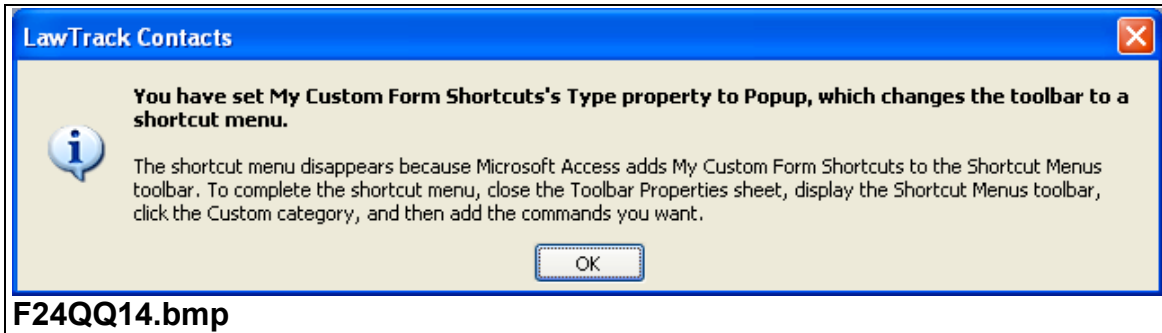
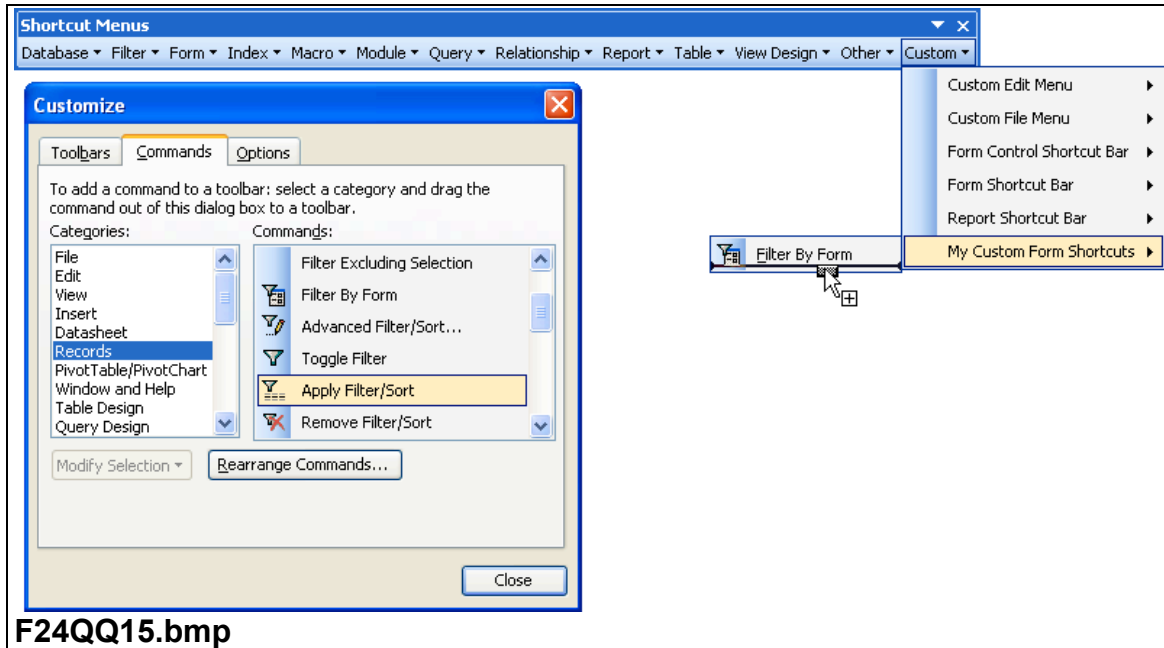


Figure 24-14.

Access tells you that you can find your new shortcut menu bar on the Shortcut Menus toolbar.

You cannot delete a custom shortcut menu by opening the Shortcut Menus toolbar, finding your toolbar on the Custom list, and dragging the toolbar off the list. The only way to delete a custom shortcut menu is to click Properties on the Toolbars tab of the Customize dialog box, change the Selected Toolbar to the name of your custom shortcut menu, and change Type to either Menu Bar or Toolbar. You can then select the toolbar in the Toolbars list of the Customize dialog box and click the Delete button to remove it.

Now that you have a new custom shortcut menu, you must open the Shortcut Menus toolbar by selecting it in the Toolbars list on the Toolbars tab in the Customize dialog box. You can customize the shortcut menu by dragging commands from the Commands tab or from another menu bar. Figure 24-15 shows adding the Apply Filter/Sort command to a custom form shortcut menu bar by dragging the command from the Records category on the Commands tab of the Customize dialog box.



F24QQ15.bmp

Figure 24-15.

Adding a command to a custom shortcut menu.

Setting Form, Form Control, and Report Menu and Toolbar Properties

After you create the custom menus and toolbars you need, you can set the Menu Bar and Toolbar properties for your forms and reports and the Shortcut Menu Bar property for your forms, form controls, and reports. Doing this one form, form control, or report at a time is a tedious task. However, if you have defined menu bars and toolbars and shortcut menu bars that work for all your forms, form controls, and reports, you can run some Visual Basic code to set all the properties at once.

For the sample applications, I created one custom menu bar, one custom toolbar, and one custom shortcut menu for all forms, and I created one custom shortcut menu that would work for all form controls. I also created one custom menu bar, one custom toolbar, and one custom shortcut menu for all reports. I then wrote some Visual Basic procedures to set the appropriate properties for all forms, form controls, and reports. You

can find these procedures in the *modExamples* module in both the LawTrack Contacts and Housing Reservations sample databases—SetFormToolbars for forms and

SetReportToolbars for reports—and you can adapt these procedures for your own use.

Here's the code for SetFormToolbars:

```
Public Sub SetFormToolbars()  
    Dim objFrm As AccessObject, frm As Form, ctl As Control  
    ' Go through every form in the database  
    For Each objFrm In CurrentProject.AllForms  
        ' Skip fixing any "plain" examples  
        If Right(objFrm.Name, 5) <> "plain" Then  
            DoCmd.OpenForm FormName:=objFrm.Name, View:=acDesign, _  
                WindowMode:=acHidden  
            ' Set a pointer to the form just opened  
            Set frm = Forms(objFrm.Name)  
            ' Set custom menu and toolbars for the form object  
            ' .. only for primary forms  
            If Left(frm.Name, 3) = "frm" Then  
                ' Clear any saved filter while we're at it  
                Forms(frm.Name).Filter = ""  
                ' Set the custom menu bar  
                Forms(frm.Name).MenuBar = "Custom Form Menu Bar"  
                ' Set the custom toolbar  
                Forms(frm.Name).Toolbar = "Custom Form Toolbar"  
                ' Set the custom shortcut menu bar  
                Forms(frm.Name).ShortcutMenuBar = "Form Shortcut Bar"  
            End If  
            ' Loop through all controls  
            For Each ctl In frm.Controls  
                ' Skip control types that don't have a ShortcutMenuBar property  
                If (ctl.ControlType <> acCustomControl) And _  
                    (ctl.ControlType <> acSubform) And _  
                    (ctl.ControlType <> acRectangle) And _  
                    (ctl.ControlType <> acLabel) And _
```

```
        (ctl.ControlType <> acLine) And _  
        (ctl.ControlType <> acPageBreak) Then  
            ' Set the custom control shortcut menu  
            ctl.ShortcutMenuBar = "Form Control Shortcut Bar"  
        End If  
  
        ' Loop through all controls on the form  
    Next ctl  
  
    ' Fix AllowDesignChanges while we're at it  
    frm.AllowDesignChanges = False  
  
    ' Close and save the result  
    DoCmd.Close acForm, objFrm.Name, acSaveYes  
End If  
  
' Loop to get the next form  
Next objFrm  
  
' Clean up  
Set ctl = Nothing  
Set frm = Nothing  
Set objFrm = Nothing  
End Sub
```

This code takes advantage of the AllForms collection that you learned about in Chapter 22 to open each form in Design view, and set the form's MenuBar, ToolBar, and ShortcutMenuBar properties to the custom form toolbars that I created. The code also examines each control on each form and sets the ShortcutMenuBar property for selected control types. (Some control types, such as lines and rectangles, do not have a ShortcutMenuBar property.) The code also resets the AllowDesignChanges property that you'll learn about later in this chapter and then saves the changes. This code takes a few seconds to run from the Visual Basic Immediate window, and it is much easier than opening each form individually to set the properties.

Fine-Tuning with the Performance Analyzer Wizard

Even the most experienced database designers (including me) don't always take advantage of all the techniques available to improve performance in an Access application. Fortunately, Access provides a Performance Analyzer Wizard to help you do a final analysis after you build most of your application. In this section, we'll let the wizard analyze the Housing Reservations sample database (*Housing.mdb*). To start the wizard, switch to the Database window, choose Analyze from the Tools menu, and then choose Performance from the submenu. Access opens the window shown in Figure 24-16.

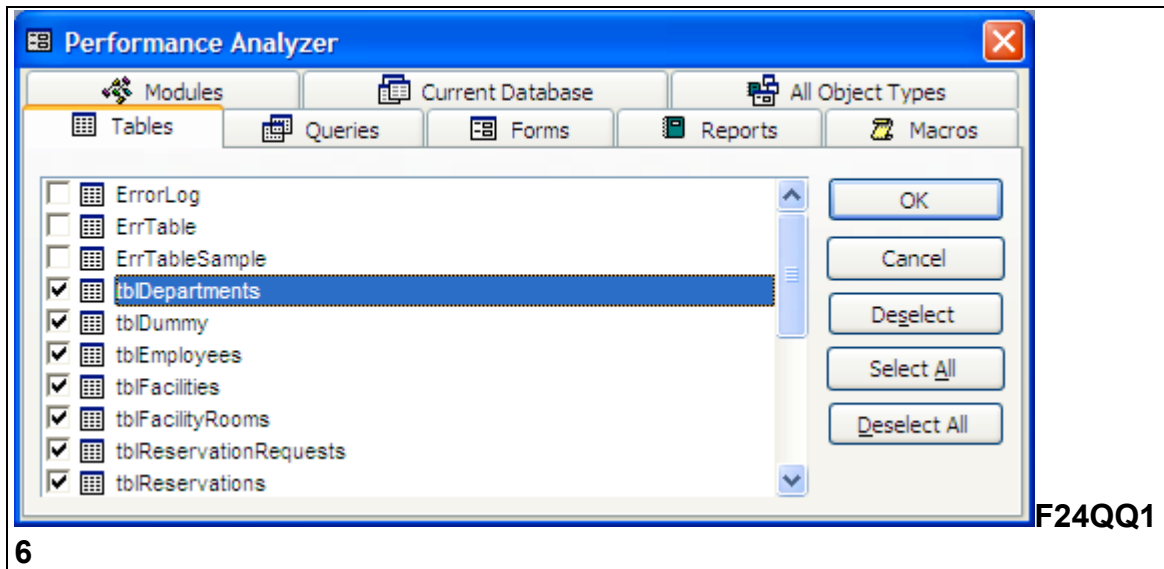


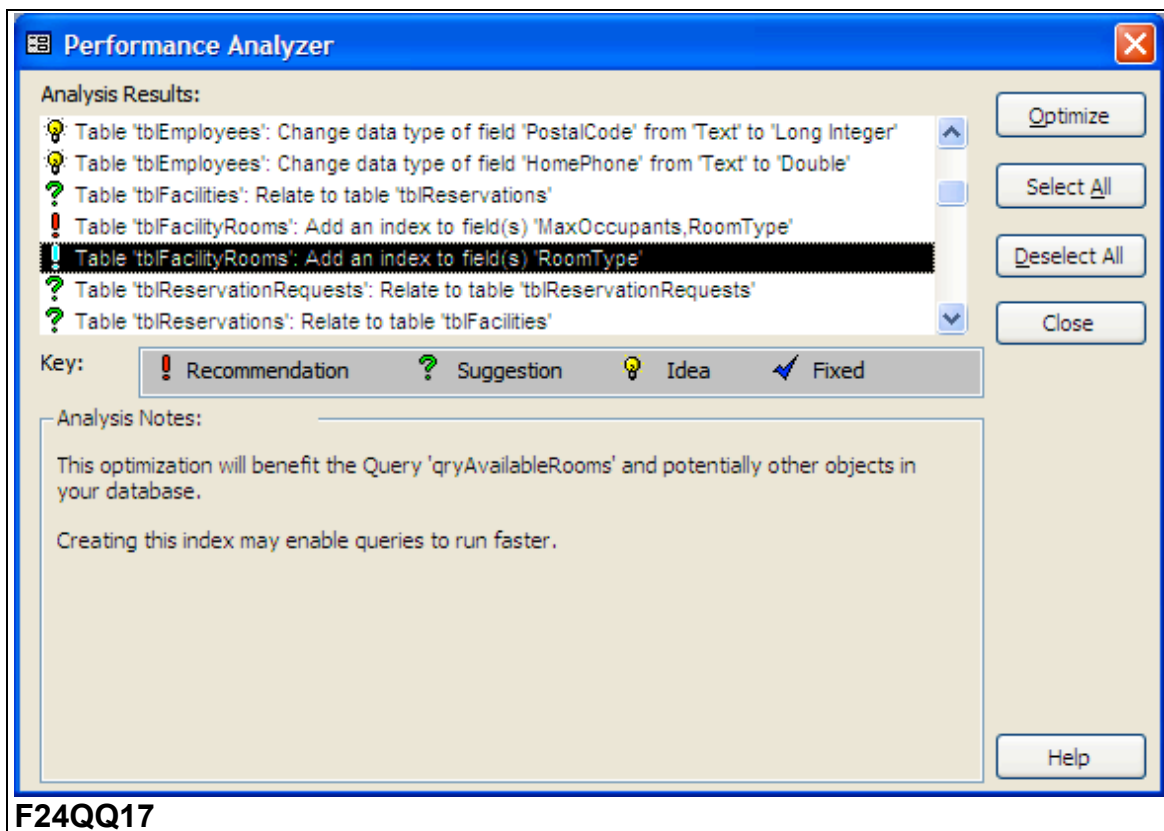
Figure 24-16.

The main selection window of the Performance Analyzer Wizard.

You can select a specific category of objects to analyze—Current Database (which lets you analyze table relationships), Tables, Queries, Forms, Reports, Macros, Modules, or All Object Types. Within a category, you can click the check box next to an object name to select it for analysis. You can click the Select All button to ask the wizard

to examine all objects or click Deselect All if you made a mistake and want to start over. In this example, I chose the All Object Types tab, clicked Select All, and then clicked on sample tables, queries, forms, and reports that aren't part of the actual application (all the extra examples I built for the book) to deselect them.

Click OK to run the wizard. The wizard opens a window that shows you its progress as it analyzes the objects you selected. When it is finished, the wizard displays the results of its analysis, similar to those shown in Figure 24-17.



F24QQ17

Figure 24-17.

Analysis results from the Performance Analyzer Wizard.

You can scan the list of recommendations, suggestions, and ideas displayed by the wizard. (Notice the key below the Analysis Results list.) Click any *recommendation* or *suggestion* that you like, and then click the Optimize button to have the wizard

implement the change on the spot. After the wizard implements a change, you'll see a check mark next to the item. If you like, you can click the Select All button to highlight all the recommendations and suggestions and then click Optimize to implement the fixes.

Personally, I'd rather choose the ones I want one at a time. For example, the idea to change the PostalCode field from text to long integer won't work if you're storing Canadian or European postal codes in your database. Also, I know that tblReservations is already related to tblFacilities through the tblFacilityRooms table, so adding a direct relationship between tblFacilities and tblReservations would be redundant.

Although you can implement recommendations and suggestions directly from the wizard, you can't do so with *ideas*. Most ideas are changes that could potentially cause a lot of additional work. For example, changing a data type of a field in a table might improve performance slightly, but it might also cause problems in dozens of queries, forms, and reports that you've already built using that table field. Other ideas are fixes that the wizard isn't certain will help; they depend on how you designed your application. I recommend that you look at the recommendations and suggestions and implement the ones that make the most sense for your application.

Disabling Form Design View

You might have noticed as you built new forms in your Access 2003 databases that Access sets the new Allow Design Changes property to All Views by default. This is a handy feature while you build forms because it allows you to change any form or control property directly from Form view and see the results immediately. Figure 24-18 shows you one of the sample forms from the **ContactsDataCopy.mdb** database opened in

Form view. You can see that Access shows you the Form Design toolbars and menus and lets you open the property sheet to make changes.

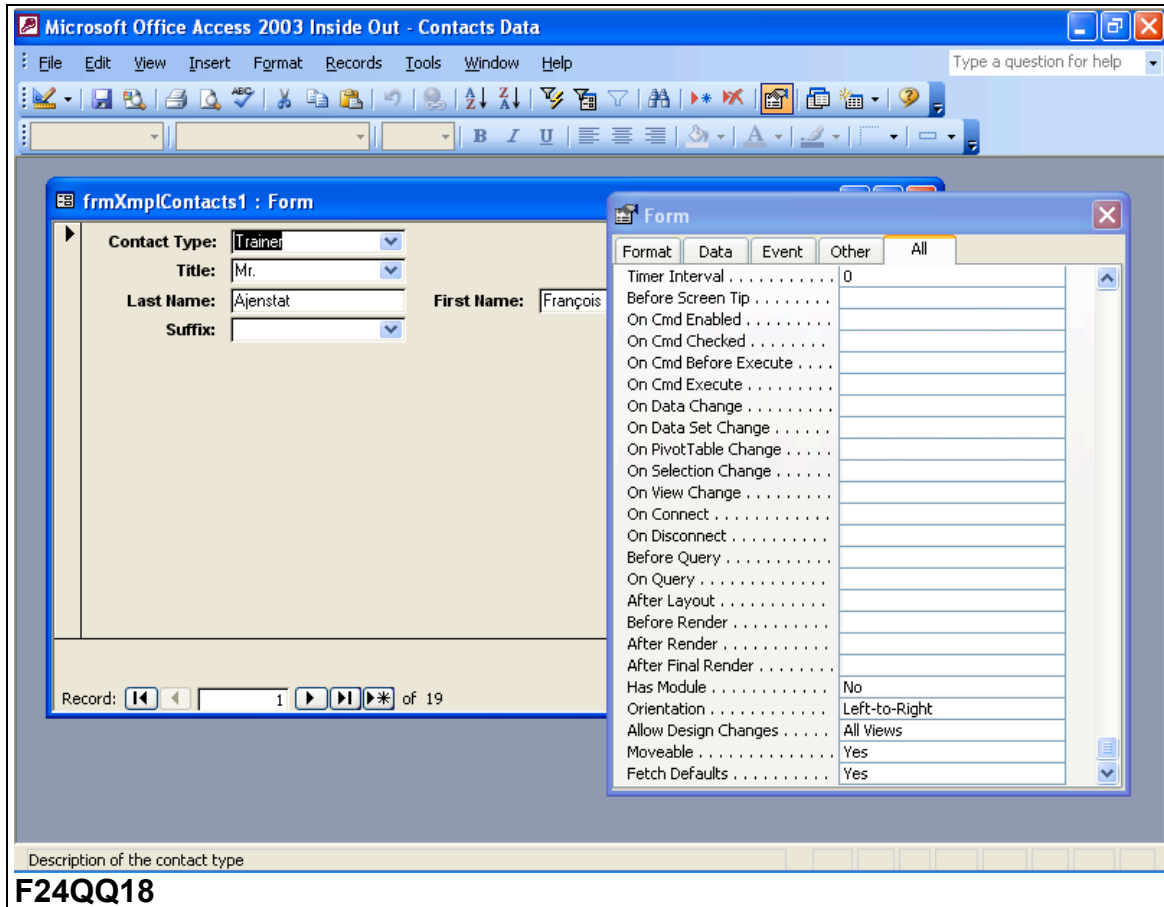


Figure 24-18.
The sample frmXmplContacts1 form from Chapter 13 opened in Form view.

When you're ready to put your application in production, however, you need to reset this new property to Design View Only for all your forms so the users see your forms as you intended. (The actual property value stored with the form is True if you select All Views and False if you select Design View Only.) You could open every form in Design view, change the property, and save the form. But why do it the hard way? You can also code a simple Visual Basic procedure to open all your forms and reset the property for you.

To help you out, I included a sample procedure in the modExamples module in both the LawTrack Contacts and Housing Reservations sample databases. You can copy the FixAllowDesign procedure into your database to take care of this tedious process.

Here's the procedure:

```
Sub FixAllowDesign()  
Dim objFrm As AccessObject, frm As Form  
    ' Go through every form in the database  
For Each objFrm In CurrentProject.AllForms  
    ' Open the form in Design view  
DoCmd.OpenForm FormName:=objFrm.Name, _  
    View:=acDesign  
    ' Set the form object for efficiency  
Set frm = Forms(objFrm.Name)  
    ' Check and reset the AllowDesignChanges property  
If frm.AllowDesignChanges = True Then  
    frm.AllowDesignChanges = False  
    ' Save the change  
DoCmd.RunCommand acCmdSave  
End If  
    ' Close the form  
DoCmd.Close acForm, objFrm.Name  
    ' Loop to the next form  
Next objFrm  
End Sub
```

This procedure takes advantage of the AllForms collection to find the names of all your forms. To examine a form property or modify a property, you have to open the form in Design view. Although the Allow Design Changes property setting in the form design Properties window shows All Views and Design View Only, these actually correspond to True and False internal property values, respectively. So, if AllowDesignChanges is True, the procedure sets the value to False and saves the result. The code closes the current form before moving on to the next one.

Defining Switchboard Forms

Usually the last forms that you build are the switchboard forms that give the user direct access to the major tasks in your application.

Designing a Switchboard Form from Scratch

Your main switchboard form should be a simple form with a logo, a title, and perhaps as many as eight command buttons. The command buttons can be used to open the forms that you defined in the application. Figure 24-19 shows the main switchboard form for the LawTrack Contacts database in Design view.



Figure 24-19.
The main switchboard form for the LawTrack Contacts database.

One feature worth mentioning here is the use of the ampersand (&) character when setting each control's Caption property. You can use the ampersand character to define a shortcut key for the control. In the Caption property for the Companies command button, for example, the ampersand precedes the letter *C*. The letter *C* becomes the shortcut key, which means that you can choose the Companies button by pressing Alt+C as well as by more traditional methods such as clicking the button with the mouse or tabbing to the button and pressing the Spacebar or the Enter key. You must be careful, however, not to duplicate another shortcut key letter. For example, the shortcut key for the Contacts command button in this example is *O*, to avoid conflict with the *C* access key for the Companies command button.

[New_Icon] If you choose Options from the Tools menu and select the **Keyboard shortcut errors** option on the Error Checking tab, Access displays a smart tag next to any command button or label that has a duplicate shortcut key defined.

You can use a shortcut key to make it easier to select any control that has a caption. For command buttons, the caption is part of the control itself. For most other controls, the caption is in the attached label. For example, you can define shortcut keys to select option buttons or toggle buttons in an option group by including an ampersand in the caption for each button in the group.

For each command button, you need a simple event procedure to handle the Click event and to open the appropriate form. Here is the procedure for the Products button.

```
Private Sub cmdProducts_Click()  
    ' Open the Product edit/create form
```

```
DoCmd.OpenForm "frmProducts"  
End Sub
```

If you have a custom form menu bar, you should set the Menu Bar property of your switchboard form to point to the name of the custom menu bar. If you also have a custom form toolbar, you should set the form's Toolbar property to it. In the LawTrack Contacts and Housing Reservations applications, a single custom form toolbar is used for all forms.

Using the Switchboard Manager to Design Switchboard Forms

If your application is reasonably complex, building all the individual switchboard forms you need to provide the user with navigation through your application could take a while. Access has a Switchboard Manager utility that helps you get a jump on building your switchboard forms. This utility uses a creative technique to handle all switchboard forms by dynamically modifying a single form. It uses a driver table named Switchboard Items to allow you to define any number of switchboard forms with up to eight command buttons each. Information in the table tells the code behind the switchboard form how to modify the buttons displayed and what to do when the user clicks each of the buttons.

The LawTrack Contacts application ([Contacts.mdb](#)) has a SwitchboardSample form that I created with the Switchboard Manager. Because I renamed the objects and fixed the code behind the SwitchboardSample form to use the SwitchboardDriver table, you can't use the Switchboard Manager to modify this form. You can open the SwitchboardSample form to see how a form built by the Switchboard Manager works, and you can follow along with the steps in this chapter to build your own switchboard.

To start the Switchboard Manager, choose Database Utilities from the Tools menu and then choose Switchboard Manager from the submenu. The utility checks to see if you already have a switchboard form and a Switchboard Items table in your database. If you don't have these, the Switchboard Manager displays the message box shown in Figure 24-20, which asks you if you want to build them.

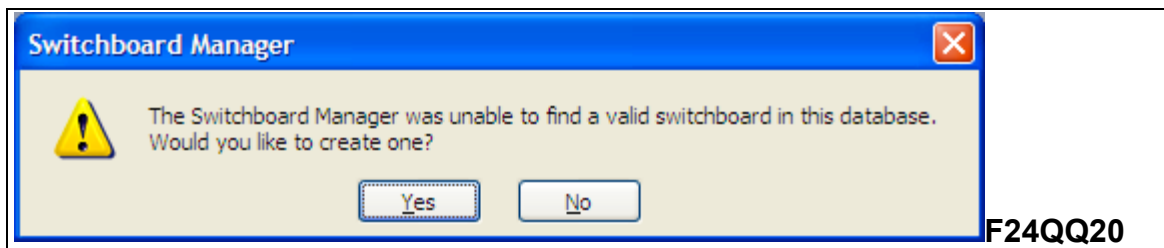


Figure 24-20.

The message box that appears if the Switchboard Manager does not find a valid switchboard form and Switchboard Items table in your database.

Click Yes to allow the Switchboard Manager to continue. After it builds a skeleton switchboard form and a Switchboard Items table (or after it establishes that you already have these objects in your database), it displays the main Switchboard Manager window. Unless your application is very simple, you won't be able to provide all the navigation your users need on one switchboard with eight options—especially considering that you should use one of the options to provide a way for the user to exit the application. So, you should plan additional switchboards that the user can navigate to from the main switchboard. One way to lay out the additional switchboards is to plan one switchboard for each major subject or group of similar features in your database.

You should first define all the switchboards that you need (called *pages* in the wizard) because a page must be defined before you can create a button to navigate to or from the page. To build an additional switchboard page, click the New button and enter a

name for the new switchboard in the Create New dialog box, as shown in Figure 24-21.

Click OK to create the page.

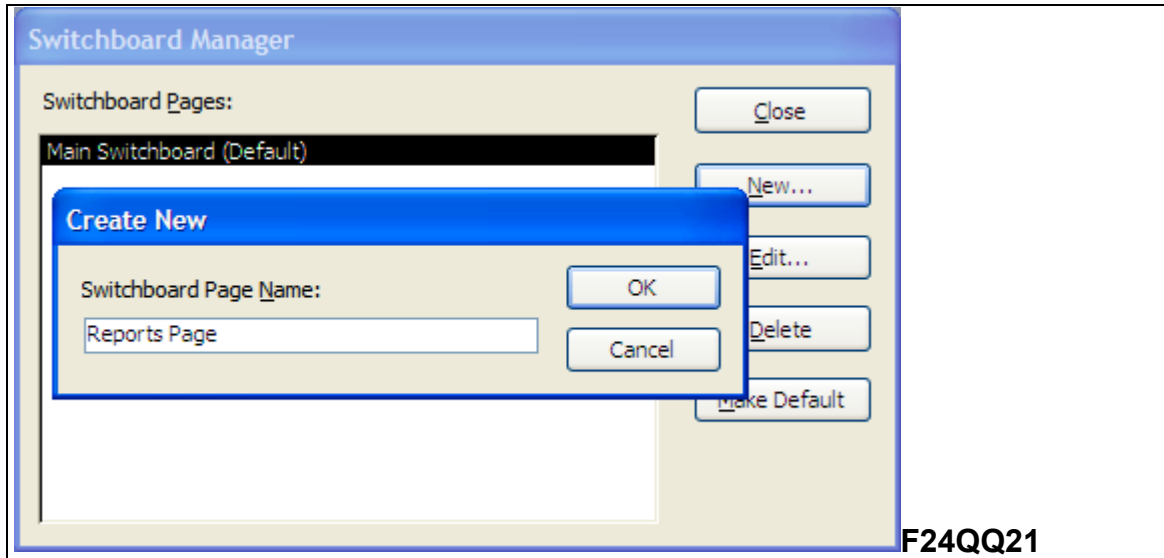


Figure 24-21.

Adding an additional switchboard page to the main switchboard form.

After you create the additional switchboard pages that you need, you can select one in the main Switchboard Manager window and click the Edit button to begin defining actions on the page. You'll see a window similar to the one shown in the background in Figure 24-22. Use this window to create a new action, edit an existing action, or change the order of actions. Figure 24-22 shows a new action being created. When you create a new action, the Switchboard Manager places a button on the switchboard page to execute that action. In the Text box, enter the caption that you want displayed next to the button. Note that you can enter an ampersand (&) before one of the letters in the caption to make that letter the access key for the button. Choose the action you want from the Command drop-down list. The Switchboard Manager can create actions such as moving to another switchboard page, opening a form in add or edit mode, opening a report, switching to Design view, exiting the application, or running a macro or a Visual Basic procedure. If

you choose an action such as opening a form or report, the Edit Switchboard Item dialog box shows you a list of appropriate objects in the second drop-down list.

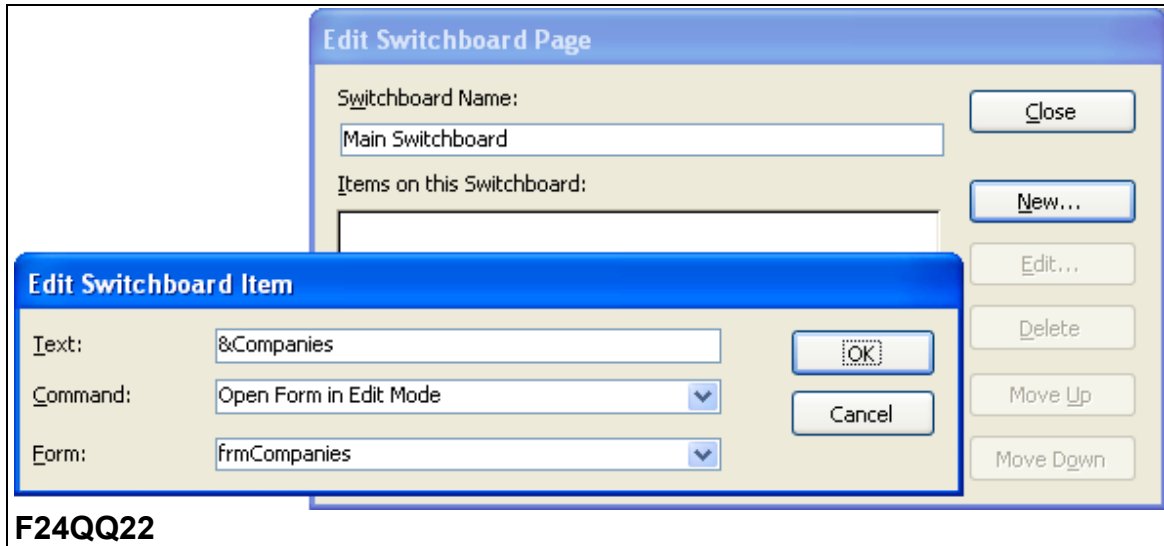


Figure 24-22.
Creating a new action on a switchboard page.

On the main switchboard page, you should create actions to open other pages and an action to exit the application. On each subsequent page, you should always provide at least one action to move back through the switchboard-page tree or to go back to the main switchboard page, as shown in Figure 24-23.

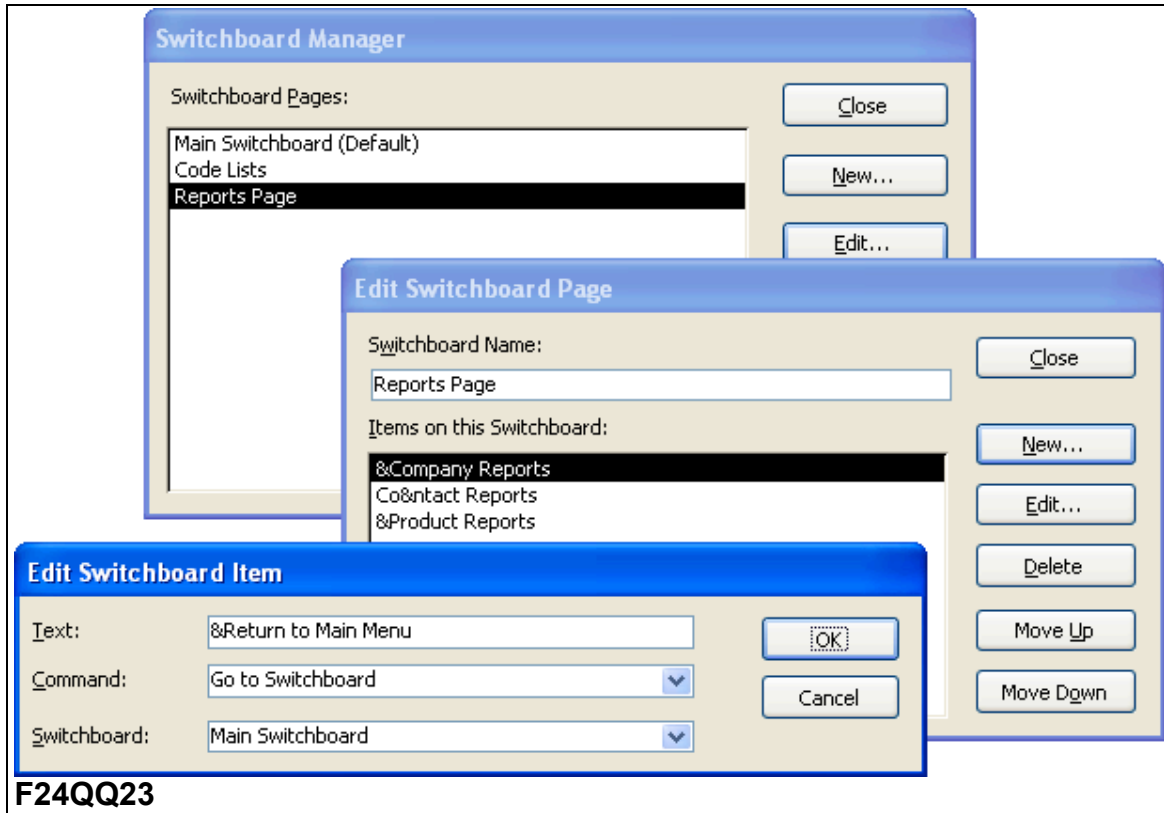


Figure 24-23.

Creating an action to return to the main switchboard form from another switchboard form.

On the SwitchboardSample form, I created buttons to open Companies, Contacts, Products, Pending Events, and Invoices. I also created buttons to show code lists that can be edited and the available reports. On the Code Lists page, I added entries to open Contact Types, Event Types, Product Categories, Person Titles, Person Suffixes, and Users. I also added a button to return to the main menu. On the Reports Page, I added entries to open Company Reports, Contact Reports, and Product Reports as well as a button to return to the main menu.

After you finish, the Switchboard Manager saves the main switchboard form with the name *Switchboard*. You can rename this form if you want to. If you want to rename the Switchboard Items table, be sure to edit the Visual Basic procedures stored with the

Switchboard form so that they refer to the new name. You'll also need to change the record source of the form.

Figure 24-24 shows an example Switchboard form for the LawTrack Contacts database (the SwitchboardSample form). I edited the form design to add the LawTrack logo. You can further customize the look of this form as long as you don't remove any of the option buttons or attached labels or change the names of these controls.

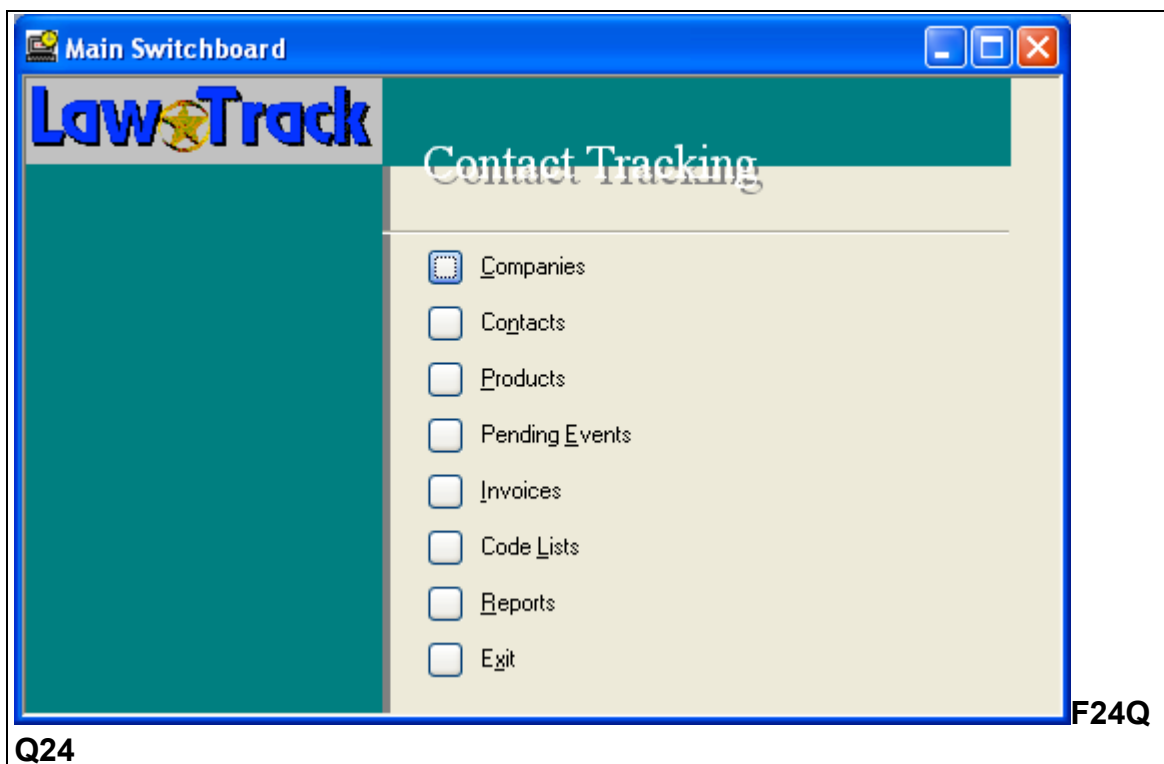


Figure 24-24.

The resulting main switchboard form.

I personally prefer to design my own switchboard forms so that I can add specialized code behind some of the command buttons. But you can see that this wizard is a handy way to quickly design a complex set of switchboard pages without a lot of work.

Controlling How Your Application Starts and Runs

Especially if you're distributing your application for others to use, you probably want your application to automatically start when the user opens your database. As noted in the previous section, you should design switchboard forms to help the user navigate to the various parts of your application. You should also set properties and write code to ensure that your user can cleanly exit your application.

Setting Startup Properties for Your Database

At this point, you know how to build all the pieces you need to fully implement your database application. But what if you want your application to start automatically when you open your database? One way to do that is to create a macro named *Autoexec*—Microsoft Access always runs this macro if it exists when you open the database (unless you hold down the Shift key when you open the database). However, a better way to start your application is to specify an opening form in the startup properties for the database. You can set these properties by switching to the Database window and then choosing Startup from the Tools menu. Access opens the Startup dialog box, as shown in Figure 24-25.

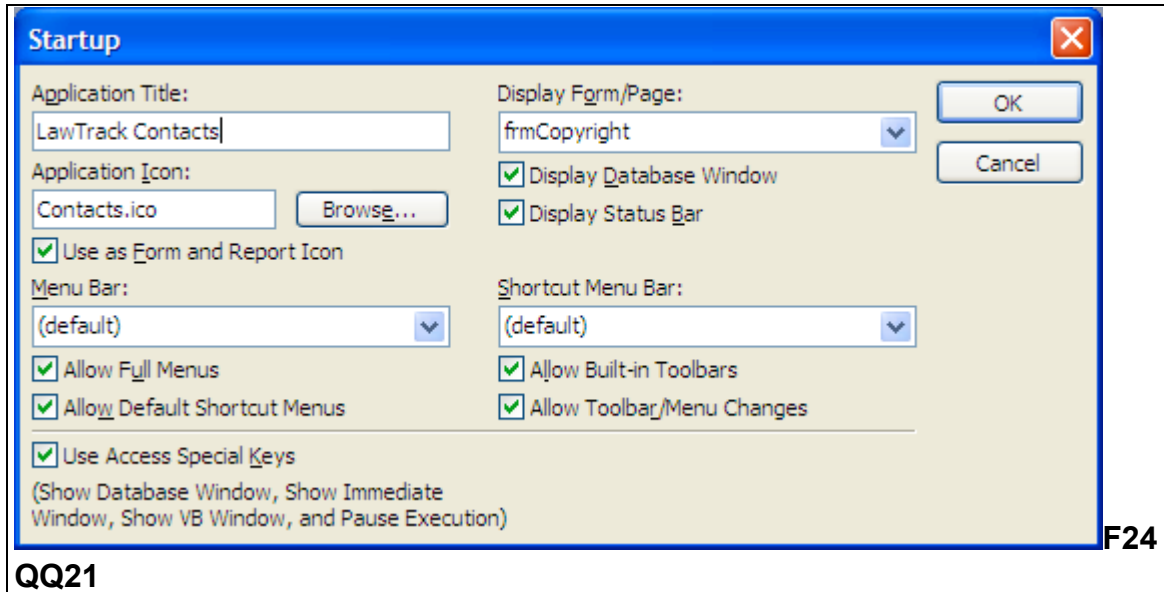


Figure 24-25.

Setting startup properties for your database.

You can specify which form or data access page opens your database by selecting a form from the Display Form/Page drop-down list. You can also specify a custom title for the application, an icon for the application, a default menu bar, and a default shortcut menu bar to override the standard menus for all forms. (I prefer to set the individual form and report menu bar, toolbar, and shortcut menu properties.) If you always open the database with its folder set to the current directory, you can simply enter the icon file name, as shown in the figure. If you're not sure what folder will be current when the application opens, you should enter a fully qualified file name location. Note that you can also ask that Access display the icon you specify as the form and report icon instead of the standard Access icons.

If you clear the Display Database Window check box, Access hides the Database window when your application starts. (As you'll learn later, you can also write code that executes in your startup form to ensure that the Database window is hidden.) You can

also hide the status bar if you want to by clearing the Display Status Bar check box. I like to use the SysCmd function to display information on the status bar, so I usually leave the Display Status Bar option selected. Access has a set of condensed built-in menus that don't provide, for example, access to design commands. If you clear the Allow Full Menus check box, Access provides these shortened menus as the default. You can also hide all the built-in toolbars (you should provide your own custom toolbars in this case), disallow toolbar changes, and disallow the default shortcut menus.

Finally, you can disable special keys—such as F11 to reveal the Database window, Ctrl+G to open the Debug window, or Ctrl+Break to halt code execution—by clearing the Use Access Special Keys check box. As you can see, you have many powerful options for customizing how your application starts and how it operates.

Starting and Stopping Your Application

Although you can set startup properties asking Access to hide the Database window, you might want to include code in the Load event of your startup form to make sure the window is hidden. All the sample databases provided with this book open the frmCopyright form as the startup form. This form displays information about the database, and code behind the form checks connections to linked tables or data access pages. In both the LawTrack Contacts and Housing Reservations sample applications, the code behind the frmCopyright form tells you to open the frmSplash form to actually start the application.

When the frmSplash form opens, code in the Load event uses the following procedure to make sure the Database window is hidden:

```
' Select the database window
DoCmd.SelectObject acForm, "frmSplash", True
' .. and hide it
RunCommand acCmdWindowHide
```

The procedure hides the Database window by selecting a known object in the Database window to give the Database window the focus and then executing the WindowHide command. The splash form waits for a timer to expire (the Timer event procedure) and then opens a form to sign on to the application. When you sign on successfully, the frmMain form finally opens.

The frmMain form in the LawTrack Contacts application has no Close button and no Control-menu button. The database also has an AutoKeys macro defined that intercepts any attempt to close a window using the Ctrl+F4 keys. (You'll learn about creating an AutoKeys macro in the next section.) So, you must click the Exit button on the frmMain form to close the application. On the other hand, the frmMain form in the Housing Reservations application does allow you to press Ctrl+F4 or click the Close button to close the form and exit the application.

You should always write code to clean up any open forms, reset variables, and close any open recordsets when the user asks to exit your application. Because the user can't close the frmMain form in the LawTrack Contacts application except by clicking the Exit button, you'll find such clean-up code in the command button's Click event. In the frmMain form in the Housing Reservations database, the clean-up code is in the form's Close event procedure. The code in both forms is similar, so here's the exit code in the LawTrack Contacts sample application:

```
Private Sub cmdExit_Click()  
    Dim intErr As Integer, frm As Form, intI As Integer  
    Dim strData As String, strDir As String  
    Dim lngOpen As Long, datBackup As Date  
    Dim strLowBkp As String, strBkp As String, intBkp As Integer  
    Dim db As DAO.Database, rst As DAO.Recordset  
    If vbNo = MsgBox("Are you sure you want to exit?", _  
        vbYesNo + vbQuestion + vbDefaultButton2, _  
        gstrAppTitle) Then  
        Exit Sub  
    End If  
    ' Trap any errors  
    On Error Resume Next  
    ' Make sure all forms are closed  
    For intI = (Forms.Count - 1) To 0 Step -1  
        Set frm = Forms(intI)  
        ' Don't close myself!  
        If frm.Name <> "frmMain" Then  
            ' Use the form's "Cancel" routine  
            frm.cmdCancel_Click  
            DoEvents  
        End If  
        ' Note any error that occurred  
        If Err <> 0 Then intErr = -1  
    Next intI  
    ' Log any error beyond here  
    On Error GoTo frmMain_Error  
    ' Skip backup check if there were errors  
    If intErr = 0 Then  
        Set db = CurrentDb  
        ' Open ztblVersion to see if we need to do a backup  
        Set rst = db.OpenRecordset("ztblVersion", dbOpenDynaset)  
        rst.MoveFirst  
        lngOpen = rst!OpenCount  
        datBackup = rst!LastBackup  
        rst.Close
```

```
Set rst = Nothing

' If the user has opened 10 times
' or last backup was more than 2 weeks ago...
If (lngOpen Mod 10 = 0) Or ((Date - datBackup) > 14) Then
    ' Ask if they want to backup...
    If vbYes = MsgBox("LawTrack highly recommends backing up " & _
        "your data to avoid " & _
        "any accidental data loss. Would you like to backup now?", _
        vbYesNo + vbQuestion, gstrAppTitle) Then
        ' Get the name of the data file
        strData = Mid(db.TableDefs("ztblVersion").Connect, 11)
        ' Get the name of its folder
        strDir = Left(strData, InStrRev(strData, "\"))
        ' See if the "BackupData" folder exists
        If Len(Dir(strDir & "BackupData", vbDirectory)) = 0 Then
            ' Nope, build it!
            MkDir strDir & "BackupData"
        End If
        ' Now find any existing backups - keep only three
        strBkp = Dir(strDir & "BackupData\LawTrackBkp*.mdb")
        Do While Len(strBkp) > 0
            intBkp = intBkp + 1
            If (strBkp < strLowBkp) Or (Len(strLowBkp) = 0) Then
                ' Save the name of the oldest backup found
                strLowBkp = strBkp
            End If
            ' Get the next file
            strBkp = Dir
        Loop
        ' If more than two backup files
        If intBkp > 2 Then
            ' Delete the oldest one
            Kill strDir & "BackupData\" & strLowBkp
        End If
        ' Now, setup new backup name based on today's date
        strBkp = strDir & "BackupData\LawTrackBkp" & _
```

```
        Format(Date, "yymmdd") & ".mdb"
    ' Make sure the target file doesn't exist
    If Len(Dir(strBkp)) > 0 Then Kill strBkp
    ' Create the backup file using Compact
    DBEngine.CompactDatabase strData, strBkp
    ' Now update the backup date
    db.Execute "UPDATE ztblVersion SET LastBackup = #" & _
        Date & "#", dbFailOnError
    MsgBox "Backup created successfully!", vbInformation, gstrAppTitle
End If
' See if error log has 20 or more entries
If db.TableDefs("ErrorLog").RecordCount > 20 Then
    ' Don't ask if they've said not to...
    If Not (DLookup("DontSendError", "tblUsers", _
        "UserName = '" & gstrThisUser & "'")) Then
        DoCmd.OpenForm "fdlgErrorSend", WindowMode:=acDialog
    Else
        db.Execute "DELETE * FROM ErrorLog", dbFailOnError
    End If
End If
End If
Set db = Nothing
End If

' Restore original keyboard behavior
' Disabled in this sample
' Application.SetOption "Behavior Entering Field", gintEnterField
' Application.SetOption "Move After Enter", gintMoveEnter
' Application.SetOption "Arrow Key Behavior", gintArrowKey
' We're outta here!
frmMain_Exit:
    On Error GoTo 0
    DoCmd.Close acForm, Me.Name
    ' In a production application, would quit here
    DoCmd.SelectObject acForm, "frmMain", True
    Exit Sub
frmMain_Error:
```

```
ErrorLog "frmMain", Err, Error  
Resume frmMain_Exit  
End Sub
```

After confirming that the user really wants to exit, the code looks at every open form. All forms have a public `cmdCancel_Click` event procedure that this code can call to ask the form to clear any pending edits and close itself. The `DoEvents` statement gives that code a chance to complete before going on to the next form. Notice that the code skips the form named `frmMain` (the form where this code is running).

If there were no errors closing all the forms, then the code opens a table that contains a count of how many times this application has run and the date of the last backup. Every tenth time the application has run or every two weeks since the last backup, the code offers to create a backup of the application data. If the user confirms creating a backup, the code creates a `Backup` subfolder if it does not exist, deletes the oldest backup if there are three or more in the folder, and then backs up the data using the `CompactDatabase` method of the `DBEngine`.

Next, the code checks to see if more than 20 errors have been logged by code running in the application. If so, it opens a dialog box that gives the user the option to e-mail the error log, print out the error log, skip printing the error log this time, or turn off the option to print the log. Because the error log option form opens in `Dialog` mode, this code waits until that form closes. Finally, the code closes this form and selects an object in the `Database` window to reveal that window. If this weren't a demonstration application, the code would use the `Quit` method of the `Application` object to close and exit Access.

This might seem like a lot of extra work, but taking care of details like this really gives your application a professional polish.

Creating an AutoKeys Macro

As noted earlier, the LawTrack Contacts sample application (*Contacts.mdb*) has an AutoKeys macro defined to intercept pressing Ctrl+F4. You can normally press this key combination to close any window that has the focus, but the application is designed so that you must close the frmMain form using the Exit button, not Ctrl+F4. You can create an AutoKeys macro to define most keystrokes that you want to intercept and handle in some other way. You can define something as simple as a StopMacro action to effectively disable the keystroke, create a series of macro actions that respond to the keystrokes, or use the RunCode action to call complex Visual Basic code. Figure 24-26 shows you the AutoKeys macro in the LawTrack Contacts database open in Design view.

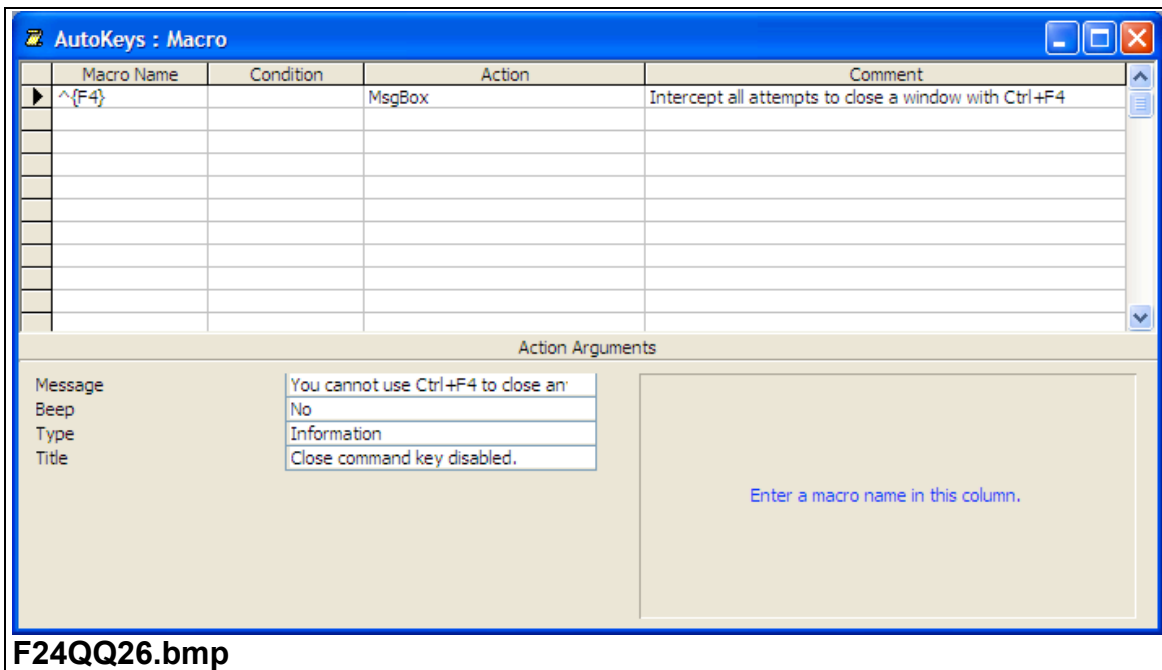


Figure 24-26.
The design of an AutoKeys macro.

The critical part of a macro defined in an AutoKeys macro group is the macro name. When the name is a special combination of characters that match a key name, the macro executes whenever you press those keys. Table 23-1 shows you how to construct macro names in an AutoKeys macro group to respond to specific keys.

Table 23-1. AutoKeys Macro Key Codes	
AutoKeys Macro Name	Key Intercepted
<i>^letter or ^number</i>	Ctrl+[the named letter or number key]
<i>{Fn}</i>	The named function key (F1–F12)
<i>^{Fn}</i>	Ctrl+[the named function key]
<i>+{Fn}</i>	Shift+[the named function key]
<i>{Insert}</i>	Insert
<i>^{Insert}</i>	Ctrl+Insert
<i>+{Insert}</i>	Shift+Insert
<i>{Delete} or {Del}</i>	Delete
<i>^{Delete} or ^{Del}</i>	Ctrl+Delete
<i>+{Delete} or +{Del}</i>	Shift+Delete

Keep in mind that you can also intercept any keystroke on a form in the KeyDown and KeyPress events when you want to trap a particular key on only one form or control.

Performing a Final Visual Basic Compile

The very last task you should perform before placing your application in production is to compile and save all your Visual Basic procedures. When you do this, Access stores a compiled version of the code in your database. Access uses the compiled code when it needs to execute a procedure you have written. If you don't do this, Access has to load and interpret your procedures the first time you reference them, each time you start your application. For example, if you have several procedures in a form module, the form will open more slowly the first time because Access has to also load and compile the code.

To compile and save all the Visual Basic procedures in your application, open any module—either a module object or a module associated with a form or report. Choose Compile from the Debug menu, as shown in Figure 24-27. If your code compiles successfully, be sure to save the result by choosing Save from the File menu or by clicking the Save button on the toolbar. (If you have errors in any of your code, the compiler halts on the first error it finds, displays the line of code, and displays an error message dialog box.) After a successful compilation, close your database and compact it as described in Chapter 5, “Modifying Your Table Design.” In Access for Windows 95, renaming your database always “decompiled” your code. This meant that if you changed the database name, you would have to compile your code again. With Access 97 and later, you don't have to recompile the code after renaming the database file.

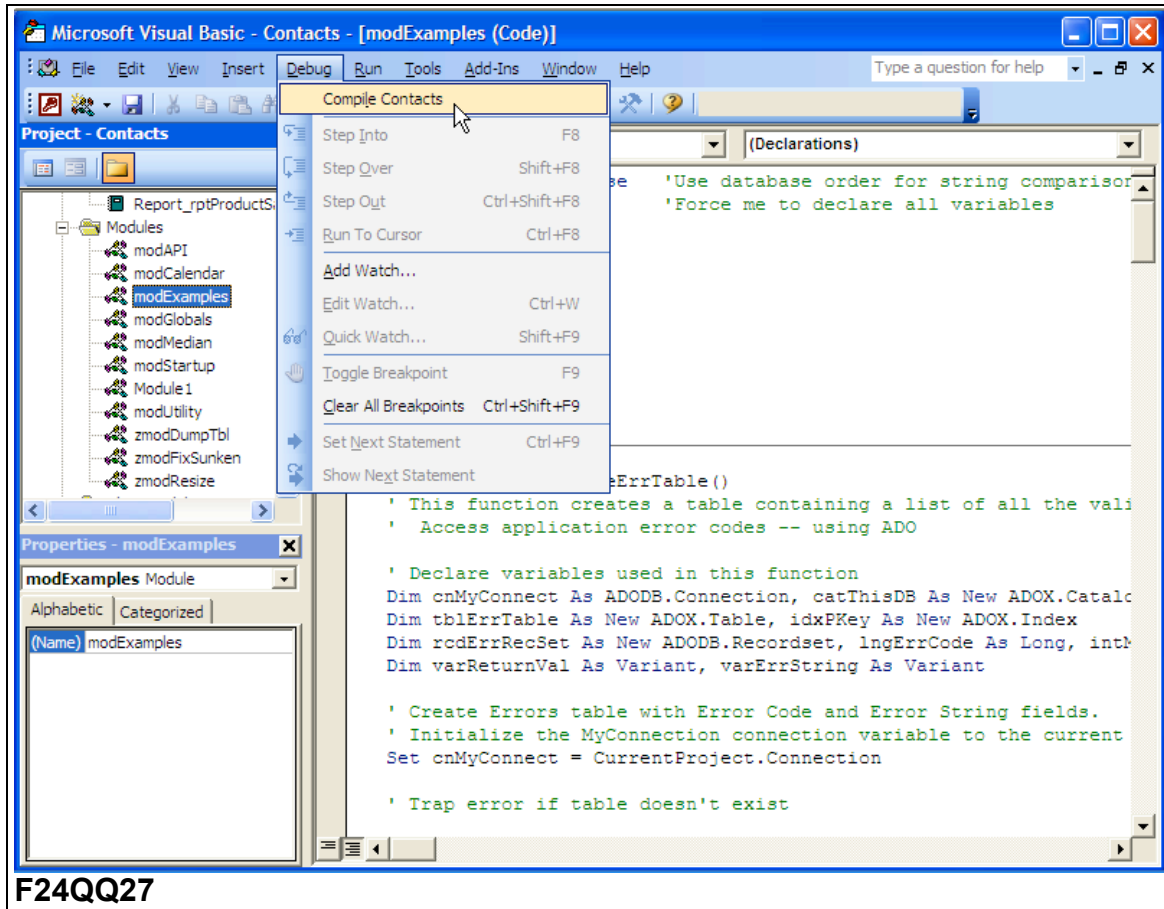


Figure 24-27.

Choose the Compile command from the Debug menu to compile all the Visual Basic procedures in your database.

As you've seen in this book, you can quickly learn to build complex applications. You can use the relational database management system in Access to store and manage your data locally or on a network, and you can access information in other popular database formats or in any server-hosted or mainframe-hosted database that supports the Open Database Connectivity (ODBC) standard. You can get started with macros to become familiar with event-oriented programming and to prototype your application. With a little practice, you'll soon find yourself writing Visual Basic event procedures like

a pro. In the next part of this book, you can learn about working with your Access applications on the Web. In the final part, you can learn how to implement your Access application in a client/server environment, secure your database, and prepare your database for distribution.