CHAPTER 29

Building Reports in an Access Project

Understanding Report Differences in an Access Project 1567
Working with Server Filters 1570
Working with Input Parameters 1572

As you learned in the previous chapters about Access projects, 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). You need to consider only a few differences when working with forms—and even fewer when working with reports.

As with forms, reports in an Access project are nearly identical to reports in an Access desktop database. You can get started building most reports without knowing any details about the differences. In the previous chapter, you learned about how different types of queries used as record sources can affect the way you build your forms. The same is true for reports. Because you are working with data from Microsoft SQL Server, it is important to understand the key differences of all the query types and how to maximize their efficiency when using them as record sources in your reports.

Note

This chapter assumes that you have already read the earlier chapters on report design (Chapter 14, "Using Reports"; Chapter 15, "Constructing a Report"; and Chapter 16, "Advanced Report Design") and have a good grasp of the basic report design concepts. You should be familiar with the report designer and know how to create reports that allow you to view grouped and aggregate data. 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 Report Differences in an Access Project

To understand the report differences in an Access project, open the Conrad Systems Contacts project file (Contacts.adp), and then open the rptProductsPlain report in Print Preview. Figure 29-1 shows the report in Print Preview. (This is one of the few reports in the sample project file that does not have a custom Ribbon.)

C29623252.indd 1567 2/23/2007 9:48:28 PM

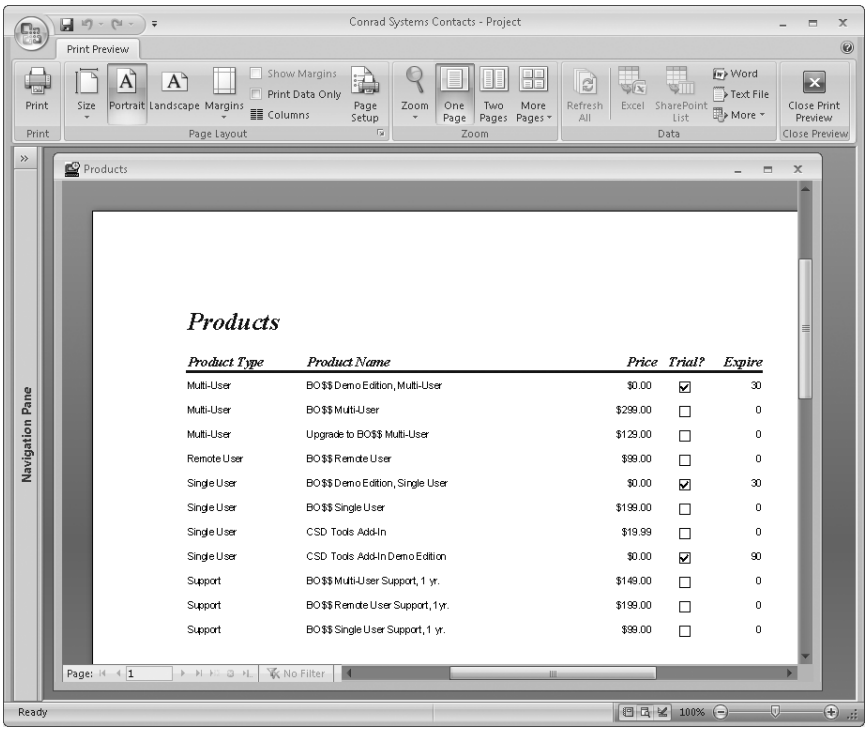

Figure 29-1 Here is a sample report in a project file displayed in Print Preview.

Notice that the navigation bar at the bottom of the report includes the Stop Refresh and Set The Maximum Record Count buttons that you can see on forms in a project, but they are disabled. When you're viewing reports in a project file, these buttons are always disabled. Unlike forms, Access project reports always fetch all records before displaying any data. Keep in mind that a report represents a complete summary of information that is meant to be displayed or printed, not updated. A report might include grouping or totals that require the entire recordset in order to be calculated correctly. If a report returned only a partial recordset, totals and averages included in the report would be inaccurate.

Because reports always return all the rows in the record source, it is important to limit the record source to only the rows that the user needs to see. If a large report returns mostly unnecessary information, a lot of time and server resources will be wasted. Instead, try to create recordsets that return only the data that the user needs to view. For example, you might want to print a report about companies that have a training department or the invoices for a specific company and date. 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) in a form before opening the report. Using methods like these makes the database easier to use and also lessens the strain you place on the server.

Setting Project Report Properties

Let's take a look at some of the new properties you can set for a report in an Access project. Click the arrow in the Views group on the Home tab, and click Design View from the list of available views to switch the rptProductsPlain report to Design view. (Alternatively, if you didn't open the report previously, right-click the report name in the Navigation Pane, and then click 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 report, as shown in Figure 29-2.

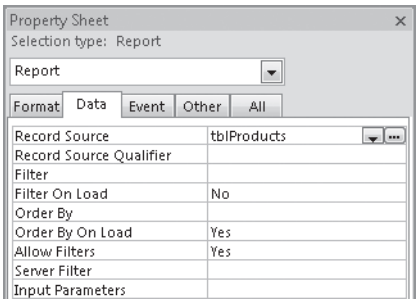

Figure 29-2 A report in an Access project file has a few more data properties than a desktop database does.

> **Note**
>
> You cannot open reports in Layout view when using Access project (.adp) files. In Microsoft Office Access 2007, you can make design-level changes to reports in project files only by using Design view.

Record Source Qualifier, Server Filter, and Input Parameters are properties that you'll find in reports in an Access project (.adp) but not in reports in a desktop database (.accdb).

If you have read Chapter 28, "Designing Forms in an Access Project," these properties should already be familiar to you. Table 29-1 shows all the properties that are different for a report in an Access project. You can refer to this table for a brief definition of how you can use each property.

Table 29-1 Additional Report Properties in an Access Project

Property	Tab	Description
Record Source Qualifier	Data	Identifies the SQL Server owner of the record source for this report. Unless a different owner has been specified by an administrator on SQL Server, this value is normally *dbo*. (short for database owner).
Server Filter	Data	Specifies a filter that Access sends to the server to limit the rows from the server when the user opens the report.
Input Parameters	Data	Allows you to specify the values of any input parameters that are used in the SQL record source of the report.

Working with Server Filters

As you learned in Chapter 28, when you specify a server filter, Office Access 2007 includes the filter in the request it sends to the server. The server selects the rows that meet the criteria in the server filter and sends only the requested data back to Access 2007. Server filters are advantageous because they reduce the size of the recordset, fetching only the data the user needs to see. You can also set the Filter property on a report, but Access doesn't apply that filter until it has fetched all the unfiltered data from the server. Using a server filter cuts down on the time a server must spend processing the recordset and eliminates the time that Access would otherwise spend applying the filter on your client computer. You can imagine that using a server filter can have a significant impact on the efficiency of your reports, particularly when the underlying tables needed for a report contain hundreds of thousands of rows.

Although server filters are very advantageous, you can use them only when your report has a table or a view as its record source. You can define a server filter for a report that is bound to an in-line function, but Access ignores it. If you try to define a server filter for a report bound to a stored procedure, Access displays a warning message when you try to open the report: "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 report 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 report to a function or stored procedure.

To see how server filters work, open the rptXmplCompanyContactsServerFilter report in Print Preview. Access displays a result, as shown in Figure 29-3.

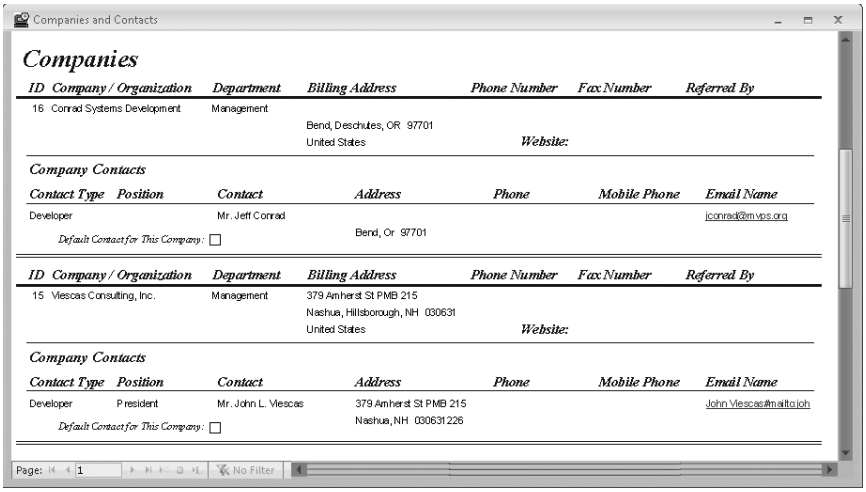

Figure 29-3 This sample report is based on a view that has a server filter defined.

Notice that the report displays only two of the 16 companies in the database. If you switch to Design view and look at the Server Filter property, you'll find the following:

```
Department = 'Management'
```

This example is similar to the server filter example shown in Chapter 28 and illustrates how you can use the server filter effectively in both forms and reports. 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 of a single criterion:

```
StateOrProvince LIKE 'N%'
```

You can change the Server Filter property to this criterion and then change to the Report window. Notice that the report now displays four companies instead of two. You can also specify criteria as complex as necessary to get the job done. This report uses a subreport to display the contact information for each company. If you want to filter on the contact information, you must use a subquery. (See Article 2, "Understanding SQL," on the companion CD for details about using subqueries to filter data.) For example, if you want to list all companies that have a contact who is contact type Customer in the state of Pennsylvania, you could specify the following filter:

```
CompanyID IN (SELECT CompanyID FROM tblCompanyContacts
INNER JOIN tblContacts
ON tblCompanyContacts.ContactID = tblContacts.ContactID
WHERE tblContacts.ContactType = 'Customer'
AND tblContacts.WorkStateOrProvince = 'PA')
```

Using the preceding filter returns the two companies in Pennsylvania that have one or more contacts categorized as customers.

INSIDE OUT

Reduce Network Traffic by Using Server Filters

Access provides the option to specify both server filters and regular filters in your reports. Remember that server filters are always more efficient because the server applies the filter before returning the data. Unless you have a very compelling reason to filter the data after it is returned to the report, always use server filters instead.

Working with 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 results from 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 report, you can let Access prompt the user for the parameters when the user opens the report. However, reports 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 report's record source.

INSIDE OUT

Choosing the Type of Query Used as the Record Source of a Report

When considering what type of query to use as the record source of a report, the question really boils down to using server filters versus using input parameters. You can use either method to return the data you want to display, but each method has its own advantages.

Server filters are easy to define and are best used when you know you are going to be filtering the data the same way every time you view the report. Examples include filtering products by a particular category, displaying a list of active employees, or showing the current sales tax rate for the 50 states. Use a view or a table whenever you use server filters.

Input parameters are versatile and are best used when you need to allow your users to specify criteria to limit the recordset of the report. Examples include showing all the employees for a selected company, displaying a list of all sales within a specified date range, or looking up the detailed information for a particular contact. Use an in-line function or a stored procedure whenever you use input parameters.

If you don't intend to use either server filters or input parameters, use a view as your record source because it is simple and efficient.

In the sample database, you can find a simple stored procedure, spXmplInvoiceParm, that returns rows from the tblCompanies and tblInvoices tables and includes parameters to filter the company name and the date of the invoice. In Chapter 28, you learned the various techniques that you can use to resolve the parameter when you use this query in a form using three sample forms. In the sample Contacts.adp project file, you can find three sample reports that illustrate these same techniques when you're working with reports in an Access project. The first report, rptXmplInvoicesParmFromSP, doesn't have any special property settings to help resolve the parameters. When you open the report, Access prompts you for the input parameters, one at a time, as shown in Figure 29-4.

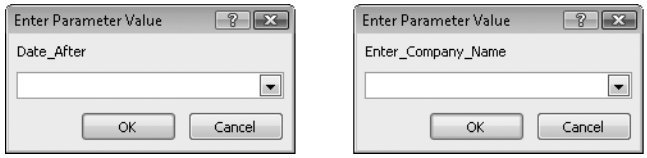

Figure 29-4 When you open a simple report bound to a parameter query, Access prompts you for the parameter values.

Although the parameter names are relatively self-explanatory, the Date_After parameter doesn't tell you that you'll see invoices produced on or after the date you enter. Also, the Enter_Company_Name parameter doesn't give you any clue that you can enter all or part of a company name. For example, if you type 1/15/2007 for the date and the letter *C* for the company name and click OK, Access displays a result as shown in Figure 29-5. (You should see 18 invoices.)

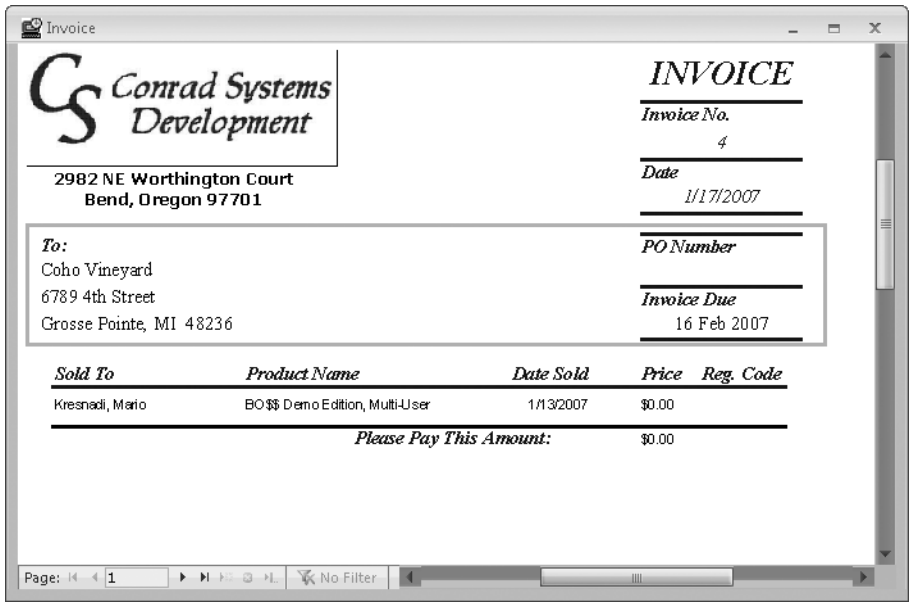

Figure 29-5 These are the results you'll see with the sample report based on a parameter query displaying all invoices created on or after January 15, 2007, for companies that have a name beginning with the letter *C*.

If you don't want the user to see the parameter prompt at all, you can set the Input Parameters property of the report. Open the rptXmplInvoicesParmFromSP report in Design view. In the Property Sheet window, click the Data tab, and locate the Input Parameters property. When a report 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 report is @Enter_Company_Name. If you want this report to always display invoices created within the last 90 days and only for companies that have names beginning with the letter *C*, enter the following in the Input Parameters property:

```
@Enter_Company_Name nvarchar(50) = 'C',@Date_After datetime = (Date()-90)
```

Notice that you're taking advantage of the Access built-in Date function to fetch the current date and then subtract 90 days to provide one of the parameter values. The report should now look like Figure 29-6.

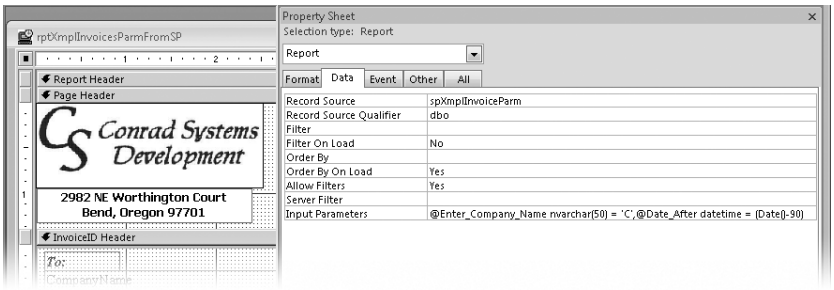

Figure 29-6 You can resolve a parameter by setting a report's Input Parameters property.

The sample database contains records only up through July 7, 2007, so to get any results, you might have to temporarily set your system clock back. Set your system date to June 15, 2007, and then switch to Print Preview by clicking the arrow in the Views group on the Home tab and clicking Print Preview from the list of available views. Access shouldn't prompt you for any parameters, and you should see 12 invoices.

Notice that the report 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 report that uses a table or view as its record source. The report sends a filter to the server, and the user cannot change the filter. When you close the report, 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 for the parameters when you open the report.

> **Note**
>
> You might notice that the examples and the steps for applying server filters and using input parameters are very similar in both forms and reports. Keep in mind that both server filters and input parameters are properties that modify the result returned by the record source. After you understand the principles behind server filters and input parameters, you can use them effectively in both your forms and your reports.

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

```
@Date_After datetime = [Invoices on or after date:],
@Enter_Company_Name nvarchar(50) = [Enter all or part of a company name:]
```

As you learned in the previous chapter, the previous settings accomplish a substitution of each parameter name in the query with a parameter value that provides more information. We created a sample report in the database that includes this parameter substitution in the Input Parameters property, rptXmplInvoicesCustomParameter. When the user opens this report, Access prompts for the substituted parameter names, not the original names, as shown in Figure 29-7. Now your users might have a better idea of what information is being requested.

Figure 29-7 You can use more descriptive requests for parameter values.

If you read the previous chapter and followed the examples on input parameters, everything discussed so far for reports should seem familiar. You can use input parameters to control the data the server returns. If the parameter name is too cryptic, you can substitute a more descriptive request for information. However, you can still run into problems if your users don't know what company they are looking for or can't remember how to spell the company name.

Chapter 28 proposed using a form with a combo box to select the name of the company you want to view. This solution also works great for reports. You can build a form similar to the one shown in Chapter 28, except now you're using it to open a report instead of a form, and the form needs to prompt for two parameter values. This method is especially useful if you want to limit the data in the report to a specific date range by using a form to supply the beginning and end date parameters to your report. Now let's take a look at an example of a form that can be used to select a starting date and a company name to view in a report.

In the sample database, you can find a form, fdlgXmplInvoiceParm, that provides a box to enter a date (called txtDate) and a list of company names in a combo box (called cmbCompany). You can open the form, enter a date, and select a company name from the list, as shown in Figure 29-8.

Figure 29-8 You can enter a date and choose a value from a combo box on a form that resolves stored procedure parameters.

Click OK, and a report opens to display the invoices after the date you entered for the company you selected, as shown in Figure 29-9. This report, rptXmplInvoicesParmFromForm, uses the same parameter stored procedure (spXmplInvoiceParm) as its record source. However, now Access doesn't need to prompt the user for the report parameters.

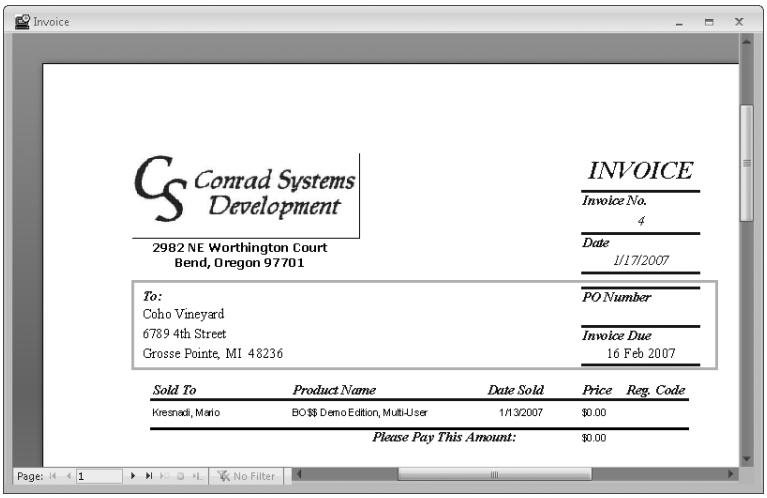

Figure 29-9 This sample report uses a parameter query, and the parameter is resolved from a value chosen from a list in another form.

As you learned in the previous chapter, the secret is that the report's Input Parameters property has a special setting as follows:

```
@Date_After datetime = [Forms]![fdlgXmplInvoiceParm]![txtDate],
@Enter_Company_Name nvarchar(50) = [Forms]![fdlgXmplInvoiceParm]![cmbCompany]
```

The interesting values assigned to the parameters are references to the two controls on the fdlgXmplInvoiceParm form. This special syntax *[Forms!]* asks Access to look at all the open forms, find the form named fdlgXmplInvoiceParm, and then find the control on the form named txtDate and the control named cmbCompany. The report doesn't open on its own, however. Special code attached to the buttons on the fdlgXmplInvoiceParm form opens the report when the user clicks the OK button. For now, you need to understand only that once the report is opened, the input parameter of the report knows to look at the fdlgXmplInvoiceParm form for the needed input parameter information. For more details about Visual Basic, see Chapter 19, "Understanding Visual Basic Fundamentals."

INSIDE OUT

Make Sure That Any Subreports or Subforms That You Use in a Report Are Also Efficient

As you begin to work with more advanced report designs in your Access project, you might want to start using embedded subforms and subreports to display more complex information. Remember that you need to keep server efficiency in mind when designing your subforms and subreports as well. Use parameters and server filters where appropriate to restrict the data to only what the user needs to see. If you are using a subform in your report (for example, to display a PivotChart), be sure to set the Recordset Type property to Snapshot (instead of Updatable Snapshot) because this requires fewer server resources.

The properties for reports that are unique to an Access project have a lot in common with those for forms. The key point to keep in mind is that forms and reports are different in an Access project only because of the way they need to interact with SQL Server. The major design elements are the same, so once you understand how forms and reports interact with SQL Server, you can design your forms and reports using the same techniques you used when building these objects in Access desktop databases.

This concludes all the material on designing objects in Access projects. You should now have a good understanding of how an Access project is different from an Access desktop database, and you should also understand the power and versatility that SQL Server can offer to your application design. If you are thinking about converting an existing Access desktop database to an Access project, Access provides an upsizing wizard to help with this task.

C29623252.indd 1578 2/23/2007 9:48:34 PM