22. VB Programming Fundamentals – Data Access with Data Objects

22.1 Data Access Object

MS Data Access Object (DAO) enables you to use a programming language to access and manipulate data in local or remote databases, and to manage databases, their objects, and their structure. There are 17 different DAO object types. You can declare new DAO object variables for any of the object types.

For example, the following Visual Basic for Applications (VBA) code creates object variables for a **Database** object, a dynaset-type **Recordset** object, and a **Field** object:

```
Dim dbsExample As Database
Dim rstExample As Recordset
Dim fldExample As Field
```

Here is an example to show you how to use VB data control and DAO to access manipulate data of plot and tree tables in timber cruising we have build in this class. You can also generate reports.

(1) Open a VB project and name it as prjVBDataAccess.vbp. The interface of Form 1 should look like what we have in Figure 1.

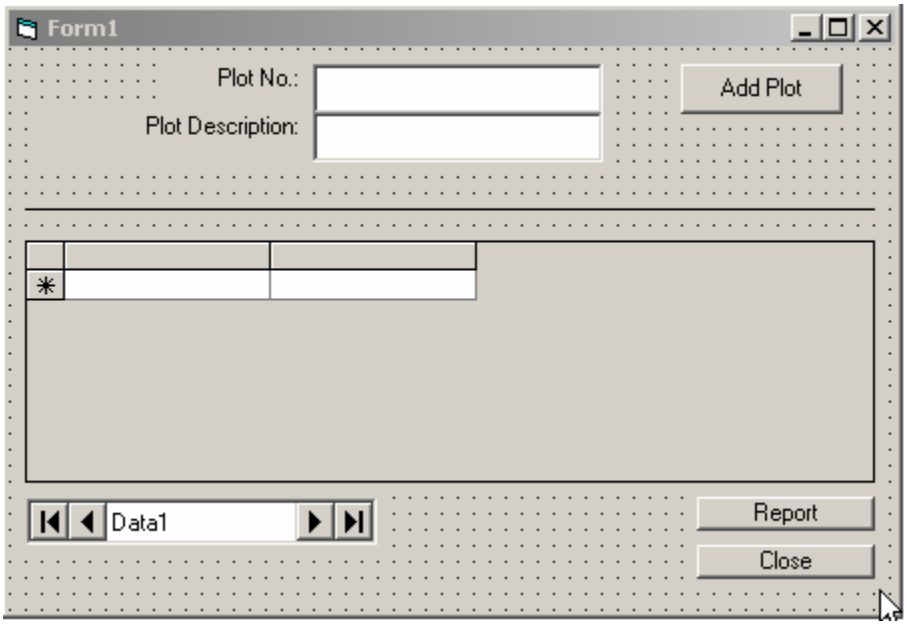

Figure 1. Interface of form 1.

(2) Reference DAO

To reference MS DAO:

a. Select References… from the Project menu, then the References dialog box will be displayed.
b. Find Microsoft DAO 3.51 Object Library in the list box and select the check box to its left.
c. Click OK button.

(3) Code in Form 1.

```
'In general declaration
Dim dbCruise As Database
Dim rsPlot As Recordset

'Command button – Add Plot
Private Sub Command2_Click()

   rsPlot.AddNew
   rsPlot.Fields("PlotNo") = Text1.Text
   rsPlot.Fields("Desc") = Text2.Text
   rsPlot.Update

   DBGrid1.Enabled = True
   DBGrid1.SetFocus

End Sub


'Command button – Report form
Private Sub Command3_Click()
Form2.Show
End Sub

'Set default value in column 2 of DB grid
Private Sub DBGrid1_Change()

   DBGrid1.Columns(1).Text = Text1.Text

End Sub

'Set database connection while loading form 1
Private Sub Form_Load()

   ChDir App.Path
   Set dbCruise = OpenDatabase(App.Path & "\dbCruise.mdb")
   Set rsPlot = dbCruise.OpenRecordset("tblPlot")

   DBGrid1.Enabled = False
   ChDir App.Path
   Data1.DatabaseName = App.Path + "\dbCruise.mdb"
   Data1.RecordSource = "Select * From tblTrees"


End Sub
```

(4) Report.

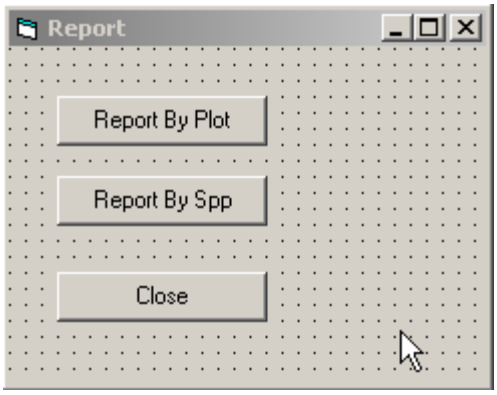

Figure 2. Report form.

(5) Code in report form.

```
‘Report by plot
Private Sub Command1_Click()

Call LoadAccessRpt("rptBAVolPlot")
End Sub


‘Report by species
Private Sub Command2_Click()
Call LoadAccessRpt("rptBAVolSpp")
End Sub
```

(6) Report procedure in standard module

```
Public Sub LoadAccessRpt(rptName As String)

   Dim objAccess As Object
   Set objAccess = GetObject(App.Path & "\dbCruise.mdb", "Access.Application")
   objAccess.docmd.openreport rptName, 2     'Open the report in print preview
   objAccess.Visible = True                  'make Access visible
   objAccess.docmd.maximize                  'maximize the report window
   Set objAccess = Nothing                   'end the OLE Automation session

End Sub
```

(7) Run your application

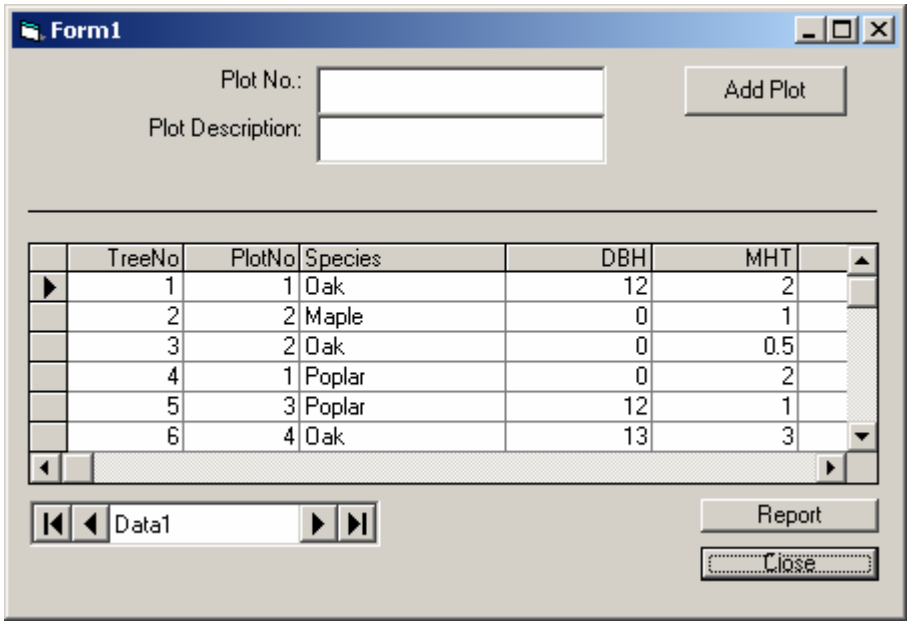

Figure 3. Output of DAO application.

22.2 ActiveX Data Object

Since we already knew how to use DAO, this lecture here especially focuses on ActiveX Data Objects (ADO). There is a lot to learn about ADO, however, we will only cover the followings:

- The ADO object model
- How to open a connection to the database and how to close it when the job is done
- Using a command to query the database
- Using recordset to manipulate data

ADO Object Model:

There are three main objects in ADO:

- The Connection object, which is designed to handle the connection to the database
- The Command object, which is designed to help us handle SQL commands
- The Recordset object, which we use to hold data and manipulate it.

In addition to these three main objects, ADO makes use of four subsidiary objects – Property, Error, Parameter, and Field – and four associated collections, which are used access these subsidiary objects.

What is ADODB?

The full implementation of ADO that provides full access to the complete ADO object model.

Example:

Let me give you an example of using ADO. Suppose we continue to work on our previous project of calculating basal area and volume of trees. In the example here instead of using direct file access, we are going to create an Access database to hold the tree data and result, then use ADO to access the database and manipulate the data in it.

(1) Create a database: create a database named dbTreeData which should have the following two tables:
   a. tblTreeData
      i. TreeNo, Integer, primary key
      ii. DBH, single
      iii. NofLogs, single
   b. tblResult
      i. TreeNo, Integer, primary key
      ii. DBH, single
      iii. NofLogs, single
      iv. BA, single
      v. Vol, single

   Remember to input the data of 10 trees in table tblTreeData.

(2) Start a new Vb project: add a list box, a label, and three command buttons to form1 (Figure 4).

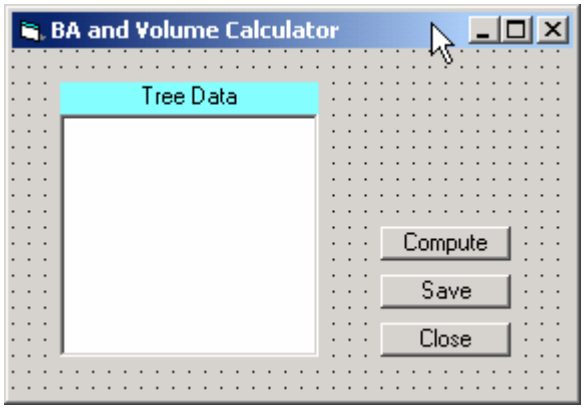

Figure 4. Interface of ADO project.

(3) Set a Reference to ADO:

Like DAO, you need to add a reference to ADO in VB project before attempting to connect to any database using ADO. VB 6.0 was released with version 2.x of ADO.

To add the ADO reference to your project, open your VB project and select Project | References… from the menu bar. The References dialog will open, scroll down and click the checkbox labeled Microsoft ActiveX Objects 2.6 Library. Then click OK button, an ADO reference is added to you project. Once you have added the reference to ADO, declare the follows in the General Declaration:

```
Dim objConn As ADODB.Connection
Dim objComm As ADODB.Command
Dim objRec As ADODB.Recordset
Dim objRecRlt As ADODB.Recordset

Dim aryTreeNo(), aryDBH(), aryNLogs(), aryBA(), aryVol()
Dim NofTrees As Integer
```

(4) Connect to a Database

In ADO, the connection is represented by a Connection object. Because the connection is fundamental to database access, the Connection object is sometimes considered to be the most important item while working with database. There are quite a few ways you can use to connect a database using ADO object. Here is a simple way that is especially used for connecting an Access database.

```
Set objConn = New ADODB.Connection

objConn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;" & _
                    "Data Source=" & App.Path & "\" & "dbTreeData.mdb"

objConn.Open
```

(5) Work on the Recordsets

We can use either the Connection object's Execute method or the Recordset object's Open method to retrieve the data in a table. Here are the syntaxes:

```
Set Recordset = Connection.Execute(CommandText, [RecordsAffected], [Options])
```

Command Text can be a SQL query or a table. Optional Record Affected allows you to specify a variable there, and the provider will return the number of affected records to that variable. Optional Options can be either of adCmdText, adCmdTable, adCmdStoredProc, adCmdUnknown.

```
Recordset.Open([Source], [ActiveConnection], [CursorType], [LockType], [Options])
```

LockType can be adLockReadOnly or adLockOptimistic. In our example here, I used both methods.

```
‘Method 1
 Set objRec = objConn.Execute("tblTreeData", , adCmdTable)

‘Method 2
Set objComm = New ADODB.Command
objComm.CommandText = "tblResult"
 objComm.CommandType = adCmdTable
 objComm.ActiveConnection = objConn

 Set objRecRlt = New ADODB.Recordset
 Set objRecRlt.Source = objComm
 objRecRlt.LockType = adLockOptimistic
 objRecRlt.Open
```

(6) Use methods of Recordset object

AddNew
Update
Delete
MoveFirst
MoveNext
MovePrevious
MoveLast

(7) Add code

```
‘In General Declaration

Dim objConn As ADODB.Connection
Dim objComm As ADODB.Command
Dim objRec As ADODB.Recordset
Dim objRecRlt As ADODB.Recordset

Dim aryTreeNo(), aryDBH(), aryNLogs(), aryBA(), aryVol()
Dim NofTrees As Integer


‘For form_load event

Private Sub Form_Load()


   Set objConn = New ADODB.Connection

   objConn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;" & _
                    "Data Source=" & App.Path & "\" & "dbTreeData.mdb"

   objConn.Open
```

```
        Set objRec = objConn.Execute("tblTreeData", , adCmdTable)

        If Not (objRec.EOF And objRec.BOF) Then
           objRec.MoveFirst
           While Not objRec.EOF
              List1.AddItem objRec.Fields("TreeNo") & ", " & objRec.Fields("DBH") & ", " &
objRec.Fields("NofLogs")
              i = i + 1
              ReDim Preserve aryTreeNo(i), aryDBH(i), aryNLogs(i), aryBA(i), aryVol(i)
              aryTreeNo(i) = objRec.Fields("TreeNo")
              aryDBH(i) = objRec.Fields("DBH")
              aryNLogs(i) = objRec.Fields("NofLogs")
              objRec.MoveNext
           Wend
        End If

        NofTrees = i

        Command3.Enabled = False


    End Sub


    'For command button 1
    Private Sub Command1_Click()

        Dim j
        ReDim aryBA(NofTrees), aryVol(NofTrees)

        For j = 1 To NofTrees
           aryBA(j) = CalBA(aryDBH(j))
           aryVol(j) = CalVol(aryDBH(j), aryNLogs(j))
        Next j
        Command3.Enabled = True
        MsgBox "Calculation was done!"

    End Sub

    'For command button 2
    Private Sub Command2_Click()

        Dim i

        Set objComm = New ADODB.Command
        objComm.CommandText = "tblResult"
        objComm.CommandType = adCmdTable
        objComm.ActiveConnection = objConn

        Set objRecRlt = New ADODB.Recordset
        Set objRecRlt.Source = objComm
        objRecRlt.LockType = adLockOptimistic
        objRecRlt.Open

        If Not (objRecRlt.EOF And objRecRlt.BOF) Then
```

```
        objRecRlt.MoveFirst
        While Not objRecRlt.EOF
            objRecRlt.Delete
            objRecRlt.MoveNext
        Wend
    End If

    For i = 1 To NofTrees
        objRecRlt.AddNew
        objRecRlt.Fields("TreeNo") = aryTreeNo(i)
        objRecRlt.Fields("DBH") = aryDBH(i)
        objRecRlt.Fields("NofLogs") = aryNLogs(i)
        objRecRlt.Fields("BA") = aryBA(i)
        objRecRlt.Fields("Vol") = aryVol(i)
        objRecRlt.Update
    Next i

    MsgBox "Results were saved!"

End Sub

'For command button 3
Private Sub Command3_Click()

    Unload Me

End Sub

'Function to calculate BA
Private Function CalBA(dbh As Variant) As Variant

    CalBA = 0.005454154 * dbh * dbh

End Function

'Function to calculate the volume
Private Function CalVol(d As Variant, l As Variant) As Variant

        CalVol = ((0.55743 * l ^ 2 + 41.51275 * l - 29.37337) + _
            (2.78043 - 0.04516 * l ^ 2 - 8.77272 * l) * d + _
            (0.04177 - 0.01578 * l ^ 2 + 0.59042 * l) * d ^ 2)

End Function
```

(8) Run your application

- Click “Compute” button
- Click “Save” button
- Go back MS Access to check tblResult table in your database

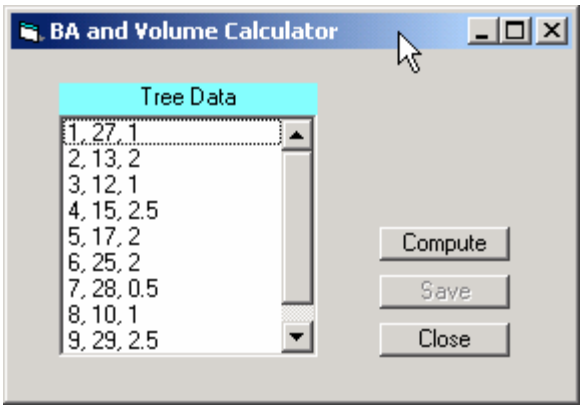

Figure 5. Running the application.