Add Oracle 10g XE Database Into Visual Basic.NET

In this section we discuss how to add an Oracle 10g XE database into Visual Basic.NET applications using the Design Tools and Wizards - Data Source windows.

Before we can perform this database addition, a sample oracle database CSE_DEPT should have been created with a user account named CSE_DEPT. Refer to section 2.11 in Chapter 2 and Appendix D to get more detailed information in how to create a new user and the associated customer database.

In the following we use a sample project OracleInsertWizard to illustrate how to add an oracle 10g XE customer database into this Visual Basic.NET application using the Data Source tools.

Open the project OracleInsertWizard and the Data Source window, and then click the Add New Data Source link to open the Data Source Configuration Wizard. Keep the default selection Database unchanged and click the Next button to go to the next window.

On the next window, click the **New Connection** button since we need to create a new connection to our sample database CSE_DEPT. On the opened Add Connection dialog, which is shown in Figure E-1, click the **Change** button that is located at the right of the Data source textbox to open the Data Source list. Select the Oracle Database item from the list and click the OK button. Enter our Oracle server name 'XE' into the Server name box, and enter 'CSE_DEPT' and 'reback' into the Username and Password boxes, respectively. Your finished Add Connection dialog box should match one that is shown in Figure E-1.

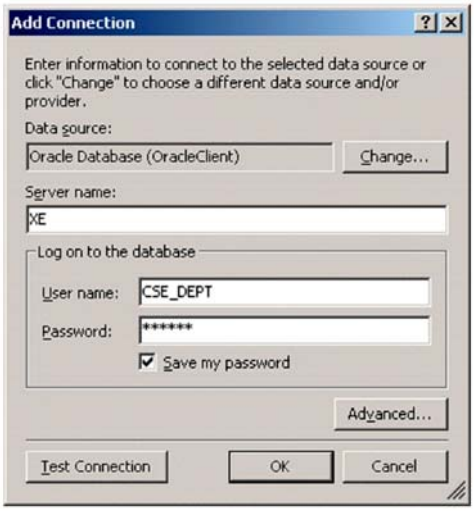

Figure E-1. The Add Connection dialog box

You can click the Save my password checkbox if you like, and then click the **Test Connection** button to test this connection. A Connection Success message box will be displayed if this connection is fine.

Then click the OK button to go to the next window, which is shown in Figure E-2.

Keep the default data source XE.CSE_DEPT unchanged and click Yes radio button to include the whole connection information in the connection string. You can also expand

the Connection String by clicking the plus sign before it to see the content of this connection string, as shown in Figure E-2.

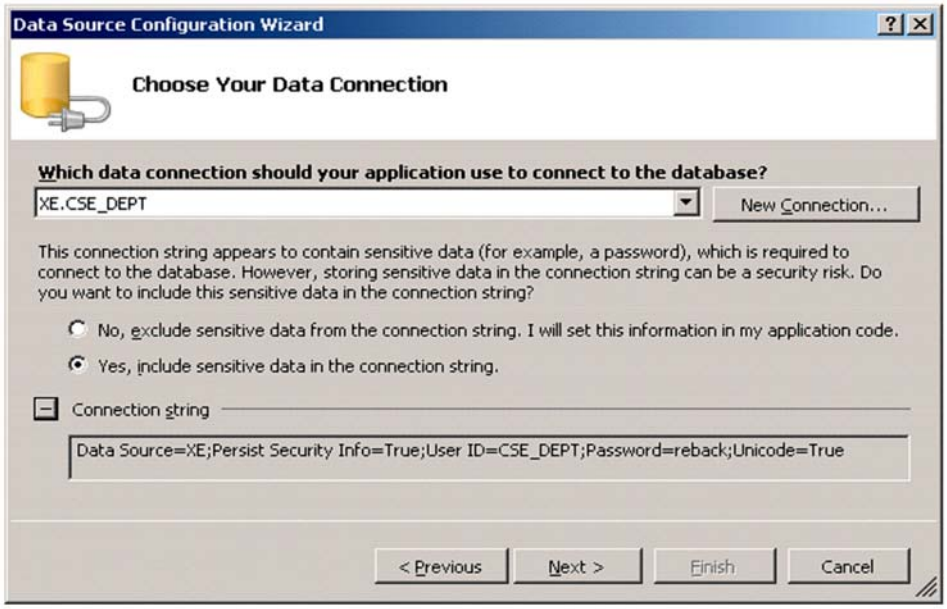

Figure E-2. Data connection dialog

Click the Next button to go to the next window.

In the next window, you are prompted to save this connection string into your configure file in your application. It is a good habit to integrate all staff together to form a complete project. So click Yes to it and modify the name of the connection string to **OracleXEConnString**, which is shown in Figure E-3.

Figure E-3. Modify the name of the connection string

Click the Next button to go to the next window.

In the opened next window, first change the DataSet name to 'CSE_DEPTDataSet' and then expand the Tables object to list all tables available to our application, which is shown in Figure E-4.

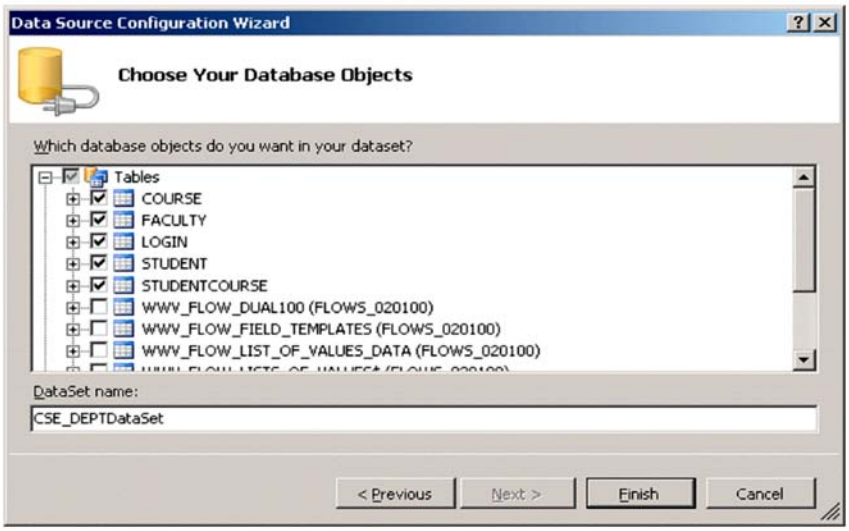

Figure E-4. Select desired data Tables and Views

We only have interests in those five tables we created and they are related to our application. Select those five tables by checking them one by one, which is shown in Figure E-4. Your finished Table Selection window should match one that is shown in Figure E-4. Click the Finish button to complete this database addition process.

Immediately you can find that five selected tables have been added into the Data Source window. Right click on any place inside the Data Source window and select Edit DataSet with Designer item from the popup menu to open the DataSet Designer window, which is shown in Figure E-5. You can find all five tables and relationships between them.

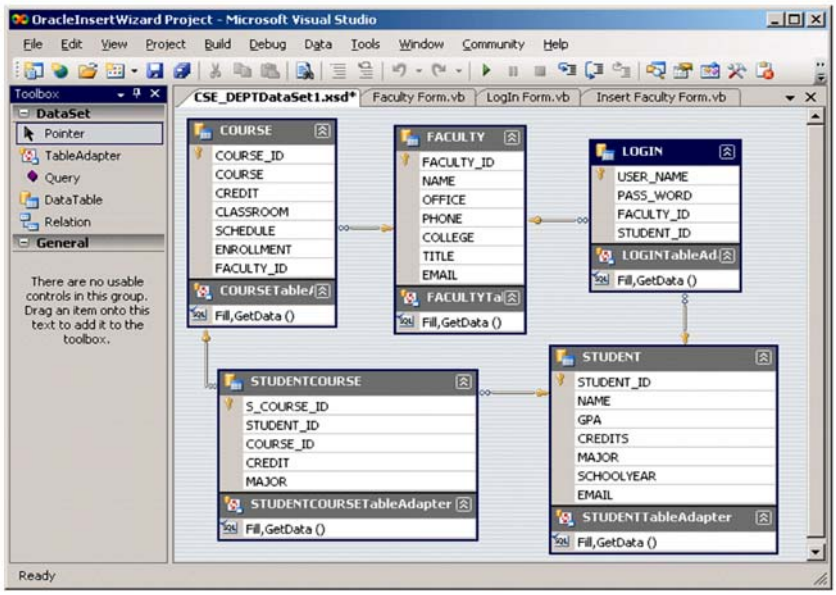

Figure E-5. The added five tables

In the following we discuss how to use the Design Tools and Wizards to develop data-driven application with the help of this added database CSE_DEPT. We divide our discussion into four sections:

1. Develop DataSet coding for the LogIn form
2. Develop DataSet coding for the Faculty form
3. Develop DataSet coding for the Course form
4. Develop DataSet coding for the Insert Faculty form
5. Develop DataSet coding for the Student form

Let's start from the LogIn form.

E.1 Develop DataSet coding for the LogIn form

The coding for this form is simple and the only job needed to do is to check the username and password entered by the users and compare them with those items in our sample database. If a match found, the login process is successful and the Selection form will be displayed to allow user to continue to perform other functionalities.

To perform this comparison, either the default method Fill() should be modified or a new method will be added into the TableAdapter. We prefer to use the second way to perform this comparison – adding a new method.

Open the Data Source window by going to Data|Show Data Sources menu item, and then open the DataSet Designer by right clicking on any place inside the Data Source window and select Edit DataSet with Designer item. Right click the last item from the LogIn table and select the item Add Query to open the TableAdapter Query Configuration Wizard. Keep the default selection Use SQL statements unchanged, and click the Next button to go to the next window. Still keep the default item SELECT which returns rows unchanged and click the Next button to go to the next window.

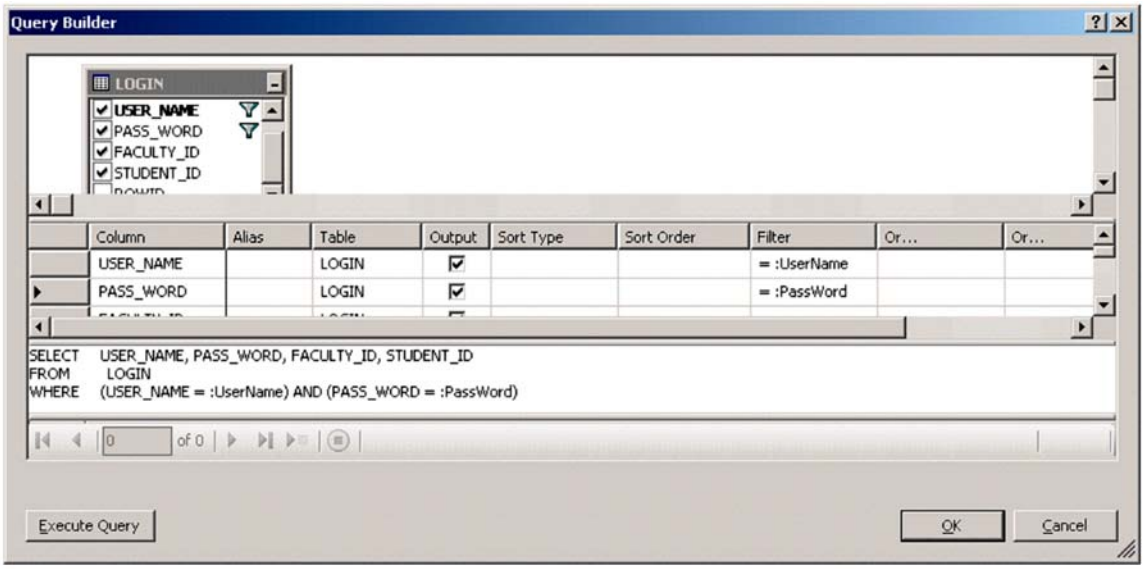

Figure E-6. The Query Builder window

Click the Query Builder button to open the Query Builder window to build our desired query method, which is shown in Figure E-6.

Type a question mark on Filter column for both USER_NAME and PASS_WORD rows in the mid-pane. Also modify the default parameters names to UserName and PassWord, respectively. Your finished Query Builder window should match one that is shown in Figure E-6.

Click the OK and then the Next buttons to go to the next window.

Change the method name to FillByUserNamePassWord in the next window, as shown in Figure E-7, and then click the Next and then Finish buttons to complete this query method building process.

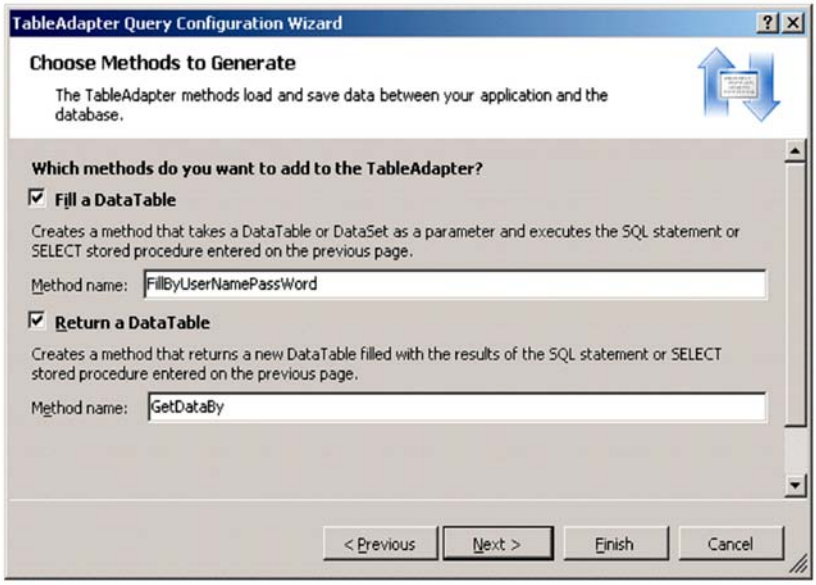

Figure E-7. Modify the default Fill method

The VB coding for the LogIn button click event procedure is straightforward and it is shown in Figure E-8.

cmdLogIn ▼ Click ▼

```
Private Sub cmdLogIn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdLogIn.Click
A       Dim LogInTableApt As New CSE_DEPTDataSetTableAdapters.LogInTableAdapter
B       Dim selForm As New SelectionForm
C       LogInTableApt.ClearBeforeFill = True
D       LogInTableApt.FillByUserNamePassWord(CSE_DEPTDataSet.LogIn, txtUserName.Text, txtPassWord.Text)
        If CSE_DEPTDataSet.LogIn.Count = 0 Then
            MessageBox.Show("No matched username/password found!")
            txtUserName.Focus()
E           Exit Sub
        End If
        selForm.Show()
        Me.Hide()
End Sub
```

Figure E-8. The coding for the LogIn button click event procedure

First a TableAdapter object LogInTableApt is created since we need to use its method FillByUserNamePassWord() we built in the Query Builder to perform the data query from the LogIn table to compare the username and password entered by the user with those items in our database. The Selection form will be display if a match is found from our database. As for DataBinding processes of this LogIn form, refer to section 4.7 in Chapter 4 to get more detailed information for this issue.

E.2 Develop DataSet coding for the Faculty form

The functionality of the Faculty form is to pick up the desired faculty information based on the input faculty name entered by the user. We need to build another query method to perform this query functionality.

Open the DataSet Designer and right click on the last item from the Faculty table on the opened Designer Wizard, and select Add Query item from the popup menu to open the TableAdapter Query Configuration Wizard. Keep the default selections for the first two windows and click the Next button to go to the next window. Click the Query Builder button to open the Query Builder dialog to build our query method, which is shown in Figure E-9.

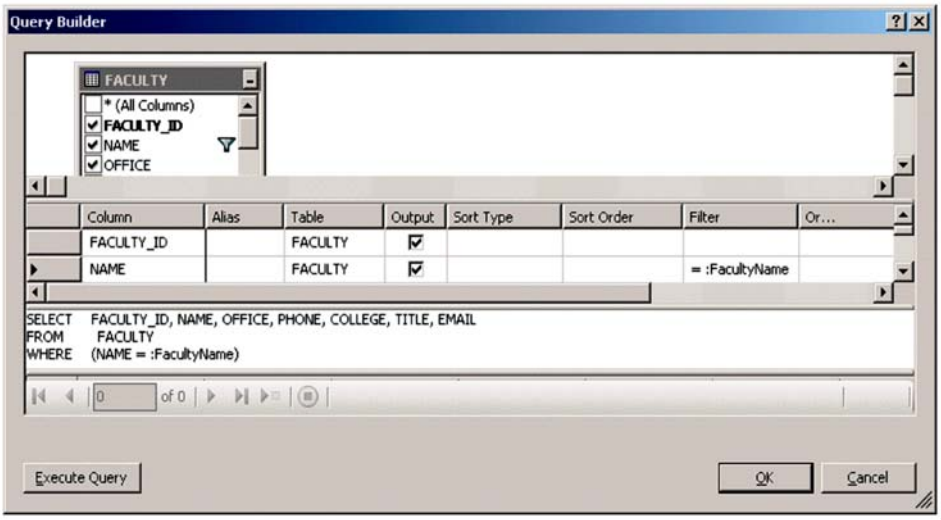

Figure E-9. The Query Builder

Type a question mark in the Filter column along the NAME row in the mid-pane and press the Enter key from your keyboard to create a dynamic parameter. Change this parameter's name to FacultyName (don't touch the symbol =:) and then click the OK and the Next buttons to go to the next window.

Change the default name of the FillBy method to FillByFacultyName, as shown in Figure E-10. Click the Next and Finish button to complete this query method building process.

The associated coding is developed in the Select button event procedure on the Faculty form window. The functionality of this piece of codes is: as the project runs, as the user selected the desired faculty member from the Faculty Name combo box, and then clicks the Select button. The query method FillByFacultyName will be executed to pick

up all matched records from the Faculty table in our sample database CSE_DEPT. The returned matched faculty information is displayed in five labels in the Faculty form window.

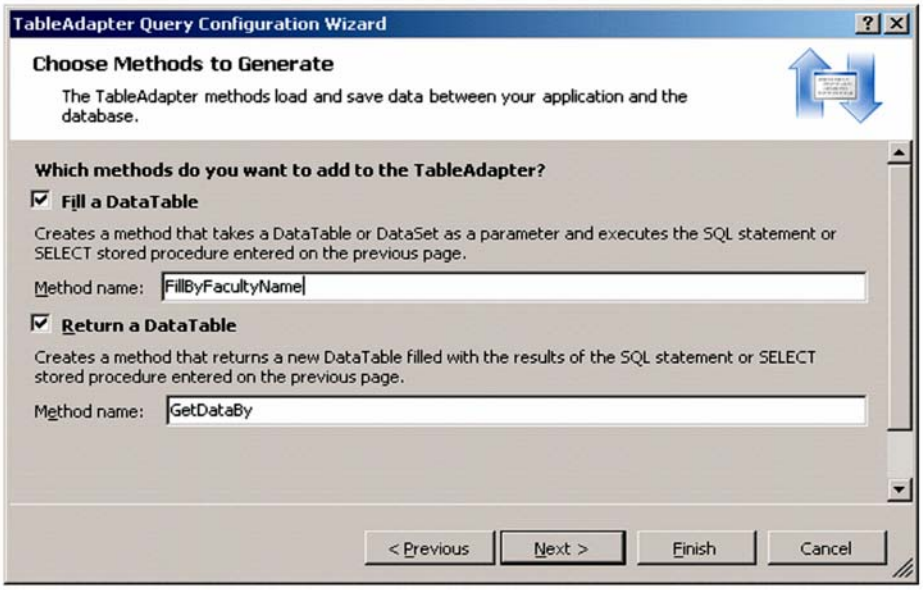

Figure E-10. The Query Builder

Open this Select button click event procedure and enter the following codes that are shown in Figure E-11.

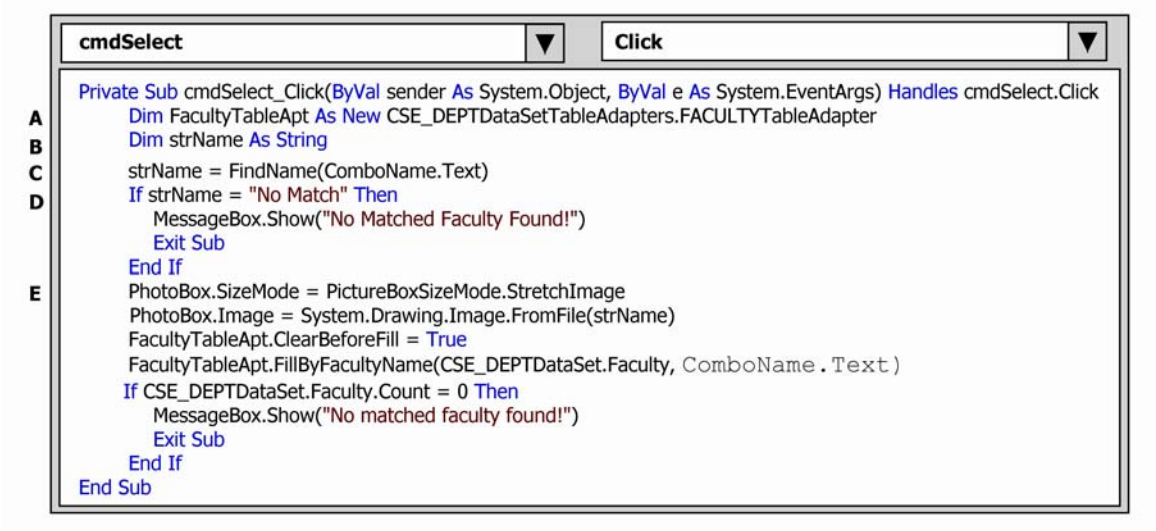

Figure E-11. The coding for the Select button click event procedure

This coding is identical with that we did for the SQL server database programming in section 4.12 in Chapter 4. refer to that part to get more detailed information for this coding. As for DataBinding processes of this Faculty form, refer to section 4.11 in Chapter 4 to get more detailed information for this issue.

E.3 Develop DataSet coding for the Course form

The functionality of the Course form is to retrieve back all matched courses taught by the selected faculty based on the input faculty name selected by the user. We need to build two query methods to perform this functionality: the first query is used to pick up the faculty_id that is related to the selected faculty member, and the second query is to pick up all matched courses taught by the selected faculty based on the faculty_id we obtained from the first query.

The reason we used two queries is that there is no Faculty Name column available in the Course table, and the only available column in the course table is the faculty_id. Therefore we need first get the matched faculty_id from the Faculty table based on the input faculty name, and then we can retrieve back all matched courses taught by the selected faculty based on the faculty_id.

To build the first query, we need to use the Faculty table. Open the DataSet Designer and right click on the last item from the Faculty table, and select Add Query item to open the TableAdapter Query Configuration Wizard. Keep the default setting for the first window, and select the item SELECT which returns a single value for the second window and click the Next button to go to the next window. Because we only need to get a single data faculty_id from the Faculty table based on the input faculty name, therefore we select this setup (refer to Figure E-12).

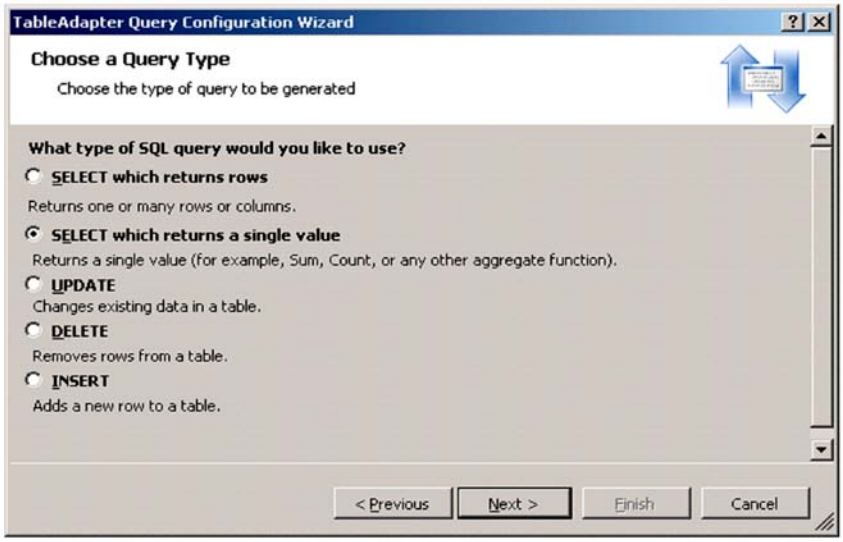

Figure E-12. SELECT which returns a single value dialog

Click the Query Builder button to open the Query Builder dialog. Highlight all content from the mid-pane and delete it. Right click on the top pane and select Add Table to add our Faculty table into this pane. Click the faculty_id and NAME from the Faculty table in the top pane, uncheck the checkbox in the Output column for the NAME row. Type a question mark in the Filter column along the NAME row and press the Enter key from your keyboard to create a dynamic parameter. Change this parameter's name to FacultyName (don't touch the symbol =:). Your finished Query Builder should match one that is shown in Figure E-13.

Click the OK and then Next buttons to go to the next window.

Change the default query method's name to FindFacultyIDByName, as shown in Figure E-14.

Click the Next and then the Finish buttons to complete this query method building process.

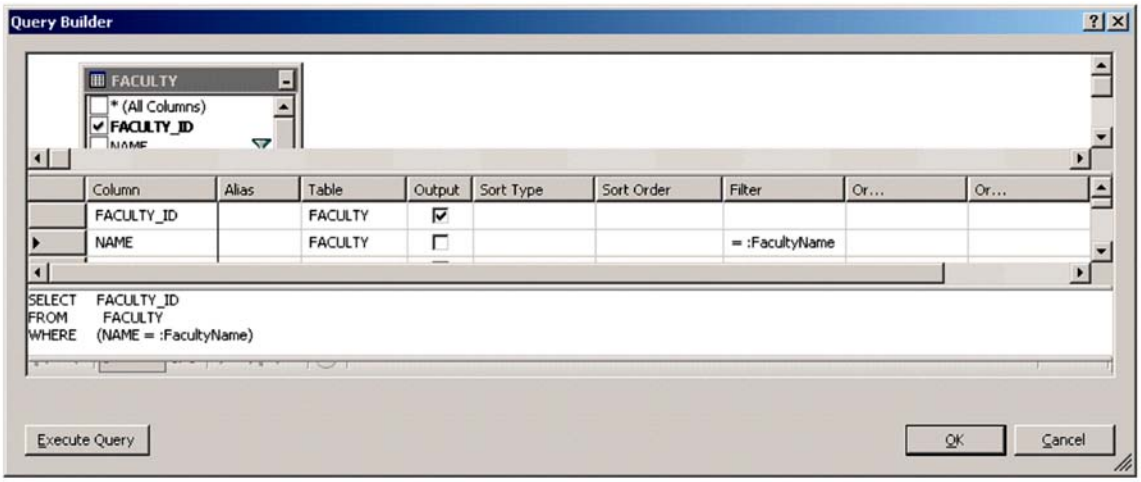

Figure E-13. The Query Builder

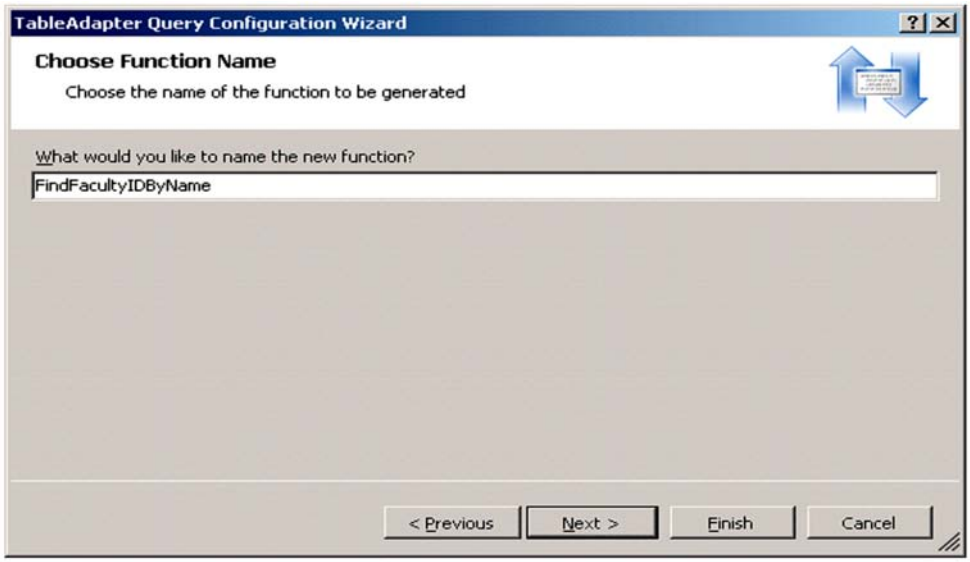

Figure E-14. Change the default query method's name

To build the second query, we need to use the Course table. Open the DataSet Designer and right click on the lat item from the Course table on the opened Designer Wizard. Select Add Query item from the popup menu to open the TableAdapter Query Configuration Wizard. Keep the default selections for the first two windows and click the Next button to go to the next window. Click the Query Builder button to open the Query Builder dialog to build our query method, which is shown in Figure E-15.

Type a question mark in the Filter column along the FACULTY_ID row in the mid-

pane and press the Enter key from your keyboard to create a dynamic parameter. Change this parameter's name to FacultyID (don't touch the symbol =:) and then click the OK and the Next buttons to go to the next window.

Change the default name of the FillBy method to FillByFacultyID, as shown in Figure E-16. Click the Next and Finish button to complete this query method building process.

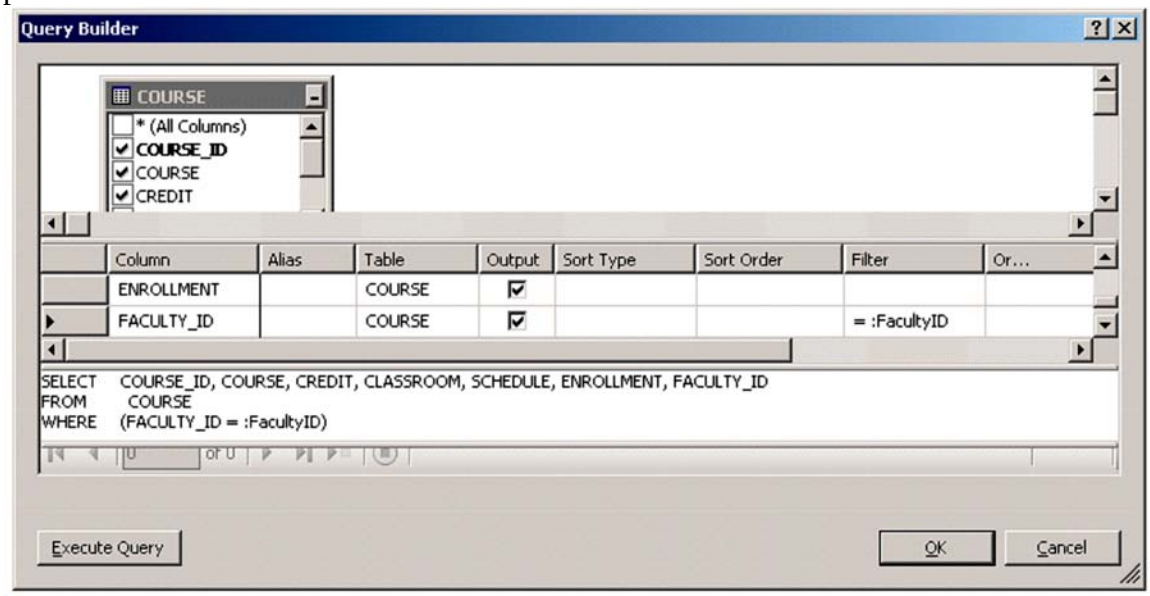

Figure E-15. The Query Builder

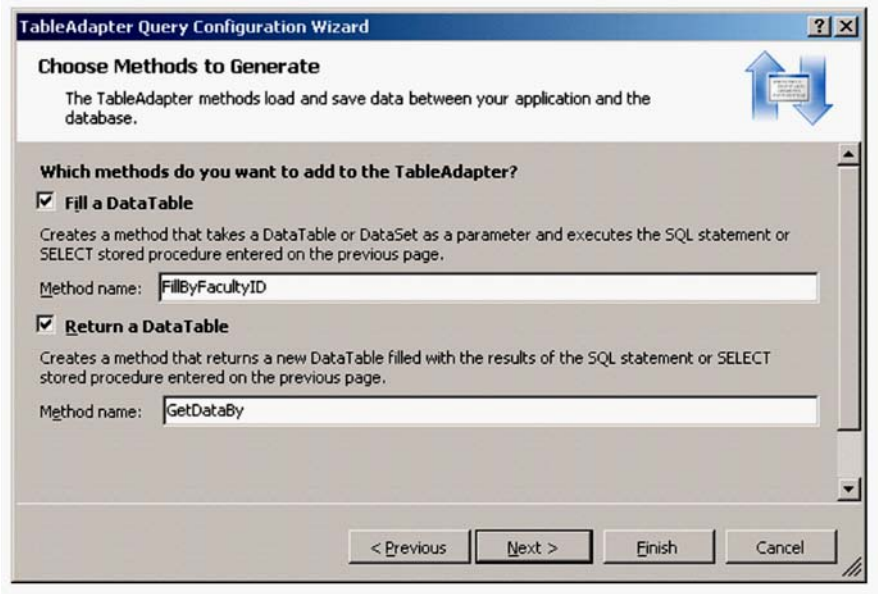

Figure E-16. Change the Query Method's name

The Visual Basic.NET coding is identical with that we did for the SQL server database programming in section 4.15 in Chapter 4. Refer to that part to get more detailed information for this coding. For your convenience, we list that coding in this section

again, which is shown in Figure E-17. As for the DataBinding process in this Course form, refer to section 4.14 in Chapter 4 to get more detailed information for this issue.

cmdSelect ▼ Click ▼

```
Private Sub cmdSelect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSelect.Click
A       Dim CourseTableApt As New CSE_DEPTDataSetTableAdapters.CourseTableAdapter
B       Dim FacultyTableApt As New CSE_DEPTDataSetTableAdapters.FacultyTableAdapter
C       Dim strFacultyID As String
D       FacultyTableApt.ClearBeforeFill = True
        strFacultyID = FacultyTableApt.FindFacultyIDByName(ComboName.Text)
        If strFacultyID = String.Empty Then
            MessageBox.Show("No matched faculty_id found!")
E           Exit Sub
        End If
        CourseTableApt.FillByFacultyID(CSE_DEPTDataSet.Course, strFacultyID)
        If CSE_DEPTDataSet.Course.Count = 0 Then
            MessageBox.Show("No Matched Courses Found!")
            Exit Sub
        End If
End Sub
```

Figure E-17. The coding for the Select button event procedure in Course form

E.4 Develop DataSet coding for the Insert Faculty form

The functionality of this Insert Faculty form is to insert a piece of new faculty information into the Faculty table in our sample database CSE_DEPT. We need to develop a new query method attached to the Faculty TableAdapter to perform this data insertion.

Open the DataSet Designer wizard and right click on the last item from the Faculty table Adapter and select the Add Query item to open the TableAdapter Query Configuration Wizard. Keep the default setting for the first window and select the item **INSERT** from the second window since we need to perform an insertion query. Click the Next button to go to the next window. On the opened next window, click the Query Builder button to build our query, which is shown in Figure E-18.

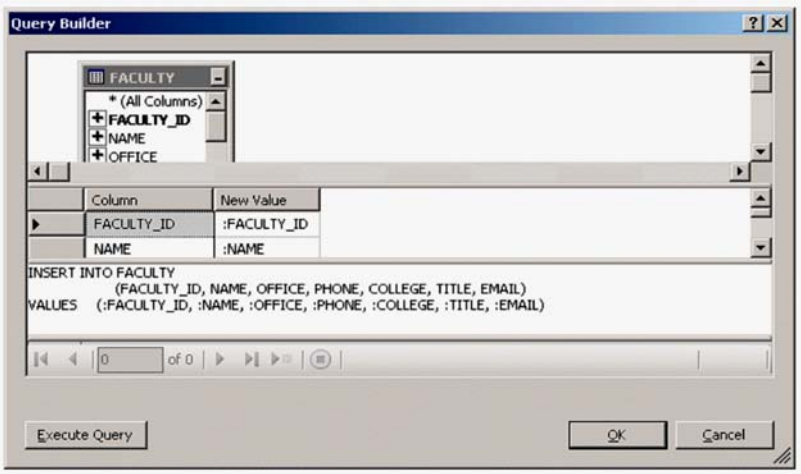

Figure E-18. The Insert Query

The query string is identical with our desired one, so click the OK and then the Next buttons to go to the next window. Change the query method's name to InsertFaculty on the next window and then click the Next and Finish buttons to complete this query building process.

As for the DataBinding of objects in the Insert Faculty form, refer to section 5.2.9.2 in Chapter 5, and the same binding process is needed for this Insert Faculty form.

The major coding for this Insert Faculty form is performed inside the Insert button click event procedure. This coding is identical with that we did in section 5.2.8 in Chapter 5. Refer to that section to get more detailed coding information for this event procedure.

E.5 Develop DataSet coding for the Student form

Two functionalities are existed in this form. The first one is to pick up all student information such as student_id, gpa, major, credits and email based on the input student name. The second functionality is to retrieve back all courses taken by the selected student based on the input student_id. Both functionalities are triggered by a clicking of the Select button on this Student form.

To match those two functionalities, two queries are needed. Since there is no Student Name column in the StudentCourse table, therefore we need to first get the student_id from the Student table based on the input student name, and then we can get all courses taken by the selected student from the StudentCourse table based on the student_id we obtained from the first query. Two queries are built on two TableAdapters: the first query is built on the StudentTableAdapter and the second query is built on the StudentCourseTableAdapter.

Let's build the first query starting from the Student TableAdapter.

Open the DataSet Designer and right click on the last item on the Student TableAdapter, select Add Query item to open the TableAdapter Query Configuration Wizard. Keep the default settings for the first two windows, and click the Query Builder button for the third window to open the Query Builder window to build our query, which is shown in Figure E-19.

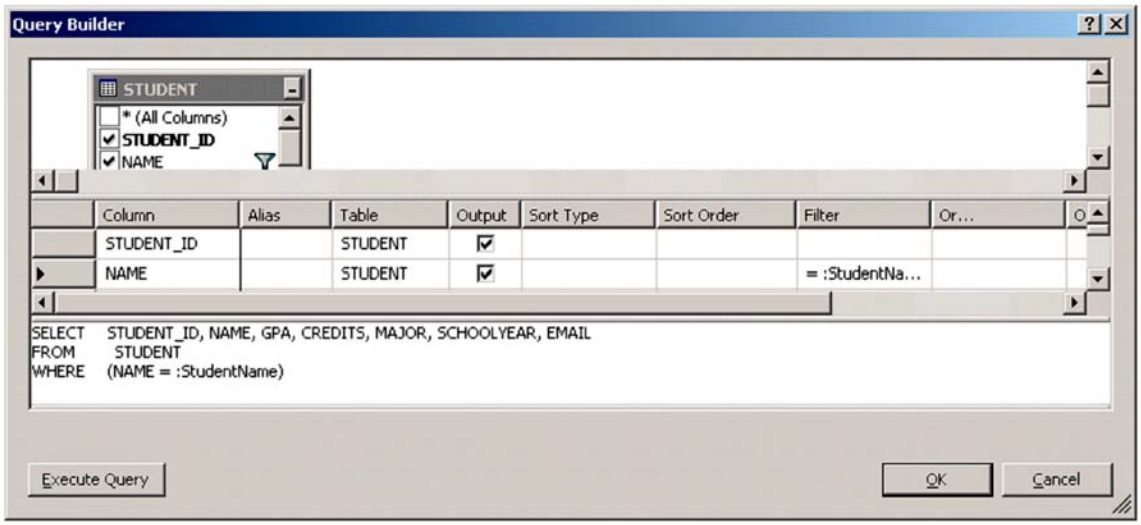

Figure E-19. The Query Builder

Type a question mark in the Filter column along the NAME row and press the Enter key from your keyboard to create a dynamic parameter. Change the name of this parameter to StudentName. Your finished Query Build dialog is shown in Figure E-19.

Click the OK and then Next button to go to the next window. Change the query method's name to FillByStudentName, as shown in Figure E-20.

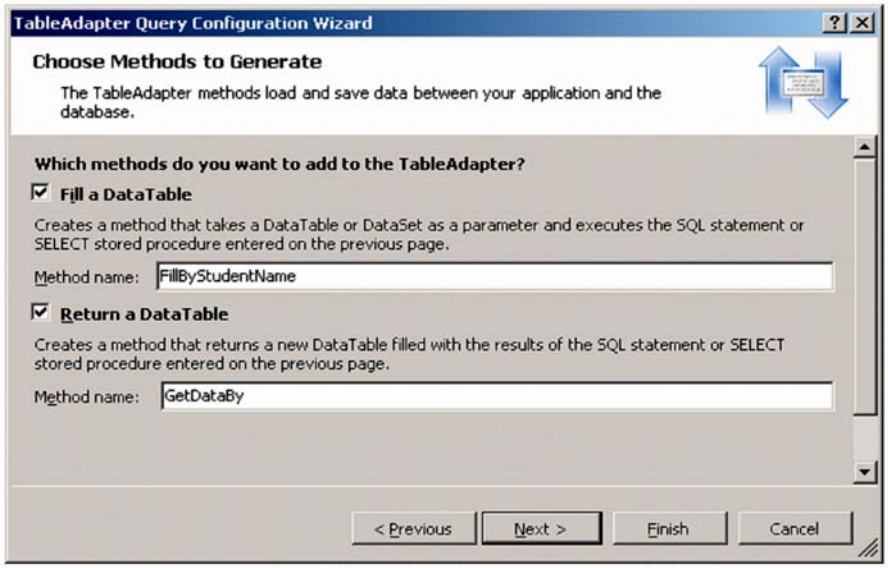

Figure E-20. Change the query method's name

Click the Next and then Finish buttons to complete this query building process.

Still in the DataSet Designer wizard, right click on the last item of the StudentCourse TableAdapter and select the Add Query item to open the TableAdapter Query Configuration Wizard. Keep the default settings for the first two windows, and click the Query Builder button for the third window to open the Query Builder window to build our query, which is shown in Figure E-21.

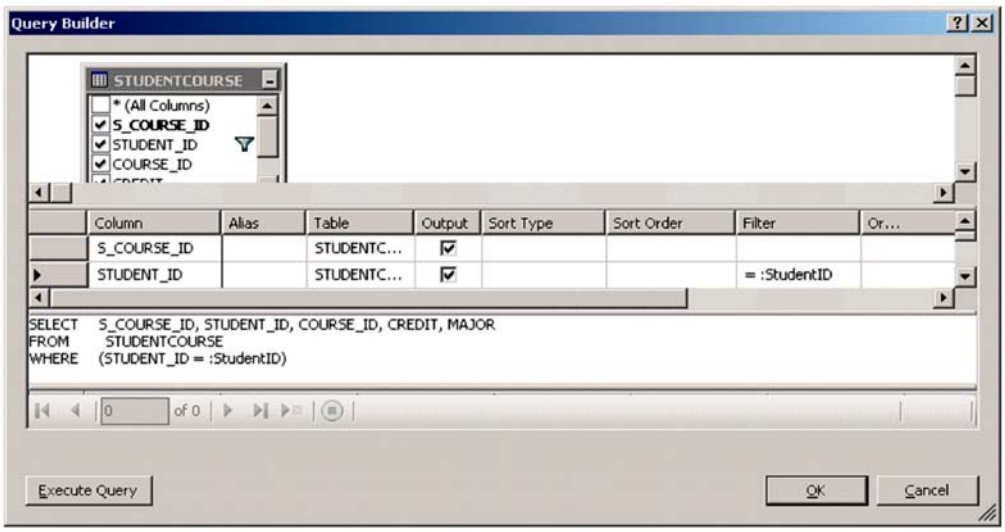

Figure E-21. The Query Builder

Type a question mark in the Filter column along the STUDENT_ID row and press the Enter key from your keyboard to create a dynamic parameter. Change the name of this parameter to StudentID. Your finished Query Build dialog is shown in Figure E-21.

Click the OK and then Next button to go to the next window. Change the query method's name to FillByStudentID, as shown in Figure E-22.

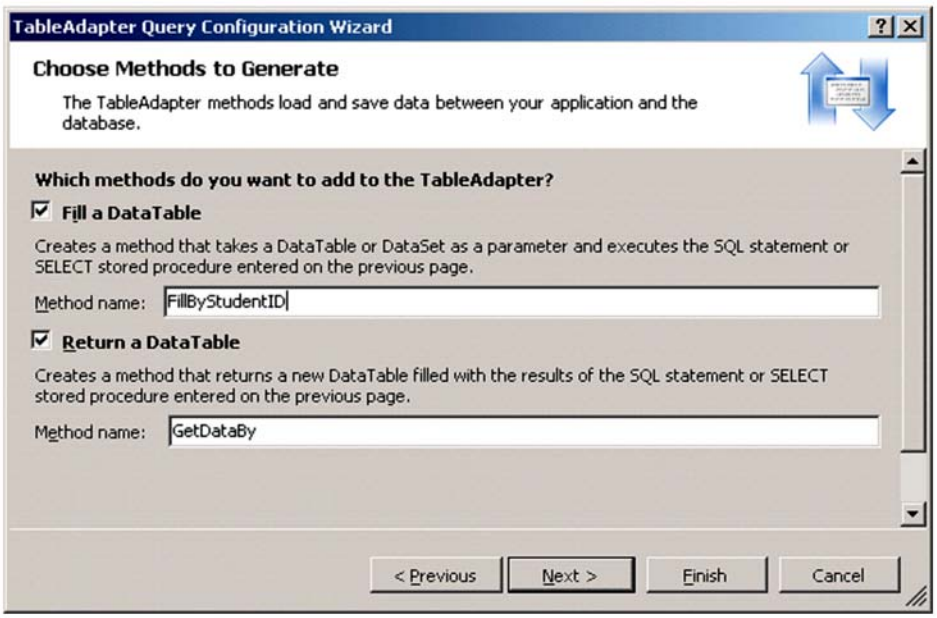

Figure E-22. Change the query method's name

Click the Next and then Finish buttons to complete this query building process.

The DataBinding between six textboxes and the data columns in the Student table is similar with those we did in section 4.11 in Chapter 4, and refer to that section to get a more clear picture in how to perform the data binding for those controls. Here we will give a quick review in how to perform the data binding between the course listbox and the StudentCourse table in the database.

The mapping between the course listbox and the StudentCourse table is one-to-many, which means that one listbox may contain multiple courses. Perform the following operations to complete this binding process.

Open the Student form window, click the course listbox, and then go to the DataSource property. Select the StudentCourseBindingSource for this property. Then go to the DisplayMember property to select COURSE_ID.

Now let's handle the coding for the Select button click event procedure in the Student form. As the project runs, after the user selected the desired student name and click the Select button, the detailed information about the selected student will be retrieved and displayed in six textboxes, and all courses taken by the selected student will be also retrieved back and displayed in the Course listbox. The detailed coding for this Select button event procedure is shown in Figure E-23. The functionalities of this piece of codes are straightforward and easy to be understood.

One point to be noticed for this coding is that two TableAdapter objects are needed for this procedure: one is the StudentTableAdapter used to pick up the student detailed information and the StudentCourseTableAdapter used to pick up all courses taken by the selected student.

cmdSelect ▼ Click ▼

```
Private Sub cmdSelect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSelect.Click
A       Dim StudentTableApt As New CSE_DEPTDataSetTableAdapters.StudentTableAdapter
B       Dim StudentCourseTableApt As New CSE_DEPTDataSetTableAdapters.StudentCourseTableAdapter
C       Dim strName As String
D       strName = FindName(ComboName.Text)
        If strName = "No Match" Then
            MessageBox.Show("No Matched Student Found!")
            Exit Sub
E       End If
        PhotoBox.SizeMode = PictureBoxSizeMode.StretchImage
        PhotoBox.Image = System.Drawing.Image.FromFile(strName)
        StudentTableApt.ClearBeforeFill = True
        StudentTableApt.FillByStudentName(CSE_DEPTDataSet.Student, ComboName.Text)
        If CSE_DEPTDataSet.Student.Count = 0 Then
            MessageBox.Show("No matched student found!")
            Exit Sub
        End If
        StudentCourseTableApt.ClearBeforeFill = True
        StudentCourseTableApt.FillByStudentID(CSE_DEPTDataSet.StudentCourse, txtID.Text)
End Sub
```

Figure E-23. The coding for the Select button event procedure in Student form