CHAPTER 28

Designing Forms in an Access Project

Understanding Form Differences in an Access Project 1547
Setting Recordset Type 1554
Understanding Max Records 1555
Working with Server Filters 1558
Setting Input Parameters 1561

The most noticeable differences between Access desktop databases (.accdb) and Access project files (.adp) are in the tables and queries (views, functions, and stored procedures). Because Access projects store all their tables and queries in Microsoft SQL Server, project tables and queries have many new design elements that you must understand to effectively use them. However, you won't notice many differences when building forms.

For the most part, forms in an Access project are identical to forms in an Access desktop database. In fact, you can successfully begin building forms that edit data from tables and views without knowing any details about the differences. However, to create a form that uses an in-line function or a stored procedure as its record source, especially if the function or procedure requires parameters, you must understand how forms in an Access project are different. The same is true if you want to create a combo box or a list box that uses a parameter query as its row source.

Note

This chapter assumes that you have already read the earlier chapters on form design (Chapter 11, "Building a Form," Chapter 12, "Customizing a Form," and Chapter 13, "Advanced Form Design") and have a good grasp of the basic form design concepts. You should be familiar with the form designer and know how to create forms that allow you to add and update data in tables. You can find the examples discussed in this chapter in the project file version of the Conrad Systems Contacts sample application, Contacts.adp, on the companion CD.

Understanding Form Differences in an Access Project

When you first open a form in a project file, you might not notice some of the subtle differences immediately. Figure 28-1 shows you the frmContactsSimple form open in the Conrad Systems Contacts project file.

C28623252.indd 1547 2/23/2007 9:49:04 PM

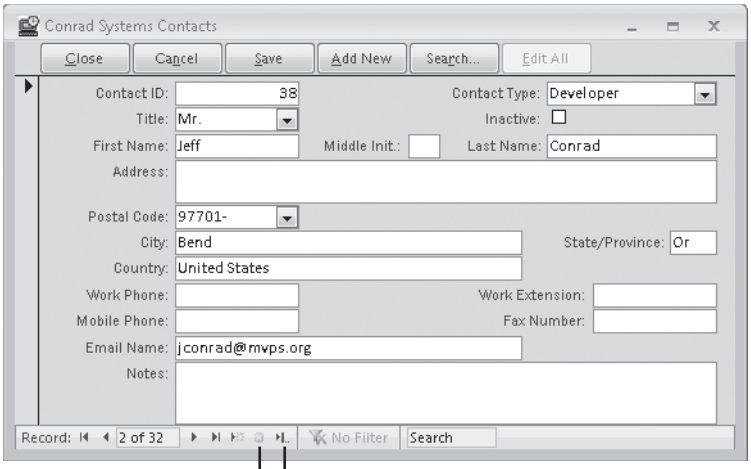

Figure 28-1 The frmContactsSimple form is a sample form in the Contacts.adp project file.

Take a close look at the navigation bar at the bottom of the form, and notice that the bar has two new buttons. The first, an X inside an oval, is the Stop Refresh button. To its right is the Set Maximum Record Count button.

When the user first opens a form in a project, Microsoft Office Access 2007 begins fetching the rows to display in the form and displays the first row immediately. When the recordset for the form has many rows, Office Access 2007 continues to fetch the data as the user begins to work with the form. Access enables the Stop Refresh button while it is still fetching rows. If the user needs only the first few rows fetched by the form, the user can click the button to ask Access to stop fetching rows, thereby freeing up resources on the server. This is a useful option when the user already has enough rows to work with and the server is taking too long to download the remaining records. If the user later decides to see all the rows, clicking Refresh in the Records group on the Home tab asks Access to re-fetch the rows. The user can click the Stop Refresh button again in the middle of the process or allow Access to fetch all the rows.

Access fetches all recordsets bound to a form in a project file as snapshots—literally copies of the data from the server. This method takes fewer resources from the server and places less strain on the network connection because Access uses the local copy of the data to display the information in the form instead of fetching data from the server when the user moves to different rows. When Access has enough information about the structure and relationships of the tables in the recordset, it allows the user to update the data in the form. When the user changes data and saves it, Access sends a specific Update command to the server to keep the data on the server synchronized with the data displayed in the form. Access does not write back the entire recordset or even the entire updated row.

To improve the overall performance of your project, you can set properties to limit the number of rows fetched for all forms or datasheets in the project. You can also set a

limit on a form-by-form basis. (You'll learn the details about these property settings later in this chapter.) When you set a limit for a form, that limit overrides any limit set for the project, but only for the record source of the form. If SQL Server has more rows to return than the limit allows, it returns as many rows as it can until it reaches the maximum limit. The rows that are actually returned are determined by the sort order of the recordset (the ORDER BY clause of the SQL statement). If there is no sort order in the recordset, the row order of the primary table takes precedence. If you have set the limit too low, the user can click the Set Maximum Record Count button, and Access displays the dialog box shown in Figure 28-2.

Figure 28-2 You can change the maximum record count for the current form by using the Set Maximum Record Count dialog box.

The user can change the Maximum Record Count value, click OK, and then click Refresh in the Records group on the Home tab to make more rows available in the form. However, if the user changes this value and then closes the form, Access asks whether the changes to the form design should be saved. The good news is you can set a form property to hide this button from your users.

Choosing Option Settings

In the Access Options dialog box, you can find settings that affect how all forms work in your Access project. With the Contacts.adp sample project file open, click the Microsoft Office Button, and then click Access Options. Click the Current Database category to see some of the options that apply to your forms, as shown in Figure 28-3.

Even when you have disabled Server Filter By Form, users can still use the standard Filter By Form feature that you learned about in Chapter 10, "Using Forms." As you might recall, the Filter By Form feature provides lists to help the user select filter values. By default, Access displays only those values it finds in the local snapshot copy of the form's recordset. You can select the Records At Server check box under Show List Of Values In The Filter Lookup Options For Contacts Database section of the Current Database category to ask Access to fetch a complete list of values (for example, all the contact LastName values) from the server. You can also limit the number of values that Access fetches either from the local snapshot or the server by setting a value for the Don't Display Lists When More Than This Number Of Records Read option. When the number of values exceeds the number you set in the option, Access offers only Is Null and Is Not Null values in the list. The default value of 1000 should be sufficient for most purposes without causing performance problems.

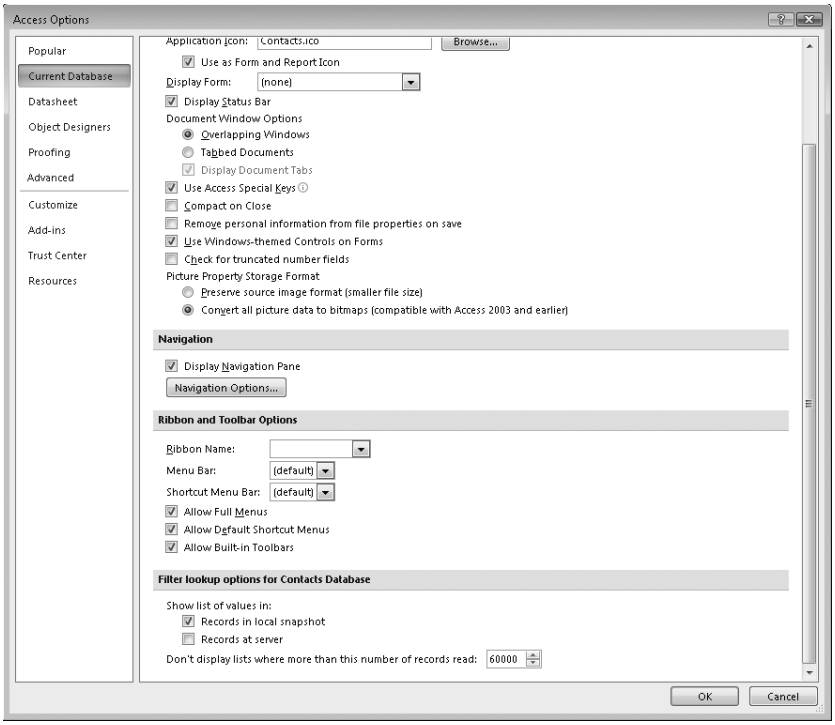

Figure 28-3 The Current Database category in the Access Options dialog box in an Access project file has options useful for forms.

Click the Advanced category to see the other option you can set to affect how forms work in your Access project file. Scroll down to the Advanced section, as shown in Figure 28-4.

In the Advanced section, you can enter a new value for Default Max Records. If you have a slow connection with the server, it is a good idea to lower the value of the default maximum records to decrease the amount of time the user has to wait for a recordset to finish downloading.

When you create a new form in an Access project, Access sets the Max Records property of the form equal to the Default Max Records setting for the project. You can always override this setting by changing the Max Records property of a form, setting the value for an individual form either higher or lower. However, the Max Records property of a form applies only to the form's record source—it does not apply to the recordsets in the row sources of any combo boxes or list boxes on the form. The default value for Default Max Records in an Access project is 10,000, but we set the value in the Conrad Systems Contacts project file to 60,000 because several of the forms have a ZIP code combo box that returns more than 50,000 rows. You'll find that the Max Records property for all forms in the sample project file is set to the lower 10,000 value.

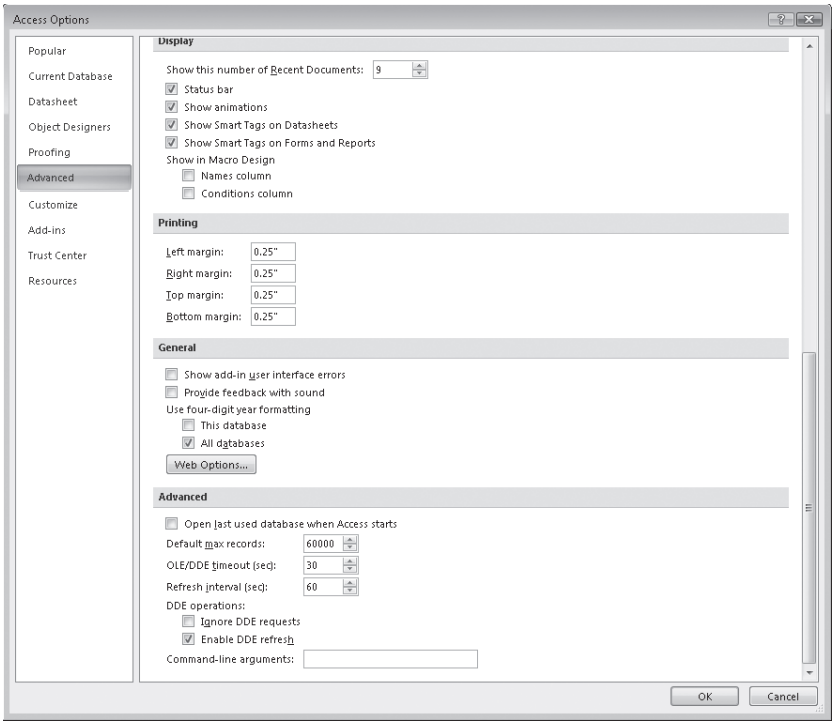

Figure 28-4 The Advanced category in the Access Options dialog box in an Access project file has additional settings that you might use for forms.

INSIDE OUT

Be Mindful of Server Resources

Normally, a user is looking for only one or two records to view or edit at any given time. If the user is browsing forms or opening reports that download the full recordset of a table, the server is wasting a lot of resources delivering records that aren't really needed. Instead, try to create recordsets that return only the amount of data that the user needs to work with. You can do this by using a variety of methods, such as building queries as record sources that limit the size of the recordset, using server filters, or using code to build or filter the recordset based on the choices of the user (searching for a record). Using methods like these makes the database easier to use and also lessens the strain you place on both the server and your network.

Setting Project Form Properties

Let's take a look at some of the new properties you can set for an individual form in an Access project. In the Contacts.adp project file, open the frmContactsSimple form in Design view. Open the Property Sheet window by clicking the Property Sheet button in the Tools group on the Design tab. Select the Data tab to see the data properties for the form, as shown in Figure 28-5.

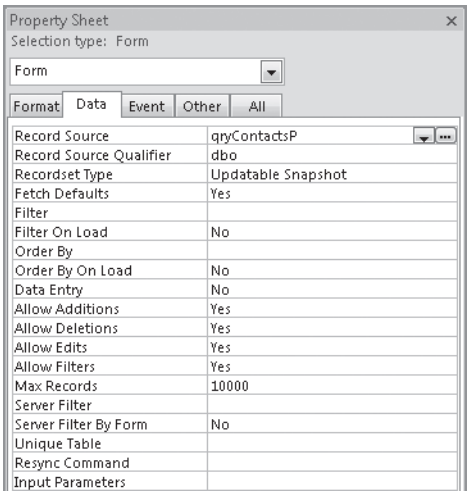

Figure 28-5 These are the data properties of a form in an Access project file.

The Record Source Qualifier property, the Recordset Type property, and the properties beginning with Max Records and ending with Input Parameters are properties that you'll find in forms in an Access project (.adp) but not in forms in a desktop database (.accdb). You can click the Other tab in the Property Sheet window to find the one other property that is unique to forms in an Access project, Max Rec Button.

TROUBLESHOOTING

Why can't I open my form in Layout view in a project file?

The new Layout view for forms and reports in Access 2007 is not supported in project (.adp) files. You can work only in Design view to modify forms and reports in project files.

Table 28-1 shows all the properties that are different for a form in an Access project. You can refer to this table for a brief definition of how each property is used.

Table 28-1 **Additional Form Properties in an Access Project**

Property	Tab	Description
Record Source Qualifier	Data	Identifies the SQL Server owner of the record source for this form. Unless a different owner has been specified by an administrator on SQL Server, this value is normally dbo (short for database owner).
Recordset Type	Data	Identifies the type of recordset. When the record source for the form is a recordset that Access considers updatable, set this property to Updatable Snapshot (the default) to be able to update data via the form. When you want the data in the form to be read-only, choose Snapshot. An updatable snapshot requires more resources because Access must fetch additional information from the server about how to update the record source.
Max Records	Data	Sets the maximum number of records that SQL Server returns from the form's record source. The value in this property can be higher or lower than the Default Max Records setting for the project, and it overrides that setting. This property does not limit the rows returned by the row source of a combo box or a list box. The record limit for the row source of a combo box or list box is determined by the Default Max Records setting for the project.
Server Filter	Data	Specifies a filter that Access sends to the server to limit the rows in the recordset when the user opens the form. You must set Server Filter By Form to Yes (see below) if you want the user to be able to change this filter.
Server Filter By Form	Data	The default setting, No, disables Server Filter By Form mode. Server Filter By Form mode is similar to Filter By Form that you learned about in Chapter 10, except that Access sends the filter to the server rather than filtering the records after fetching them. When you set this property to Yes, Access opens the form's Server Filter By Form window when the user opens the form. The user can edit any server filter saved with the form or specify a different filter. The user must click Advanced Filter Options in the Sort & Filter group on the Home tab and then click Apply Server Filter to apply the filter and see the result.
Unique Table	Data	When you connect your project to an SQL Server version 6.5 or 7.0 database and the form uses as its record source a view or stored procedure that contains two or more related tables, Access sometimes cannot determine the information it needs to update the data displayed by the form. You can set the Unique Table property to the table that provides the unique rows in the recordset—the table on the many side of a one-to-many relationship—so that Access knows how to update the data. You do not need to set this property when your project is connected to an SQL Server 2000 or later database.

Property	Tab	Description
Resync Command	Data	When you connect your project to an SQL Server version 6.5 or 7.0 database, when the form uses as its record source a view or stored procedure that contains two or more related tables, and when you had to specify the Unique Table property to enable updates, you must specify a special Transact-SQL statement to inform Access how to resynchronize the data after an update. Access uses the Resync Command to re-fetch related rows on the one side of a relationship when you change the foreign key on the many side of a relationship. Resync Command is typically a copy of the SQL from the view or stored procedure in the record source with parameters added to the WHERE clause to specify the key values of the many table in the relationship. You do not need to set this property when your project is connected to an SQL Server 2000 or later database.
Input Parameters	Data	Allows you to specify the values of any input parameters that are used in the SQL record source of the form.
Max Rec Button	Other	Specifies whether the Max Records button is visible on the form.

The remaining sections of this chapter provide more detailed explanations and examples of how these properties work in your project forms.

Setting Recordset Type

In Chapter 12 you learned how to set the Allow Edits, Allow Deletions, and Allow Additions properties of a form to restrict what data the user can update via the form. When you want to disallow all edits, deletions, and additions in a form in an Access project, you can also change the Recordset Type property. By default, the Recordset Type property of a form in a project is Updatable Snapshot. As long as Access determines that the record source of the form is updatable, the user is able to edit, delete, and add records using the form. However, when you change the Recordset Type to Snapshot, the form becomes read-only.

To see an example, open the frmXmplZIPReadOnly form in the Conrad Systems Contacts project file. Figure 28-6 shows you the form open in Form view.

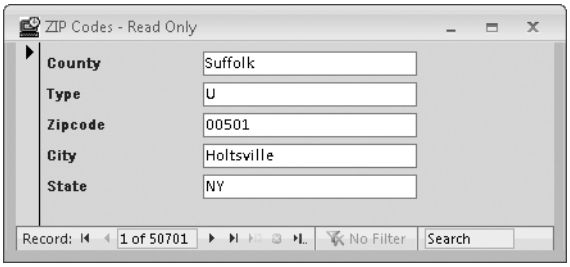

Figure 28-6 The frmXmplZIPReadOnly form is a sample form whose Recordset Type property is set to Snapshot.

Notice that the New Record button on the navigation bar is disabled. Although this might indicate that the Allow Additions property is set to No, this is usually a good indication that the recordset in the form is not updatable. Try typing in any of the fields on the form, and Access plays an alert sound and displays the message "Field '<field name>' is based on an expression and can't be edited" on the status bar. (The message doesn't make sense in this context, but that's what Access displays.)

Switch to Design view, open the Property Sheet window, and change Allow Edits, Allow Deletions, and Allow Additions to No and Recordset Type to Updatable Snapshot. Switch back to Form view, and try it. You won't be able to type in any of the fields, but Access no longer plays a sound or displays a message on the status bar. As you can see, knowing about the Recordset Type property adds one more tool to your bag of tricks. (Be sure to not save the form design changes when you close the form.)

Understanding Max Records

You learned earlier that you can set the Default Max Records property (in the Access Options dialog box) for the entire project and the Max Records property for individual forms. You already know that Default Max Records for the Contacts project is set to a very high number—60,000 records. To see the interaction of these two settings in action, open the frmXmplContacts10 sample form, and open the Postal Code combo box list, as shown in Figure 28-7.

Notice that the form's navigation bar indicates only 10 records in the recordset, but the Postal Code combo box shows a matching value in the list from the row source. ZIP code 97701 is far beyond the 10-row limit in the form, so the combo box must be working from the default max records limit set for the project.

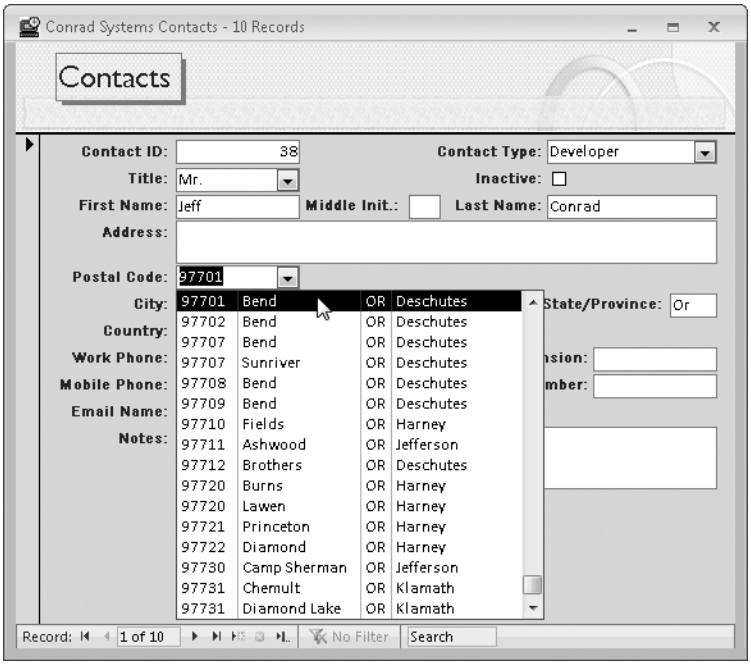

Figure 28-7 Here you can see the form has a Max Records setting of 10, but the combo box displays thousands of rows.

To see what happens when you change the Default Max Records setting for the project, try this little exercise:

1. Switch to Design view.
2. Click the Microsoft Office Button, click Access Options, and then click the Advanced category.
3. Set the Default Max Records option for the entire project to 100 in the Advanced section, and click OK.
4. Switch back to Form view, and open the list in the Postal Code combo box.

Your result should now look like Figure 28-8.

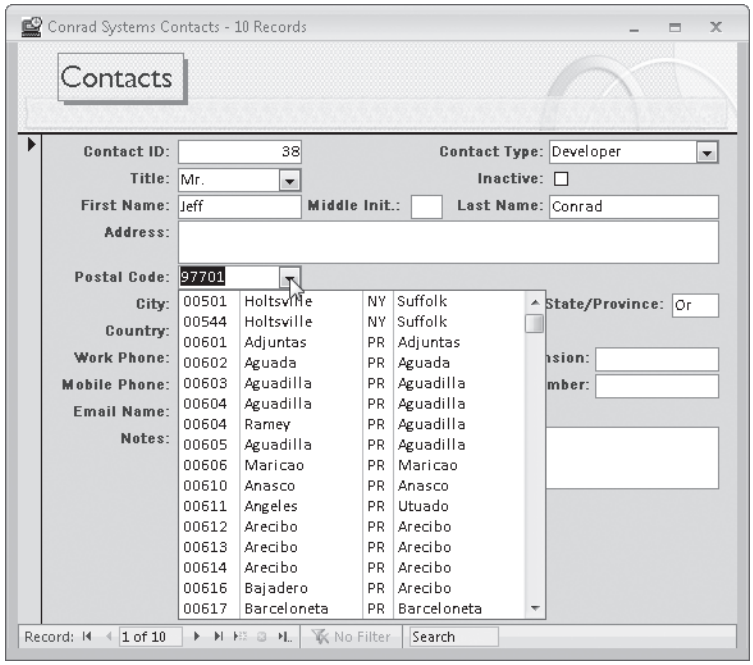

Figure 28-8 You can see the limited postal code list on the frmXmplContacts10 form after lowering the Default Max Records setting for the project.

Notice that the value for Postal Code in the current record no longer matches a value in the list. The combo box list does not have a match for the current value of the field because the list is now limited to 100 rows, the default maximum records for the entire project. But the form is still limited to 10 records—the overriding limit set for the form recordset in the form's Max Records property. You can scroll down the combo box list and find that the list ends at ZIP code 00771—the 100th value in the list.

5. Return to the Advanced category in the Access Options dialog box, and reset Default Max Records to **60000**. Click OK, and then close the form *without* saving it.

CAUTION

Be sure to return to the Access Options dialog box and change the Default Max Records setting back to **60000**. If you don't do this, none of the ZIP code combo boxes in the project will work properly. Also, don't save any changes when you close the sample form.

Working with Server Filters

You should already be familiar with using filters in forms from working through the earlier forms chapters in this book. In addition to regular filters, Access projects also include the ability to apply server filters. Server filters function similarly to regular filters except they are applied before the recordset is retrieved from the server and sent to the form. When you apply regular filters in forms, Access first fetches the recordset from the server and then applies the filter on your client machine to limit the rows actually displayed in the form. Applying filters on the server side is advantageous because it reduces the load on the server and it cuts down on network traffic.

Although server filters are very advantageous, you can use them only when your form has a table or a view as its record source. You can define a server filter for a form that is bound to an in-line function, but Access ignores it. If you try to define a server filter for a form bound to a stored procedure, Access displays a warning message when you try to open the form: "A Server Filter cannot be applied to a stored procedure Record Source. Filter not applied." Tables and views do not support parameters, so you cannot design a form that uses both parameters and server filters. As you'll learn later in this chapter, parameters can be just as effective as server filters when you bind your form to a function or stored procedure.

To see how server filters work, open the Conrad Systems Contacts project file (Contacts.adp), and then open the frmXmplContactsServFilter form. Access displays a result, as shown in Figure 28-9.

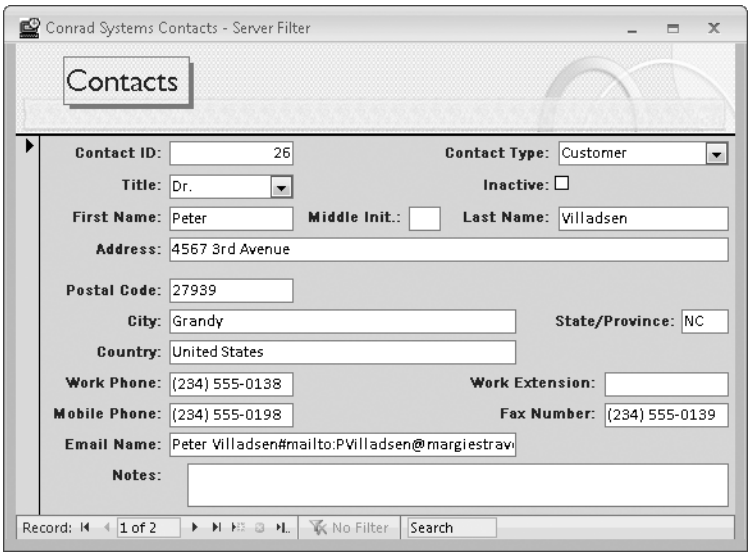

Figure 28-9 The frmXmplContactsServFilter form is based on a view that has a server filter defined.

Notice that the form displays only two of the 32 contacts in the database. If you change to Design view and look at the Server Filter property, you'll find the following:

```
LastName LIKE 'V%'
```

The Server Filter property contains criteria that you might find in a WHERE clause, but without the WHERE keyword. Remember that Access sends this filter to the server, so you must use SQL Server syntax: single quotes surrounding date/time and string literals, and % and _ wildcard characters in a LIKE comparison instead of * and ?. If you need to specify multiple criteria, you can do so by joining them with multiple AND or OR Boolean operators. Here is an example:

```
ContactType = 'Customer' AND ((WorkPostalCode BETWEEN '10000' AND '15999') OR
(WorkStateOrProvince = 'PA'))
```

You can change the Server Filter property to the criteria shown earlier and then change to Form view. Notice that the form now displays eight records instead of two. On the Home tab, in the Sort & Filter group, click the arrow on the Advanced button, and notice also that the Server Filter By Form button is unavailable—because the form's Server Filter By Form property is set to No. If you close the form and save your changes, the form will be permanently filtered using your new criteria, and the user won't be able to change it. Close the frmXmplContactsServFilter form, and do not save the form changes if Access prompts you.

INSIDE OUT

Use Server Filter By Form to Help Construct a Filter

If you are unsure how to write the server filter, try setting the form's Server Filter By Form property to Yes, changing to Form view, and then applying the filter you want using the Filter By Form interface. When you change back to Design view, you'll find the filter you defined entered in the Server Filter property of the form with correct syntax. Set the form's Server Filter By Form property to No and save your changes. Access uses the filter you created automatically every time the form is opened.

If you want to allow users to set the Server Filter property, you can set the form's Server Filter By Form property to Yes. To see how this works, open the frmXmplContactsServFilterByForm form. Rather than display records, Access shows you the Server Filter By Form interface with the currently defined filter, as shown in Figure 28-10.

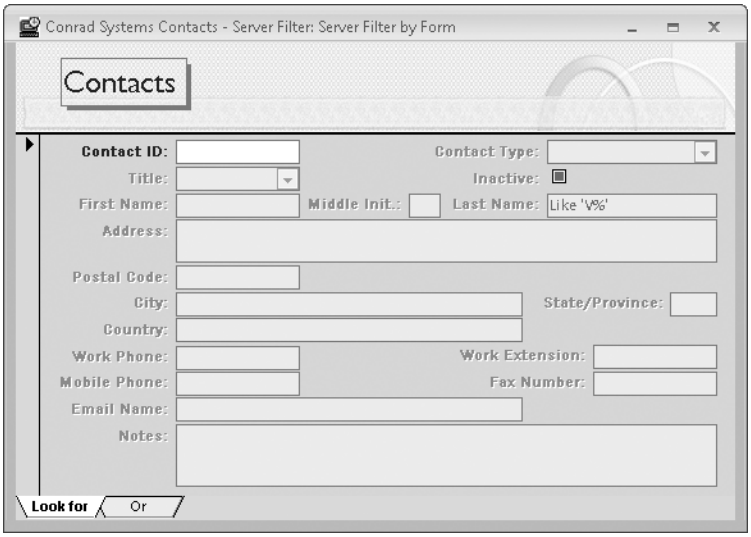

Figure 28-10 This sample form has a predefined server filter that allows Server Filter By Form.

The filter options in the Server Filter By Form window are similar to the options you have when performing a normal filter by form. You can enter filter criteria in each of the boxes and also add multiple criteria for a single column by adding criteria on the successive Or tabs. By default, the drop-down lists of filter values that Access displays for any field offer only the Is Null and Is Not Null options. If you want, you can display a full list of values for a particular text box by changing the Filter Lookup property of the text box to Always in Design view. You can also select the Records At Server option under Filter Lookup Options for <project name> Database in the Current Database category of the Access Options dialog box, as shown earlier in Figure 28-3.

Close the current Server Filter By Form interface by clicking the Close button in the upper-right corner of the form. Access then displays the form filtered, showing the two contacts with a last name starting with the letter *V*, as shown in Figure 28-11. Notice that the Server Filter By Form button is available under Advanced Filter Options in the Sort & Filter group on the Home tab so that you can return to Server Filter By Form mode. To apply a new server filter, click the Server Filter By Form button under Advanced Filter Options. Access opens a new Server Filter By Form interface with the current filter listed in the LastName field. Simply enter any new criteria you want in the appropriate text boxes, click Advanced Filter Options in the Sort & Filter group on the Home tab, and then click the Apply Server Filter button to apply the filter. For more information on filtering by forms, refer to Chapter 10.

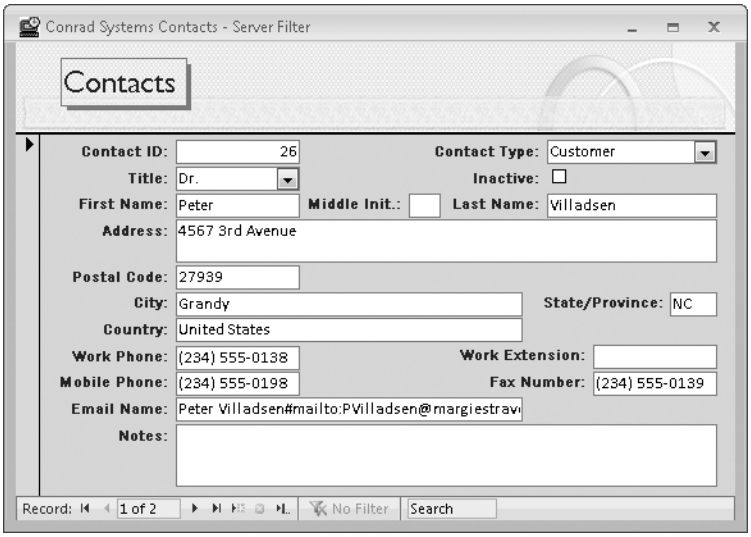

Figure 28-11 When you apply the current server filter on the frmXmplContactsServFilterByForm form, you'll see two records.

Setting Input Parameters

Server filters are great if you need to filter the data from a table or view. However, they don't work if your recordset is an in-line function or a stored procedure. Fortunately, in-line functions and stored procedures can use input parameters to control the data they return.

When you use a parameter query as the record source for a form, you can let Access prompt the user for the parameters when the user opens the form. However, forms in an Access project also have an input parameter that allows you to specify an alternative way to resolve parameters that are required in the in-line function or stored procedure that serves as the form's record source.

In the sample database, you can find a simple stored procedure, spXmplCompany-Parameter, that returns rows from the tblCompanies table and includes a parameter to filter the company name. To demonstrate the various techniques that you can use to resolve the parameter when you use this query in a form, we built three forms. The first, frmXmplCompaniesParmFromSP, doesn't have any special property settings to help resolve the parameter. When you open the form, Access prompts you for the input parameter, as shown in Figure 28-12.

Figure 28-12 When you open a simple form bound to a parameter query, Access prompts you for the parameter values.

Although the parameter name is relatively self-explanatory, it doesn't really give you any clue that you can enter all or part of a company name. For example, if you type the letter *C* and click OK, Access displays a result, as shown in Figure 28-13.

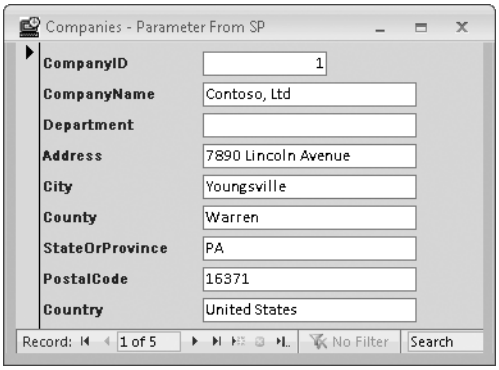

Figure 28-13 The form based on a parameter query now displays all companies that have a name beginning with the letter *C*.

If you don't want the user to see the parameter prompt at all, you can set the Input Parameters property of the form. Open the frmXmplCompaniesParmFromSP form in Design view. In the Property Sheet window, click the Data tab, and locate the Input Parameters property. When a form is bound to a record source that accepts input parameters, you can enter a value for each of the parameters in this property. (Separate the parameter declarations with commas.) To specify a parameter, you need to supply the parameter name, the data type of the parameter, and a value for the parameter. As you have already seen, the name of the parameter for the stored procedure used by this form is @Enter_Company_Name. If you want this form to always display only companies that have names beginning with the letter *C*, enter the following in the Input Parameters property:

```
@Enter_Company_Name nvarchar(50) = 'C'
```

The form's Property Sheet window should now look like Figure 28-14.

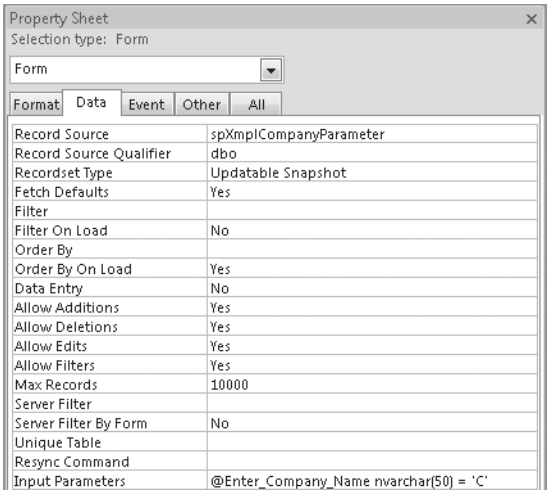

Figure 28-14 You can resolve a parameter by setting a form's Input Parameters property.

Switch to Form view by clicking the arrow under the View button in the Views group on the Ribbon and clicking Form View from the list of available views. Notice that the form automatically displays the rows for companies that have a name beginning with the letter *C*. If you think about it, this is similar to specifying a server filter for a form that uses a table or view as its record source. The form sends a filter to the server, and the user cannot change the filter. When you close the form, Access asks whether you want to save the change you just made. Keep in mind that if you save this change, Access no longer prompts you for the parameter when you open the form.

INSIDE OUT

Use Server Filter or Input Parameters Rather Than Separate Queries with Filters

Instead of creating a form that defines a fixed filter or parameter value, you could certainly create a separate query that includes the filter as part of its WHERE clause and use that query for your form. However, you might want to create your application so that it opens different forms depending on the user's authority. Some users might be allowed to view all rows. For others, you might want to always provide a filtered view of the data. You can design two forms, one that allows the user to view all rows or specify a parameter to view any rows and another that lets a more restricted user view only specific rows. You can use the same query as the record source for both forms. In the form that you open for the restricted user, create a static server filter or provide a fixed input parameter value that the user cannot change. By using the same query for both forms, you simplify the number of queries that you must define to support your application.

What if you wanted to allow the user to specify a different company every time the form opens? You could certainly leave the Input Parameters property blank and let Access prompt the user for the parameter. However, if the variable name is too cryptic or you want something that would be easier for your users to understand, you can enter a more descriptive term in the Input Parameters property as follows:

```
@Enter_Company_Name nvarchar(50) = [Enter all or part of a company name:]
```

Wait a minute! What's going on here? What the previous setting accomplishes is a substitution of the parameter name in the query with a parameter value that provides more information. We created a sample form in the database that includes this parameter substitution in the Input Parameters property, frmXmplCompaniesCustomParameter. When the user opens this form, Access prompts for the substituted parameter name, not the original name, as shown in Figure 28-15. Now your users might get a better idea of what information is being requested.

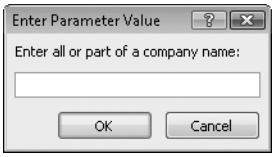

Figure 28-15 By using the Input Parameters property, you can display a more descriptive request for a parameter value.

But what if they don't know the name of the company? What if a company name begins with an article, such as The Microsoft Corporation, and the user gets frustrated when entering the letter *M* produces an incorrect result? One solution is to create a separate form with a combo box or list box that provides a list containing all the company names. Then the user could select a value from the list, and you can specify an Input Parameters setting that resolves the query parameter by referencing the value that the user selects.

In the sample database, you can find a form, fdlgXmplCompanyParm, that provides a list of company names in a combo box called cmbCompany. You can open the form and select a company name from the list, as shown in Figure 28-16.

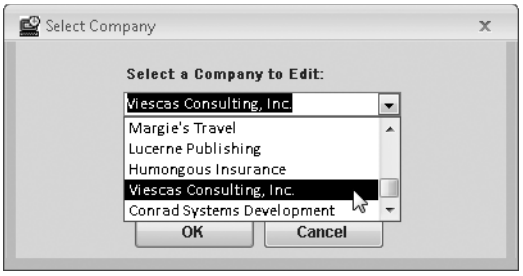

Figure 28-16 You can choose a value from a combo box on a form that resolves a stored procedure parameter.

Click OK, and a form opens to display the company you selected, as shown in Figure 28-17. This form, frmXmplCompanesParmFromForm, uses the same parameter stored procedure as its record source. But notice that Access didn't prompt for the parameter.

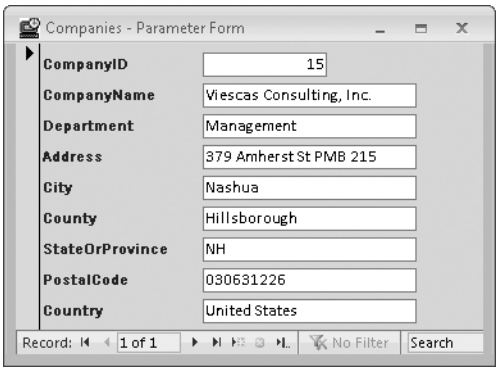

Figure 28-17 This form uses a parameter query, and the parameter is resolved from a value chosen from a list in another form.

The secret is the form's Input Parameters property has a special setting, as follows:

```
@Enter_Company_Name nvarchar(50) = [Forms]![fdlgXmplCompanyParm]![cmbCompany]
```

The interesting value assigned to the parameter is a reference to the combo box on the fdlgXmplCompanyParm form. This special syntax asks Access to look at all the open forms ([Forms]!), find the form named fdlgXmplCompanyParm, and then find the control on the form named cmbCompany. In truth, the fdlgXmplCompanyParm form has a little bit of Visual Basic code behind the two command buttons to make the example run smoothly. But the critical lesson is the special syntax that you can use in input parameters to redirect resolution of the parameter value to the value in a control on another form. For more details about Visual Basic and the rules for referencing objects in Access, see Chapter 19, "Understanding Visual Basic Fundamentals."

You should now have a good understanding of the differences between forms in an Access project file (.adp) and those in a desktop database (.accdb). In the next chapter, you'll learn about the differences for reports.