CHAPTER 27

Distributing Your Application

Although you can certainly use Microsoft Access 2010 to create database applications only for your personal use, most serious users of Access 2010 eventually end up building applications to be used by others. If you have created a stand-alone desktop client database application and your users all have the same version of Access installed on their computers, you can simply give them a copy of your database file to run. However, most database applications become really useful when multiple users share the data.

In Chapter 8, "Importing and Linking Data," you learned about linking tables from another database and using the Linked Table Manager. You also learned about some of the advantages of using a client-server architecture to allow multiple users running the same application to share data. Both these topics help you understand how to design and secure a multi-user application. However, they don't provide you with the techniques you can employ in your application design to ensure that your application installs and runs smoothly and to ensure that your users can't tamper with your code. In this chapter, you'll learn

- How to split a desktop client database to use shared data on a server and linked tables on each client computer and how to create code that verifies and corrects linked table connection properties when your application starts
- The advantages of runtime mode and design considerations for using it
- How to create an execute-only version of your application so that users can't tamper with your code
- Techniques for creating application shortcuts to simplify starting your application
- How to apply an encrypted database password to your database
- How to package your database and digitally sign it
- Tools you can use in Access 2010 Developer Extensions and Runtime to distribute your application to users who don't have Access

Using Linked Tables in a Desktop Database

Your first foray into the world of shared applications will most likely involve copying your completed database to a file server and instructing users to open the database from the server. That's basically not a good idea because Access client doesn't run on the server—it runs on each user's desktop. If you have several users sharing a database file, the copy of Access running on each user computer has to load all your "code" definitions—the queries, forms, reports, macros, and modules—over the network. If one user applies a sort or filter to a form and then closes it, Access will try to save the changed definition in the copy on the server, and it might run into locking or corruption problems if another user has the form open at the same time. Also, if your application needs to keep information about how each user works with the application, it would be more complicated to store this information in the database because all users share the same file.

The solution is to create a data-only .accdb file on a server and use linked tables in a desktop client application that you install on multiple desktop computers and link to the data server. When you separate the data tables into a shared file on a server, you're actually building a simple client-server application, much like a published Access Services application running on Microsoft SharePoint 2010. The main advantage to splitting your database over simply sharing a single desktop client database is that your application needs to retrieve only the data from the tables over the network. Because each user will have a local copy of the queries, forms, reports, macros, and modules, Access running on each user workstation will be able to load these parts of your application quickly from the local hard drive. Using a local copy of the desktop application on each client computer also makes it easy to create local tables that save settings for each user. For example, the Conrad Systems Contacts application allows each user to set a preference to open a search dialog box for companies, contacts, and invoices or to display all records directly.

INSIDE OUT

Is Your Desktop Application Designed for Client-Server?

When you build a desktop database application, it's all too easy to design forms and reports that always display all records from your tables when the user opens them. It's also tempting to create combo boxes or list boxes that display all available values from a lookup table. These issues have little to no impact when you're the only user of the application or you share your application with only a few other users. However, fetching all rows by default can have serious performance implications when you have multiple users who need to share a large amount of data over a network.

A successful client-server application fetches only the records required for the task at hand. You can design an application so that it never (or almost never) opens a form to edit data or a report to display data without first asking the user to specify the

records needed for the task at hand. For example, the Conrad Systems Contacts application opens a list of available companies, contacts, or invoices from which the user can choose only the desired records. This application also offers a custom query by form search to filter specific records based on the criteria the user enters.

You can also design the application so that it uses information about the current user to filter records. For example, the Housing Reservations database always filters employee and reservations data to display information only for the currently signed on employee. When a department manager is signed on, the application shows data only for the current manager's department.

Even so, the two main sample applications aren't perfect examples. Both applications use a ZIP code table that contains more than 50,000 records to help users enter valid address data, and this huge table is the row source for several combo boxes. In the desktop database version, each user has a local copy of the ZIP code table, so the performance impact is minimal. If you decide to split your database application into a client-server architecture, you need to think about keeping any tables similar to this in the local database on each user's computer. You could certainly do this with a ZIP code table because the data is relatively static.

The bottom line is you should take a look at the way your desktop application fetches data for the user. If it always fetches all records all the time, it's probably not a good candidate for upsizing to a client-server application.

Taking Advantage of the Database Splitter Wizard

You could split out the tables in your application into a separate file and link them into the database that contains all your queries, forms, reports, and modules "by hand" by first creating a new empty database (see Chapter 4, "Designing Client Tables") and then importing all your tables into that database using the techniques described in Chapter 8. You could then return to your original database and delete all the tables. Finally, you could move the data database (the one containing the tables) to a file server and then link these tables into your original code database (the one containing all your queries, forms, reports, and modules), again using the techniques in Chapter 8.

Fortunately, there's an easier way to do this in one step using the Database Splitter wizard. Open your original database, and on the Database Tools tab, in the Move Data group, click Access Database. (You can try this with one of your own databases or a backup copy of the Housing.accdb sample database. Note that you might need to hold down the Shift key while opening the copy of the Housing.accdb to bypass all the startup options, which ensures that Access is not running any background processes before starting the wizard.) The wizard displays the window shown in Figure 27-1.

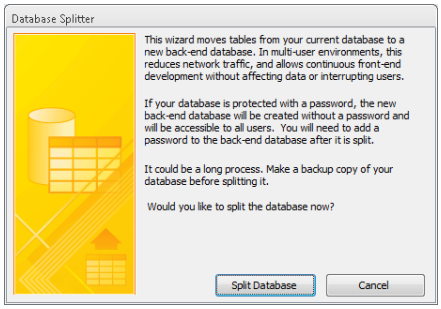

Figure 27-1 The Database Splitter wizard helps you move the tables into a separate database.

When you click Split Database, the wizard opens a second window where you can define the name and location of the back-end, or data-only, database. Be sure to choose a location for this database on a network share that is available to all potential users of your application. Click the Split button in that window, and the wizard exports all your tables to the new data-only database, deletes the tables in your original database, and creates links to the moved tables in your original database. You can now give each user a copy of the code database—containing your queries, forms, reports, modules, and linked table objects pointing to the new data-only database—to enable them to run the application using a shared set of tables.

One disadvantage to using the Database Splitter wizard is that it splits out all tables that it finds in your original desktop database. If you take a look at the desktop database version of the Conrad Systems Contacts application (Contacts.accdb), you can see that the application also uses some local tables (tables that remain in the code database). For example, the ErrorLog table contains records about errors encountered when the user runs the application. If the error was caused by a failure in the link to the server, the code that writes the error record wouldn't be able to write to this table if the table was in the server database. The database also contains local copies of lookup tables that aren't likely to change frequently, such as the tlkpStates table, which contains U.S. state postal abbreviations and names, and the ztblYears table, which provides a list of years for the frmCalendar form. Access can fetch data from local tables faster than it can from ones linked to a database on a server, so providing local copies of these tables improves performance.

Although splitting a database application makes it easier for multiple users to share your application, this technique works well only for applications containing a moderate amount of data (less than 200 MB is a good guideline) with no more than 20 simultaneous users. Remember that Access is fundamentally a desktop database system. All the work—including solving complex queries—occurs on the client computer, even when you have placed all the data on a network share. Each copy of Access on each client computer uses the file sharing and locking mechanisms of the server operating system. Access sends many

low-level file read, write, and lock commands (perhaps thousands to solve a single query) from each client computer to the file server rather than sending a single Structured Query Language (SQL) request that the server solves. When too many users share the same application accessing large volumes of data, many simple tasks can start, taking minutes instead of seconds to complete.

Creating Startup Code to Verify and Correct Linked Table Connections

If you were careful when you created your linked tables, you used a Universal Naming Convention (UNC) path name instead of a physical or logical drive letter. Unless the network share name is different on various client computers, this should work well to establish the links to the data file when the user opens your application. However, you can't always assume everything will go perfectly all the time.

In Chapter 8, you learned how to use the Linked Table Manager to repair any broken connections. However, you can't expect your users to run this wizard if the linked table connections are broken. You should include code that runs when your startup form opens that verifies and corrects the links if necessary. Also, over time, you might make changes to the structure of the data tables and issue an updated version of the client desktop database that works with the newer version of the tables. Your startup code can open and check a version table in the shared data database and warn the user if the versions don't match.

You can find sample code that accomplishes all these tasks in the desktop database version of the Conrad Systems Contacts database (Contacts.accdb). Open the database, and then open the modStartup module. Select the ReConnect function, where you'll find the following code:

```
Public Function ReConnect()
Dim db As DAO.Database, tdf As DAO.TableDef
Dim rst As DAO.Recordset, rstV As DAO.Recordset
Dim strFile As String, varRet As Variant, frm As Form
Dim strPath As String, intI As Integer
' This is a slightly different version of reconnect code
' Called by frmSplash - the normal start-up form for this application
    On Error Resume Next
    ' Point to the current database
Set db = CurrentDb
    ' Turn on the hourglass - this may take a few secs.
    DoCmd.Hourglass True
    ' First, check linked table version
    Set rstV = db.OpenRecordset("ztblVersion")
    ' Got a failure - so try to reattach the tables
    If Err <> 0 Then GoTo Reattach
    ' Make sure we're on the first row
    rstV.MoveFirst
```

```
' Call the version checker
If Not CheckVersion(rstV!Version) Then
    ' Tell caller that "reconnect" failed
    ReConnect = False
    ' Close the version recordset
    rstV.Close
    ' Clear the objects
    Set rstV = Nothing
    Set db = Nothing
    ' Done
    DoCmd.Hourglass False
    Exit Function
End If
' Versions match - now verify all the other tables
' NOTE: We're leaving rstV open at this point for better efficiency
' in a shared database environment.
' JET will share the already established thread.
' Turn on the progress meter on the status bar
varRet = SysCmd(acSysCmdInitMeter, "Verifying data tables...", _
    db.TableDefs.Count)
' Loop through all TableDefs
For Each tdf In db.TableDefs
    ' Looking for attached tables
    If (tdf.Attributes And dbAttachedTable) Then
        ' Try to open the table
        Set rst = tdf.OpenRecordset()
        ' If got an error - then try to relink
        If Err <> 0 Then GoTo Reattach
        ' This one is OK - close it
        rst.Close
        ' And clear the object
        Set rst = Nothing
    End If
    ' Update the progress counter
    intI = intI + 1
    varRet = SysCmd(acSysCmdUpdateMeter, intI)
Next tdf
' Got through them all - clear the progress meter
varRet = SysCmd(acSysCmdClearStatus)
' Turn off the hourglass
DoCmd.Hourglass False
' Set a good return
ReConnect = True
' Edit the Version table
rstV.Edit
' Update the open count - we check this on exit to recommend a backup
rstV!OpenCount = rstV!OpenCount + 1
' Update the row
rstV.Update
' Close and clear the objects
rstV.Close
Set rstV = Nothing
```

```
    Set db = Nothing
    ' DONE!
    Exit Function

Reattach:
    ' Clear the current error
    Err.Clear
    ' Set a new error trap
    On Error GoTo BadReconnect
    ' Turn off the hourglass for now
    DoCmd.Hourglass False
    ' ... and clear the status bar
    varRet = SysCmd(acSysCmdClearStatus)
    ' Tell the user about the problem - about to show an open file dialog
    MsgBox "There's a temporary problem connecting to the CSD data." & _
      "  Please locate the CSD data file in the following dialog.", _
      vbInformation, "CSD Contacts Manager"
    ' Establish a new ComDlg object
    With New ComDlg
        ' Set the title of the dialog
        .DialogTitle = "Locate CSD Contacts Data File"
        ' Set the default file name
        .FileName = "ContactsData.accdb"
        ' ... and start directory
        .Directory = CurrentProject.Path
        ' ... and file extension
        .Extension = "accdb"
        ' ... but show all accdb files just in case
        .Filter = "CSD File (*.accdb)|*.accdb"
        ' Default directory is where this file is located
        .Directory = CurrentProject.Path
        ' Tell the common dialog that the file and path must exist
        .ExistFlags = FileMustExist + PathMustExist
        If .ShowOpen Then
            strFile = .FileName
        Else
            Err.Raise 3999
        End If
    End With
    ' Open the "info" form telling what we're doing
    DoCmd.OpenForm "frmReconnect"
    ' ... and be sure it has the focus
    Forms!frmReconnect.SetFocus
    ' Attempt to re-attach the Version table first and check it
    Set tdf = db.TableDefs("ztblVersion")
    tdf.Connect = ";DATABASE=" & strFile
    tdf.RefreshLink
    ' OK, now check linked table version
    Set rst = db.OpenRecordset("ztblVersion")
    rst.MoveFirst
    ' Call the version checker
    If Not CheckVersion(rst!Version) Then
```

```
            ' Tell the caller that we failed
            ReConnect = False
            ' Close the version recordset
            rst.Close
            ' ... and clear the object
            Set rst = Nothing
            ' Bail
            Exit Function
        End If
        ' Passed version check - edit the version record
        rst.Edit
        ' Update the open count - we check this on exit to recommend a backup
        rst!OpenCount = rst!OpenCount + 1
        ' Write it back
        rst.Update
        ' Close the recordset
        rst.Close
        ' ... and clear the object
        Set rst = Nothing
        ' Now, reattach the other tables
        ' Strip out just the path name
        strPath = Left(strFile, InStrRev(strFile, "\") - 1)
        ' Call the generic re-attach function
        If AttachAgain(strPath) = 0 Then
            ' Oops - failed.  Raise an error
            Err.Raise 3999
        End If
        ' Close the information form
        DoCmd.Close acForm, "frmReconnect"
        ' Clear the db object
        Set db = Nothing
        ' Return a positive result
        ReConnect = True
        ' ... and exit
Connect_Exit:
        Exit Function
BadReconnect:
        ' Oops
        MsgBox "Reconnect to data failed.", vbCritical, _
          "CSD Contacts Manager"
        ' Indicate failure
        ReConnect = False
        ' Close the info form if it is open
        If IsFormLoaded("frmReconnect") Then DoCmd.Close acForm, "frmReconnect"
        ' Clear the progress meter
        varRet = SysCmd(acSysCmdClearStatus)
        ' ... and bail
        Resume Connect_Exit
End Function
```

The code begins by attempting to open the linked ztblVersionTable. If that action generates an error, the code immediately jumps to the Reattach label about halfway down the listing. If the version-checking table opens successfully, the code next calls the CheckVersion function (not shown here) that compares the version value in the table with a public constant saved in the modGlobals module. If the versions don't match, that function displays an appropriate error message and returns a False value to this procedure. If the version check fails, this procedure returns a False value to the original calling procedure (in the frmSplash form's module) and exits.

If the versions do match, the code next loops through all the table definitions in the database and attempts to open a recordset on each one to verify the link. Note that the code leaves the recordset open on the version-checking table. If it didn't do this, each subsequent open and close would need to establish a new network connection to the file server, and the checking of all tables would take minutes instead of seconds. Note also that the code uses the SysCmd system function to display a progress meter on the Access status bar.

If all linked tables open successfully, the procedure returns True to the calling procedure and exits. If opening any of the tables fails, the code immediately jumps to the Reattach label to attempt to fix all the links.

The code beginning at the Reattach label clears all errors, sets an error trap, and then displays a message informing the user that there's a problem. After the user clicks OK in the dialog box, the code creates a new instance of the ComDlg class module, sets its properties to establish an initial directory and ask for the correct file type, and uses the ShowOpen method of the class to display the Open File dialog box in Windows. The class module returns a True value if the user successfully locates the file, and the code retrieves the class module's FileName property to find out the path and name of the file chosen by the user. If the ShowOpen failed, the code raises an error to be logged by the error-handling code at the end of the procedure.

Next, the code opens a form that is a dialog box informing the user that a reconnect is in progress. The code attempts to fix the link to the version-checking table using the path and file that the user selected. Notice that the code sets the Connect property of the TableDef object and then uses the RefreshLink method to reestablish the connection. If the table isn't in the file that the user selected, the RefreshLink method returns an error, and the code after the BadReconnect label near the end of the procedure executes because of the error trap.

After checking that the version of the code matches the version of the database, the code calls the AttachAgain function (not shown here) and passes it the path and file name. You can also find this function in the modStartup module. The function loops through all the

TableDef objects, resets the Connect property for linked tables, and uses RefreshLink to fix the connection. Because this sample database also has some linked Microsoft Excel worksheets, you'll find that code in the AttachAgain function checks the type of linked table and sets up the Connect property appropriately.

If you'd like to see how this code works, you can open the Contacts.accdb file and then use Windows Explorer to temporarily move the ContactsData.accdb, Fictitious Companies.xlsx, and Fictitious Names.xlsx files to another folder. Open the frmSplash form, and you should see the code prompt you to identify where you moved the ContactsData.accdb file. The code in the AttachAgain procedure assumes that the two Excel files are in the same folder as the ContactsData.accdb file.

You can study the other functions called by the ReConnect function in the modStartup module on your own. We provided comments for every line of code to help you understand how the code works.

> **Note**
>
> **In the initial release of Access 2010, there is a product bug that prevents our ReConnect function from correctly reconnecting the tblContacts linked table. The code fails to reconnect this linked table because this particular table contains a Multi-Value Field. Microsoft is aware of this issue and is looking into it. For the sample databases to work properly, you need to install them in the following directory: C:\Microsoft Press\Access 2010 Inside Out. If you cannot install the files to that directory, you'll need to reconnect your linked tables manually.**

Understanding Runtime Mode

When Access starts in runtime mode, it does not allow the user to access the Navigation pane or to use any of the built-in ribbons. Therefore, the user can only run your application, not edit any of the objects. As you might expect, many keystrokes are also unavailable, such as pressing F11 to show the Navigation pane or pressing Ctrl+Break to halt Microsoft Visual Basic code execution. You can obtain additional tools and a license to freely distribute runtime versions of your applications by downloading the Access 2010 Runtime Tools from Microsoft's website. In versions of Access before Access 2007, you had to purchase the developer tools and runtime extensions separately. The big news is that for Access 2010,

Microsoft is offering these tools at no additional charge. This set of software tools includes a royalty-free license to distribute the runtime modules for Access 2010. This allows you to distribute your database with the modules to execute in runtime mode to users who do not have Access installed on their systems. You can download the Access 2010 Runtime Tools from the following location:

http://www.microsoft.com/downloads/details.aspx?FamilyID=57a350cd-5250-4df6-bfd1-6ced700a6715&displaylang=en

You can use the Package Solution Wizard to create installation files that include your database application, the runtime modules necessary to run your application, and any supporting files (such as ActiveX controls or icons). The wizard creates a standard Windows Installer setup file (.msi). After you install the Package Solution Wizard from your Office 2010 installation DVD-ROM or CD-ROM, you can start the wizard by clicking the File tab on the Backstage view, clicking Save & Publish, and then clicking the Package Solution button under Package and Distribute. See "Choosing Options When You Have No Previous Version of the Microsoft Office System," on page 1377, for more information on installing the Package Solution Wizard. The wizard creates a standard Windows Installer setup file (.msi), but you'll need to download and install the Runtime tools to package the Runtime with your setup file.

To execute successfully in runtime mode, your application must have the following:

- All features of the application must be implemented with forms and reports. The user will not have access to the Navigation pane to execute queries or to open tables.
- The application must have a startup form or an Autoexec macro that opens a startup form.
- All forms and reports must have custom ribbons because runtime mode does not provide the built-in ribbon.
- You must implement error trapping in all your macros and Visual Basic procedures. Any untrapped errors cause the application to exit.
- If you automate your application with Visual Basic or use macro actions that are not trusted, you must ensure either that the user places your database in a trusted location or that you digitally sign the database and instruct the user how to trust the signature. For more about digitally signing a database, see "Packaging and Signing Your Database," on page 1739.

- The application should execute the Quit method of the Application object to terminate. If you simply close the final form, the user will be left staring at an empty Access workspace.
- If you automate your application with Visual Basic, you should ensure that your code compiles successfully, with no errors.

The primary sample databases, Conrad Systems Contacts (Contacts.accdb), Housing Reservations (Housing.accdb), and Back Office Software System (BOSS.accdb), meet the preceding requirements except that the startup form is set to frmCopyright to display important information each time you open one of the databases, and the Exit button on the main switchboard form merely closes the form and attempts to return to the Navigation pane.

If you would like to see what runtime mode looks like, you can test it using existing databases. Open the Housing.accdb desktop database, click the File tab on the Backstage view, click Options, and then click the Current Database category. In the Application Options section, select frmSplash in the Display Form list, and then click OK. Close the Housing.accdb database. The sample files include a shortcut, Housing Runtime, that opens this database in runtime mode. This shortcut should work so long as you installed the Office 2010 system in the default folder (C:\Program Files\Microsoft Office\Office 14) and the sample files in the default folder (C:\Microsoft Press\Access 2010 Inside Out). If necessary, you can change the shortcut by right-clicking the shortcut and selecting Properties on the shortcut menu. The target setting in this shortcut is as follows:

```
"C:\Program Files\Microsoft Office\Office14\MSACCESS.EXE" "C:\Microsoft Press
\Access 2010 Inside Out\Housing.accdb" /runtime
```

After you change the Display Form setting in the Access Options dialog box and correct any settings in the shortcut target, you can double-click the shortcut to start the application in runtime mode. Click OK in the opening message box, and then sign on as any employee of your choosing to see the main switchboard form. Try pressing F11 to see whether anything happens—the Navigation pane should not appear. You can move around in the application using the command buttons on the various switchboard forms and the buttons on the custom ribbons. When you click Exit on the main switchboard, code in the form closes all open forms and then closes the switchboard. You'll be left looking at a blank Access workspace and a very limited set of options when you click the File tab on the Backstage view. You can click Close Database from here to close this limited copy of Access or click Exit. Be sure to open the Housing.accdb file again (without the Runtime shortcut), click Exit on the sign-on form, and then change Display Form in the Access Options dialog box back to (none) before proceeding further.

INSIDE OUT

Change the File Extension to Test the Runtime Mode

You can also test your application in runtime mode by changing the file extension on your database. In Windows Explorer, right-click the Housing.accdb sample database file, click Rename, and change the file name to Housing.accdr. Windows prompts you that changing the file extension might make the file unusable. Click Yes in this message box and then open the Housing.accdr database. You'll notice you see the limited ribbon, Quick Access Toolbar, and Backstage options as you did when using the runtime shortcut.

Creating a Database Template

When you create a client or web database from the New tab on the Backstage view, you're actually creating a new database (.accdb) from a database template. A database template is a file that holds the definition of an Access database. Database templates have an .accdt file extension and are essentially zip files that contain all the data and definitions describing a complete Access database. Database templates can be reused over and over to create additional, identical copies of an Access database. For example, when you create (or *instantiate*, as Microsoft uses the term) a new Access database file from the Backstage view, you select a client or web template to use, and Access instantiates the database template to create a fully functioning database. If you select the same template again on the Backstage view, you can create additional copies of the database. You can use a database template, then, as a distribution mechanism to your users. In fact, when you download Access templates from Microsoft's website, you're using a database template.

As you learned in Chapter 4, Access 2010 now includes Application Parts that you can use to help create your client and web databases. An Application Part is a database object or group of database objects that you can add to an existing open database. Application Parts also have the .accdt file extension and essentially are just parts of a database instead of a complete database. The main difference for you to remember is that you can only instantiate an Application Part into an existing database, and you create an entirely new database file when you instantiate a database template from the Backstage view. Note that you can create database templates and Application Parts for both client and web databases.

To create a database template of any completed database application or an Application Part, open the database and close any objects. Click the File tab on the Backstage view, click Save & Publish, and then double-click the Template button under Save Database As. Access opens the Create New Template From This Database dialog box, as shown in Figure 27-2.

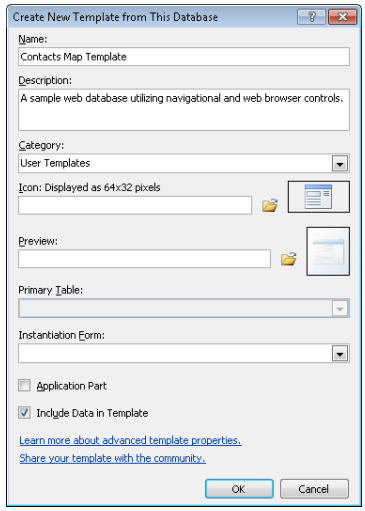

Figure 27-2 You can set your options for creating your database template in the Create New Template From This Database dialog box.

Access uses this same dialog box when you are creating a complete database template or an Application Part. If you want to create an Application Part, you need to have only the objects you want in the source database. For example, if you want to create an Application Part for a new form, just have that form by itself in an otherwise empty database container. When you create your Application Part, Access includes all objects in the database, which, in this case, would be just your one form. The options in the Create New Template From This Database dialog box are as follows:

- **Name** Enter the name of the database template you want to create. This field is required.
- **Description** Enter an optional description for the database template. The description appears in the tooltip when you hover over the Application Part in the Application Parts gallery. This description also appears in the Solutions Gallery if you upload a web database template to a server running SharePoint 2010.
- **Category** If you're creating an Application Part, you can select in which section of the Application Parts gallery your new custom Application Part appears. You can choose User Templates, the default, or type in a new category. You can use this option to organize your Application Parts.
- **Icon** If you want, you can select an icon to appear for your Application Part in the Application Parts Gallery. Access displays the icon at 64x32 pixels.

- **Preview** If you want, you can select a graphic to appear for your database template in the right task pane on the Backstage view.
- **Primary Table** If you have at least one table in your custom Application Part, Access opens the Create Relationship dialog box when you instantiate your custom Application Part from the Application Parts gallery. Use this option to tell Access which table to use as the default table choice in the Create Relationship dialog box. Note that this option is available only when you select the Application Part check box option.
- **Instantiation Form** If you want, you can select a form from the list of saved forms in your database to use as an instantiation form when a user instantiates your template. This option is useful when you want to display a splash form one time or perform some special startup routines that are necessary after Access instantiates the template. However, be warned that Access *deletes* the form immediately after the form is closed. We recommend you use this option only with extreme caution. If you have a startup form defined in the Access Options for your database, Access honors that setting when you instantiate the template and create a new database. Don't be confused by the terms *startup form* and *instantiation form*; they are not the same. If you select an instantiation form, Access only uses the form once and then deletes the form. Note that this option is not enabled unless you have at least one form in your database.
- **Application Part** Select this option, cleared by default, if you want the template to appear in the Application Parts gallery. If you leave this option cleared, the template appears only in the My Templates section on the New tab of the Backstage view.
- **Include Data in Template** Select this option, cleared by default, if you want Access to include all the data from the source database in the database template.

At the bottom of the dialog box, you can click the two links to learn more setting template properties and how to share your template on Microsoft's website. In Figure 27-2, you can see we provided a name for our new template (based on the ContactsMap.accdb sample database), a description, and selected the option to include the sample data. After you click OK, Access generates the database template or Application Part. After a few moments (or longer if your database has many objects and data) Access displays a confirmation message indicating the location of your new template file, as shown in Figure 27-3.

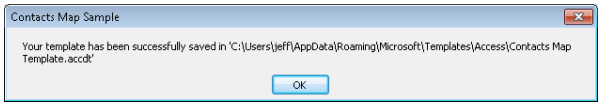

Figure 27-3 Access informs you where it saved your new template file.

Access, by default, saves the template in this location: C:\Users\<UserName>\AppData\Roaming\Microsoft\Templates\Access. You'll notice that the folder where Access saves the template—AppData—might be a hidden folder on your computer. When you click the File tab on the Backstage view, click New, and then click My Templates, you'll see your new template listed here, as shown in Figure 27-4 You can now create a new database file from this saved template. Access lists any templates it finds in the folder listed above in My Templates section of the New tab on the Backstage view. If you've created an Application Part, you can see your new option by clicking the Application Parts button in the Template group on the Create tab. Access displays your Application Part at the bottom of the Application Part Gallery in the category you created. You can send your saved database templates and Application Parts to users for them to use in their own applications.

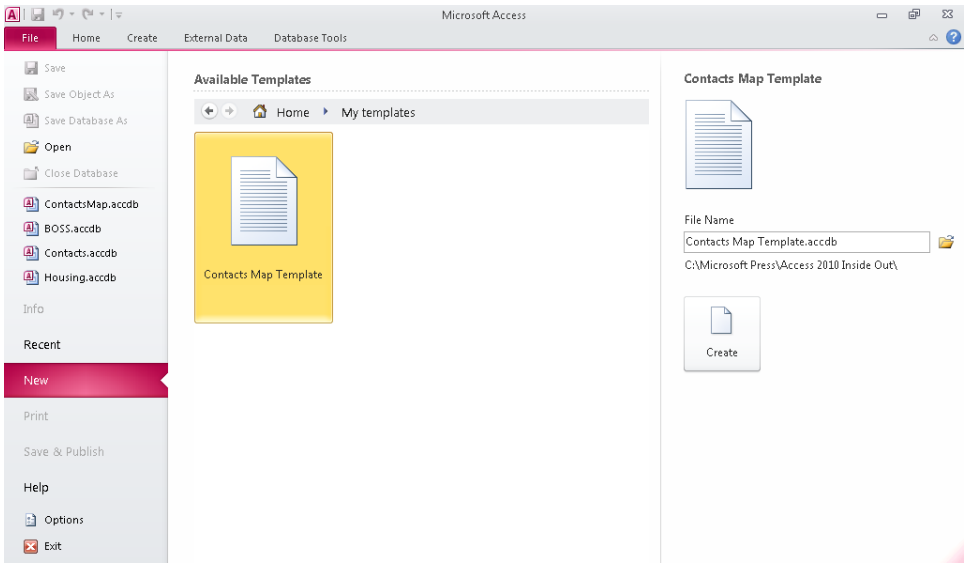

Figure 27-4 You can see your new template displayed in the Backstage view.

INSIDE OUT

Displaying Custom Templates for All Users

If you add database templates and Application Parts to the C:\Program Files\Microsoft Office\Templates\1033\Access\ folder, Access displays those templates to all users of the same computer. Note that the location path above assumes an English-language installation of Office.

Creating Custom Data Type Parts

Similar to database templates and Application Parts, you can create your own custom Data Type Parts and use them in your applications or distribute them to other users. To create a Data Type Part, open the table that contains the fields you want to use for your custom Data Type Part in Datasheet view. Next, highlight the field or fields you want to use, click the More Fields button in the Add & Delete group on the Fields tab, and then click Save Selection As New Data Type at the bottom of the gallery. Access opens the Create New Data Type From Fields dialog box, as shown in Figure 27-5.

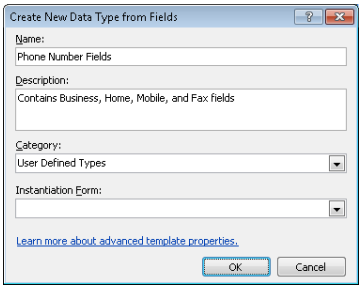

Figure 27-5 Enter your custom Data Type Part options on the Create New Data Type From Fields dialog box.

The options in this dialog box are essentially the same as the ones you saw previously when creating database templates and Application Parts. You can see in Figure 27-5 we've defined a name and description for our custom Data Type Part and left the default Category as User Defined Types. After you click OK, Access generates the custom Data Type Part and displays a confirmation message upon success. Access saves the Data Type Part with an .accft file extension in the same folder as the custom database templates and Application Parts. If you open a table in Datasheet view, you'll now see your custom Data Type Part displayed in the Data Type Part gallery, as shown in Figure 27-6.

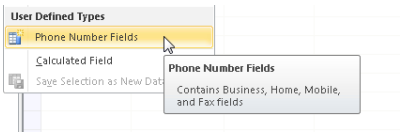

Figure 27-6 Access displays your custom Data Type Part in the Data Type Parts gallery.

INSIDE OUT

Displaying Custom Templates for All Users

If you want to remove a custom Application Part or Data Type Part from the galleries, you have two options. First, you can navigate to the hidden folder where Access stores the .accdt and .accft files and either delete the files or move them out of that folder. Second, you can right-click the gallery and click Delete Application Part From Gallery or Delete Data Type Part From Gallery. If you choose the second option, Access displays a warning message indicating that it will permanently delete the file from your hard drive. If you want to keep the file, you should choose the first option and move the file to a different location for storage.

Creating an Execute-Only Database

Even if you have secured your database, you might still want to be sure that no one can examine or change the Visual Basic procedures you created. After you have fully compiled your Visual Basic project, Access no longer needs the original text of your Visual Basic statements. You can create a special execute-only copy of your database by using one of the utilities supplied with Access. An additional advantage of an execute-only database is that it might be significantly smaller than a copy that contains all the code—particularly if you have written many Visual Basic procedures.

To create an execute-only copy of any completed database application, open the database, and close any open objects. Click the File tab on the Backstage view, click Save & Publish, and then click the Make ACCDE button under Advanced. (Note that if you're using an Access .adp project file, this button says Make ADE.) The Save As dialog box asks you for a location and name for your new database or project. It then makes sure the current database is fully compiled and saved, copies the database to a new file with the appropriate .accde or .ade extension, removes the Visual Basic source code, and compacts the new file.

If you open an .accde file (you can find a file called Contacts.accde on the companion CD), you'll find that you can't open any Visual Basic module—neither the module of any form or report module nor any module object. You also won't be able to open a form or a report in Design view or Layout view. (This is an additional benefit of an .accde file.) Figure 27-7 shows the Modules list in the Contacts.accde database file. Notice that the Design View button is disabled on the shortcut menu, indicating that you cannot view the source code of an existing module and that the only way to run code is through the database application's interface. The Module and Class Module buttons in the Macros & Code group on the Create tab are also disabled, so you cannot create new modules. You'll find that the Design View option is also disabled for all forms and reports, and all the commands in the Forms and Reports groups on the Create tab are unavailable.

> **Note**
>
> **When you create an execute-only version of your database, you should create the file using the same version of Access your users are using.**

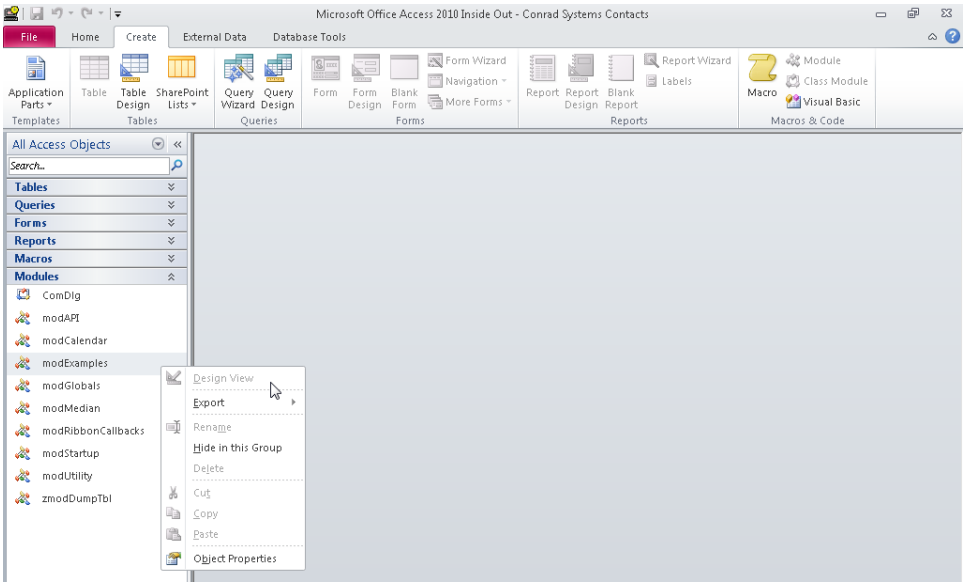

Figure 27-7 You can't edit any modules in the Contacts.accde database file.

Creating an Application Shortcut

When you're all done, you might need to create a way for users to easily start your application. If your users all have a copy of Access, you could give them your database application files and simply instruct them to open the appropriate file. But what if the user doesn't have Access, so you have to set them up to execute your application with the runtime version of Access? What if you want to also define certain utility functions that the user might need to execute from time to time? The answer is to create a shortcut.

You use shortcuts all the time in Windows to start programs on your computer. When you install an application on your computer, the setup program usually creates a shortcut that it adds to your Start menu. Some setup programs also add a shortcut on your desktop. The icon for a shortcut on your desktop has a small white box in the lower-left corner with an arrow in it. You can right-click a shortcut and choose Properties from the shortcut menu to see the definition of the shortcut.

To create a shortcut on your Windows desktop for your Access application, right-click the desktop, click New, and then click Shortcut. You can also create a shortcut in a folder by opening Windows Explorer, navigating to the folder you want, pressing the Alt key to view the menu bar, clicking New on the File menu, and then clicking Shortcut. In either case, Windows opens the Create Shortcut Wizard to help you find the program you want the shortcut to open. Click Browse, and find C:\Program Files\Microsoft Office\Office14\MSACCESS.EXE. Click OK to select the file, and then click the Next button. Give your shortcut a name, such as the name of the database you plan to open with the shortcut, and then click Finish.

Right-click your new shortcut and click Properties. You'll see a dialog box similar to Figure 27-8.

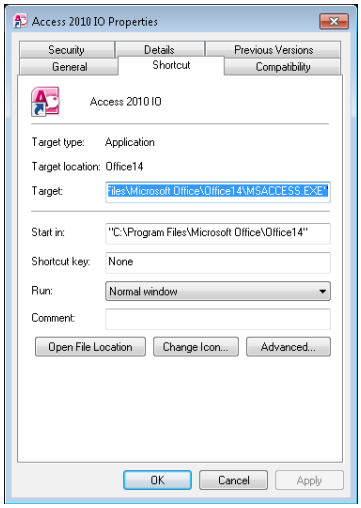

Figure 27-8 You can modify the Target setting for a Windows shortcut in the shortcut's Properties dialog box.

At the top of the Shortcut tab is the icon the shortcut displays and the name of the shortcut. You can click the General tab and enter a new name to rename your shortcut. Target Type tells you that this shortcut starts an application. Target Location displays the original location of the program that this shortcut starts. The Target box allows you to specify the program or file that you want to run. Note that at this point, your new shortcut starts only the Access program—it doesn't specify a file to open or any parameters. You can specify the database file name and enter any parameters used by the program in the Target box.

Immediately following the name of the Access program in the Target box, enter a space followed by the database you want to open (with its full path). If the path or file name

contains any blanks or special characters, you must enclose the file path in double quotes. Follow the name of the database with the options you need to perform the task you want. For example, to open the Housing.accdb sample database from the default installation, the target setting in this shortcut is as follows:

```
"C:\Program Files\Microsoft Office\Office14\MSACCESS.EXE" "C:\Microsoft Press
\Access 2010 Inside Out\Housing.accdb"
```

> **Note**
>
> **You can also specify only the name of a database file in the Target box in a shortcut, and Windows opens the program that can process this file (in this case, Access) when you double-click the shortcut. However, Access won't recognize any parameters that you include after the file name. You must specifically ask to open the Access program (MSACCESS.EXE) and add the file name and parameters.**

Table 27-1 summarizes the shortcut command-line options you can use. When you include multiple command-line options, separate each with a space.

Table 27-1 Access Shortcut Command-Line Options

Option	Description
<*database*>	Opens the specified database. If the path or file name contains blanks, you must enclose the string in double quotes. Must be the first option after the folder and file location for MSACCESS.EXE.
/cmd <*command string*>	Specifies a program parameter that can be retrieved by a Visual Basic procedure using the built-in Command function. Must be the last option on the command line.
/compact [<*target*>]	Compacts and repairs the specified database but does not open the database. If you omit the target file name, Access compacts the database into the original file name and location.
/convert <*target*>	Converts the specified version 11 or earlier database to Access 2007/2010 file format and stores it in the target file.
/excl	Opens the specified database with exclusive access. Only one user at a time can use a database that is opened exclusively.
/profile <*userprofile*>	Specifies the name of a user profile in the Windows registry. You can use a profile to override database engine settings and specify a custom application title, icon, or splash screen.

Option	Description
/pwd <*password*>	Specifies the password for the user named in the /user parameter. If the password contains the / or ; character, enter the character twice. For example, if the password is #ab/cd;de, enter #ab//cd;;de. This option applies to databases in Access 2003 and earlier (.mdb) format that have user-level security implemented.
/repair	Repairs the specified database but does not open the database.
/ro	Opens the specified database in read-only mode.
/runtime	Specifies that Access will execute with runtime version options.
/user <*userid*>	Specifies the logon user ID. This option applies to databases in Access 2003 and earlier (.mdb) format that have user-level security implemented.
/wrkgrp <*workgroupfile*>	Uses the specified workgroup file. This option applies to databases in Access 2003 and earlier (.mdb) format that have user-level security implemented.
/x macroname	Runs the specified macro after opening the specified database.

Using a command-line option, you can also create a shortcut to perform the maintenance task of compacting your database. For example, to compact the Contacts.accdb database and save the compacted version to a file named ContactsCompact.accdb in the same folder, enter the following on the Target line:

```
"C:\Program Files\Microsoft Office\OFFICE14\MSACCESS.EXE"
    "C:\Microsoft Press\Access 2010 Inside Out\Contacts.accdb" /compact
    "C:\Microsoft Press\Access 2010 Inside Out\ContactsCompact.accdb"
```

This previous text assumes you've installed the sample files in the default folders. In the Start In box, specify the starting folder for the application. In the Shortcut Key box, you can enter a single letter or number that the user can press with the Ctrl+Alt key combination to run the shortcut. The shortcut key must be unique for all shortcuts on your system. In the Run list, you can choose to start the application in a normal-size window (the default), minimized as an icon on your taskbar, or maximized to fill your screen. In the Comment box, you can enter text that appears when the user rests the mouse pointer on the shortcut.

Click Open File Location to verify that the target you entered is valid. Click Change Icon to select a different icon stored within the target program (MSACCESS.EXE has 80 available icons) or to locate an icon file on your hard disk. Click Advanced if you need to set up this shortcut to run under a specific Windows user ID.

On the Compatibility tab of the Properties dialog box, you can find an option to run the program in compatibility mode as though it's running on an older operating system such as Microsoft Windows 2000 or Microsoft Windows XP. You can also force your display to 256 colors, use 640×480 screen resolution, or disable Windows themes when this program runs. On the Security tab, you can allow or deny permissions to use this shortcut for specific Windows users or groups.

After you have completed the settings you want, click OK to save your changes to the shortcut. You can now double-click the shortcut to run the program with the options you specified.

INSIDE OUT

Creating an Object Shortcut

You can create a shortcut to a specific database object by selecting the object in the Navigation pane and then dragging it to your desktop. Access creates a shortcut file on your desktop for that object. When you double-click the shortcut, Access opens first and then opens the specific object.

Encrypting Your Database

Access 2010 includes a feature to encrypt your database with a password. You can use this feature to prompt users for a password before opening the database. When you encrypt the database, Access makes the data unreadable to tools that can read binary data stored in the physical file.

To encrypt your database with a password, you must first open your database in exclusive mode. Click the File tab on the Backstage view, and then click Open. Select your database, and then, in the Open dialog box, click the arrow on the Open button and then click Open Exclusive, as shown in Figure 27-9.

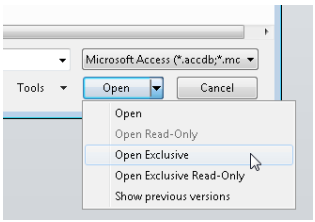

Figure 27-9 You must open your database in exclusive mode to encrypt the database with a password.

After your database opens, click the File tab on the Backstage view, click Info, and then click Encrypt With Password. Access opens the Set Database Password dialog box, as shown in Figure 27-10. Enter your password in the Password text box, and then reenter it in the Verify text box. Click OK, and Access checks to see whether the two passwords match. If the passwords match, the next time you open the database, Access prompts you for the database password. Note that you might receive a warning message when you save the password, indicating that row level locking will be ignored because you are encrypting the file. Click OK to dismiss this message.

If you want to remove the password later, you'll need to open the database in exclusive mode and then click the Decrypt Database button on the Info tab of the Backstage view. In the Unset Database Password dialog box, type your password in the Password text box, and then click OK. The next time you open the database, Access does not prompt you for a password.

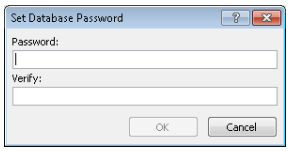

Figure 27-10 Enter your password in the Set Database Password dialog box.

INSIDE OUT How Useful Is Encryption?

Encrypting your database does provide a thin layer of security, but it's not foolproof. If you distribute this database to many users, each user has to know the password to open and use the database. Because all users need to share one common password, we don't recommend encryption for multiple-user scenarios. If you're creating this database for your own personal use, encrypting might be an option to keep other people out of your database. Just remember, however, a determined hacker could still gain access to your database, given enough time and determination. If you're really concerned about security, you should consider using network file share permissions or a server-based back-end data store such as Microsoft SQL Server or SharePoint to control permissions.

Packaging and Signing Your Database

If you want to send a database to other users, you can certainly put it in a zip file and e-mail it. However, unless the recipient really trusts that the e-mail came from you (it's easy to spoof a sending e-mail address), the recipient might not be willing to open your file. Access 2010 provides a tool that lets you compress your database file and include it inside a file that is digitally signed.

So, what is "digitally signed?" If you've surfed the Internet at all, you've probably encountered several digitally signed files. For example, when a website wants to download and install an ActiveX control and you have security enabled in your browser, your browser prompts you to decide whether to download and run the file. If the file is digitally signed, you'll see verified information about the publisher that has been authenticated over the Internet by a commercial certificate authority such as VeriSign, GeoTrust, or GoDaddy. In many cases, you can select an option to accept all signed files from a specific trusted source (such as Microsoft) so that you won't be prompted again if you encounter another signed file from the same source.

Access 2010 lets you package your database into a compressed Deployment file (.accdc) and then sign it with a digital certificate that is ready to send to your users. When a user attempts to open your file, Access 2010 uses the digital certificate to verify the source of the file and that all objects in the database have not been changed since the database was signed. If the user trusts the digital certificate, Access 2010 opens and extracts your database file ready for the user to run.

But there's one catch. If you need to distribute this database to other users, you must purchase a digital certificate from a commercial certificate authority, and they're not inexpensive. When you own a commercial certificate, you can use it to "sign" any file that you publish (perhaps on your website) or send to others. The program that the recipient uses to open the file can send the certificate information over the Internet to the validating authority. The validating authority verifies the certificate and sends back information about the publisher of the file. The recipient can decide to trust the information to avoid being prompted in the future, decide to open the file anyway, or cancel the request.

INSIDE OUT Using a Self-Signing Certificate

If you want to test how packaging and signing works, you can create and use a self-signing certificate. The Office 2010 system includes a tool to create self-signing digital certificates—SelfCert—that you can use for packaging databases. These certificates, however, are valid only for the computer on which you create them. To create a digital certificate for yourself in Windows, click the Start button, click All Programs, click your Microsoft Office folder, click Microsoft Office Tools, and then click Digital Certificate For VBA Projects. In the Create Digital Certificate dialog box, enter the name of the certificate you want to create and then click OK. Because a self-signing digital certificate is valid only on the computer on which you create it, if you package and sign a database with a self-signing certificate and then send it to someone else, the certificate is no longer valid.

To package and digitally sign your database, open the database, click the File tab on the Backstage view, click Save & Publish, and then click the Package And Sign command under Advanced. Access opens the Windows Security dialog box, as shown in Figure 27-11. Click the link displayed in the middle of the dialog box to review all the details of the selected certificate. Select the certificate you want to use from the list, and then click OK. (In this example, we used a self-signing certificate for demonstration purposes.)

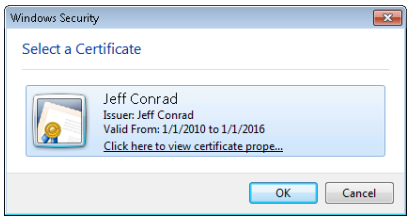

Figure 27-11 Select the digital certificate you want to use to sign the package.

Access opens the Create Microsoft Access Signed Package dialog box, as shown in Figure 27-12. Enter or browse to the location in which you want to save your signed database package. In the File Name box, enter a name for this new packaged file and then click Create. Access compresses your database, "signs" the file using the digital certificate you selected, and places the database and signature into an .accdc file in the location you specified.

Note

You can package and sign only those databases saved in the .accdb file format. In addition, you can include only one database in a package. If you want to digitally sign the Visual Basic code in an .mdb or .adp file, open the Visual Basic Editor, and click Digital Signature on the Tools menu. Your VBA project must be compiled. If you make any further changes to your database after signing it, the digital signature becomes invalid.

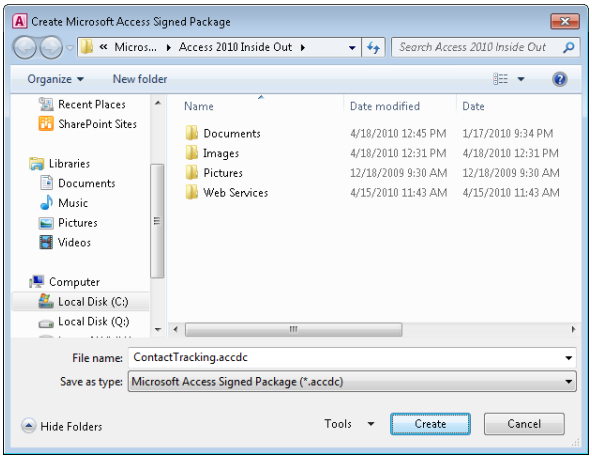

Figure 27-12 Enter a file name and location for your packaged database.

When you or your user opens a signed database, Access displays the Microsoft Access Security Notice dialog box (shown in Figure 27-13) if you have not previously trusted this publisher. If you're unsure of the source of this certificate, you can click the Show Signature Details link to examine all the details about the publisher. If you click Trust All From Publisher, Access always trusts any files from this source. You can see a list of trusted publishers in the Trusted Publishers list in the Trust Center, which you can access from the Access Options dialog box.

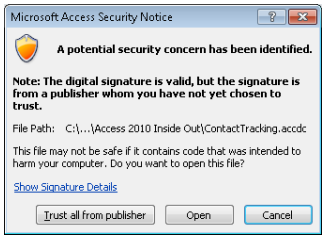

Figure 27-13 Click Open if you trust the publisher and want to open the database.

Click Open if you trust the publisher, and Access opens the Extract Database To dialog box, as shown in Figure 27-14. Enter a name in the File Name text box, select a location to save the extracted database, and then click OK. Access extracts the database from the .accdc file, saves it to the location you specified, and then opens the extracted file. Note that Access might still disable content in this database based upon your settings in the Trust Center.

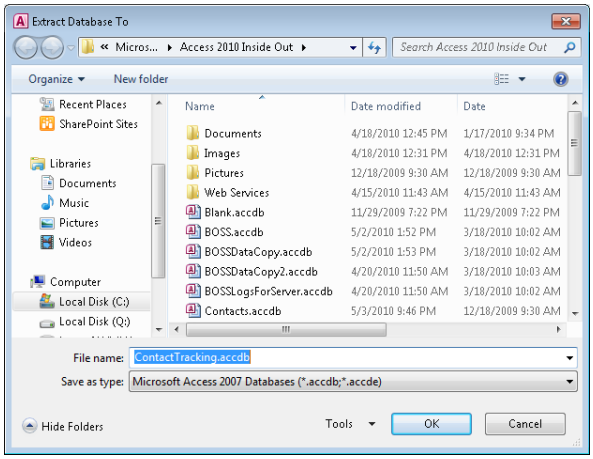

Figure 27-14 Select a location to extract the packaged database.

This concludes the CD chapter portion of Microsoft Access 2010 Inside Out. Be sure to also read the reference articles on the CD that you'll find essential for increasing your knowledge about building applications using Access.