Lab 4: Advanced Database Development

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

Objectives

In this lab, you will add functionality to a project that uses advanced data access concepts.

After completing this lab, you will be able to:

- Handle referential integrity.
- Understand and work with a multi-user environment.
- Access external data with ODBCDirect.

Prerequisites

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

- The data access objects (DAO) object model.
- The contents of this chapter.

Lab Setup

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

- The data access objects (DAO) object model.
- The contents of this chapter.

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 4.

Exercise 1: Handling Referential Integrity Violations

In this exercise, you will trap a run-time error produced by violating referential integrity rules.

Exercise 2: Multi-User Issues

In this exercise, you will trap run-time errors that occur when multiple users try to edit the same database information.

Exercise 3 (Optional): Using ODBCDirect

In this exercise, you will access external data stored in a Microsoft SQL Server database with ODBCDirect.

To complete this lab, you need the following setup:

- Visual Basic version 5.0 or later.

Exercise 1: Handling Referential Integrity Violations

In this exercise, you will trap a run-time error produced by violating referential integrity rules.

▶ **Check the referential integrity error**

1. In the folder \Labs\Lab04, open NWindAdv.vbp.
2. Run the application.
3. Try to delete the first record. What happens?
4. End the application.

▶ **Trap the referential integrity violation**

1. Open the Categories form.
2. In the Click event of the **Delete** button, add an error-handling routine to trap and handle the error produced above. In the handler, display a message box saying that the item was not deleted.

```
HandleError:
    If Err.Number = 3200 Then
        MsgBox "Can not delete a Category which has " & _
            "related products in the Products Table."
    Else
        MsgBox "Error " & Err.Number & ": " & Err.Description
    End If
```

▶ **Test the application**

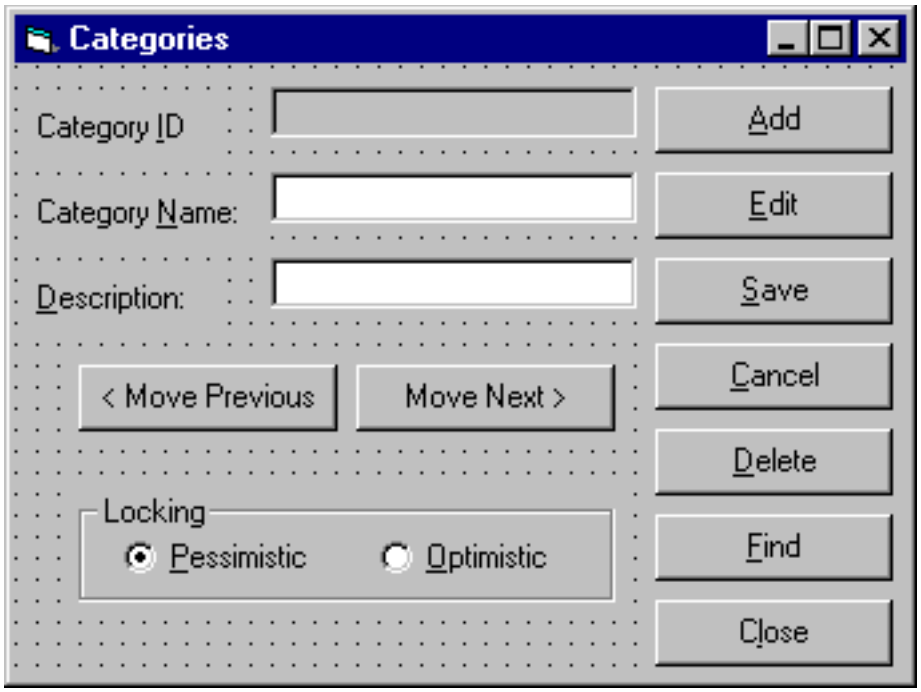
1. Run the application.
2. Try to delete the first category. What happens?

Exercise 2: Multi-User Issues

In this exercise, you will trap run-time errors that occur when multiple users try to edit the same database information.

▶ **Add locking options**

1. Add option buttons to the Categories form, as shown in the following illustration.



2. Set the **Value** property of the **Pessimistic** option button to **True**.
3. To set the **LockEdits** property, add code to each of the option buttons.
4. Add the following code to the **Save** button after the **Update** method, and to the **Cancel** button after the **CancelUpdate** method.

   ```
   DBEngine.Idle dbFreeLocks
   ```

   This will free any database locks that were set during the add or edit process.
5. Save the project.

▶ Test for multi-user errors

1. Create an executable for the application.
3. Run two instances of the application.
4. Using the **Pessimistic** locking option for both instances, try to edit the same record in both instances of Visual Basic, and note the error that occurs.

The error should display as "3260: Couldn't update; currently locked by user 'name' on machine 'name'."

5. Using the **Optimistic** locking option for both instances, edit the same record in both instances of Visual Basic.

   In the first instance, change the Category Name and save the record.

   In the second instance, change the Category Name and try to save the record.

   Note the error that occurs.

The error should display as "3197: The Microsoft Jet database engine stopped the process because you and another user are attempting to change the same data at the same time."

6. In the first instance of Visual Basic, set locking to **Pessimistic**, and edit the first record.

   In the second instance of Visual Basic, set locking to **Optimistic**, edit the first record, change the Category Name, and save the record.

   Note the error that occurs.

The error should display as "3260: Couldn't update; currently locked by user 'name' on machine 'name'."

7. End both instances of the application, and close the second instance of Visual Basic.

▶ Trap multi-user run-time errors

1. Add code to the **Edit** command to check for the error encountered in Step 4 of the previous procedure, and notify the user if an error occurs.

```
Private Sub cmdEdit_Click()
On Error GoTo EditErrorHandler
    recCategories.Edit
    ButtonEditAddMode
Exit Sub
EditErrorHandler:
    Select Case Err.Number
    Case 3260 'page currently locked
        MsgBox "Record is currently locked. Try again later."
    Case 3197 'data has changed
        MsgBox "Data has been changed and will be refreshed."
        'get updated data
        recCategories.Bookmark = recCategories.Bookmark
        FillFields
        recCategories.Edit
    Case Else
        MsgBox Err.Number & ": " & Err.Description
    End Select
End Sub
```

2. Add an error-handling routine to the **Save** command Click event.

   In the error-handling code, check for the error encountered in Step 5 of the previous procedure. If the error is found, notify the user and cancel the update.

   In the error-handling routine, check for the error encountered in Step 6 of the previous procedure. If the error is found, notify the user.

```
Private Sub cmdSave_Click()
On Error GoTo SaveErrorHandler
    ' save the record
    recCategories.Fields("CategoryName") = txtCategoryName.Text
    recCategories.Fields("Description") = txtDescription.Text
    recCategories.Update
    recCategories.Bookmark = recCategories.LastModified
    ButtonNonEditAddMode
    DBEngine.Idle dbFreeLocks
Exit Sub
SaveErrorHandler:
    Select Case Err.Number
    Case 3260
        MsgBox "Record is currently locked. Try Save again later or
cancel changes."

    Case 3197
        MsgBox "Data has been changed by another user. Your changes
have been canceled."
        cmdCancel_Click

    Case Else
        MsgBox Err.Number & ": " & Err.Description
```

```
    End Select
End Sub
```

3. Save the application.
4. Repeat the steps in this procedure to test the application.

Exercise 3: (Optional) Using ODBCDirect

In this exercise, you will access external data stored in a Microsoft SQL Server database by using ODBCDirect.

▶ Query the external database

For this exercise, you will need a server running Microsoft SQL Server and its sample Pubs database.

1. Create a new Visual Basic standard EXE project.
2. Create a new form with a command button and a list box.

   Your form should resemble the following illustration.

3. In the Click event for the command button, do the following:
   a. Create an ODBCDirect workspace.
   b. Create a new connection to the workspace.
   c. Create a recordset based on the Authors table.

      To see a code sample of the SQL statement, click this icon.

```
SELECT au_lname FROM Authors ORDER BY au_lname
```

   d. Populate the list box with the last names (field au_lname) from the Authors table.