

## Worksheet 2 - Database Management in Visual Basic

### Exercise:

1. **Copy** the file Library2000.mdb from Samples\Infosys\bco3150\VBWkShop2 to a new folder called Library on your floppy disk.

- ☐ tblBooks = @ISBN + CatNum + Title + Author + PurchasePrice + DatePurchased
- ☐ tblCategories = @CatNum + CatName

2. **Start a new VB project** named Library.vbp and **create the form** as shown below -

- ☐ Open **VB** and add a blank form to an exe project - Use **Project/Add Form**.
- ☐ The **form** should be named frmBooks in the Properties box.
- ☐ The textboxes, command buttons, labels for the heading and the Date should be named as shown below.

Note that in order to simplify the code the frames and command buttons should be placed on the form in a strict order:

1. Draw cmdSave and cmdCancel on the form first. Use the Format/Order menu to Send To The Back
2. a) Then draw **fraNavigation**.  
b) Draw cmdFirst, cmdNext, cmdPrevious, cmdLast and cmd BookSearch on fraNavigation.
3. a) Then draw **fraFunctions** on the form to cover cmdSave and cmdCancel.  
b) Draw cmdEdit, cmdDelete, cmdAdd, cmdStats, cmdPurchases and cmdExit on fraNavigation. They will cover cmdSave and cmdCancel.

The screenshot shows a Visual Basic form titled "Maintain Books". The form has a blue title bar and a grey background. At the top, there is a blue rectangular area with the word "Books" in white. Below this, there are several text boxes and labels. The labels are: "ISBN:", "Title:", "Author:", "Purchase Price:", "Date Purchased:", and "Category Number:". The text boxes are: "txtISBN", "txtTitle", "txtAuthor", "txtPrice", "txtDate", and "txtCatNum". At the bottom of the form, there are two frames. The left frame is titled "Navigation" and contains five buttons: "First", "Next", "Previous", "Last", and "Search". The right frame is titled "Functions" and contains six buttons: "Edit", "Delete", "Add", "Stats", "Purch's", and "Exit". Labels with leader lines point to various controls: "lblHeading" points to the "Books" label, "lblDate" points to the "Date Purchased:" label, "txtISBN" points to the ISBN text box, "txtTitle" points to the Title text box, "txtAuthor" points to the Author text box, "txtPrice" points to the Purchase Price text box, "txtDate" points to the Date Purchased text box, "txtCatNum" points to the Category Number text box, "fraNavigation" points to the Navigation frame, and "fraFunction" points to the Functions frame.

*Note that some of the controls on the form will serve no purpose at this stage. They will be given functionality over the following Workshop exercises, but it is desirable to include them in the design now.*

- ❑ **IMPORTANT.** Make sure that your project can use the ActiveX Data Objects (ADO) by checking the box in **Project / References** for the **Microsoft ActiveX Data Object Library 2.1 or 2.5 or 2.6 or 2.7**, whichever is the latest version
- ❑ Set the **StartUp** Object to **Sub Main** under **Project / Properties**.
- ❑ Create a Module using **Project/Add Module** and add the following procedures and declarations.

#### Option Explicit

```
'''Declare a New ADODB connection object variable  
Public gcnLibrary As New ADODB.Connection
```

#### Sub Main()

```
'''Set the connection object to the database once only in a project, assuming '''that  
it is in the same folder as the .vbp file - portability is ensured using '''ChDrive  
and ChDir. Check if it is a network drive, looking for \\  
If Left(App.Path, 2) <> "\\\" Then  
    ChDrive App.Path  
End If  
ChDir App.Path  
'''Open the connection to the database  
gcnLibrary.Open "Provider = Microsoft.Jet.OLEDB.4.0;Data Source = Library2000.mdb"  
frmBooks.Show 1  
End Sub
```

#### Sub CloseConnection()

```
gcnLibrary.Close  
Set gcnLibrary = Nothing  
End Sub
```

- ❑ Open the **frmBooks** code window by clicking on it in Project Explorer
- ❑ Add the following code to the **'General Declarations'** section of the form:

#### Option Explicit

```
'''Declare a module-level variable for the recordset which is to be used by more  
'''than one procedure in the form module.  
Private mrsBooks As New ADODB.Recordset
```

- ❑ Add the following code to the **Form\_Load** and **Form\_Activate** event procedures.

#### Private Sub Form\_Load()

```
'''Define the recordset. Note that pstBooks is only used in this procedure,  
'''whereas mrsBooks is used by a number of procedures.  
Dim pstBooksSQL As String  
pstBooksSQL = "SELECT * FROM tblBooks ORDER BY ISBN"  
mrsBooks.Open pstBooksSQL, gcnLibrary, adOpenStatic, adLockOptimistic, adCmdText  
frmBooks.WindowState = 2 'What does this mean? Look up the form's Properties.  
lblDate.Caption = Format(Now, "Long date")  
'''Display the first record, if there is one; show nothing if there isn't one.  
If mrsBooks.RecordCount > 0 Then  
    Call DisplayData  
Else  
    'Call ClearData 'include this line now. The "" will be removed later.  
End If  
End Sub
```

#### Private Sub Form\_Activate()

```
''' Set the focus to the exit button - this can't done on Form_Load.  
cmdExit.SetFocus
```

**End Sub**

- ❑ Write a new procedure named **'DisplayData'** as a 'Sub' procedure.

**Sub DisplayData()**

*'\*\*There are 3 ways to write the field's name in a recordset as shown below -  
'\*\*Which of the following 3 ways would you use?*

```
txtISBN.Text = mrsBooks("ISBN")  
txtTitle.Text = mrsBooks![Title]  
txtAuthor.Text = mrsBooks!Author  
txtPrice.Text = Format(mrsBooks!PurchasePrice, "Currency")  
txtDate.Text = Format(mrsBooks!DatePurchased, "Short Date")  
txtCatNum.Text = "" & mrsBooks!CatNum    'What does the "" & do?
```

**End Sub**

- ❑ Next, add the code to respond to the click of the Exit button as an Event procedure

**Private Sub cmdExit\_Click()**

*'\*\*Close the recordset and database before exiting.*

```
mrsBooks.Close  
Set mrsBooks = Nothing  
Call CloseConnection  
Unload frmBooks
```

**End Sub**

- ❑ You should now save and run your project.
- ❑ It should display the first record in your database and it should close correctly.
- ❑ No other functions have been coded yet

- ❑ Now add the following code to the click event of the Next navigation button:

**Private Sub cmdNext\_Click()**

*'\*\*Move the record pointer to the next record if not already at the End Of the  
'\*\*File.*

```
mrsBooks.MoveNext  
If mrsBooks.EOF Then  
    MsgBox "There are no more Book records.", vbCritical, "End of table"  
    mrsBooks.MoveLast
```

```
Else  
    Call DisplayData
```

**End If**

**End Sub**

- ❑ Write the event procedure for the click of cmdPrev. It will be similar to the above.

- ❑ To add a new record to the database there are 2 things that need to occur:
  - 1. The user indicates that a new record is to be added, so the textboxes are cleared.
    - ❑ The cmdAdd button does this when clicked by calling a sub procedure named 'ClearData'. [Activate the Call line of code in Form\_Load()]
  - 2. The user types in the data and indicates that the record should be saved to the database. The cmdSave button does this when it is clicked.

**Private Sub cmdAdd\_Click()**

*'\*\*Clear all text boxes and change the status of the command buttons.*

```
Call ClearData
fraNavigation.Visible = False
fraFunctions.Visible = False
End Sub
```

#### **Sub ClearData()**

```
''' This procedure places an empty string into the textboxes.
txtISBN.Text = ""
txtTitle.Text = ""
txtAuthor.Text = ""
txtPrice.Text = ""
txtDate.Text = ""
txtCatNum.Text = ""
txtISBN.SetFocus '''Move the cursor to the ISBN field.
```

**End Sub**

#### **Private Sub cmdSave\_Click()**

```
'''Move the record-pointer to the blank record at the end of the recordset,
'''take data from the screen to the recordset, and then update the database.
Dim pinResponse As Integer
''' compare the use of With ....End With to the code in the DisplayData procedure.
With mrsBooks
    .AddNew
    !ISBN = txtISBN.Text
    !Title = txtTitle.Text
    !Author = txtAuthor.Text
    !PurchasePrice = txtPrice.Text
    !DatePurchased = txtDate.Text
    !CatNum = txtCatNum.Text
    .Update
    '''Get the revised / updated recordset into the mrsBooks variable.
    .Requery
End With
'''Tell the user that the record has been saved and find out what to do next
pinResponse = MsgBox("The added book has been saved. Do you want to add  
another one?", vbYesNo, "Record Saved")
If pinResponse = vbYes Then
    Call ClearData
Else
    Call DisplayData
    '''Set command buttons back to their original state.
    fraNavigation.Visible = True
    fraFunctions.Visible = True
End If
End Sub
```

#### **Private Sub cmdCancel\_Click()**

```
''' Move back to the start of the recordset
mrsBooks.MoveFirst
''' Display the record
Call DisplayData
'''Set the command buttons back to their original state
fraNavigation.Visible = True
fraFunctions.Visible = True
End Sub
```

### Additional Requirements:

1. Identify the repetition of a series of lines of code. How might **sub procedures** be used to reduce the amount of code in this program – similar to DisplayData() and ClearData(). Write them and replace the redundant code with Calls. One might be to hide/show the command buttons.
2. Using similar code to the navigation buttons' code above you could add functionality to the buttons to move directly to the **first** and **last** records in the recordset. Use the **.MoveFirst** and **.MoveLast** methods of the recordset object. Of course there is no reason to check for the EOF or BOF in these event procedures.
3. **As there are a number of different states or modes that this form has (View, Add, Edit, Delete) it is essential to know which one the form is currently in – eg. whether to send the cursor to cmdExit when the control gets focus.**
  - a. Create a form level variable called mstMode and use it so that the program can find out which mode the form is in at any point in time.
4. **There is no Edit or Delete or Print function available on the form.**
  - a. **Edit** is the same as Add except that the .AddNew line of code is omitted.
  - b. **Delete** requires the recordset's method **.Delete** (without Update)
  - c. In both cases make sure that the record-pointer has not moved.
5. **Data-entry controls (eg. textboxes ) have not been locked when in 'View' mode.**
  - a. Do this on Form\_Load(), and on the GotFocus event for each textbox use **.SetFocus** to move focus directly to cmdExit.
  - b. When cmdAdd is clicked don't forget to unlock the controls.
6. **The heading does not indicate that 'Add' mode has been entered when cmdAdd is clicked. –**
  - a. Fix this.

### Also

- ☐ **The user cannot yet search for a record except by using the Next and Previous buttons. – This will be rectified in WorkShop 4.**
- ☐ **There is no validation of the data that has been entered when in 'Add' mode. – This will be rectified in WorkShop 6.**
- ☐ **There is no checking for blank textboxes when in 'Add' mode. - This will be rectified in WorkShop 6.**
- ☐ **There are more forms to be included in the Project.**
- ☐ **etc.**