Lab 3: Using Data Access Objects

For background information on this lab, click each of these topics:

Objectives

In this lab, you will build and test forms that use data access objects (DAO).

After completing this lab, you will be able to:

- Open a database by using DAO.
- Navigate and manipulate a recordset by using DAO.
- Find information in a table.
- Create a recordset based on a query.
- Use a parameter query.

Prerequisites

Before working on this lab, you should be familiar with the following concepts:

- The data access object model.
- The contents of this chapter.

Lab Setup

To complete this lab, you need the following setup:

- Visual Basic version 5.0 or later.

To see a demonstration of the completed lab solution, click this icon.

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with each lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs folder of the *Mastering Microsoft Visual Basic 5* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 3.

Exercise 1: Creating and Navigating a Recordset

In this exercise, you will use data access objects (DAO) to open a database and create a recordset.

Exercise 2: Adding and Editing Records

In this exercise, you will add new records and edit existing records using various DAO methods.

Exercise 3: Finding Records

In this exercise, you will let the user search for specific records in the Categories table, based on text within the Description field.

Exercise 4: Using Queries

In this exercise, you will use a saved query to return information from the Nwind.mdb database.

Exercise 5 (Optional): Disabling Buttons During Edit

In this exercise, you will enable and disable the appropriate command buttons while the user is navigating or editing data on the Categories form.

Exercise 6 (Optional): Using a Parameter Query

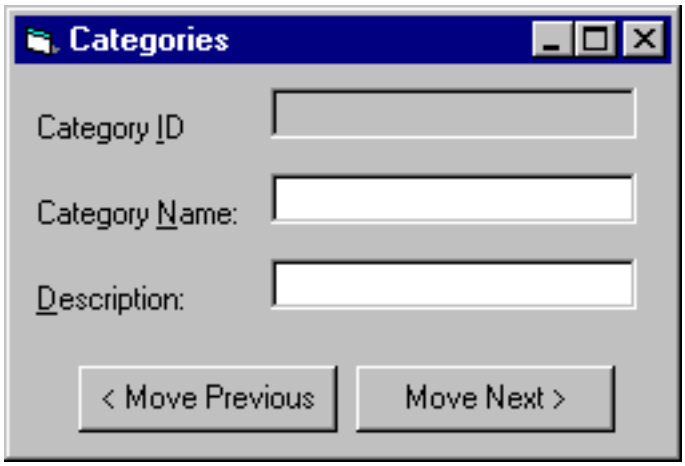
In this exercise, you will use a parameter query to return sales information from the Nwind.mdb database.

Exercise 1: Creating and Navigating a Recordset

In this exercise, you will use data access objects (DAO) to open a database and create a recordset.

▶ **Create Categories form**

1. Create a new Visual Basic Standard EXE project.
2. Name the default form frmCategories.
3. Add controls to the form, as shown in the following illustration.



Note Because the ID data will not be edited by the user, the Category ID field is a **Label** control.

▶ **Open a database and create a recordset**

1. On the Categories form, declare two module-level variables, as shown in the following code:

```
Dim dbCurrent As Database
Dim recCategories As Recordset
```

2. Set a reference to the Microsoft DAO 3.5 Object Library.
3. In **Load** event procedure of the Categories form, follow these steps:
   a. Open \Program Files\DevStudio\VB\Nwind.mdb and save the resulting database object in the variable dbCurrent.
   b. Create a recordset by using the Categories table in the Nwind.mdb database, and capture the resulting **Recordset** object in the variable recCategories.
   c. Move to the first record in the recordset.
   d. Read the Category ID, Category Name, and Description fields into the appropriate controls. To see an example of how your code should look, click this icon.

```
Private Sub Form_Load()
```

```
    Set dbCurrent = OpenDatabase("C:\Program
Files\DevStudio\VB\Nwind.mdb")
    Set recCategories = dbCurrent.OpenRecordset("Categories")
    recCategories.MoveFirst
    FillFields
End Sub
Sub FillFields()
    ' this procedure will populate the fields on the form.
    ' it will be called from multiple procedures.
    lblCategoryID.Caption = recCategories.Fields("CategoryID")
    txtCategoryName.Text = recCategories.Fields("CategoryName")
    txtDescription.Text = recCategories.Fields("Description")
End Sub
```

4. In the form **Unload** event procedure, **Close** the database.
5. Test the application. Is the first record in the Categories table displayed correctly?

▶ Write code for the Move Previous and Move Next buttons

1. Add code to make the **Move Previous** and **Move Next** command buttons functional.
   - After a move, fill in the fields on the form with data.
   - Test for the beginning and the end of the recordset. To see an example of how your code should look, click this icon.

```
Private Sub cmdPrevious_Click()
    recCategories.MovePrevious
    If recCategories.BOF Then
        Beep
        recCategories.MoveFirst
    Else
        FillFields
    End If
End Sub

Private Sub cmdNext_Click()
    recCategories.MoveNext
    If recCategories.EOF Then
        Beep
        recCategories.MoveLast
    Else
        FillFields
    End If
End Sub
```

▶ Test the project

1. Run the form.
2. Click the **Move Previous** button.
3. Click the **Move Next** button.

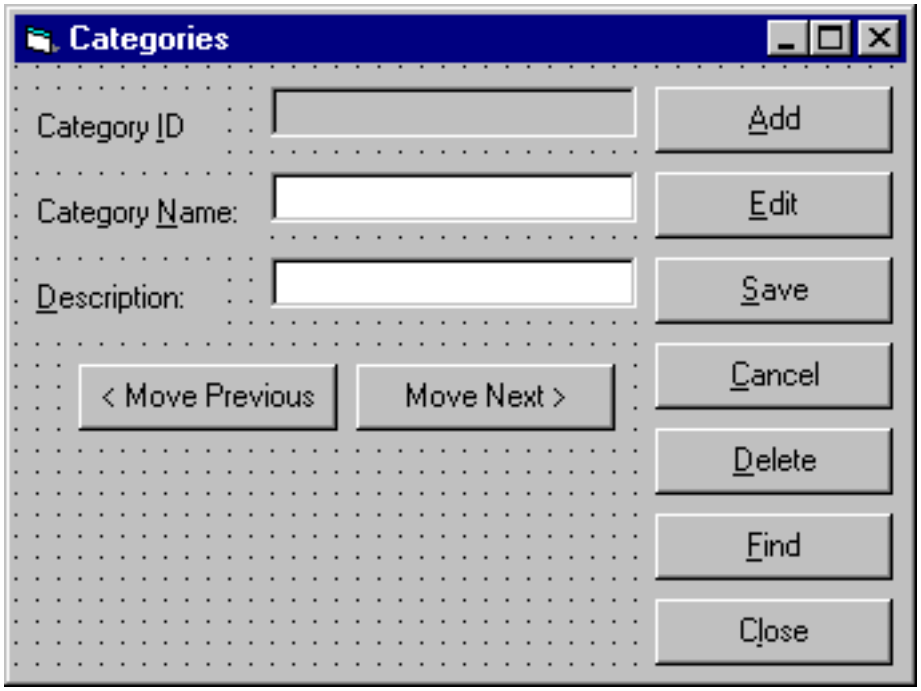
   You should see different Category records.

Exercise 2: Adding and Editing Records

In this exercise, you will add new records and edit existing records using various DAO methods.

▶ Modify the form

1. Modify the Categories form by adding the command buttons shown in the following illustration.



▶ **Write code for the Add, Edit, Save, Cancel, and Delete buttons**

1. To clear the text fields and enter Add mode, write code for the **Add** button.

   Note Because the Category ID field is a counter field, it will automatically contain a new value as soon as you issue the **AddNew** method. You can then place the value for this field in the Category ID label. For example, execute the following code immediately after the **AddNew** method.

   ```
   lblCategoryID.Caption = recCategories("CategoryID")
   ```

2. To enter Edit mode, write code for the **Edit** button. For more information about editing records, see Updating Records.
3. To write changes from the text boxes back to the recordset and update it, write code for the **Save** button.
4. To cancel the pending update and update the fields with the current record in the recordset, write code for the **Cancel** button.
5. To delete the current record, write code for the **Delete** button. Move to the next record and retrieve the fields.

   To see an example of how your code should look, click this icon.

```
Private Sub cmdEdit_Click()
    recCategories.Edit
End Sub
Private Sub cmdSave_Click()
    recCategories.Fields("Category Name") = txtCategoryName.Text
    recCategories.Fields("Description") = txtDescription.Text
    recCategories.Update
    recCategories.Bookmark = recCategories.LastModified
End Sub
Private Sub cmdCancel_Click()
    recCategories.CancelUpdate
```

```
    FillFields
End Sub
Private Sub cmdDelete_Click()
    recCategories.Delete
    recCategories.MoveNext
    If recCategories.EOF Then
        recCategories.MoveLast
    End If
    FillFields
End Sub
```

▶ **Test the application**

1. Add a new record.
2. Cancel the Add. Is the previous information restored correctly?
3. Add and save a new record.
4. To ensure that you are on the correct record after the Add, test the **Move Next** and **Move Previous** buttons.
5. Delete the new record.
6. Try to delete the first category.

   The referential integrity constraints on this table should cause a run-time error. In a later lab, you will trap for that error.

Exercise 3: Finding Records

In this exercise, you will let the user search for specific records in the Categories table, based on text within the Description field.

▶ **Write code for the Find button**

1. Use the **InputBox** function to prompt users for a string that represents a portion of the Description field.
2. Using the string returned from the Input box, create an SQL string to select the matching records from the Categories table, as shown in this example:

   ```
   "select * from categories where [Description] like '*pasta*'"
   ```

3. Create a recordset based on the SQL query, and assign the results to the **recCategories** variable. This enables the other buttons to operate in the new recordset.
4. If no records are found (recCategories.RecordCount = 0), display a message saying that no records were found. Otherwise, display the first record in the new recordset.

   To see an example of how your code should look, click this icon.

```
Private Sub cmdFind_Click()
    Dim strSQL As String
    Dim strAnswer As String
    strAnswer = InputBox("Enter any portion of the Description", "Find
Records")
    strSQL = "select * from categories where [Description] like " & _
        "'*" & strAnswer & "*'"
    Set recCategories = dbCurrent.OpenRecordset(strSQL)
    If recCategories.RecordCount = 0 Then 'no records found
        MsgBox "No matching records found. Displaying all records."
        Set recCategories = dbCurrent.OpenRecordset("Categories")
    Else 'at least one record was found
        recCategories.MoveFirst
        FillFields
```

```
    End If
End Sub
```

▶ **Test the application**

1. Run the form.
2. Click the **Find** button.
3. Enter **sweet** in the Input box.

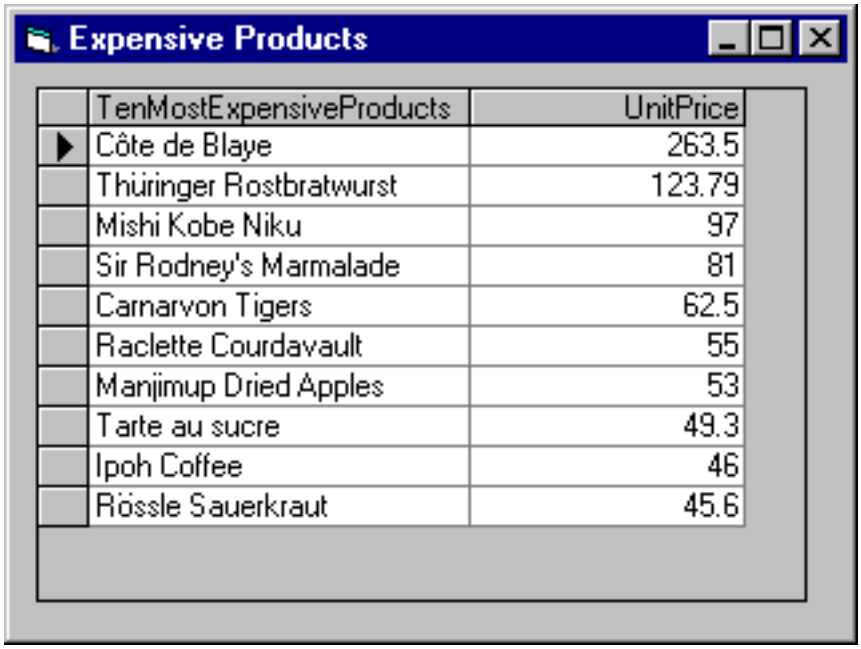
   You should see two records that contain the text **sweet** in the description.

Exercise 4: Using Queries

In this exercise, you will use a saved query to return information from the Nwind.mdb database.

▶ **Query for the ten most expensive products**

1. Create a new form.
2. The saved query **Ten Most Expensive Products** in the Nwind.mdb database returns the ten highest priced products. In the **Form_Load** event procedure on the new form, create a recordset based on this query, and display the results in a **DBGrid** control, as shown in the following illustration.



To see examples of how your code should look, click this icon.

```
Private Sub Form_Load()
    Dim dbCurrent As Database
    Set dbCurrent = DBEngine.Workspaces(0).OpenDatabase("C:\Program
Files\DevStudio\VB\Nwind.mdb")
    Set datProducts.Recordset = OpenRecordset("Ten Most Expensive
Products")
End Sub
```

Place a hidden **Data** control on the form and set the **DataSource** property of the **DBGrid** control to the **Data** control. Then, you can set the **Recordset** property of the **Data** control equal to the recordset returned by the saved query.

▶ **Test the application**

1. On the **Project** menu, click **Project Properties**.
2. Change the Startup Object to the new form, and then click OK.
3. Run the form.

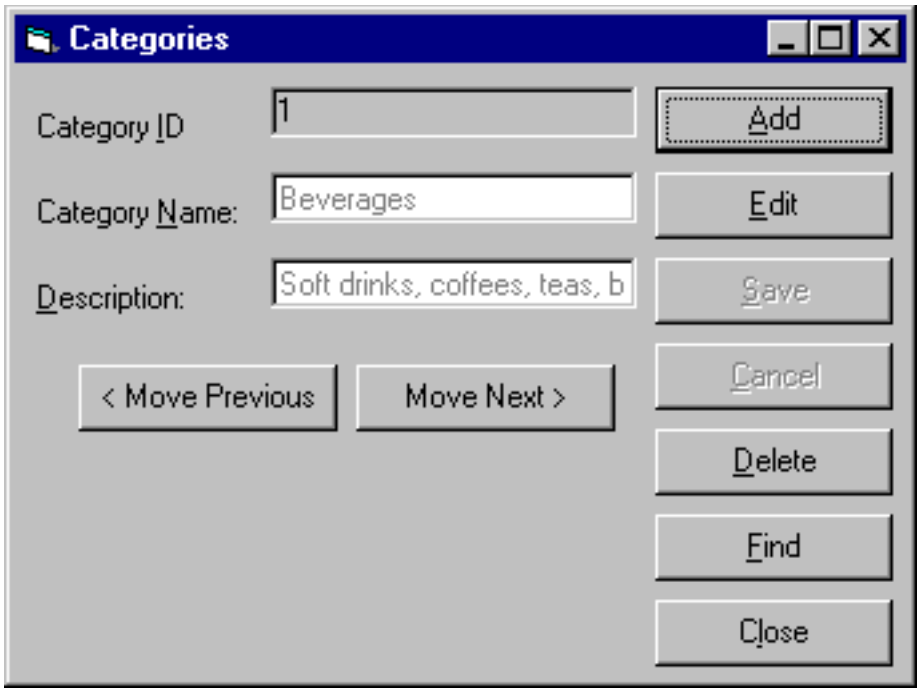
   You should see the ten most expensive products in the grid.

Exercise 5: (Optional) Disabling Buttons During Edit

In this exercise, you will enable and disable the appropriate command buttons while the user is navigating or editing data on the Categories form.

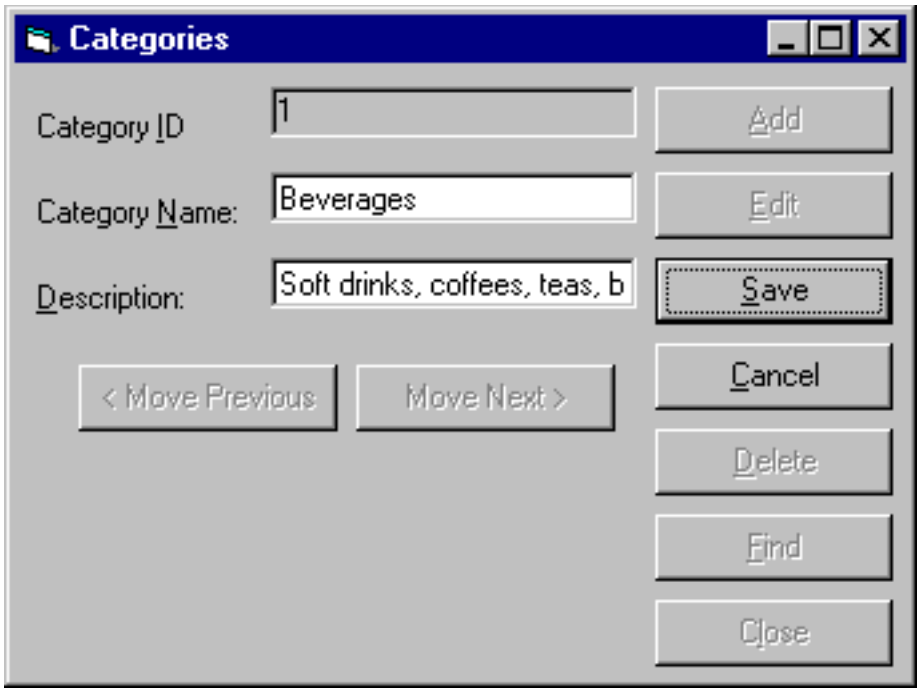
▶ **Enable and disable controls**

1. When a user navigates in the Categories form, disable the text boxes, and the **Save** and **Cancel** button, so changes cannot be made to the data unless the user selects the **Edit** or **Add** button, as shown in the following illustration.



2. When a user selects the **Edit** or **Add** button, enable the text boxes, and the **Save** and **Cancel** buttons. Disable all other buttons on the form.

   After you have finished the form, it should resemble the following illustration.



To see an example of how your code should look, click this icon.

```
Private Sub cmdEdit_Click()
    ButtonEditAddMode
    rs.Edit
End Sub

Sub ButtonEditAddMode()
    cmdSave.Enabled = True
    cmdCancel.Enabled = True
    cmdAdd.Enabled = False
    cmdEdit.Enabled = False
    cmdDelete.Enabled = False
    cmdFind.Enabled = False
    cmdClose.Enabled = False
    cmdPrevious.Enabled = False
    cmdNext.Enabled = False
    txtCategoryName.Enabled = True
    txtDescription.Enabled = True
End Sub
```

3. In the **Save** and **Cancel** buttons, add code that:
   - Enables the command buttons **Add**, **Edit**, **Delete**, **Find**, **Move Previous**, and **Move Next**.
   - Disables the text boxes.
   - Disables the **Save** and **Cancel** buttons.

▶ **Test the application**

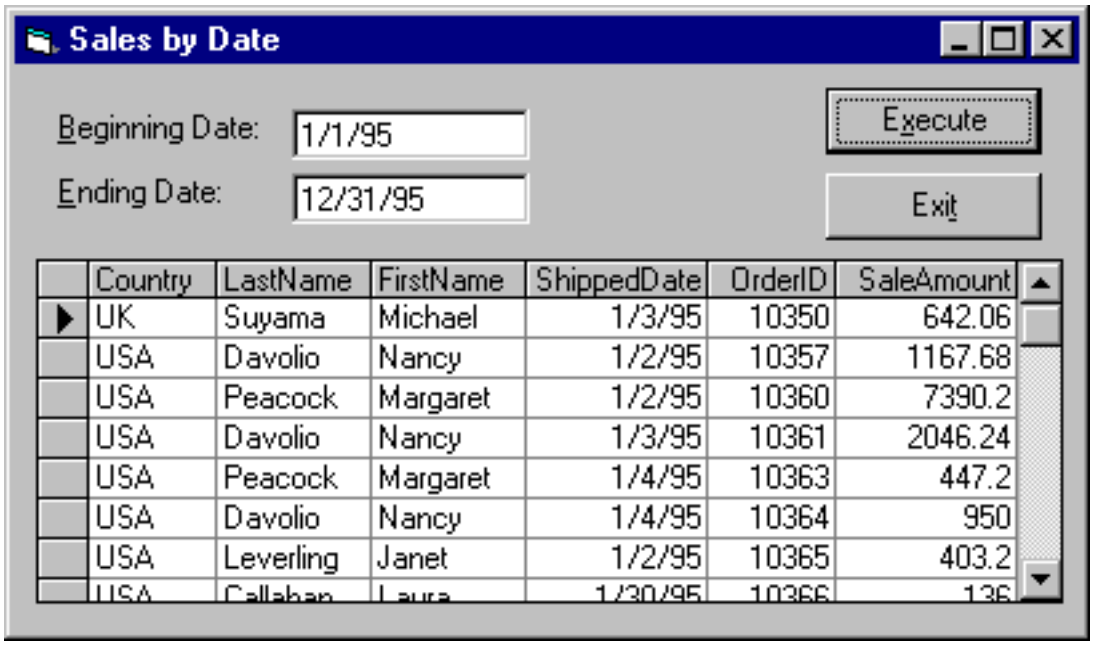
1. Add a new record and save your changes.
2. Edit the new record and save your changes.
3. Edit the new record again and cancel your changes.

Exercise 6: (Optional) Using a Parameter Query

In this exercise, you will use a parameter query to return sales information from the Nwind.mdb database.

▶ Use a parameter query

1. Create a new form that uses the **Employee Sales by Country** parameter query to display sales records in a date range, as shown in the following illustration.



   The parameter names for the query are **Beginning Date** and **Ending Date**. To see an example of how your code should look, click this icon.

```
Private Sub cmdExecute_Click()
    Dim strSQL As String
    Dim dbCurrent As Database
    Dim recSales As Recordset
    Dim qrySales As QueryDef


    Set dbCurrent = DBEngine.Workspaces(0).OpenDatabase _
                    ("C:\Program Files\DevStudio\VB\Nwind.mdb")
    Set qrySales = dbCurrent.QueryDefs("Employee Sales by Country")
    qrySales.Parameters("Beginning Date") = CDate(txtBegin)
    qrySales.Parameters("Ending Date") = CDate(txtEnd)
    Set recSales = qrySales.OpenRecordset()
    Set datSales.Recordset = recSales
End Sub
```

2. Test the application.