

MỤC LỤC

Dùng Đồ Họa (Phần II)	3
Dùng Đồ Họa (Phần III)	12
Cơ sở dữ liệu (Database)	23
Dùng Control Data	35
Lập trình với kỹ thuật DAO	46
Lập trình với ADO (phần I)	58

Dùng Đồ Họa (Phần II)

In trên màn ảnh

VB6 có **method Print** cho ta in thẳng trên Form, PictureBox hay Printer. Ba loại control này được coi như những khung vải mà họa sĩ vẽ lên.

Bạn hãy khởi động một chương trình VB6 mới. Đặt lên form một PictureBox tên Picture1 và một button tên CmdPrintTenLines với Caption **Print Ten Lines**. DoubleClick lên button này và viết code dưới đây:

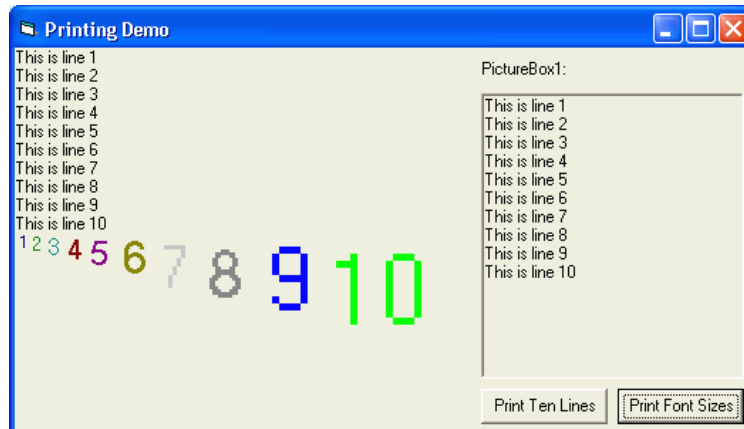
```
Private Sub CmdPrintTenLines_Click()  
    Dim i As Integer  
    ' String variable used for display  
    Dim strLine As String  
    ' Write 10 lines, one under the other  
    For i = 1 To 10  
        strLine = "This is line " & CStr(i)  
        Me.Print strLine ' Print on Form  
        Picture1.Print strLine ' Print on Picture1  
    Next  
End Sub
```

Bạn hãy chạy thử program rồi click nút **Print Ten Lines**. Trong trường hợp này ta dùng default Font và Color để in 10 hàng. Sau mỗi Print, chương trình tự động xuống hàng.

Kế đó, thêm một button tên CmdPrintFontSizes với Caption **Print Font Sizes**. DoubleClick lên button này và viết code dưới đây:

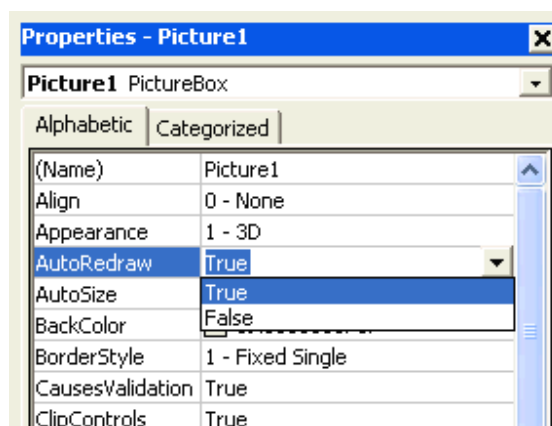
```
Private Sub CmdPrintFontSizes_Click()  
    Dim i As Integer  
    ' Print numbers 1 to 10, one after the other on the same line  
    For i = 1 To 10  
        ' Define Font size  
        Me.Font.Size = Me.Font.Size + i  
        ' Define Color using Function QBColor  
        Me.ForeColor = QBColor(i)  
        ' Print without moving to next line. Note the semicolon ";"  
        Me.Print Str(i);  
    Next  
End Sub
```

Trong Sub CmdPrintFontSizes_Click, ta thay đổi cỡ kiểu chữ để cho các con số được in ra lớn lên dần dần và thay đổi màu của các con số bằng cách dùng **function QBColor**. Để in các con số liên tục không xuống hàng ta dùng method Print với semicolon (;). Bạn hãy chạy chương trình lại. Click nút **Print Ten Lines** rồi click nút **Print Font Sizes**, kết quả sẽ giống như dưới đây:



Bây giờ bạn thử **minimize** cửa sổ của chương trình, kể đó restore nó lại kích thước cũ. Bạn sẽ thấy các hàng ta in lúc nãy không còn trong form hay PictureBox nữa.

Lý do là khi ta Print lên form hay PictureBox, các hình ấy được vẽ trong graphic địa phương chứ không được VB6 kể là một phần của cửa sổ. Muốn tránh trở ngại này ta phải dặn VB6 nhớ vẽ lại bằng cách set **property AutoRedraw** của form và PictureBox ra **True**.



Hệ thống tọa độ

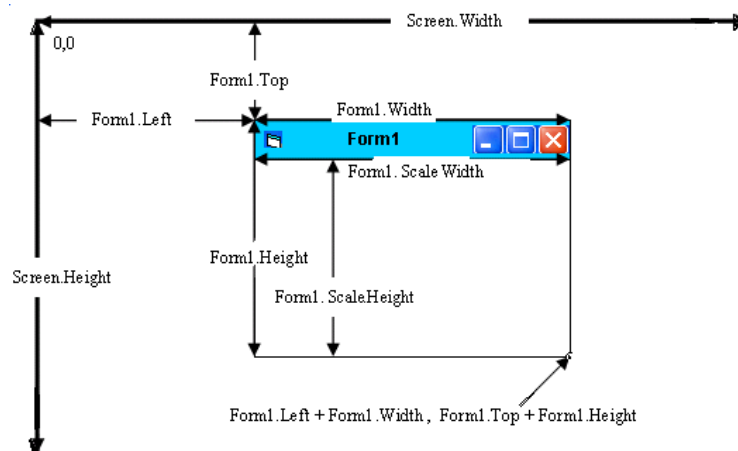
Khi đặt một Object hay vẽ một cái gì lên màn ảnh (screen) hay form .v.v.. ta cần phải chỉ định Object ấy nằm chỗ nào kể từ (with reference to) cái góc Trên Trái (Top Left) của màn ảnh hay form.

Cái góc Trên Trái là Trung tâm tọa độ của screen hay form. Ở đó tọa độ X và Y đều bằng 0, ta viết là **0,0**. Nếu ta đi lần qua phải theo chiều rộng của screen thì tọa độ X tăng lên. Nếu ta đi dọc xuống dưới theo chiều cao của screen thì tọa độ của Y tăng lên.

Kể đến là đơn vị đo lường ta dùng để biểu diễn khoảng cách. Trong bài trước ta đã nói đến độ mịn của màn ảnh (screen resolution) dựa vào **pixel**. Ta có thể dùng đơn vị pixel để nói một Object có tọa độ X và Y mỗi chiều bao nhiêu pixels tính từ trung tâm tọa độ.

Như thế, ngay cả trên cùng một màn ảnh khi ta tăng độ mịn nó lên thì một Object đã được đặt lên màn ảnh theo đơn vị pixel sẽ xích qua trái và lên trên một ít vì kích thước một pixel bây giờ nhỏ hơn lúc trước một chút.

Hình dưới đây minh họa các kích thước của màn ảnh và Form.

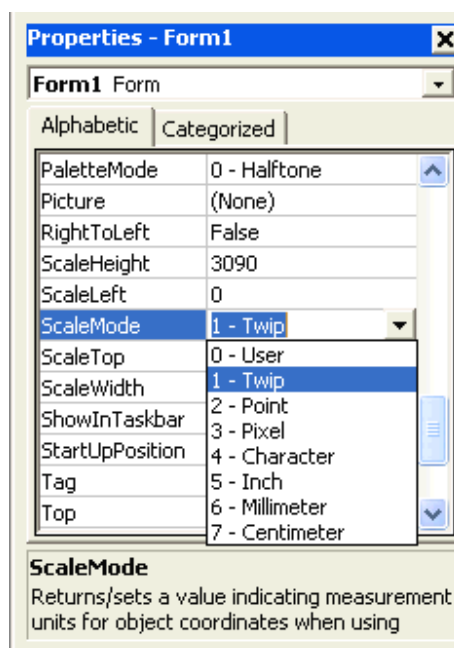


Điểm cần biết là có những phần như **title bar** và **border** của một form ta không thể vẽ lên được. Do đó diện tích còn lại của form được gọi là **Client Area**. Chiều rộng và chiều cao của Client Area được gọi là **ScaleWidth** và **ScaleHeight**.

Nếu muốn khoảng cách từ một Object đến trung tâm tọa độ, hay kích thước của chính Object, không hề thay đổi dù ta có tăng, giảm độ mịn của màn ảnh hay in hình ra printer (thí dụ ta muốn nó luôn luôn dài 5cm chẳng hạn) thì ta dùng hệ thống tọa độ theo đơn vị **twips** của form.

Twips là Default Coordinate System của VB6. Trong hệ thống này mỗi điểm là tương đương với 1/567 centimeter. Do đó, nếu bạn vẽ một đường dài 567 twips nó sẽ hiển thị dài 1cm trên màn ảnh, và khi bạn in nó ra, nó cũng dài 1cm trên giấy. Tức là độ dài thật của Object không tùy thuộc vào loại màn ảnh (độ mịn cao hay thấp) hay printer. Người ta nói nó là **Device independent** coordinate system (Hệ thống tọa độ độc lập với dụng cụ). Nói một cách khác Twips cho ta thật sự **what you see is what you get (WYSIWYG - thấy sao có vậy)**, rất thích hợp với Desktop publishing.

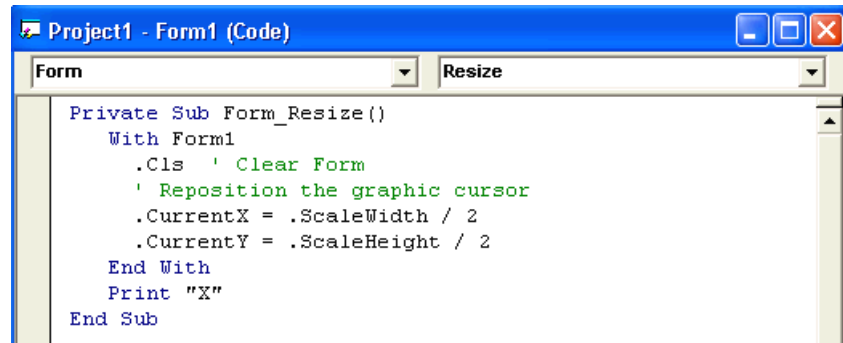
Bạn có thể thay đổi hệ thống tọa độ của một form bằng cách edit **property ScaleMode** qua cửa sổ Properties như sau:



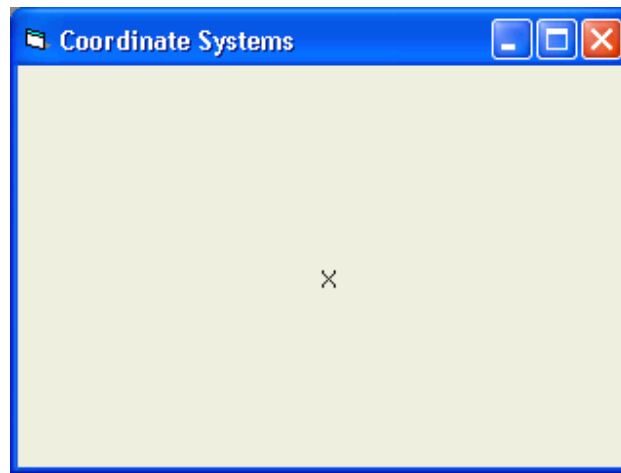
Ghi chú: Thay đổi trị số ScaleMode không có hiệu lực ngay mà chỉ ảnh hưởng những gì được thiết kế sau đó.

Giống như khi ta Edit Text trong Notepad, Text Cursor (thanh | chớp chớp) là vị trí hiện tại, nơi sẽ hiển thị cái chữ ta đánh sắp tới, trong graphic ta có một Cursor vô hình, nơi sẽ hiển thị cái gì ta sắp **Print**. Ta chỉ định vị trí của graphic cursor ấy bằng cách cho trị số của **CurrentX** và **CurrentY**.

Bạn hãy khởi động một dự án VB6 mới và viết code cho **Event Resize** của form chính như sau:



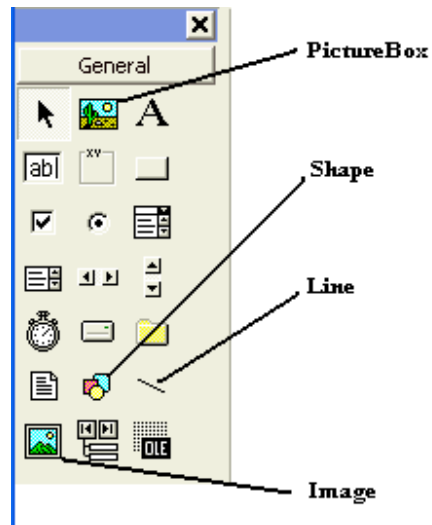
Thử chạy chương trình và Resize form. Mỗi khi bạn Resize form, chữ **X** sẽ được dời đến vị trí khoảng chính giữa của Client Area của form.



Dùng Graphics

Đã có một chút căn bản về graphics của VB6, bây giờ ta có thể đặt những graphics lên form. Có hai cách để làm chuyện ấy:

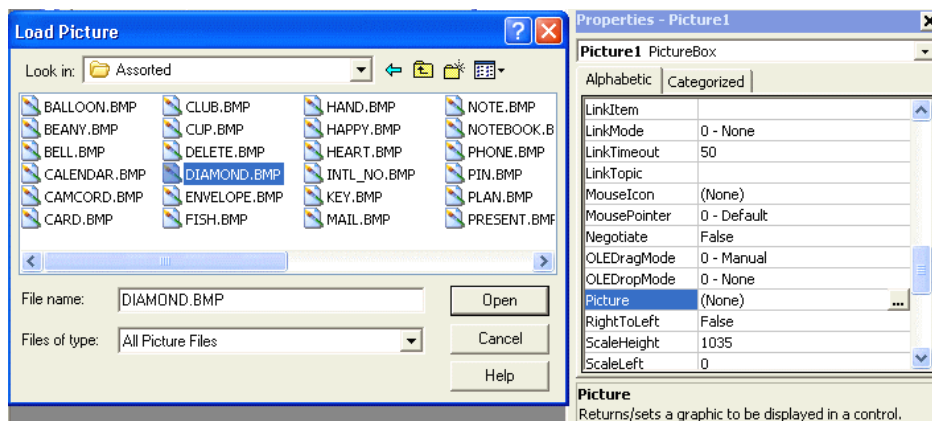
- Dùng **Graphical Controls**: Ta có **PictureBox** và **Image** có thể chứa hình ảnh. Trong khi **Line** và **Shape** có thể vẽ đường thẳng hay các hình chữ nhật, tròn .v.v.. trên form, lúc thiết kế.
- Dùng **Graphics Methods**: Đây là những mệnh lệnh cho ta vẽ trực tiếp lên form lúc run-time. Các mệnh lệnh VB6 cho ta là **Cls**, **Pset**, **Point**, **Line** và **Circle**.



Tùy theo hoàn cảnh, bạn có thể lựa chọn cách nào tiện dụng.

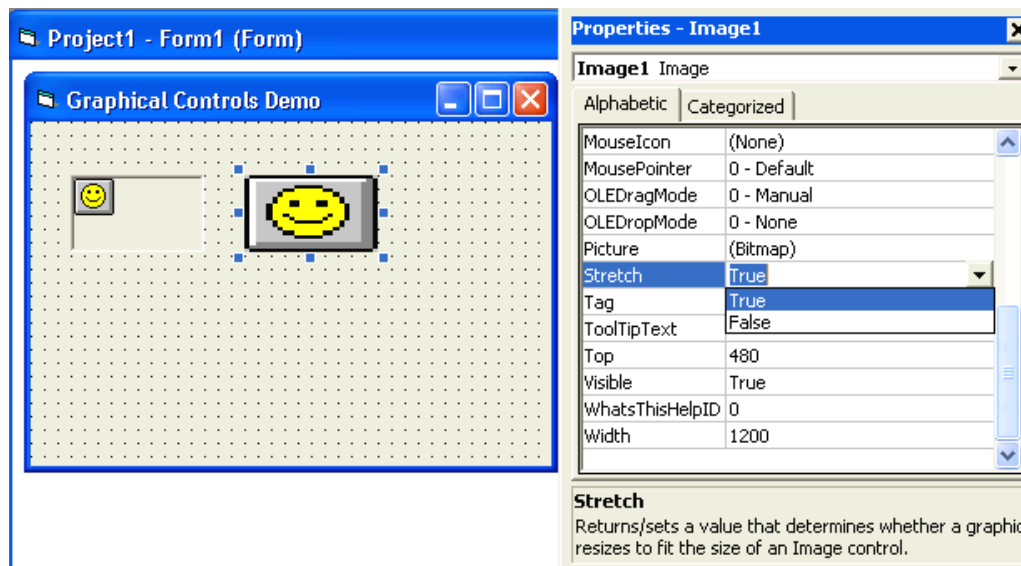
PictureBox và Image

Dùng PictureBox hay Image là cách dễ nhất để hiển thị một graphic trong form. Lúc thiết kế, bạn có thể đánh thẳng tên của graphic vào **property Picture** trong cửa sổ Properties. Form cũng nhận property Picture. Bạn cũng có thể click lên bên phải chữ property Picture để browse và chọn một graphic, thường là Bitmap hay Icon.



Sự khác biệt chính giữa Image và PictureBox là Image có **property Stretch** mà ta có thể set thành True để kéo giãn graphic ra cho chiếm trọn diện tích của Image. Image là một graphic control **lightweight** (nhẹ ký), tức là nó không đòi hỏi nhiều memory và chạy nhanh hơn PictureBox. Lý do là PictureBox là một container, tức là nó có thể chứa các controls khác. Ngoài ra, PictureBox cũng cho phép ta vẽ lên trên nó giống như trên form.

Trong hình dưới đây, trong lúc thiết kế ta đặt một PictureBox và một Image cùng một cỡ lên cùng một form. Kể đó ta assign cùng một picture hình **happy.bmp** cho cả hai. Riêng với Image, ta set property **Stretch** của nó ra **True**.



Chỉ định hình ảnh lúc run-time

Trong lúc program đang chạy, ta có thể thay đổi hình ảnh chứa trong PictureBox hay Image bằng cách dùng **Function LoadPicture**. Nhớ là ta không thể assign trực tiếp vào Property Picture của hai graphical controls này. Lý do là Property Picture chỉ là một cách thân thiện cho ta chỉ định một graphic trong lúc thiết kế. Khi một hình ảnh đã được chỉ định rồi, VB6 chứa cả hình ấy vào file có cùng tên với file của form nhưng với extension **.frx**. Tức là nếu tên của form là **Form1** thì graphic của Property Picture được chứa chung với các graphics khác của form trong file **Form1.frx**.

Do đó, vì VB6 program chứa luôn graphic chung với nó, ta không cần phải nhắc đến tên của graphic file khi dùng hay deploy, tức là không cần đính kèm tên graphic file trong Setup file cho người ta install. Dưới đây là code mẫu để lúc run-time ta load một graphic tên **sad.bmp** nằm trong Subfolder tên **images** của App.path vào Image control tên Image1.

```
Private Sub CmdLoad_Click()
    Dim LocalDir As String
    ' Assign Folder where program resides to LocalDir
    LocalDir = App.Path
    ' Append right backslash if last character is not "\"
    If Right(LocalDir, 1) <> "\" Then
        LocalDir = LocalDir & "\"
    End If
    Image1.Picture = LoadPicture(LocalDir & "images\sad.bmp")
End Sub
```

```

End If
' Load graphic "sad.bmp" from SubFolder "images" into Image1
Image1.Picture = LoadPicture(LocalDir & "images\sad.bmp")
End Sub

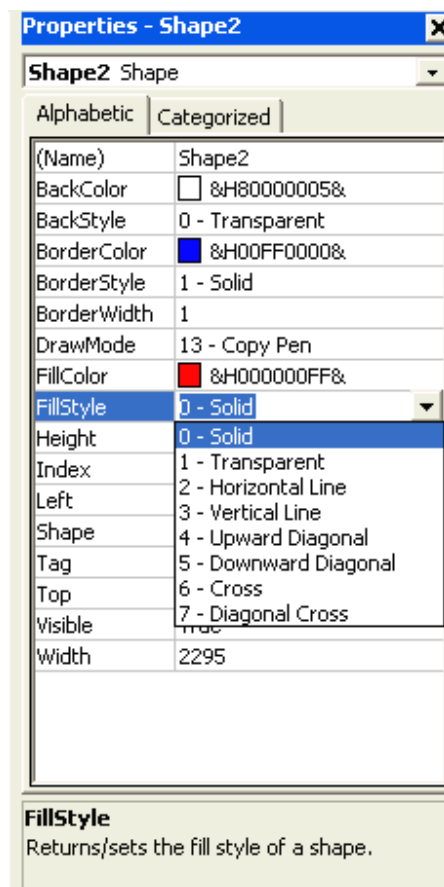
```

Dĩ nhiên, nếu ta muốn load graphic lúc run-time thì phải cung cấp graphic file riêng.

Control Shape

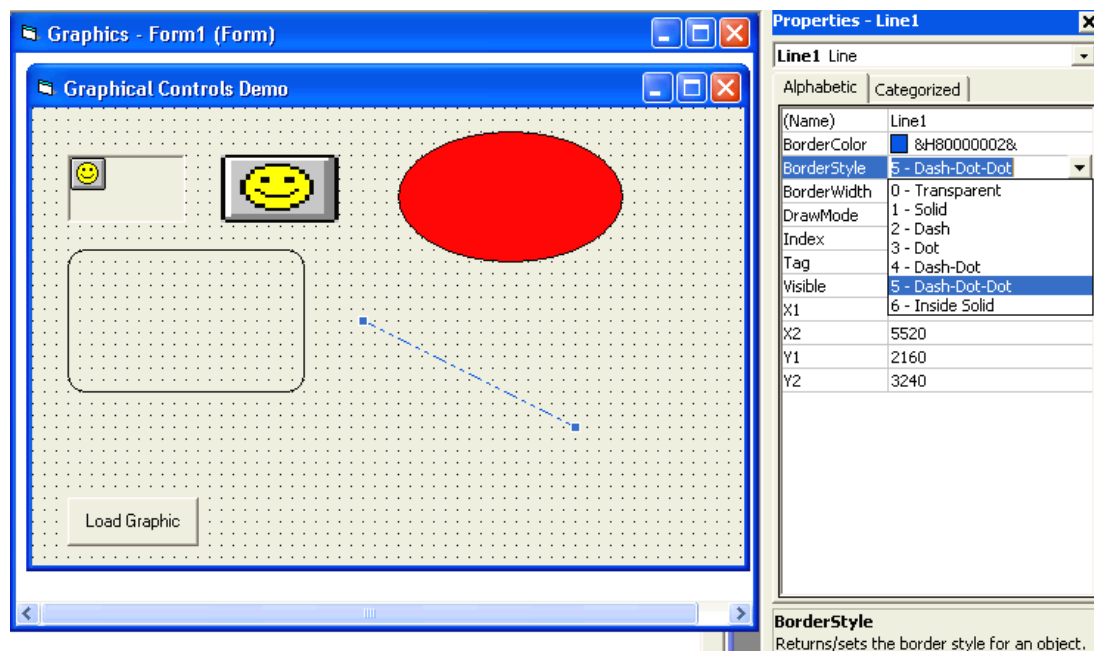
Control Shape cho phép bạn vẽ những hình đơn giản như đường thẳng, hộp, vòng tròn trên form, lúc thiết kế. Sau khi DoubleClick lên control Shape trong Toolbox để thêm một control Shape vào form, bạn chọn loại Shape của nó từ cửa sổ Properties rồi nằm vào một góc của Shape trên form drag lớn nhỏ tùy ý.

Muốn sơn bên trong một Shape, bạn chọn màu từ **property FillColor**. Property FillColor cũng giống như BackColor của các controls khác, nhưng nó chỉ có hiệu lực khi bạn cho **property FillStyle** một trị số khác hơn là **1- Transparent** (trong suốt), thí dụ như **0- Solid** (dày đặc).



Control Line

Tương tự với các properties Fill của Shape, đối với **Line** bạn có các **properties BorderColor, BorderStyle và BorderWidth**. Border color chỉ định màu của chính đường thẳng, BorderStyle để cho bạn lựa đường liên tục hay gạch chấm, và BorderWidth để làm cho đường dày to hơn. Các properties này cũng áp dụng cho chu vi (đường bao quanh) của các hình chữ nhật, tròn .v.v.



Dùng Đồ Họa (Phần III)

Graphics Methods

Trong khi các Graphical Controls như Shape, Line cho ta vẽ hình lúc thiết kế thì **Graphics Methods** cho ta vẽ những thứ ấy lúc run-time. Ta cũng có thể chấm từng đốm (pixel) hay copy cả một Picture từ chỗ này đến chỗ khác.

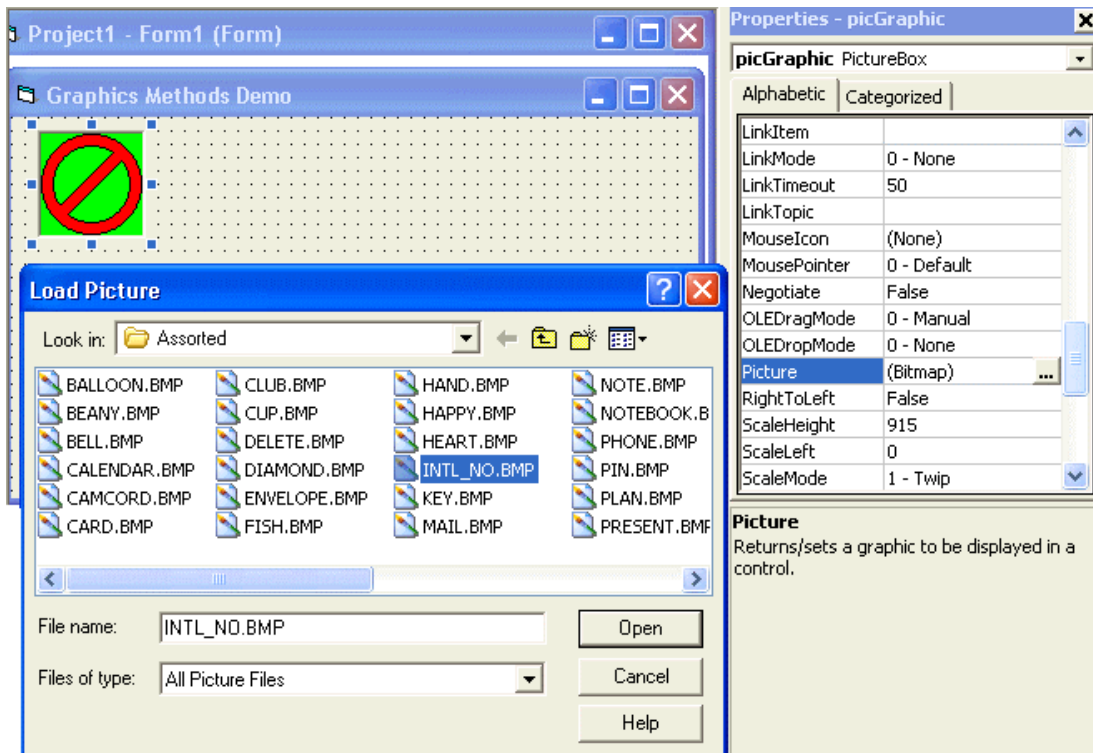
Chỉ cần một chút kinh nghiệm bạn có thể làm hoạt họa (animation) hay tạo visual effects tuyệt diệu mà không cần phải đụng đến **Windows API (Application Programming Interface)** để dùng **Function BitBlt**.

Method PaintPicture

Method PaintPicture cho phép bạn copy rất nhanh một khối dữ kiện đồ họa, nói nôm na là một khu vực trong một hình graphic trên form, PictureBox hay Printer đến một nơi khác. Thí dụ bạn copy một hình từ chỗ này đến chỗ khác trong form, hay từ form/PictureBox ra Printer Object để một chốc sau bạn in nó ra.

Bạn hãy khởi động một dự án VB6 mới và DoubleClick lên PictureBox Icon trong Toolbox để đặt một PictureBox lên form. Đặt tên PictureBox ấy là **picGraphic** và set property Visible của nó ra False để ta không thấy nó lúc run-time.

Bây giờ load một hình vào property Picture của picGraphic bằng cách Browse một Bitmap file từ cửa sổ Properties. Ở đây ta chọn **INTL_NO.BMP** từ folder **\Program Files\Microsoft Visual Studio\Common\Graphics\Bitmaps\Assorted**



Trong chương trình này ta muốn để khi đè nút trái của Mouse xuống và di chuyển Mouse cursor thì khi cursor đi đến đâu, hình INTL_NO được vẽ đến đó.

Ta sẽ dùng một **Flag** để đánh dấu nút-trái-của-Mouse-Down, đặt tên là **flgMouseDown**. Khi nhận được **Event MouseDown** ta set flgMouseDown thành **True**, và khi nhận được **Event MouseUp** ta reset flgMouseDown thành **False**. Mỗi lần nhận được Event MouseMove thì nếu flgMouseDown là True ta sẽ PaintPicture INTL_NO.

Để xóa background của form, ta thêm một button tên **CmdClearForm** để chạy graphic **method Cls**. Dưới đây là liệt kê code mẫu:

```
' Flag that indicates that the Mouse's left button is depressed
Dim flgMouseDown As Boolean

Private Sub Form_Load()
    ' Initialise flgMouseDown to False
    flgMouseDown = False
End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Set Flag flgMouseDown
    flgMouseDown = True
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As
```

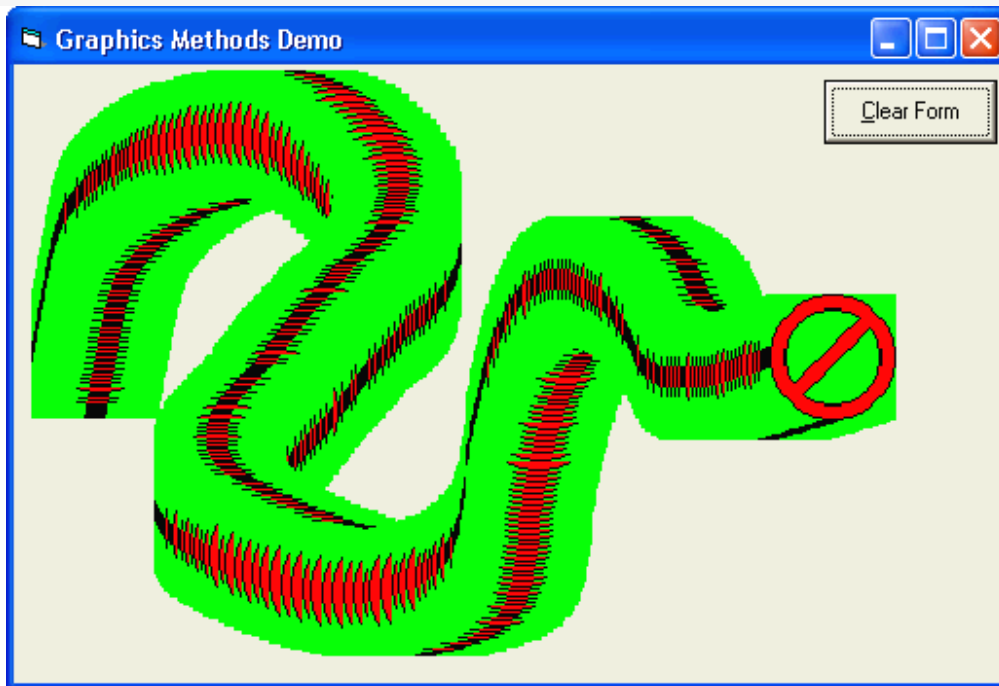
```

Single, Y As Single)
    ' Paint picGraphic if flgMouseDown is True
    If flgMouseDown Then
        ' Paint full-size picGraphic at Mouse cursor location
        PaintPicture picGraphic.Picture, X, Y, picGraphic.Width,
picGraphic.Height
    End If
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)
    ' Reset Flag flgMouseDown
    flgMouseDown = False
End Sub

Private Sub CmdClearForm_Click()
    ' Clear the form
    Cls
End Sub

```



Lưu ý là bạn phải declare variable **flgMouseDown** bên ngoài các Subs để mọi Sub đều thấy và có thể dùng nó. Muốn biết thêm chi tiết về cách dùng **method PaintPicture**, trong VB6 IDE DoubleClick lên chữ PaintPicture trong code editor để highlight chữ ấy rồi bấm nút **F1**.

Method PSet

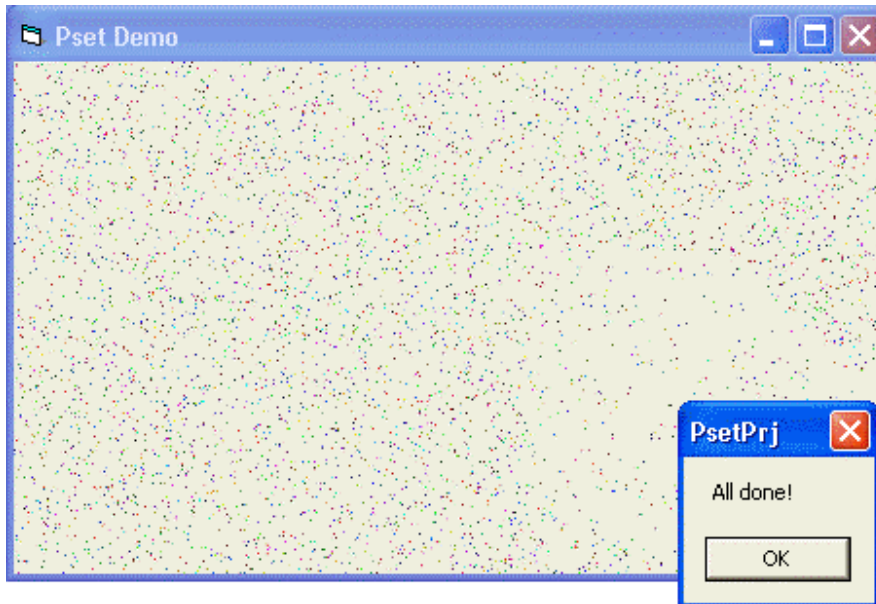
Ta dùng **method PSet** (đến từ chữ **Point Set**) để vẽ một pixel lên form. Ta cần cho biết PSet ở đâu và với màu gì, tức là ta cho nó tọa độ X,Y của pixel và một màu tính từ **function RGB**.

Dưới đây là code để vẽ pixels đủ màu lên form một cách bất chừng (randomly) về vị trí và màu sắc khi user clicks lên form chính:

```
Private Sub Form Click()  
    Dim i As Integer  
    ' Variables for pixel coordinates  
    Dim iXCoord As Integer  
    Dim iYCoord As Integer  
    ' Variable for primary colours  
    Dim iRed As Integer  
    Dim iGreen As Integer  
    Dim iBlue As Integer  
    ' Start the Random number generation  
    Randomize  
    ' Plot 2000 dots randomly  
    For i = 1 To 2000  
        ' get a random XCoord.  
        ' Note that Rnd(1) returns a real number between 0 and 1, eg:  
0.384  
        iXCoord = Int(Rnd(1) * ScaleWidth)  
        ' get a random YCoord.  
        iYCoord = Int(Rnd(1) * ScaleHeight)  
        ' Get a random number between 0 and 255 for each primary colour  
        iRed = Int(Rnd(1) * 255)  
        iGreen = Int(Rnd(1) * 255)  
        iBlue = Int(Rnd(1) * 255)  
        ' Plot the pixel at iXCoord,iYCoord  
        PSet (iXCoord, iYCoord), RGB(iRed, iGreen, iBlue)  
    Next  
    MsgBox ("All done!")  
End Sub
```

Trong thí dụ trên ta dùng **method Randomize** để generate sẵn trong bộ nhớ các con số real bất chừng từ 0 đến 0.999. Sau đó mỗi lần ta gọi **Function Rnd(1)** là nó sẽ trả về một con số real lấy bất chừng từ bộ số do method Randomize generated. Do đó, **Rnd(1) * ScaleWidth** sẽ cho ta một con số real có trị số từ 0 đến ScaleWidth. Muốn đổi con số real đó ra Integer, ta dùng **Function Int**.

Khi khởi động chương trình và Click lên form ta sẽ có hình giống như dưới đây:



Mách nước: Để xóa một đốm bạn Pset lại tại chỗ ấy một đốm mới có cùng màu với BackColor của form.

Method Line

Method Line vẽ một đường thẳng từ một tọa độ này đến một tọa độ khác trong màu do ta chỉ định. Với hai methods PSet và Line ta có thể làm được rất nhiều chuyện. Thí dụ muốn cho một vật di động, ta xóa vật ấy bằng cách vẽ lại nó với cùng màu của BackColor của form, rồi vẽ vật ấy ở vị trí mới. Muốn vẽ một đa giác như tam giác hay chữ nhật ta ráp nhiều đường thẳng lại với nhau, đầu của mỗi đường thẳng là cuối của đường thẳng vừa mới được vẽ trước. Muốn sơn Shade bên trong một hình chữ nhật ta dùng PSet..v.v.

Có ba cách để chỉ định tọa độ của hai đầu của một đường thẳng ta muốn vẽ:

1. Cho biết tọa độ của đầu và cuối đường thẳng:
thí dụ: **Line (50, 100)-(3000, 4000)**
Khi đường này được vẽ xong thì vị trí của graphic cursor có tọa độ là vị trí của cuối đường, tức là CurrentX=3000 và CurrentY=4000 trong trường hợp này.
2. Chỉ cho biết tọa độ cuối đường thẳng:
thí dụ: **Line -(3600, 4500), vbMagenta**
Trong trường hợp này vị trí của graphic cursor (CurrentX, CurrentY) được lấy làm tọa độ của đầu đường thẳng khi vẽ. Tức là nếu trước khi execute dòng code này CurrentX=3000 và CurrentY=4000 thì dòng code tương đương với:

Line (3000,4000)-(3600,4500), vbMagenta

3. Dùng chữ **Step** để nói sự khác biệt từ CurrentX và CurrentY:
thí dụ: **Line Step(400, 600)-Step(800, -500), vbGreen**
Nếu trước khi execute dòng code này CurrentX=3600 và CurrentY=4500 thì dòng code tương đương với:

Line (4000,5100)-(4800,4600), vbGreen

Trong thí dụ dưới đây, một hình tam giác được vẽ bằng hai cách coding khác nhau. Khi chạy program để thử, bạn hãy lần lượt click **Triangle METHOD I** và **Triangle METHOD II** để thấy cả hai cách vẽ đều y như nhau, chỉ khác màu thôi.

```
Private Sub CmdTrianI_Click()
    ' Drawing a black triangle: METHOD I
    Line (700, 500)-(2800, 2400)
    Line (2800, 2400)-(1800, 900)
    Line (1800, 900)-(700, 500)
End Sub

Private Sub CmdTrianII_Click()
    ' Drawing a red triangle: METHOD II
    ' Draw a red line from Location(700, 500) to Location (2800, 24000)
    Line (700, 500)-(2800, 2400), vbRed
    ' Draw a red line from Location(2800,2400) to Location (1800,900)
    Line -(1800, 900), vbRed
    ' Draw a red line from Location(1800,900) to Location (700,500)
    Line -(700, 500), vbRed
End Sub
```

Để vẽ một hình chữ nhật, cách tiện nhất là dùng Step như dưới đây:

```
Private Sub Rectangle(ByVal X1 As Integer, ByVal Y1 As Integer, ByVal
X2 As Integer, ByVal Y2 As Integer)
    ' Draw a rectangle
    Line (X1, Y1)-(X2, Y1)
    Line -(X2, Y2)
    Line -(X1, Y2)
    Line -(X1, Y1)
End Sub
```

Ta còn có thể vẽ một hình chữ nhật với bốn góc tròn như sau:

```
Private Sub RoundCornerRectangle(ByVal X1 As Integer, ByVal Y1 As
Integer, ByVal X2 As Integer, ByVal Y2 As Integer)
    Const Delta = 50
    ' Draw a rectangle with round corner
    Line (X1 + Delta, Y1)-(X2 - Delta, Y1)
    Line -Step(Delta, Delta)
    Line -(X2, Y2 - Delta)
```

```

Line -Step(-Delta, Delta)
Line -(X1 + Delta, Y2)
Line -Step(-Delta, -Delta)
Line -(X1, Y1 + Delta)
Line -Step(Delta, -Delta)
End Sub

```

Ta cũng có thể sơn Shade bên trong hình chữ nhật bằng cách dùng method PSet để chấm các đốm cách nhau chừng 50 pixels như sau:

```

Private Sub Shade(ByVal X1 As Integer, ByVal Y1 As Integer, ByVal X2 As Integer, ByVal Y2 As Integer)
    ' Shade a roundcorner rectangle by plotting dots using method Pset
    Const Delta = 50
    Dim i As Integer
    Dim j As Integer
    ' Make sure that X1 is less than X2
    ' Swap values of X1, X2 if X1 > X2
    If X2 < X1 Then
        Temp = X1
        X1 = X2
        X2 = Temp
    End If
    ' Make sure that Y1 is less than Y2
    ' Swap values of Y1, Y2 if Y1 > Y2
    If Y2 < Y1 Then
        Temp = Y1
        Y1 = Y2
        Y2 = Temp
    End If
    ' Plotting dots inside the rectangle at 50 pixels apart
    For i = X1 + Delta To X2 - Delta Step 50
        For j = Y1 + Delta To Y2 - Delta Step 50
            PSet (i, j)
        Next
    Next
End Sub

```

Muốn Shade đậm hơn, bạn có thể chấm các đốm gần nhau hơn, thí dụ cho cách nhau 30 pixels thay vì 50 pixels. Có một cách khác là tăng trị số của **DrawWidth**, độ dày của đường vẽ hay đốm.

Bây giờ phối hợp cách vẽ hình chữ nhật với method Shade nói trên và **method Print** ta có thể viết chữ bên trong một khung màu nhạt như sau:

```

Private Sub CmdDrawFrame_Click()
    Dim X1 As Integer
    Dim Y1 As Integer
    Dim X2 As Integer
    Dim Y2 As Integer
    ' Initialise Coordinates of rectangle

```

```

X1 = 4200: Y1 = 1000
X2 = 6200: Y2 = 2000
' Draw a roundcorner rectangle
RoundCornerRectangle X1, Y1, X2, Y2
' Shade the rectangle
Shade X1, Y1, X2, Y2
' Position cursor to Print some text
CurrentX = X1 + 50
CurrentY = Y1 + 50
' Define Font Size
Font.Size = 18
' Print the text at cursor location
Print "Hello there!"
End Sub

```

Khi chạy chương trình này và click tất cả các buttons trên form, bạn sẽ có hình dưới đây:



Hãy nhớ set property AutoDraw của form ra True để các graphic chương trình vẽ không bị mất khi user minimises form.

Bạn cũng có thể dùng những kỹ thuật nói trên với Object Printer để in các mẫu giấy điền chi tiết.

Method Circle

Ta dùng **Method Circle** để vẽ hình tròn, hình bầu dục và đường cung, với bên trong trống rỗng hay được sơn đầy bằng một màu ta chỉ định. Ta phải cho biết tọa độ của tâm điểm vòng tròn và bán kính của nó.

Bạn hãy khởi động một dự án VB6 mới, đặt lên form một button với tên **frmCircle** và caption **Circle & Lines**. DoubleClick lên button ấy và viết code sau đây:

```
Private Sub CmdCircleLine_Click()  
    ' Draw a circle centered at 2000,1500 with radius equal 800  
    Circle (2000, 1500), 800  
    ' Draw a vertical line from center  
    Line (2000, 1500)-Step(0, 800)  
    ' Draw a horizontal line from center  
    Line (2000, 1500)-Step(800, 0)  
End Sub
```

Bây giờ hãy đặt lên form một button khác tên **CmdArc** và caption **Draw Arc**. Thay vì vẽ nguyên một vòng tròn, ta sẽ chỉ vẽ một hình vòng cung bằng màu đỏ.

Để chỉ định rằng ta sẽ vẽ từ vị trí nào trên vòng tròn đến vị trí nào khác, thí dụ từ 45độ đến 230độ, ta cần phải đổi degree ra đơn vị Radian bằng cách dùng **Function Rads** như sau:

```
Private Function Rads(ByVal Degree As Single) As Single  
    ' Convert Degrees to Radian  
    Const PI = 22 / 7  
    Rads = Degree / 180 * PI  
End Function
```

Vòng cung luôn luôn được vẽ ngược chiều kim đồng hồ. Dưới đây là code để vẽ một đường vòng cung màu đỏ bán kính 800, tâm điểm ở (4000, 2000), từ 45độ đến 230độ:

```
Private Sub CmdArc_Click()  
    Circle (4000, 2000), 800, vbRed, Rads(45), Rads(230)  
End Sub
```

Ta có thể cho sơn bên trong các hình tròn, hay Pie Slices (một phần của hình tròn) bằng cách set **FillStyle** bằng 0 và chỉ định màu **FillColor**. Một Pie Slice là một vòng cung đóng kín bởi hai đường thẳng bán kính ở hai đầu. Muốn vẽ một Pie Slice ta đánh thêm dấu trừ ("-") trước hai trị số Radian, tức là dùng **-Rads(45), -Rads(230)** thay vì **Rads(45), Rads(230)**.

Dưới đây là code vẽ hai Pie Slices, có tâm điểm lệch nhau một tí, đồng thời thêm chú thích 87.5% và 12.5%.

```
Private Sub CmdPie_Click()  
    FillStyle = 0 ' Fill inside any closed shaped  
    FillColor = vbYellow
```

```

' Draw a Pie Slice from 90deg to 45deg in Yellow
Circle (3000, 4000), 800, , -Rads(90), -Rads(45)
' Position the graphic cursor to Print some text
CurrentX = 2800: CurrentY = 4400
Print "87.5%"
FillColor = vbBlue
' Draw a Pie Slice from 45deg to 90deg in Blue
Circle (3050, 3900), 800, , -Rads(45), -Rads(90)
' Position the graphic cursor to Print some text
CurrentX = 3400: CurrentY = 3000
Print "12.5%"
FillStyle = 1 ' No fill
End Sub

```

Cách dùng cuối cùng của method Circle là để vẽ một **hình bầu dục (Ellipse)**. Vẽ hình bầu dục giống như vẽ một hình tròn nhưng ta cần cho thêm một parameter gọi là **Aspect**. Aspect là sự liên hệ giữa bán kính vertical và bán kính horizontal. Thí dụ nếu Aspect=2 thì chiều cao của hình bầu dục gấp đôi chiều ngang, ngược lại, nếu Aspect=0.5 thì chiều ngang sẽ gấp đôi chiều cao.

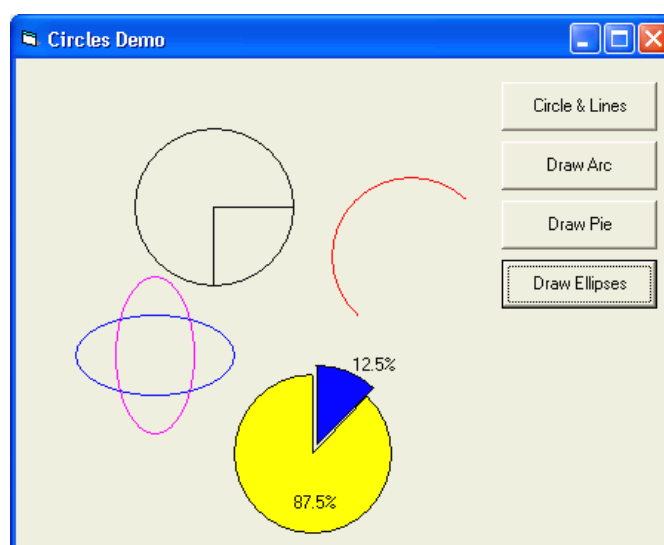
Dưới đây là code ta dùng để vẽ hai hình bầu dục cùng cỡ, một cái màu tím nằm thẳng đứng và một cái màu xanh nằm ngang.

```

Private Sub CmdEllipse_Click()
    Circle (1400, 3000), 800, vbMagenta, , , 2
    Circle (1400, 3000), 800, vbBlue, , , 0.5
End Sub

```

Nếu bạn khởi động chương trình và click cả bốn buttons bạn sẽ thấy hình sau đây:



Property DrawMode

Thông thường khi ta vẽ, trị số default của **property DrawMode** là **13- Copy Pen**. Có một trị số DrawMode rất thích hợp cho áp dụng hoạt họa là **7- Xor Pen**. Muốn xóa một hình vừa vẽ xong ta chỉ cần vẽ lại hình ấy trong DrawMode Xor Pen, không cần biết trước đó background như thế nào, nó sẽ hiện ra trở lại.

Cơ sở dữ liệu (Database)

Table, Record và Field

Nói đến cơ sở dữ liệu, ta lập tức nghĩ đến SQLServer, Access hay Oracle .v.v., những nơi chứa rất nhiều dữ liệu để ta có thể lưu trữ hay lấy chúng ra một cách tiện lợi và nhanh chóng. Hầu hết các chương trình ta viết đều có truy cập cơ sở dữ liệu, và ta dùng nó như một công cụ để làm việc với rất nhiều dữ liệu trong khi tập trung vào việc lập trình phần giao diện với người dùng (users).

Do đó ta cần có một kiến thức căn bản về kiến trúc của cơ sở dữ liệu để hiểu lý do tạo sao ta thiết kế hay truy cập nó theo những cách nhất định.

Ta sẽ dùng Access Database **biblio.mdb**, nằm ở **C:\Program Files\Microsoft Visual Studio\VB98\biblio.mdb** để minh họa các ý niệm cần biết về cơ sở dữ liệu.

Trong database này có 4 **tables**: **Authors** (tác giả), **Publishers** (nhà xuất bản), **Titles** (đề mục) và **Title Author**.

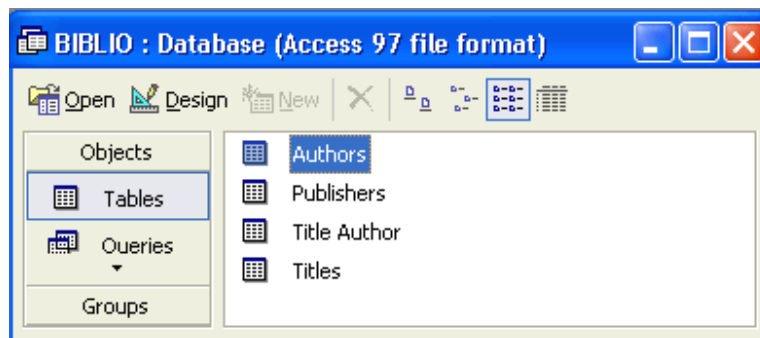
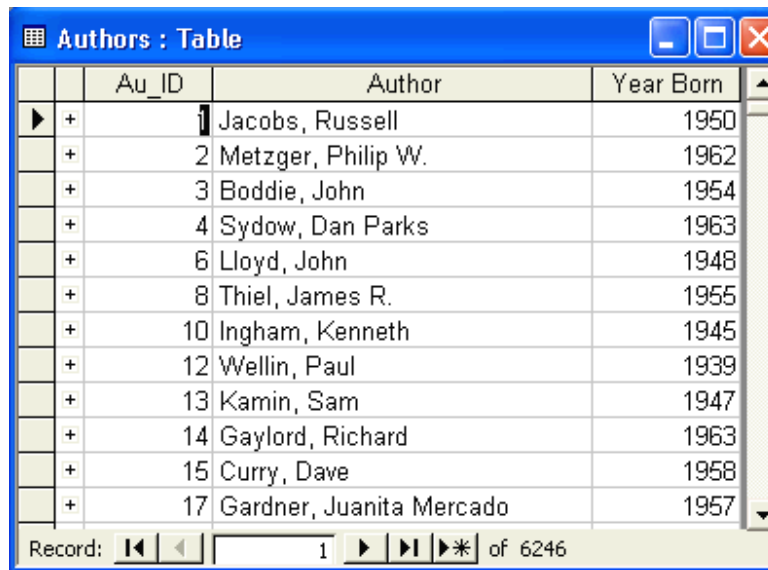


Table Authors chứa nhiều **records**. Mỗi record trong table Authors chứa 3 **fields**: **Au_ID**, **Author** và **Year Born** (năm sanh). Ta có thể trình bày Table Authors dưới dạng một spreadsheet như sau:



	Au_ID	Author	Year Born
▶ +	1	Jacobs, Russell	1950
+	2	Metzger, Philip W.	1962
+	3	Boddie, John	1954
+	4	Sydow, Dan Parks	1963
+	6	Lloyd, John	1948
+	8	Thiel, James R.	1955
+	10	Ingham, Kenneth	1945
+	12	Wellin, Paul	1939
+	13	Kamin, Sam	1947
+	14	Gaylord, Richard	1963
+	15	Curry, Dave	1958
+	17	Gardner, Juanita Mercado	1957

Record: 1 of 6246

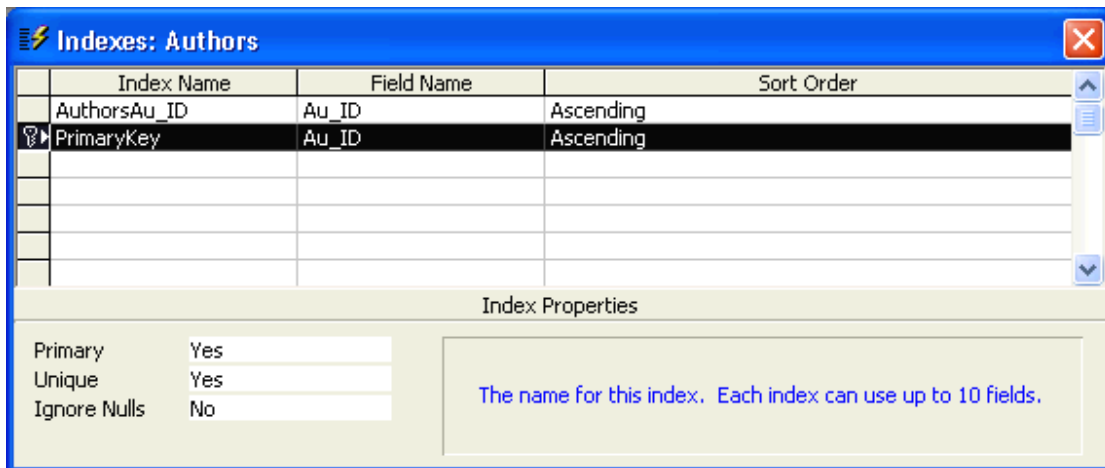
Vì cùng một field của các records hiển thị trong cùng một cột của spreadsheet, nên ta cũng nói đến một field như một **column** (cột). Và vì mỗi data record chiếm một row (hàng) của spreadsheet, nên có khi ta cũng nói đến một record như một **row**.

Thật tình mà nói, ta không cần phải có một computer để lưu trữ hay làm việc với một table như Authors này. Ta đã có thể dùng một hộp cật, trên mỗi cật ta ghi các chi tiết Au_ID, Author và Year Born của một Author. Như thế mỗi tấm cật tương đương với một record và nguyên cái hộp là tương đương với Table Authors.

Ta sẽ sắp các cật trong hộp theo thứ tự của số Au_ID để có thể truy cập record nhanh chóng khi biết Au_ID. Chỉ khổ một nỗi, nếu muốn biết có bao nhiêu tác giả, trong số 300 cật trong hộp, già hơn 50 tuổi thì phải mất vài phút mới có thể trả lời được. Database trong computer nhanh hơn một hệ thống bằng tay (Manual) là ở chỗ đó.

Primary Key và Index

Để tránh sự trùng hợp, thường thường có một field của record, thí dụ như Au_ID trong Table Authors, được dành ra để chứa một trị số độc đáo (unique). Tức là trong Table Authors chỉ có một record với field Au_ID có trị số ấy mà thôi. Ta gọi nó là **Primary Key**.

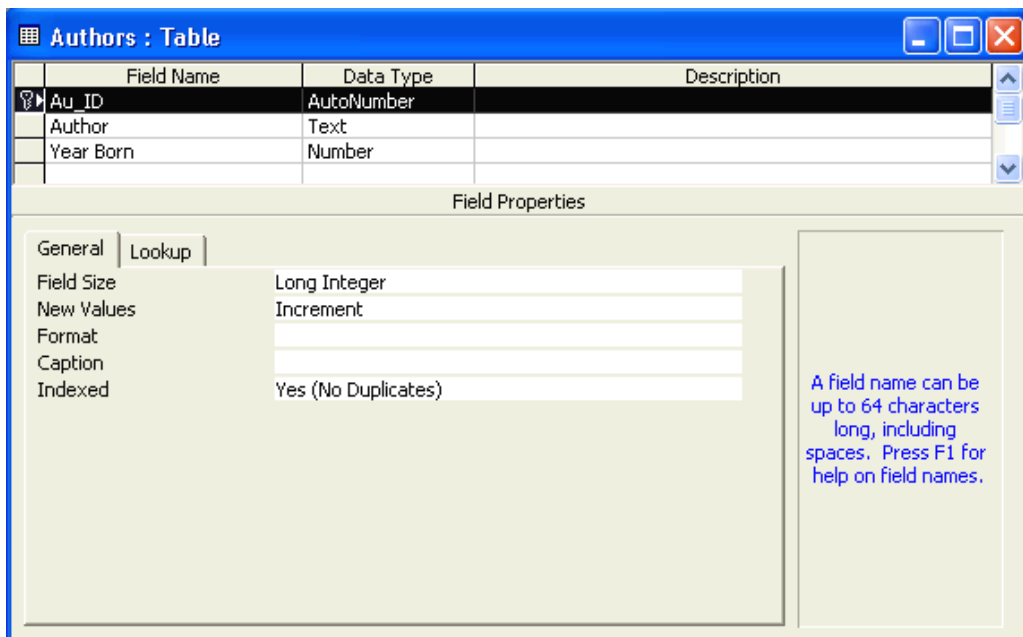


Không phải lúc nào ta cũng muốn truy cập một record Author dựa vào Au_ID. Nhiều khi ta muốn dùng chính tên của Author để truy cập, do đó ta cũng cần phải sort sẵn các records theo thứ tự alphabet. Ta cũng có thể hợp nhiều fields lại để sort các records. Thật ra, chính các records không cần phải được dời đi để nằm đúng vị trí thứ tự. Ta chỉ cần nhớ vị trí của nó ở đâu trong table là đủ rồi.

Cái field hay tập hợp của nhiều fields (thí dụ surname và firstname) để dùng vào việc sorting này được gọi là **Index** (ngón tay chỉ). Một Table có thể có một hay nhiều Index. Mỗi Index sẽ là một table nhỏ của những **pointers**, chứa vị trí của các records trong Table Authors. Nó giống như mục lục index ở cuối một cuốn sách chứa trang số để chỉ ta đến đúng phần ta muốn tìm trong quyển sách.

Khi thiết kế một Table ta chỉ định **Datatype** của mỗi field để có thể kiểm tra data cho vào có hợp lệ hay không. Các Datatypes thông dụng là Number, String (để chứa Text), Boolean (Yes/No), Currency (để chứa trị số tiền) và Date (để chứa date/time). Datatype Number lại gồm có nhiều loại datatypes về con số như Integer, Long (integer chiếm 32 bits), Single, Double, .v.v.

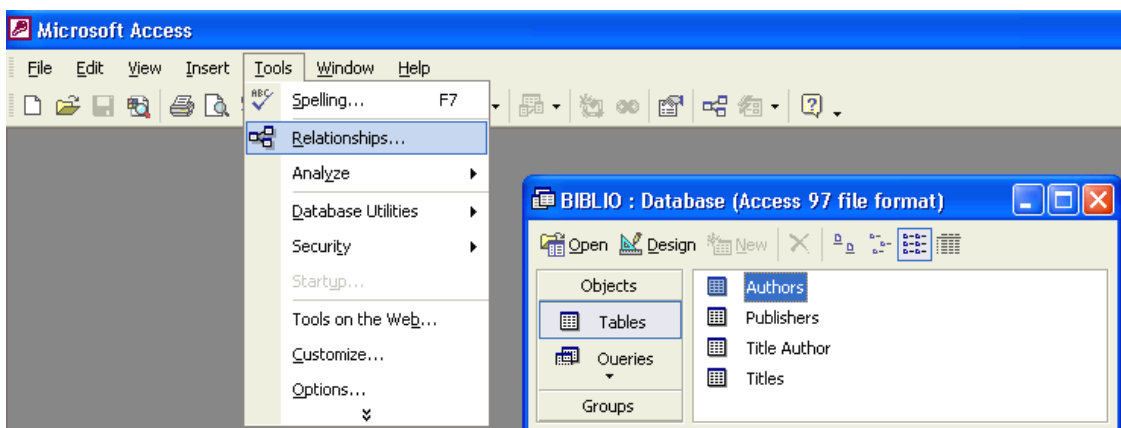
Dưới đây là Datatypes của các fields trong record Author:



Có loại Datatype đặc biệt tên là **AutoNumber**. Thật ra nó là Long nhưng trị số được phát sinh tự động mỗi khi ta thêm một record mới vào Table. Ta không làm gì hơn là phải chấp nhận con số ấy.

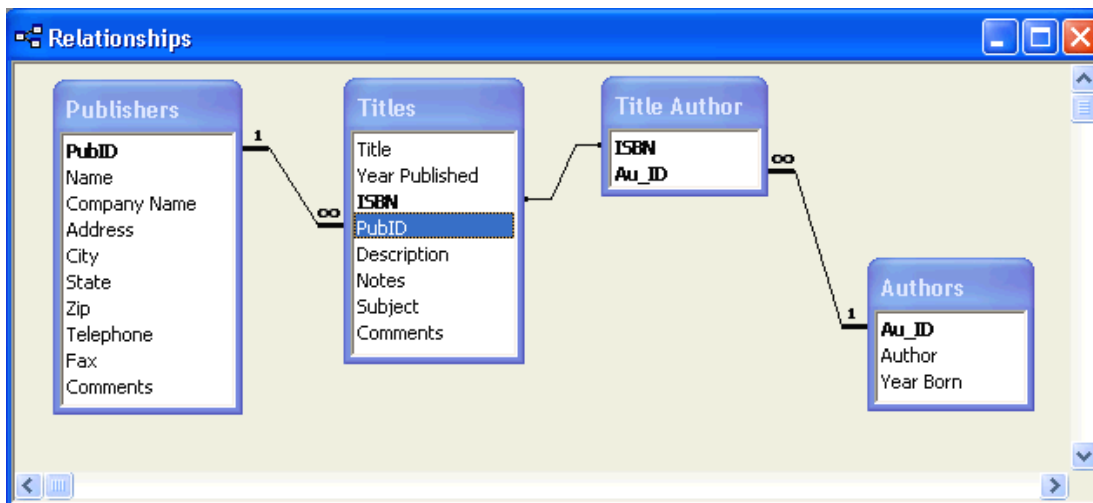
Relationship và Foreign Key

Bây giờ, nếu bạn đang chạy Microsoft Access để quan sát database biblio.mdb, bạn có thể dùng Menu Command **Tools | Relationships** như sau để xem sự liên hệ (relationships) giữa các tables.



Access sẽ hiển thị giao thoại Relationships, trong đó mỗi table có chứa tên các fields. Mỗi table lại có một hay hai sợi dây nối qua các tables khác. Mỗi sợi dây là một mối liên hệ (relationship), nó nối một field trong một table với một field có cùng tên trong table kia.

Thí dụ như giữa hai tables **Publishers** và **Titles** có mối liên hệ dựa trên field **PubID** (**P**ublisher **I**Dentification - số lý lịch của nhà xuất bản). Hơn nữa, nếu để ý bạn sẽ thấy ở đầu dây phía table Publishers có con số **1**, còn ở đầu dây bên phía table Titles có dấu vô cực (∞). Ta gọi mối liên hệ (**1- ∞**) là **one-to-many**, ý nói **một** nhà xuất bản có thể phát hành **nhều** đề mục sách/CD.



Tương tự như vậy, trong mối liên hệ one-to-many giữa table Authors và Title Author, ta thấy một tác giả (bên đầu có con số 1) có thể sáng tác nhiều tác phẩm được đại diện bởi các record Title Author.

Trong khi đó giữa hai tables Titles và Title Author, ta có một mối liên hệ **one-to-one**, tức là tương ứng với mỗi record Title chỉ có một record Title Author. Câu hỏi đặt ra là các mối liên hệ one-to-many có cái gì quan trọng.

Tưởng tượng khi ta làm việc với table Titles (tạm gọi là Tác phẩm), nhiều khi ta muốn biết chi tiết của nhà xuất bản của tác phẩm ấy. Thật ra ta đã có thể chứa chi tiết của nhà xuất bản của mỗi tác phẩm ngay trong table Titles. Tuy nhiên, làm như thế có điểm bất lợi là records của các tác phẩm có cùng nhà xuất bản sẽ chứa những dữ liệu giống nhau. Mỗi lần muốn sửa đổi chi tiết của một nhà xuất bản ta phải sửa chúng trong mỗi record Title thuộc nhà xuất bản ấy. Vì muốn chứa chi tiết của mỗi nhà xuất bản ở một chỗ duy nhất, tránh sự lặp lại, nên ta đã chứa chúng trong một table riêng, tức là table Publishers.

Nếu giả sử ta bắt đầu thiết kế database với Table Titles, rồi quyết định tách các chi tiết về nhà xuất bản để vào một table mới, tên Publishers, thì kỹ thuật ấy

được gọi là **normalization**. Nói một cách khác, normalization là thiết kế các tables trong database làm sao để mỗi loại mảnh dữ kiện (không phải là Key) chỉ xuất hiện ở một chỗ.

Trong mỗi liên hệ one-to-many giữa tables Publishers và Titles, field PubID là Primary Key trong table Publishers. Trong table Titles, field PubID được gọi là **Foreign Key**, có nghĩa rằng đây là Primary Key của một table lạ (foreign). Hay nói một cách khác, trong khi làm việc với table Titles, lúc nào cần chi tiết một nhà xuất bản, ta sẽ lấy chìa khóa lạ (Foreign Key) dùng làm Primary Key của Table Publishers để truy cập record ta muốn. Để ý là chính Table Titles có Primary Key **ISBN** của nó.

Relational Database

Một database có nhiều tables và hỗ trợ các liên hệ, nhất là one-to-many, được gọi là **Relational Database**. Khi thiết kế một database, ta sẽ tìm cách sắp đặt các dữ liệu từ thế giới thật bên ngoài vào trong các tables. Ta sẽ quyết định chọn các cột (columns/fields) nào, chọn Primary Key, Index và thiết lập các mối liên hệ, tức là đặt các Foreign Key ở đâu.

Các lợi ích

Trong số các lợi ích của một thiết kế Relational Database có:

- Sửa đổi dữ kiện, cho vào records mới hay delete (gạch bỏ) records có sẵn rất hiệu quả (nhanh).
- Truy cập dữ kiện, làm báo cáo (Reports) cũng rất hiệu quả.
- Vì dữ kiện được sắp đặt thứ tự và có quy củ nên ta có thể tin cậy tính tình của database (không có ba trộn, khi thì thế này, khi thì thế khác - giựt giựt).
- Vì hầu hết dữ kiện nằm trong database, thay vì trong chương trình ứng dụng, nên database tự có documentation (tài liệu cắt nghĩa).
- Dễ sửa đổi chính cấu trúc của các tables.

Integrity Rules (các quy luật liên chính)

Integrity Rules được dùng để nói về những qui luật cần phải tuân theo trong khi làm việc với database để đảm bảo là database còn tốt. Có hai loại quy luật: luật tổng quát (General Integrity Rules) và luật riêng cho database (Database-Specific Integrity Rules). Các luật riêng này thường tùy thuộc vào các quy luật về mậu dịch (Business Rules).

General Integrity Rules

Có hai quy luật liên chính liên hệ hoàn toàn vào database: Entity (bản thể) Integrity Rule và Referential (chỉ đến) Integrity Rule.

Entity Integrity Rule nói rằng **Primary Key** không thể thiếu được, tức là không thể có trị số **NULL**. Quy luật này xác nhận là vì mỗi Primary Key đưa đến một row độc đáo trong table, nên dĩ nhiên nó phải có một trị số đẳng hoàng.

Lưu ý là Primary Key có thể là một **Composite Key**, tức là tập hợp của một số keys (columns/fields), nên nhất định không có key nào trong số các columns là NULL được.

Referential Integrity Rule nói rằng database không thể chứa một Foreign Key mà không có Primary Key tương ứng của nó trong một table khác. Điều ấy hàm ý rằng:

- Ta không thể thêm một Row vào trong một Table với trị số Foreign Key trong Row ấy không tìm thấy trong danh sách Primary Key của table bên phía **one** (1) mà nó liên hệ.
- Nếu có thay đổi trị số của Primary Key của một Row hay delete một Row trong table bên phía **one** (1) thì ta không thể để các records trong table bên phía **many** (∞) chứa những rows trở thành mồ côi (orphans).

Nói chung, có ba nhiệm ý (options) ta có thể chọn khi thay đổi trị số của Primary Key của một Row hay delete một Row trong table bên phía **one** (1):

1. **Disallow** (không cho làm): Hoàn toàn không cho phép chuyện này xảy ra.
2. **Cascade** (ảnh hưởng dây chuyền): Nếu trị số Primary Key bị thay đổi thì trị số Foreign Key tương ứng trong các records của table bên phía **many** (∞) được thay đổi theo. Nếu Row chứa Primary Key bị deleted thì các records tương ứng trong table bên phía **many** (∞) bị deleted theo.
3. **Nullify** (cho thành NULL): Nếu Row chứa Primary Key bị deleted thì trị số Foreign Key tương ứng trong các records của table bên phía **many** (∞) được đổi thành NULL, để hàm ý đừng có đi tìm thêm chi tiết ở đâu cả.

Database-Specific Integrity Rules

Những quy luật liên chính nào khác không phải là Entity Integrity Rule hay Referential Integrity Rule thì được gọi là Database-Specific Integrity Rules. Những quy luật này dựa vào chính loại database và nhất là tùy thuộc vào các

quy luật về mậ dịch (Business Rules) ta dùng cho database, thí dụ như mỗi record về tiền lương của công nhân phải có một field Số Thuế (Tax Number) do sở Thuế Vụ phát hành cho công dân. Lưu ý là các quy luật này cũng quan trọng không kém các quy luật tổng quát về liên chính. Nếu ta không áp dụng các Database-Specific Integrity Rules nghiêm chỉnh thì database có thể bị hư và không còn dùng được.

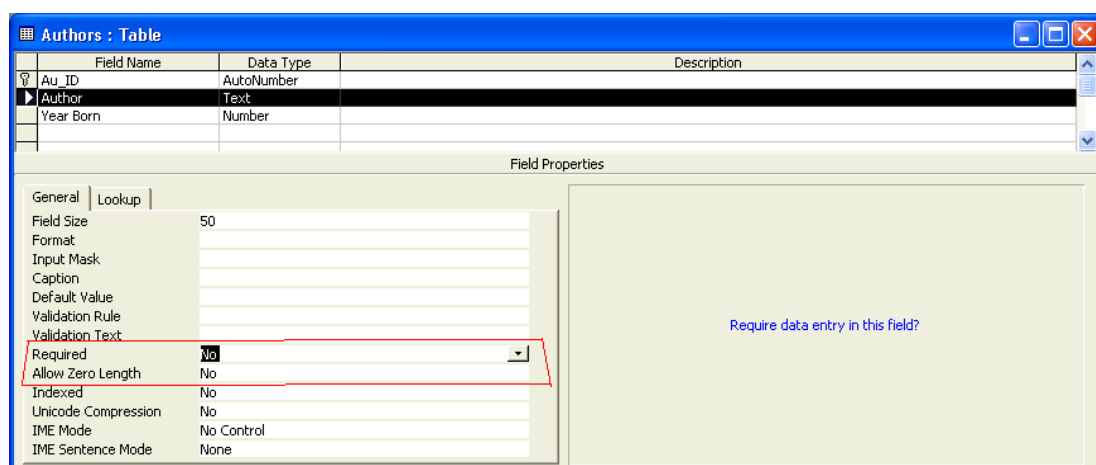
Microsoft Access Database Management System (MSAccess DBMS)

Microsoft Access Database Management System gồm có Database Engine và những công cụ đi chung để cung cấp cho users một môi trường làm việc thân thiện với database, như Database Design (thiết kế các tables và mối liên hệ), Data entry và báo cáo (reports). Kèm theo với Visual Basic 6.0 khi ta mua là một copy của Database Engine của MSAccess. Tên nó là **Jet Database Engine**, cái lõi của MSAccess DBMS. Các chương trình VB6 có thể truy cập database qua Jet Database Engine.

Nếu trên computer của bạn có cài sẵn MSAccess, thì bạn có thể dùng đó để thiết kế các tables của database hay cho data vào các tables.

Properties Required và Allow Zero Length

Khi thiết kế một table field, lưu ý property **Required** và nhất là property **Allow Zero Length** của Text. Nếu property **Required** của một field là **Yes** thì ta không thể update (viết) một record với field ấy có trị số NULL. Nếu một Text field có property **Allow Zero Length** là **No** thì ta không thể update một record khi field ấy chứa một empty string.

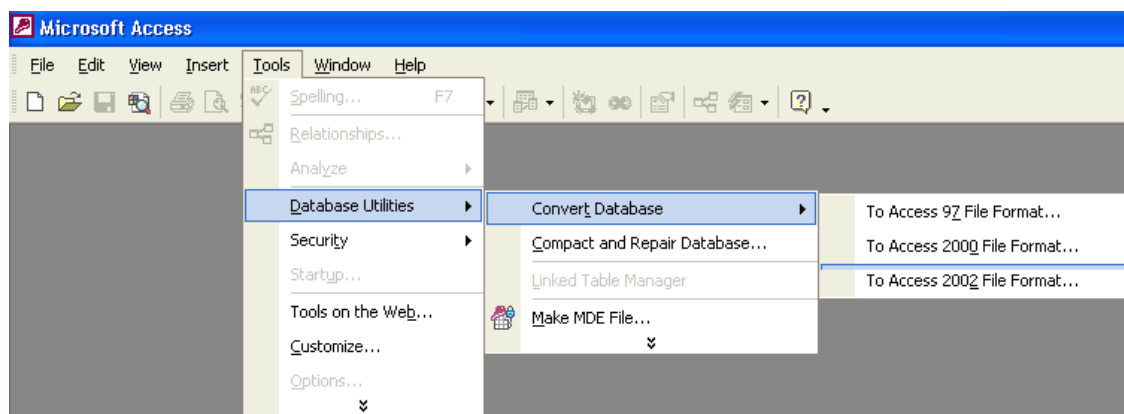


Khi ta tạo một record lần đầu, nếu không cho trị số của một field, thì field ấy có trị số là **NULL**. Thường thường, Visual Basic 6.0 không thích NULL value nên ta

phải thử một field với **Function IsNULL()** để đảm bảo nó không có trị số NULL trước khi làm việc với nó. Nếu IsNULL trả về trị số False thì ta có thể làm việc với field ấy. Nhớ là khi trị số NULL được dùng trong một expression, ngay cả khi chương trình không cho Error, kết quả cũng là NULL.

Làm việc với các versions khác nhau

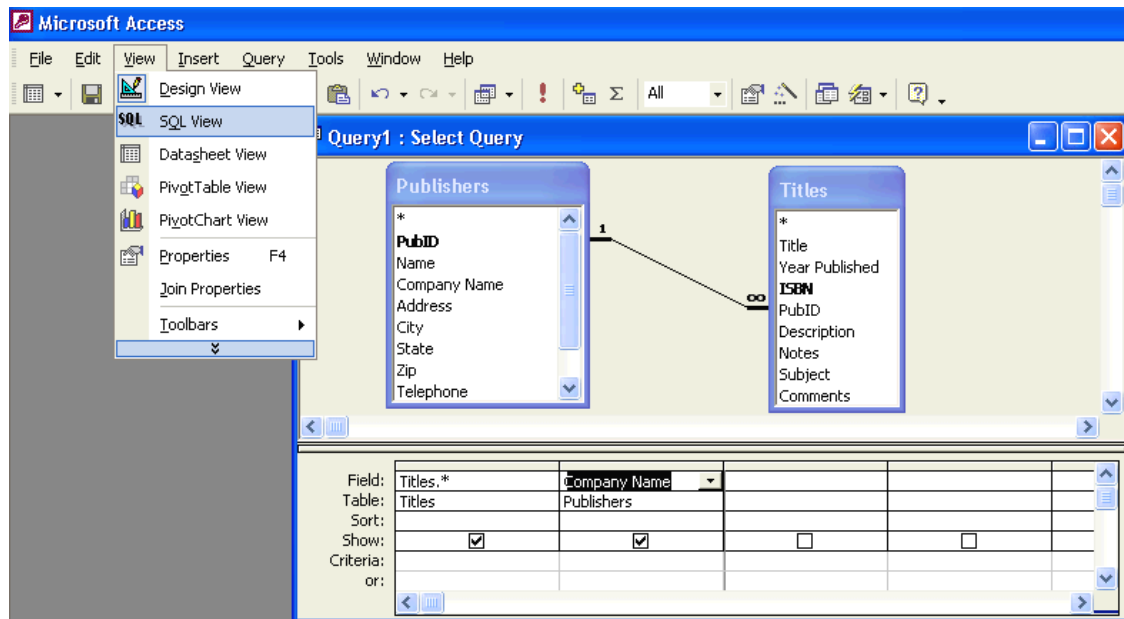
Nếu máy bạn đang chạy MSAccess2002 thì bạn có thể làm việc với Access database file version 97, 2000 và 2002. Nếu cần phải convert từ version này qua version khác, bạn có thể dùng Access DBMS Menu Command **Tools | Database Utilities | Convert Database | To Access 2002 File Format...** Nếu muốn giữ nguyên version, bạn có thể convert database qua File Format 2002 để sửa đổi, rồi sau đó convert trở lại File Format cũ.



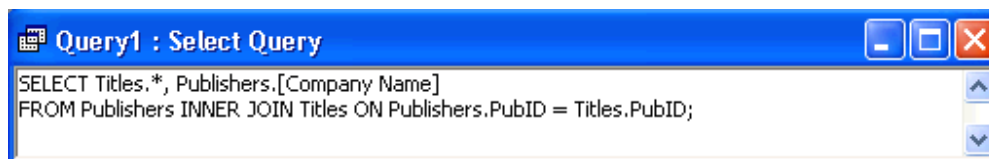
Access database file lớn lên rất nhanh, vì các records đã bị deleted vẫn còn nằm nguyên, nên mỗi tuần bạn nên nhớ nén nó lại để bỏ hết các records đã bị deleted bằng cách dùng Access DBMS Menu Command **Tools | Database Utilities | Compact and Repair Database...** hay dùng **function DBEngine.CompactDatabase** trong VB6.

Dùng Query để viết SQL

Một cách để truy cập database là dùng ngôn ngữ **Structured Query Language (SQL)** theo chuẩn do ISO/IEC phát hành năm 1992, gọi tắt là **SQL92**. Tất cả mọi database thông dụng đều hỗ trợ SQL, mặc dầu nhiều khi chúng còn cho thêm nhiều chức năng rất hay nhưng không nằm trong chuẩn. Các lệnh SQL thông dụng là **SELECT, UPDATE, INSERT** và **DELETE**. Ta có thể dùng phương tiện thiết kế Query của MSAccess để viết SQL. Sau khi thiết kế Query bằng cách drag drop các fields, bạn có thể dùng Menu Command **View | View SQL** như sau:

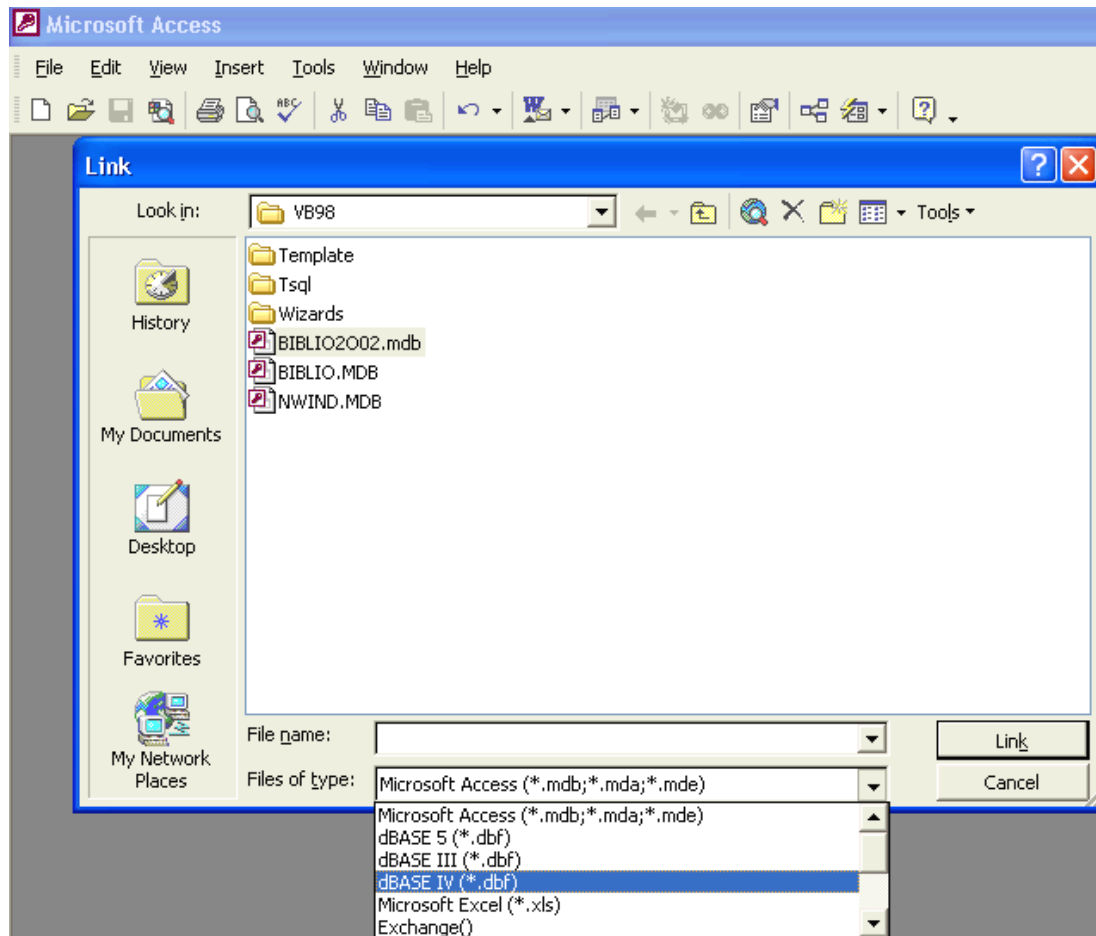


Tiếp theo đây là SQL statement của Query bên trên mà bạn có thể copy để paste vào trong code VB6:



Dùng Link Table để làm việc trực tiếp với database loại khác

Ta có thể dùng một database loại khác, như DBase, trực tiếp trong VB6 như dùng một Access database bình thường. Muốn thiết lập móc nối ấy, bạn dùng Menu Command **File | Get External Data | Link Tables...** rồi chọn loại DBase và chính file của table mà bạn muốn dùng để nhét nó vào Access database đang mở:



Database Server và một số ý niệm

Dù Jet Database Engine là một relational database rất tốt và hiệu năng, nó thuộc loại **File Based database**, tức là nó thụ động, không chạy một mình nhưng phải tùy thuộc vào chương trình dùng nó. File Based database không thích hợp với những ứng dụng có nhiều người dùng cùng một lúc.

Trong khi đó, một Database Server như **SQLServer** chạy riêng để phục vụ bất cứ chương trình khách (client) nào cần. Database Server thích hợp cho các ứng dụng có nhiều users vì chỉ có một mình nó chịu trách nhiệm truy cập dữ liệu cho mọi clients. Nó có thể chứa nhiều routines địa phương, gọi là **Stored Procedures**, để thực hiện các công tác client yêu cầu rất hiệu năng. Database Server thường có cách đối phó hữu hiệu khi có sự cố về phần cứng như đĩa hư hay cúp điện. Ngoài ra, Database Server có sẵn các phương tiện về an ninh và backup. Nó cũng có thêm các chức năng để dùng cho mạng.

Ngày nay ta thu thập dữ liệu dưới nhiều hình thức như Email, Word documents, Spreadsheet. Không nhất thiết dữ liệu luôn luôn được chứa dưới dạng table của

những records và không nhất thiết dữ liệu luôn luôn được lưu trữ trong một database đang hoạt động. Dù vậy, chúng vẫn được xem như database dưới mắt một chương trình ứng dụng. Do đó, ta dùng từ **Data Store** (Kho dữ liệu) thay thế cho database để nói đến nơi chứa dữ liệu. Và đối với chương trình tiêu thụ dữ liệu, ta nói đến **Data Source** (Nguồn dữ liệu) thay vì database.

Khi lập trình bằng VB6 để truy cập database, ta nhìn database một cách trừu tượng, tức là đầu nó là Access, DBase, SQLServer hay Oracle ta cũng xem như nhau. Nếu có thay đổi loại database bên dưới, cách lập trình của ta cũng không thay đổi bao nhiêu.

Trong tương lai, một **XML file** cũng có thể được xem như một database nhỏ. Nó có thể đứng một mình hay là một table trích ra từ một database chính huy. XML là một chuẩn mà ta có thể dùng để import/export dữ liệu với tất cả mọi loại database hỗ trợ XML. Ta có thể trao đổi dữ liệu trên mạng Internet dưới dạng XML. Ngoài ra, thay vì làm việc trực tiếp với một database lớn, ta có thể trích ra vài tables từ database ấy thành một XML file. Kế đó ta chỉ lập trình với XML file cho đến khi kết thúc sẽ hòa (merge/reconcile) XML file với database lớn. Nếu phần lớn các chương trình áp dụng được thiết kế để làm việc cách này, thì trong tương lai ta không cần một Database Server thật mạnh.

Dùng Control Data

Control Data

Từ VB5, Visual Basic cho lập trình viên một control để truy cập cơ sở dữ liệu, tên nó chỉ đơn sơ là **Data**. Như ta biết, có một cơ sở dữ liệu Microsoft gói kèm khi ta mua VB6 - đó là **Jet Database Engine**. Jet Database Engine là cái "**phòng máy**" của chính MS Access Database Management System.

Cho đến thời VB5, Microsoft cho ta ba kỹ thuật chính:

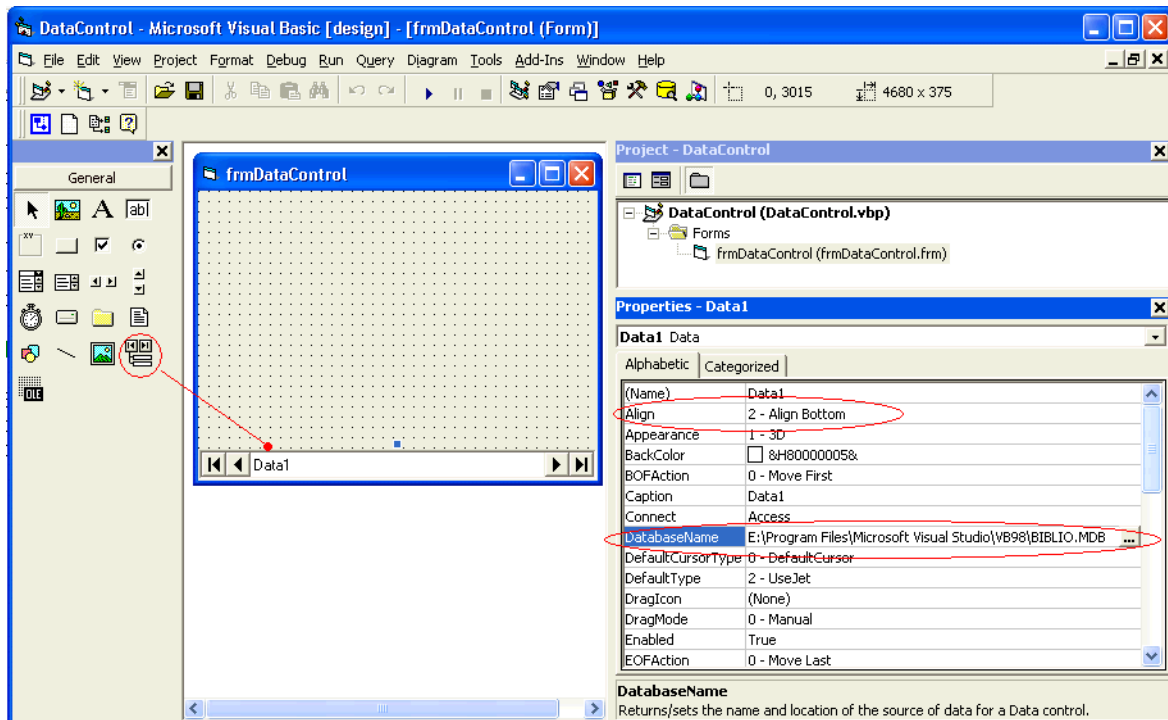
- **DAO (Data Access Objects)**: DAO là kỹ thuật bí truyền của Microsoft, chỉ để dùng với Jet Database Engine. Nó rất dễ dùng, hiệu năng và tiện, nhưng bị giới hạn trong phạm vi MS Access. Dầu vậy, nó rất thịnh hành vì có lợi ích thực tiễn.
- **ODBC (Open Database Connectivity)**: ODBC được thiết kế để cho phép users nối với đủ loại databases mà chỉ dùng một method duy nhất. Điều này cắt bớt gánh nặng cho lập trình viên, để chỉ cần học một kỹ thuật lập trình duy nhất mà có thể làm việc với bất cứ loại database nào. Nhất là khi sau này nếu cần phải thay đổi loại database, như nâng cấp từ Access lên SQLServer chẳng hạn, thì sự sửa đổi về coding rất ít. Khi dùng ODBC chung với DAO, ta có thể cho Access Database nối với các databases khác. Có một bất lợi của ODBC là nó rắc rối.
- **RDO (Remote Data Object)**: Một trong những lý do chính để RDO được thiết kế là giải quyết khó khăn về sự rắc rối của ODBC. Cách lập trình với RDO đơn giản như DAO, nhưng thật ra nó dùng ODBC nên cho phép users nối với nhiều databases. Tuy nhiên, RDO không được thịnh hành lắm.

VB6 tiếp tục hỗ trợ các kỹ thuật nói trên, và cho thêm một kỹ thuật truy cập database mới, rất quan trọng, đó là **ADO (ActiveX Data Objects)**. Trong một bài tới ta sẽ học về ADO với những ưu điểm của nó. Tuy nhiên, vì DAO rất đơn giản và hiệu năng nên ta vẫn có thể tiếp tục dùng nó rất hữu hiệu trong hầu hết các áp dụng. Do đó bài này và bài kế sẽ tập trung vào những kỹ thuật lập trình phổ biến với DAO.

Cách dùng giản tiện của control Data là đặt nó lên một Form rồi làm việc với những Properties của nó. Bạn hãy bắt đầu một dự án VB6 mới, cho nó tên **DataControl** bằng cách click tên project trong Project Explorer bên phải rồi edit property Name trong Properties Window.

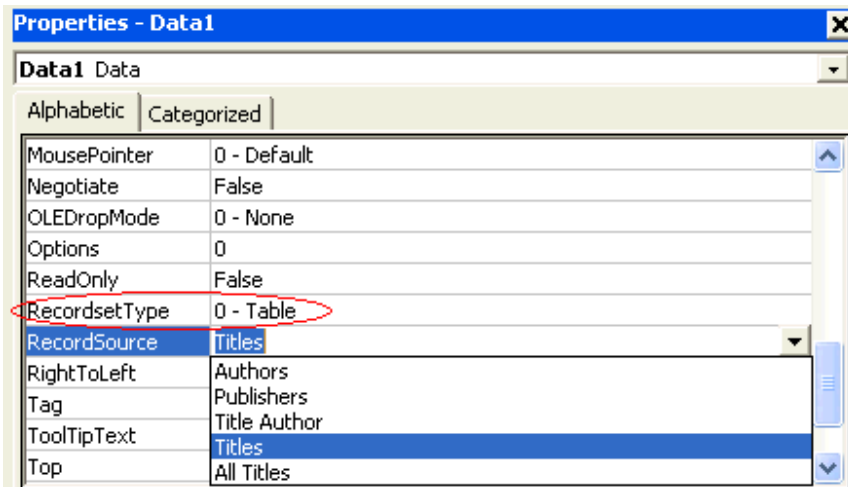
DoubleClick lên Icon của Control Data trong Toolbox. Một Control Data tên **Data1** sẽ hiện ra trên Form. Muốn cho nó nằm bên dưới Form, giống như một StatusBar, hãy set **property Align** của nó trong Properties Window thành **2 - Align Bottom**.

Click bên phải hàng **property DatabaseName**, kế đó click lên nút browse có ba chấm để chọn một file Access database từ giao thoại cho Data1. Ở đây ta chọn **E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB**, trong computer của bạn có thể nó nằm trên disk **C** hay **D**.



Trong chương trình này ta muốn làm việc với **table Titles** của database BIBLIO.MDB, để xem và edit các records. Để ý **property DefaultType** của Data1 có trị số **2- UseJet**, tức là dùng kỹ thuật DAO, thay vì dùng kỹ thuật ODBC.

Khi bạn click lên **property Recordsource** của Data1, rồi click lên cái tam giác nhỏ bên phải, một ComboBox sẽ mở ra cho ta thấy danh sách các tables trong database. Bạn hãy chọn **Titles**. Để ý **property RecordsetType** của Data1 có trị số là **0 - Table**:



Cái từ mới mà ta sẽ dùng thường xuyên khi truy cập dữ liệu trong VB6 là **Recordset** (bộ records). Recordset là một **Set of records**, nó có thể chứa một số records hay không có record nào cả. Một record trong Recordset có thể là một record lấy từ một Table. Trong trường hợp ấy có thể ta lấy về tất cả records trong table hay chỉ những records thỏa đúng một điều kiện, thí dụ như ta chỉ muốn lấy các records của những sách xuất bản trước năm 1990 (Year Published < 1990).

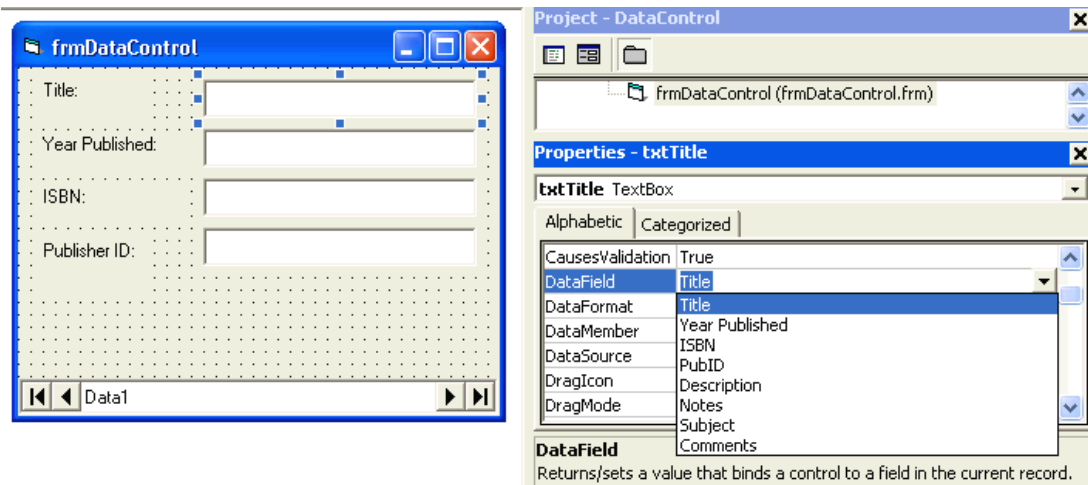
Một Record trong Recordset cũng có thể là tập hợp các cột (columns) từ hai (hay ba) tables qua các mối liên hệ one-to-one và one-to-many. Thí dụ như khi lấy các records từ table Titles, ta muốn có thêm chi tiết tên công ty (Company Name) và điện thoại (Telephone) của nhà xuất bản (table Publishers) bằng cách dùng **Foreign Key PubID** trong table Titles làm **Primary Key** trong table Publishers để lấy các chi tiết ấy.

Trong trường hợp ấy ta có thể xem như có một **virtual (ảo) table** là tập hợp của hai tables Titles và Publishers.

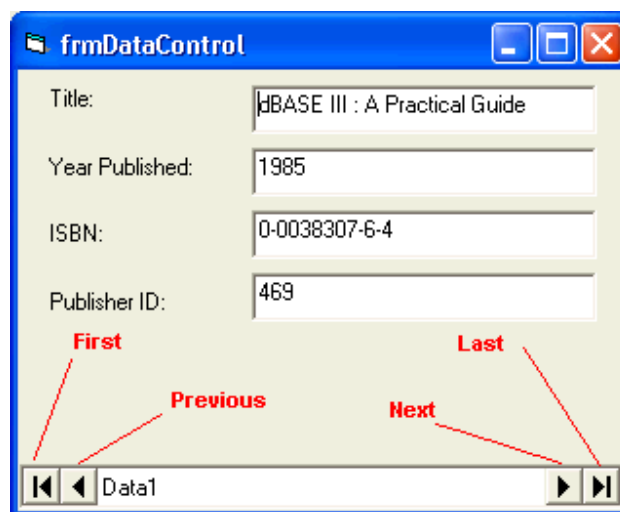
Bây giờ bạn hãy đặt lên Form 4 labels với captions: **Title, Year Published, ISBN** và **Publisher ID**. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là **txtTitle, txtYearPublished, txtISBN** và **txtPublisherID**.

Chọn textbox txtTitle, rồi set **property Datasource** của nó trong Properties Window thành **Data1**. Khi click lên **property Datafield** của txtTitle và mở ComboBox ra bạn sẽ thấy liệt kê tên các Fields trong table Titles. Đó là vì Data1 được coi như trung gian lấy table Titles từ database. Ở đây ta sẽ chọn cột Title.

Lập lại công tác này cho 3 textboxes kia, và chọn các cột Year Published (năm xuất bản), ISBN (số lý lịch trong thư viện quốc tế), và PubID (số lý lịch nhà xuất bản) làm Datafield cho chúng.



Tới đây, mặc dầu chưa viết một hàng code, ta có thể chạy chương trình được rồi. Nó sẽ hiển thị chi tiết của record đầu tiên trong table Titles như dưới đây:



Bạn có thể bấm các nút di chuyển **Navigator Buttons** để đi đến các record **đầu (first)**, **trước (previous)**, **kế (next)** và **cuối (last)**. Mỗi lần bạn di chuyển đến một record mới là chi tiết của record ấy sẽ hiển thị. Nếu không dùng các Navigator Buttons, ta cũng có thể code để làm công tác tương đương bằng cách gọi các Recordset **methods** **MoveFirst**, **MovePrevious**, **MoveNext** và **MoveLast**.

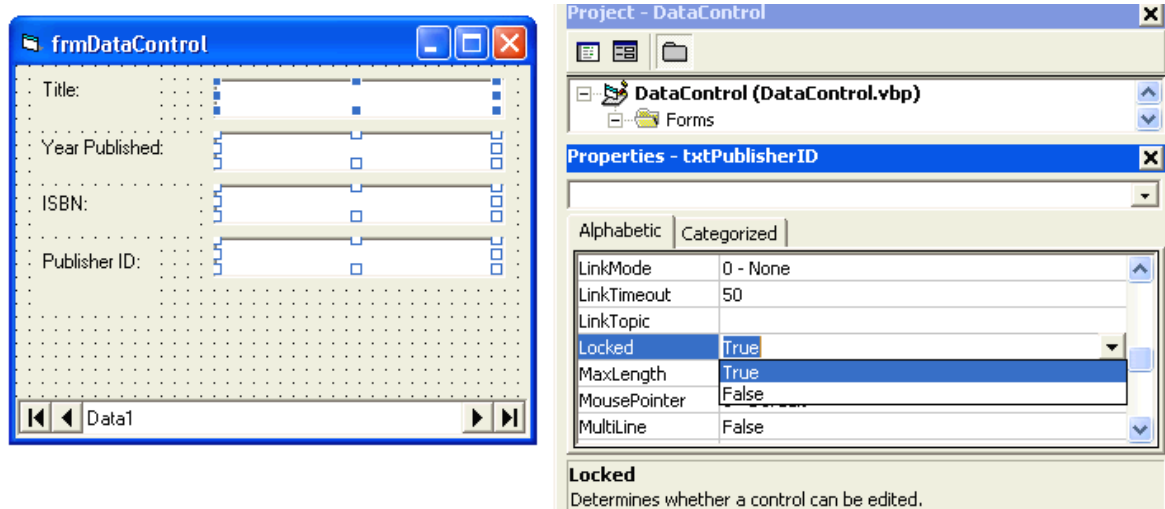
Khi record cuối của Recordset đang hiển thị, nếu ta gọi method `MoveLast` thì **property EOF (End-Of-File)** của Recordset trở thành `True`. Tương tự như vậy, khi record thứ nhất của Recordset đang hiển thị, nếu ta gọi method `MovePrevious` thì **property BOF (Begin-Of-File)** của Recordset trở thành `True`.

Nếu một Recordset không có chứa một record nào cả thì cả hai properties EOF và BOF đều là True.

Đặc tính hiển thị dữ liệu trong các textbox theo đúng record hiện thời (**current record**) được gọi là **data binding** hay **data bound** (buộc vào dữ liệu) và control TextBox hỗ trợ chức năng này được nói là **Data Aware** (biết bà con dữ liệu).

Khi record đầu tiên đang hiển thị, nếu bạn edit **Year Published** để đổi từ 1985 thành **1983** rồi click Navigator button Next để hiển thị record thứ nhì, kế đó click Navigator button Previous để hiển thị lại record đầu tiên thì bạn sẽ thấy là field Year Published của record đầu tiên đã thật sự được thay đổi (updated) thành 1983.

Điều này có nghĩa rằng khi Data1 navigates từ record này đến record khác thì nếu record này đã có sự thay đổi vì user edited, nó lưu trữ sự thay đổi đó trước khi di chuyển. Chưa chắc là bạn muốn điều này, do đó, nếu bạn không muốn user tình cờ edit một record thì bạn có thể set **property Locked** của các textboxes ấy thành True để user không thể edit các textboxes như trong hình dưới đây:



Chỉ định vị trí Database lúc chạy chương trình

Cách chỉ định tên DatabaseName trong giai đoạn thiết kế (at design time) ta đã dùng trước đây tuy tiện lợi nhưng hơi nguy hiểm, vì khi ta cài chương trình này lên computer của khách, chưa chắc file database ấy nằm trong một folder có cùng tên. Thí dụ trên computer mình thì database nằm trong folder E:\Program Files\Microsoft Visual Studio\VB98, nhưng trên computer của khách thì database nằm trong folder C:\VB6\DataControl chẳng hạn. Do đó, khi chương trình khởi động ta nên xác định lại vị trí của database. Giả dụ ta muốn để database trong

cùng một folder với chương trình đang chạy, ta có thể dùng **property Path** của Application Object **App** như sau:

```
Dim AppFolder As String
Private Sub Form_Load()
    ' Fetch Folder where this program EXE resides
    AppFolder = App.Path
    ' make sure it ends with a back slash
    If Right(AppFolder, 1) <> "\" Then AppFolder = AppFolder & "\"
    ' Assign Full path database filename to Data1
    Data1.DatabaseName = AppFolder & "BIBLIO.MDB"
End Sub
```

Với cách code nói trên ta sẽ đảm bảo chương trình tìm thấy file database đúng chỗ, không cần biết người ta cài chương trình bạn ở đâu trong hard disk của computer khách.

Nếu bạn đang học VB6 từ xa, khi nộp bài database cho giám thị VB6 mà bạn hardcode (viết chết cứng) vị trí của file database trong lúc thiết kế thì giám thị (tutor) cũng gặp cùng sự khó khăn này vì chưa chắc giám thị sẽ chứa database trong một folder có cùng tên như trong harddisk của bạn.

Thêm bớt các Records

Chương trình trên dùng cũng tạm được, nhưng nó không cho ta phương tiện để thêm (add), bớt (delete) các records. Bây giờ bạn hãy để vào Form 5 buttons tên: **cmdEdit**, **cmdNew**, **cmdDelete**, **cmdUpdate** và **cmdCancel**.

Mặc dầu bạn không thấy, nhưng thật ra Control Data **Data1** có một **property Recordset** và khi ta dùng Navigator buttons là di chuyển từ record này đến record khác trong Recordset ấy. Ta có thể nói đến nó bằng Notation (cách viết) **Data1.Recordset**, và mỗi lần muốn lấy Recordset mới nhất từ database ta dùng **method Refresh** như **Data1.Recordset.Refresh**.

Lúc chương trình mới khởi động, user đang xem (browsing) các records thì hai buttons **Update** và **Cancel** không cần phải làm việc. Do đó ta sẽ nhân tiện Lock (khóa) các textboxes và disable (làm cho bất lực) hai buttons này vì không cần dùng chúng.

Trong **Sub SetControls** dưới đây, ta dùng một parameter gọi là **Editing** với trị số False hay True tùy theo user đang Browse hay Edit, ta gọi là **Browse mode** và **Edit mode**. Trong **Edit mode**, các Textboxes được unlocked (mở khóa) và các nút **cmdNew**, **cmdDelete** và **cmdEdit** trở nên bất lực:


```

Sub SetControls(ByVal Editing As Boolean)
    ' Lock/Unlock textboxes
    txtTitle.Locked = Not Editing
    txtYearPublished.Locked = Not Editing
    txtISBN.Locked = Not Editing
    txtPublisherID.Locked = Not Editing
    ' Enable/Disable buttons
    CmdUpdate.Enabled = Editing
    CmdCancel.Enabled = Editing
    CmdDelete.Enabled = Not Editing
    cmdNew.Enabled = Not Editing
    CmdEdit.Enabled = Not Editing
End Sub

```

Trong **Browse mode**, Form có dạng như sau:

Sub SetControls được gọi trong **Sub Form_Load** khi chương trình khởi động và trong **Sub CmdEdit_Click** khi user click nút **Edit** như sau:

```

Private Sub Form_Load()
    ' Fetch Folder where this program EXE resides
    AppFolder = App.Path
    ' make sure it ends with a back slash
    If Right(AppFolder, 1) <> "\" Then AppFolder = AppFolder & "\"
    ' Assign Full path database filename to Data1
    Data1.DatabaseName = AppFolder & "BIBLIO.MDB"
    ' Place controls in Browse Mode
    SetControls (False)
End Sub

Private Sub CmdEdit_Click()
    ' Place controls in Edit Mode
    SetControls (True)
End Sub

```

Khi ta Delete một record trong recordset, vị trí của record hiện tại (current record) vẫn không thay đổi. Do đó, sau khi delete một record ta phải **MoveNext**. Khổ nỗi, nếu ta vừa delete record cuối của Recordset thì sau khi MoveNext, **property EOF** của Recordset sẽ thành True. Thành ra ta phải kiểm tra điều đó, nếu đúng vậy thì lại phải **MoveLast** để hiển thị record cuối của Recordset như trong code của **Sub cmdDelete_Click** dưới đây:

```
Private Sub CmdDelete_Click()
    On Error GoTo DeleteErr
    With Data1.Recordset
        ' Delete new record
        .Delete
        ' Move to next record
        .MoveNext
        If .EOF Then .MoveLast
    End With
    Exit Sub
DeleteErr:
    MsgBox Err.Description
    Exit Sub
End Sub
```

Trong lúc code, ta Update (cập nhật hóa) một record trong Recordset bằng method **Update**. Nhưng ta chỉ có thể gọi method Update của một Recordset khi Recordset đang ở trong **Edit hay AddNew mode**. Ta đặt một Recordset vào Edit mode bằng cách gọi **method Edit** của Recordset, thí dụ như **Data1.Recordset.Edit**. Tương tự như vậy, ta đặt một Recordset vào AddNew mode bằng cách gọi **method AddNew** của Recordset, thí dụ như **Data1.Recordset.AddNew**.

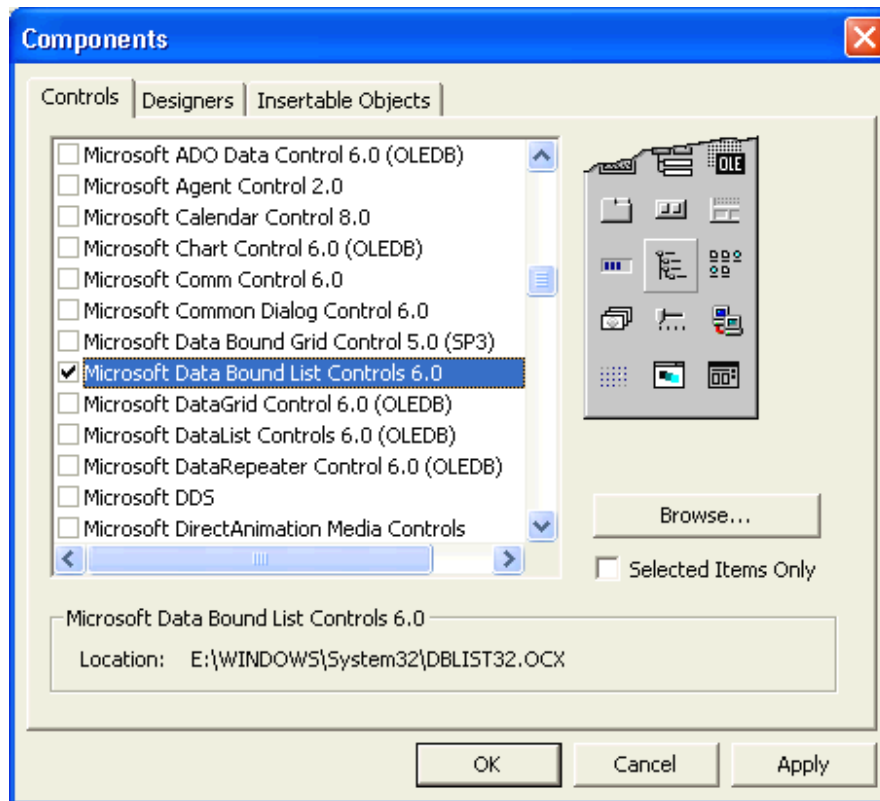
```
Private Sub cmdNew_Click()
    ' Place Recordset into Recordset AddNew mode
    Data1.Recordset.AddNew
    ' Place controls in Edit Mode
    SetControls (True)
End Sub
```

Sau khi Recordset gọi method Update thì Recordset ấy ra khỏi AddNew hay Edit modes. Ta cũng có thể tự thoát ra khỏi AddNew hay Edit modes, hay nói cho đúng hơn là hủy bỏ mọi pending (đang chờ đợi) Update bằng cách gọi **method CancelUpdate**, thí dụ như **Data1.Recordset.CancelUpdate**.

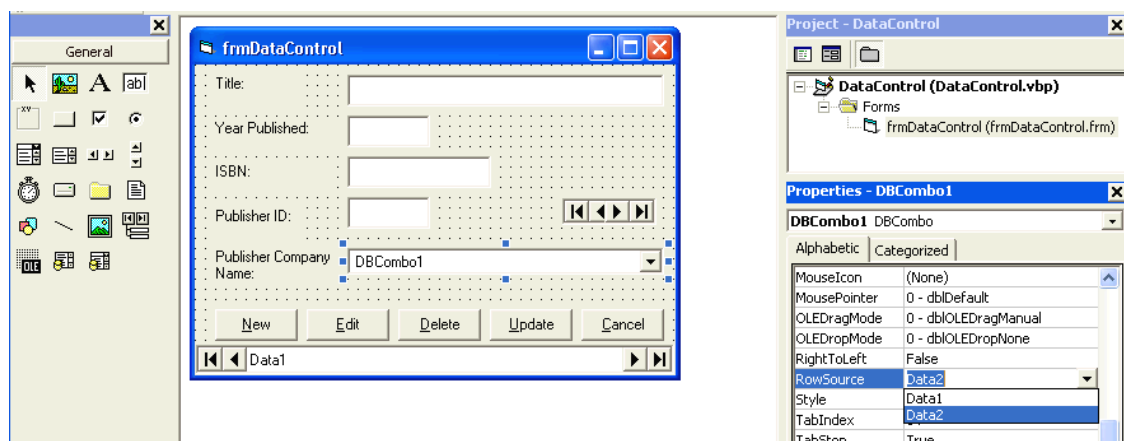
Dùng DataBound Combo

Trong chương trình hiện tại ta chỉ hiển thị lý lịch nhà xuất bản (PubID) của Title, chứ không có thêm chi tiết. Phải chi mặc đầu chương trình lưu trữ **PubID**, nhưng hiển thị được **Company Name** của nhà xuất bản cho ta làm việc để khỏi phải nhớ các con số thì hay quá. Ta có thể thực hiện điều đó bằng cách dùng

Control **DBCombo (Data Bound Combo)**. Bạn hãy dùng IDE Menu Command **Project | Components...** để chọn **Microsoft Data Bound List Controls 6.0** rồi click **Apply**.



Kế đó, thêm một DBCombo tên **DBCombo1** vào Form. Vì ta cần một Recordset khác để cung cấp Table Publisher cho DBCombo1, nên bạn hãy thêm một control Data thứ nhì tên **Data2** vào Form. Cho Data2, hãy set property DatabaseName thành **E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB** và property RecordSource thành **Publishers**. Để không cho người ta thấy hình Data2 lúc run-time, bạn hãy set **property Visible** nó thành False.



Cái mục đích của chúng ta khi dùng DBCombo1 là hiển thị Company Name của nhà xuất bản, nhưng đằng sau lưng thì không có gì thay đổi, tức là ta vẫn làm việc với PubID cho các record Title của Data1. Khi user click lên DBCombo1 để chọn một nhà xuất bản, thì ta theo Company Name đó mà chứa PubID tương ứng trong record Title của Data1. Do đó có nhiều thứ ta phải sắp đặt cho DBCombo1 như sau:

Property	Value	Chú thích
RowSource	Data2	Đây là datasource của chính DBCombo1. Nó cung cấp table Publishers.
Listfield	Company Name	Khi RowSource phía trên đã được chọn rồi, Combo của property Listfield này sẽ hiển thị các fields của table Publishers. Company Name là field của RowSource mà ta muốn hiển thị trên DBCombo1.
DataSource	Data1	Đây là datasource của record mà ta muốn. edit, tức là record của table Titles
Datafield	PubID	Field (của record Title) sẽ được thay đổi.
BoundColumn	PubID	Field trong RowSource (table Publishers) tương ứng với item user chọn trong DBCombo1 (Company Name).

Khi trong Edit mode user chọn một Company Name khác trong DBCombo1 rồi click nút Update bạn sẽ thấy Textbox txtPublisherID cũng đổi theo và hiển thị con số lý lịch PubID mới. Nếu trước khi Update bạn muốn thấy PubID mới hiển thị trong Textbox txtPublisherID thì bạn có thể dùng Event Click của DBCombo1 như sau:

```
Private Sub DBCombo1_Click(Area As Integer)
    ' Display new PuBID
    txtPublisherID.Text = DBCombo1.BoundText
End Sub
```

Property BoundText của DBCombo1 là trị số của BoundColumn mà ta có thể truy cập (viết hay đọc) được. Thí dụ như bạn muốn mỗi khi thêm một record Title mới thì default PubID là 324, tức là Company Name= "GLOBAL ENGINEERING". Bạn có thể assign trị số 324 vào property BoundText của DBCombo1 trong Sub cmdNew_Click như sau:

```
Private Sub cmdNew_Click()
    ' Place Recordset into Recordset AddNew mode
    Data1.Recordset.AddNew
    ' Default Publisher is "GLOBAL ENGINEERING", i.e. PubID=324
```

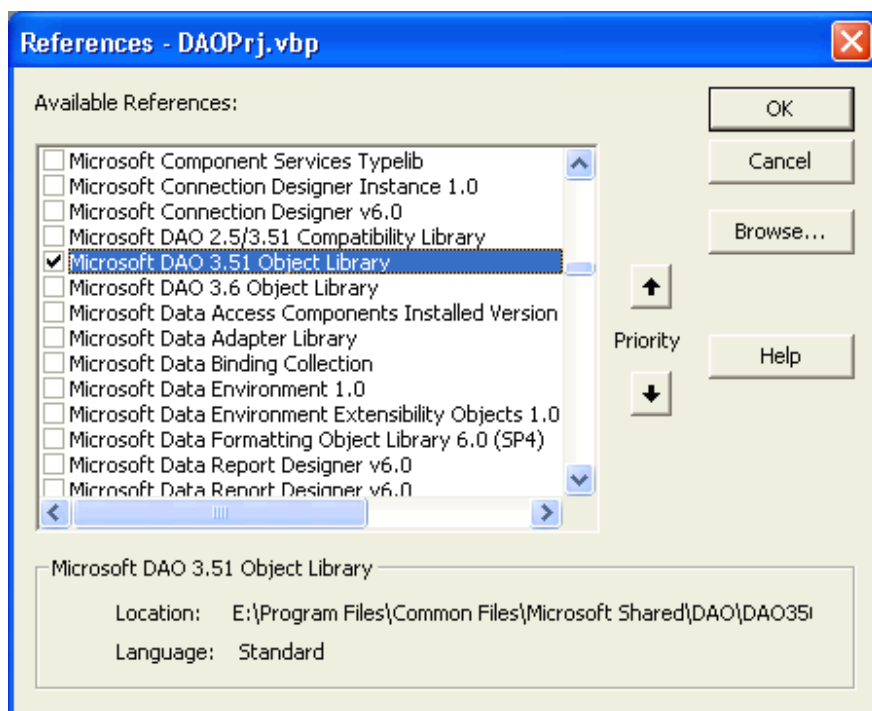
```
DBCombo1.BoundText = 324  
' Place controls in Edit Mode  
SetControls (True)  
End Sub
```

Trong bài tới ta sẽ học thêm về cách coding để dùng Recordset trong kỹ thuật DAO.

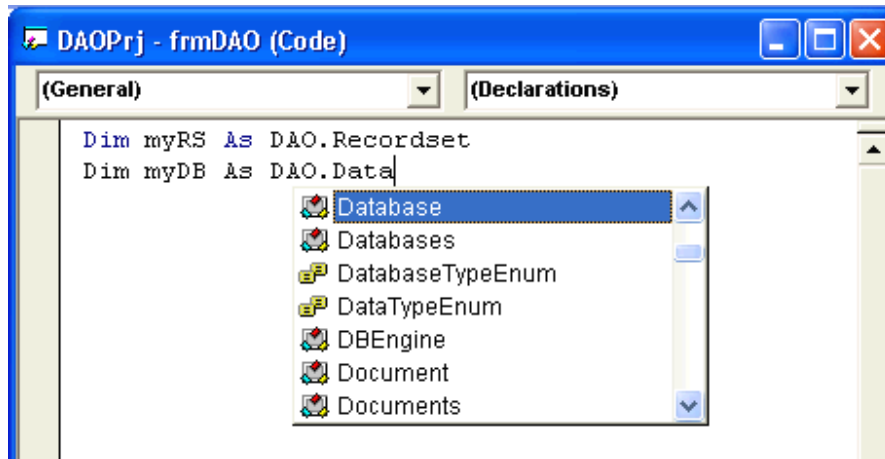
Lập trình với kỹ thuật DAO

Reference DAO

Trong bài này ta sẽ học những cách lập trình căn bản với MS Access database qua kỹ thuật DAO mà không cần dùng đến **Control Data** như trong bài trước. Ta sẽ cần đến vài Objects trong thư viện DAO, do đó nếu bạn mở một dự án VB6 mới thì hãy dùng Menu Command **Project | References...** để chọn **Microsoft DAO 3.51 Object Library** bằng cách click cái checkbox bên trái như trong hình dưới đây. (Một cách để nhớ tên của Object này là nhớ câu *"thằng cha của ĐÀO 35 con dê"*).



Sau đó trong code của Form chính ta sẽ declare variable **myDatabase** cho một instance của **DAO database** và variable **myRS** cho một **DAO recordset**. Ở đây ta nói rõ Database và Recordset là thuộc loại **DAO** để phân biệt với Database và Recordset thuộc loại **ADO (ActiveX Data Object)** sau này. Để ý là Intellisense giúp ta trong lúc viết code:



Bây giờ bạn hãy đặt lên Form chính, tên **frmDAO**, 4 labels với captions: **Title**, **Year Published**, **ISBN** và **Publisher ID**. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là **txtTitle**, **txtYearPublished**, **txtISBN** và **txtPublisherID**.

Điều ta muốn làm là khi Form mới được loaded, nó sẽ lấy về từ database một Recordset chứa tất cả records trong **table Titles** theo thứ tự về mẫu tự (alphabetical order) của **field Title** và hiển thị record đầu tiên.

Dùng keyword SET

Chuyện trước hết là mở một Database Object dựa vào tên đầy đủ (full path name) của Access database:

```
' Open main database
Set myDB = OpenDatabase(AppFolder & "BIBLIO.MDB")
```

Để ý chữ **Set** trong câu code trên. Đó là vì myDB là một **Pointer** đến một Object. Mặc dầu từ rày về sau ta sẽ dùng myDB như một Database theo cách giống như bất cứ variable thuộc data type nào khác, nhưng khi chỉ định lần đầu là nó từ đâu đến thì ta dùng chữ Set, để nói rằng thật ra myDB không phải là Object Database, nhưng là Pointer đến Object Database. Điểm này càng nói đến càng khó hiểu.

Đại khái là VB6 runtime dynamically allocates (dành ra cho khi cần) một phần trong bộ nhớ (memory) để chứa Object Database khi ta nhận được nó từ execution của **Method OpenDatabase**. Dầu vị trí chỗ chứa Object Database trong bộ nhớ không nhất định, nhưng vì ta nắm cái cán chỉ đến vị trí ấy nên ta vẫn có thể làm việc với nó một cách bình thường. Cái cán ấy là value (trị số) của variable myDB. Vì value này không phải là Object, nhưng nó chứa **memory**

address chỉ đến (**point to** hay **refer to**) Object Database, nên ta gọi nó là Pointer.

Lập trình dùng Pointer nói chung rất linh động là hiệu năng trong các ngôn ngữ như C, Pascal, C++ ,v.v.. Tuy nhiên, lập trình viên phải nhớ trả lại Operating System phần memory mình dùng khi không còn cần nó nữa để Operating System lại allocate cho Object khác. Nếu công việc quản lý dùng lại memory không ổn thỏa thì có những mảnh memory nằm lang bang mà Operating Sytem không biết. Lăn lăn Operating System sẽ không còn memory dư nữa. Ta gọi hiện tượng ấy là **memory leakage (rỉ)**. Các ngôn ngữ sau này như Java, C# đều không dùng Pointer nữa. Visual Basic không muốn lập trình viên dùng Pointer. Chỉ trong vài trường hợp đặc biệt VB6 mới lộ ra cho ta thấy thật ra ở trong hậu trường VB6 Runtime dùng Pointer, như trong trường hợp này.

Tương tự như vậy, vì Recordset là một Pointer đến một Object, ta cũng dùng **Set** khi chỉ định một DAO Recordset lấy về từ **Method OpenRecordset** của database myDB.

```
'Open recordset
Set myRS = myDB.OpenRecordset("Select * from Titles ORDER BY Title")
```

Cái parameter loại String ta dùng cho method OpenRecordset là một **Lệnh (Statement) SQL**. Nó chỉ định cho database lấy tất cả mọi fields (columns) (**Select ***) của mỗi record từ Table Titles (**from Titles**) làm một Recordset và sort các records trong Recordset ấy theo alphabetical order của field Title (**ORDER BY Title**).

Nhớ là Recordset này cũng giống như **property Recordset** của một Control Data mà ta dùng trong bài trước. Bây giờ có Recordset rồi, ta có thể hiển thị chi tiết của record đầu tiên nếu Recordset ấy có ít nhất một record. Ta kiểm tra điều ấy dựa vào **property RecordCount** của Recordset như trong code dưới đây:

```
Private Sub Form_Load()
    ' Fetch Folder where this program EXE resides
    AppFolder = App.Path
    ' make sure it ends with a back slash
    If Right(AppFolder, 1) <> "\" Then AppFolder = AppFolder & "\"
    ' Open main database
    Set myDB = OpenDatabase(AppFolder & "BIBLIO.MDB")
    'Open recordset
    Set myRS = myDB.OpenRecordset("Select * from Titles ORDER BY Title")
    ' if Recordset is not empty then display the first record
    If myRS.RecordCount > 0 Then
        myRS.MoveFirst ' move to first record
        Displayrecord ' display details of current record
    End If
End Sub
```


Sau khi dùng **method MoveFirst** của Recordset để position **current record** ở Record đầu tiên, ta hiển thị trị số các fields của record bằng cách assign chúng vào các textboxes của Form như sau:

```
Private Sub Displayrecord()
    ' Assign record fields to the appropriate textboxes
    With myRS
        ' Assign field Title to textbox txtTitle
        txtTitle.Text = .Fields("Title")
        txtYearPublished.Text = .Fields("[Year Published]")
        txtISBN.Text = .Fields("ISBN")
        txtPublisherID.Text = .Fields("PubID")
    End With
End Sub
```

Để ý vì field **Year Published** gồm có hai chữ nên ta phải đặt tên của field ấy giữa hai dấu ngoặc vuông ([]). Để tránh bị phiền phức như trong trường hợp này, khi bạn đặt tên database field trong lúc thiết kế một table hãy dán dính các chữ lại với nhau, đừng để rời ra. Thí dụ như dùng **YearPublished** thay vì **Year Published**.

Các nút di chuyển

Muốn có các nút Navigators tương đương với của một Control Data, bạn hãy đặt lên Form 4 buttons mang tên **CmdFirst**, **CmdPrevious**, **CmNext** và **CmdLast** với captions: <<, <, >, >>.

Code cho các nút này cũng đơn giản, nhưng ta phải coi chừng khi user muốn di chuyển quá record cuối cùng hay record đầu tiên. Ta phải kiểm tra xem **EOF** có trở thành True khi user click CmdNext, hay **BOF** có trở thành True khi user click CmdPrevious:

```
Private Sub CmdNext_Click()
    myRS.MoveNext ' Move to next record
    ' Display record details if has not gone past the last record
    If Not myRS.EOF Then
        Displayrecord ' display details of current record
    Else
        myRS.MoveLast ' Move back to last record
    End If
End Sub

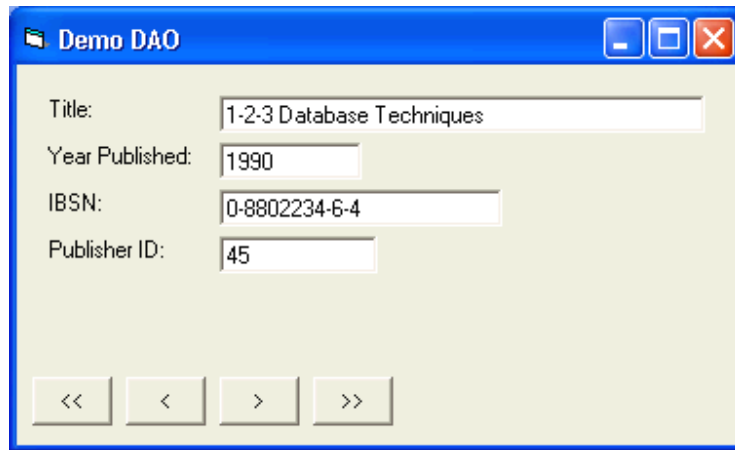
Private Sub CmdPrevious_Click()
    myRS.MovePrevious ' Move to previous record
    ' Display record details if has not gone past the first record
    If Not myRS.BOF Then
        Displayrecord ' display details of current record
    Else
        myRS.MoveFirst ' Move back to first record
    End If
End Sub
```

```
End If
End Sub

Private Sub CmdFirst_Click()
    myRS.MoveFirst ' Move back to first record
    Displayrecord ' display details of current record
End Sub

Private Sub CmdLast_Click()
    myRS.MoveLast ' Move back to last record
    Displayrecord ' display details of current record
End Sub
```

Khi chạy chương trình bạn sẽ thấy nó hiển thị chi tiết của Record đầu tiên khác với trong bài trước đây vì các records đã được sorted:



Bạn hãy thử dùng các Navigator buttons cây nhà, lá vườn của mình xem chúng làm việc có đúng không.

Tới đây, không biết bạn có để ý là dù user có vô tình sửa đổi một chi tiết nào trong các textboxes, không có record nào bị cập nhật hóa trong database khi user di chuyển từ record này đến record khác. Lý do là các Textboxes không có Data Bound với các Fields của Recordset.

Thêm bớt các Records

Giống như chương trình trong bài rồi, ta sẽ thêm phương tiện để thêm (add), bớt (delete) các records. Bây giờ bạn hãy để vào Form 5 buttons tên: **cmdEdit**, **cmdNew**, **cmdDelete**, **cmdUpdate** và **cmdCancel**.

Chỗ nào trong chương trình trước ta dùng **Data1.Recordset** thì bây giờ ta dùng **myRS**.

Ta sẽ dùng lại **Sub SetControls** với parameter **Editing** có trị số False hay True tùy theo user đang Browse hay Edit. Trong **Browse mode**, các Textboxes bị Locked (khóa) và các nút **cmdUpdate** và **cmdCancel** trở nên bất lực. Trong **Edit mode**, các Textboxes được unlocked (mở khóa) và các nút **cmdNew**, **cmdDelete** và **cmdEdit** trở nên bất lực.

Vì ở đây không có Data Binding nên đợi cho đến khi **Update (cập nhật hóa)** ta mới đặt Recordset vào **AddNew** hay **Edit mode**. Do đó ta chỉ cần nhớ là khi user edits là đang Edit một record hiện hữu hay thêm một Record mới. Ta chứa trị số Boolean ấy trong variable **AddNewRecord**. Nếu user sắp thêm một record mới thì AddNewRecord = True, nếu User sắp Edit một record hiện hữu thì AddNewRecord = False.

Ngoài ra, khi User sắp thêm một record mới bằng cách click nút New thì ta phải tự clear (làm trắng) hết các textboxes bằng cách assign Empty string vào text property của chúng như sau:

```
' If Editing existing record then AddNewRecord = False
' Else AddNewRecord = true
Dim AddNewRecord As Boolean

Private Sub ClearAllFields()
    ' Clear all the textboxes
    txtTitle.Text = ""
    txtYearPublished.Text = ""
    txtISBN.Text = ""
    txtPublisherID.Text = ""
End Sub

Private Sub cmdNew_Click()
    ' Remember that this is Adding a new record
    AddNewRecord = True
    ' Clear all textboxes
    ClearAllFields
    ' Place controls in Edit Mode
    SetControls (True)
End Sub

Private Sub CmdEdit_Click()
    ' Place controls in Edit Mode
    SetControls (True)
    ' Remember that this is Editing an existing record
    AddNewRecord = False
End Sub
```

Nếu user clicks Cancel trong khi đang edit các textboxes, ta không cần gọi **method CancelUpdate** vì Recordset chưa bị đặt vào AddNew hay Edit mode. Ở đây ta chỉ cần hiển thị lại chi tiết của current record, tức là hủy bỏ những gì user đang đánh vào:

```
Private Sub CmdCancel_Click()
    ' Cancel update
    SetControls (False)
    ' Redisplay details or current record
    Displayrecord
End Sub
```

Lúc user clicks Update, bạn có dịp để kiểm tra data xem có field nào bị bỏ trống (nhất là **Primary Key ISBN** bắt buộc phải có trị số) hay có gì không valid bằng cách gọi **Function GoodData**. Nếu GoodData trả lại một trị số False thì ta không xúc tiến với việc Update. Nếu GoodData trả về trị số True thì ta đặt Recordset vào AddNew hay Edit mode tùy theo trị số của Boolean variable AddNewRecord.

Giống như khi hiển thị chi tiết của một Record ta phải assign từng Field vào textbox, thì bây giờ khi Update ta phải làm ngược lại, tức là assign property Text của từng textbox vào Record Field tương ứng. Sau cùng ta gọi **method Update** của recordset và cho các controls trở lại Browse mode:

```
Private Function GoodData() As Boolean
    ' Check Data here. If Invalid Data then GoodData = False
    GoodData = True
End Function

Private Sub CmdUpdate_Click()
    ' Verify all data, if Bad then do not Update
    If Not GoodData Then Exit Sub
    ' Assign record fields to the appropriate textboxes
    With myRS
        If AddNewRecord Then
            .AddNew ' Place Recordset in AddNew Mode
        Else
            .Edit ' Place Recordset in Edit Mode
        End If
        ' Assign text of txtTitle to field Title
        .Fields("Title") = txtTitle.Text
        .Fields("[Year Published]") = txtYearPublished.Text
        .Fields("ISBN") = txtISBN.Text
        .Fields("PubID") = txtPublisherID.Text
        ' Update data
        .Update
    End With
    ' Return controls to Browse Mode
    SetControls (False)
End Sub
```

Cũng vì không có Data Binding, nên khi User Delete một record, sau khi di chuyển qua record kế tiếp ta phải tự hiển thị chi tiết của record đó như sau:

```
Private Sub CmdDelete_Click()
    On Error GoTo DeleteErr
    With myRS
```

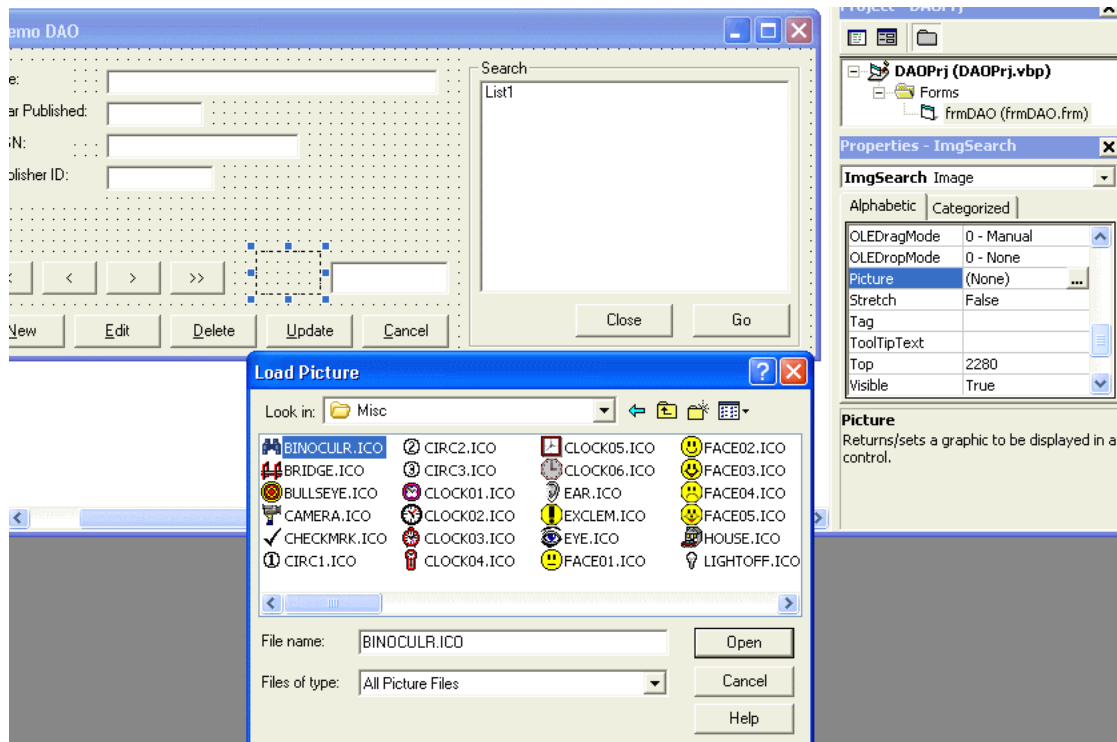
```
' Delete new record
.Delete
' Move to next record
.MoveNext
If .EOF Then .MoveLast
' Display details of current record
Displayrecord
Exit Sub
End With
DeleteErr:
MsgBox Err.Description
Exit Sub
End Sub
```

Tìm một record

Tiếp theo đây, ta muốn liệt kê các sách có tiêu đề chứa một chữ hay câu nào đó, thí dụ như chữ **"Guide"**. Kể đó user có thể chọn một sách bằng cách select tiêu đề sách ấy và click nút **Go**. Chương trình sẽ locate (tìm ra) record của sách ấy và hiển thị chi tiết của nó.

Bây giờ bạn hãy cho vào Form một textbox tên **txtSearch** và một Image tên **ImgSearch**. Kể đó đặt một frame tên **fraSearch** vào Form. Để lên frame này một listbox tên **List1** để hiển thị tiêu đề các sách, và hai buttons tên **CmdClose** và **CmdGo**, với caption Close và Go. Sau khi select một sách trong List1, user sẽ click nút **Go** để hiển thị chi tiết sách ấy. Nếu đổi ý, user sẽ click nút **Close** để làm biến mất frame fraSearch.

Bình thường frame fraSearch chỉ hiện ra khi cần, nên lúc đầu hãy set **property Visible** của nó thành False. Ta sẽ cho ImgSearch hiển thị hình một ống dòm nên bạn hãy click vào bên phải **property Picture** trong Properties Window để chọn Icon BINOCULR.ICO từ folder E:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Misc:



Cái Primary Key của table Titles là **ISBN**. Khi user select một sách ta muốn biết ISBN của sách ấy để locate (định chỗ) nó trong Recordset myRS. Do đó trong khi thêm tiêu đề của một sách vào List1, ta đồng thời thêm ISBN của sách ấy vào một Listbox thứ hai tên List2. Ta chỉ sẽ dùng List2 sau hậu trường, nên hãy set property Visible của nó thành False. Dưới đây là code để load tiêu đề sách và ISBN vào các Listboxes:

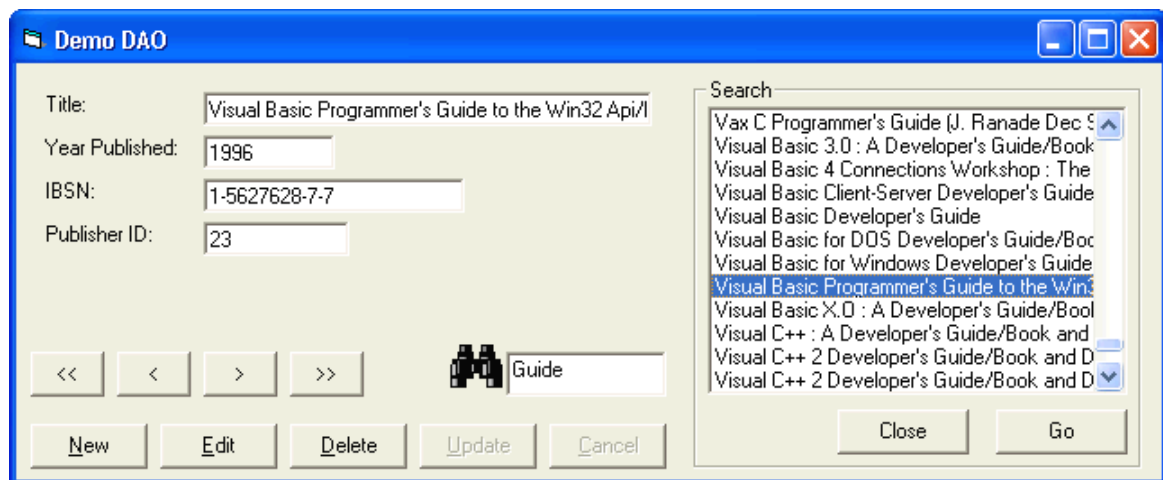
```
Private Sub ImgSearch_Click()
    ' Show Search Frame
    fraSearch.Visible = True
    Dim SrchRS As DAO.Recordset
    Dim SQLCommand As String
    ' Define SQL statement
    SQLCommand = "Select * from Titles where Title LIKE '" & "*" &
txtSearch & "*" & "' ORDER BY Title"
    ' Fetch all records having Title containing the text pattern given
    by txtSearch
    Set SrchRS = myDB.OpenRecordset(SQLCommand)
    ' If Recordset is not Empty then list the books' titles in List1
    If SrchRS.RecordCount > 0 Then
        List1.Clear ' Clear List1
        ' We use List2 to contain the Primary Key ISBN corresponding to
        the books in List1
        List2.Clear ' Clear List2
        With SrchRS
            ' Iterate through the Recordset until EOF
            Do While Not SrchRS.EOF
                ' Display Title in List1
```

```

List1.AddItem .Fields("Title")
' Store corresponding ISBN in List2
List2.AddItem .Fields("ISBN")
.MoveNext ' Move to next record in the Recordset
Loop
End With
End If
End Sub

```

Khi user Click ImgSearch với text pattern là chữ **Guide**, ta sẽ thấy hình dưới đây:



Trong SELECT statement bên trên ta dùng operator **LIKE** trên text pattern, chữ **Guide**, có **wildcard character (*)** ở hai bên. Wildcard character là chỗ có (hay không có) chữ gì cũng được. Trong trường hợp này có nghĩa là hễ có chữ Guide trong tiêu đề sách là được, không cần biết nó nằm ở đâu. Ngoài ra sự chọn lựa này **Không có Case Sensitive**, tức là chữ **guide**, **Guide** hay **GUIDE** đều được cả.

Khi user clicks nút Go, ta sẽ dùng **method FindFirst** của Recordset myRS để định chỗ của record có trị số Primary Key là hàng text trong List2 tương ứng với tiêu đề được chọn trong List1 như sau:

```

Private Sub CmdGo_Click()
    Dim SelectedISBN As String
    Dim SelectedIndex As Integer
    Dim Criteria As String
    ' Index of line selected by user in List1
    SelectedIndex = List1.ListIndex
    ' Obtain corresponding ISBN in List2
    SelectedISBN = List2.List(SelectedIndex)
    ' Define Search criteria - use single quotes for selected text
    Criteria = "ISBN = '" & SelectedISBN & "'"
    ' Locate the record, it will become the current record

```

```

myRS.FindFirst Criteria
' Display details of current record
Displayrecord
' Make fraSearch disappeared
fraSearch.Visible = False
End Sub

```

Lưu ý là trong string Criteria, vì ISBN thuộc loại text, chứ không phải là một con số, nên ta phải kẹp nó giữa hai dấu ngoặc đơn.

Bookmark

Khi di chuyển từ record này đến record khác trong Recordset, đôi khi ta muốn đánh dấu vị trí của một record để có dịp sẽ trở lại. Ta có thể thực hiện điều ấy bằng cách ghi nhớ **Bookmark** của Recordset.

Thí dụ khi user clicks nút Go, ta muốn nhớ vị trí của record lúc ấy để sau này quay trở lại khi User clicks nút **Go Back**. Bạn hãy thêm vào Form một button tên **CmdGoBack** với Caption **Go Back**. Ta sẽ thêm một variable tên **LastBookmark** loại data type **Variant**:

```
Dim LastBookMark As Variant
```

Lúc đầu button CmdGoBack invisible, và chỉ trở nên visible sau khi user clicks nút Go. Ta thêm các hàng codes sau vào Sub CmdGo_Click() như sau:

```

' Remember location of current record
LastBookMark = myRS.BookMark
CmdGoback.Visible = True

```

Dưới đây là code để quay trở lại vị trí current record trước đây trong Recordset:

```

Private Sub CmdGoback_Click()
' Reposition record to last position
myRS.BookMark = LastBookMark
' Redisplay details or current record
Displayrecord
End Sub

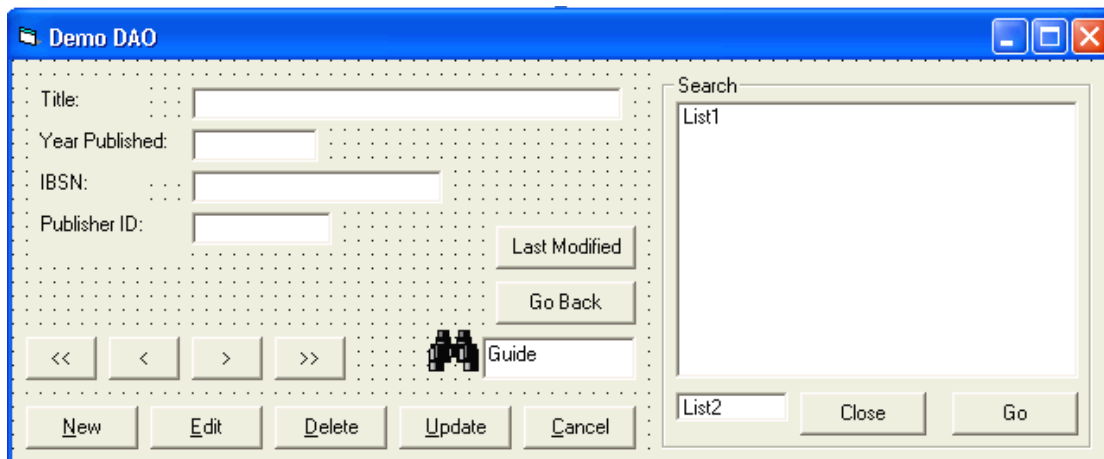
```

LastModified

LastModified là vị trí của record vừa mới được sửa đổi hay thêm vào trong Recordset. Để thử điều này bạn hãy thêm một button invisible tên **CmdLastModified** với caption là **Last Modified**. Button này chỉ hiện ra sau khi user clicks Update. Bất cứ lúc nào bạn Click nút CmdLastModified, record mới vừa được sửa đổi hay thêm vào sẽ hiển thị:


```
Private Sub CmdLastModified_Click()  
    ' Reposition record to last position  
    myRS.BookMark = myRS.LastModified  
    ' Redisplay details or current record  
    Displayrecord  
End Sub
```

Dưới đây là hình của Form lúc đang được thiết kế:



Lập trình với ADO (phần I)

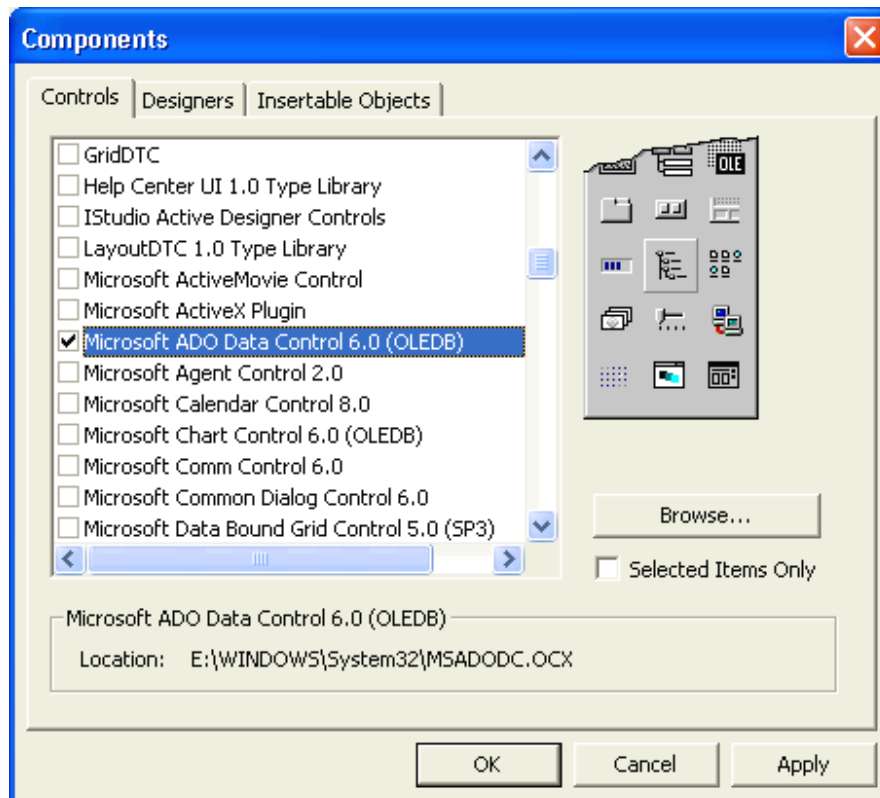
Control Data ADO

Visual Basic 6 cho ta sự lựa chọn về kỹ thuật khi lập trình với database, hoặc là dùng **DAO** như trong hai bài trước, hoặc là dùng **ADO (ActiveX Data Objects)**.

Sự khác biệt chính giữa ADO và DAO là ADO cho phép ta làm việc với mọi loại nguồn dữ kiện (data sources), không nhất thiết phải là Access database hay ODBC. Nguồn dữ kiện có thể là danh sách các địa chỉ Email, hay một file text string, trong đó mỗi hàng là một record gồm những fields ngăn cách bởi các dấu phẩy (**comma separated values**).

Nếu trong DAO ta dùng thẳng tên của MSAccess Database thì trong ADO cho ta **nối với (connect)** một database qua một Connection bằng cách chỉ định một **Connection String**. Trong Connection String có **Database Provider** (thí dụ như Jet, ISAM, Oracle, SQLServer..v.v.), tên Database, **UserName/Password** để logon một database .v.v.. Sau đó ta có thể **lấy về (extract)** những recordsets, và cập nhật hóa các records bằng cách dùng những **lệnh SQL** trên các tables hay dùng những **stored procedures** bên trong database.

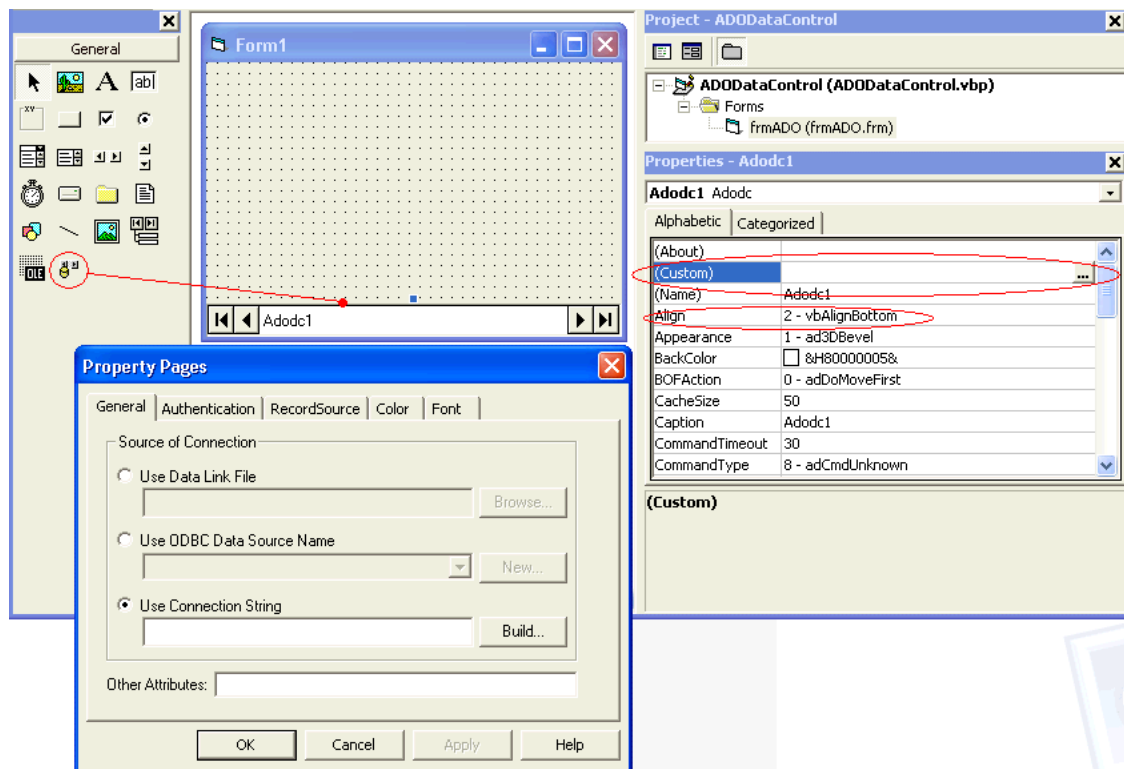
Bình thường, khi ta mới khởi động một project VB6 mới, Control **Data ADO** không có sẵn trong IDE. Muốn có nó, bạn hãy dùng Menu Command **Project | Components...**, rồi chọn **Microsoft ADO Data Control 6.0 (OLEDB)** từ giao diện Components như dưới đây:



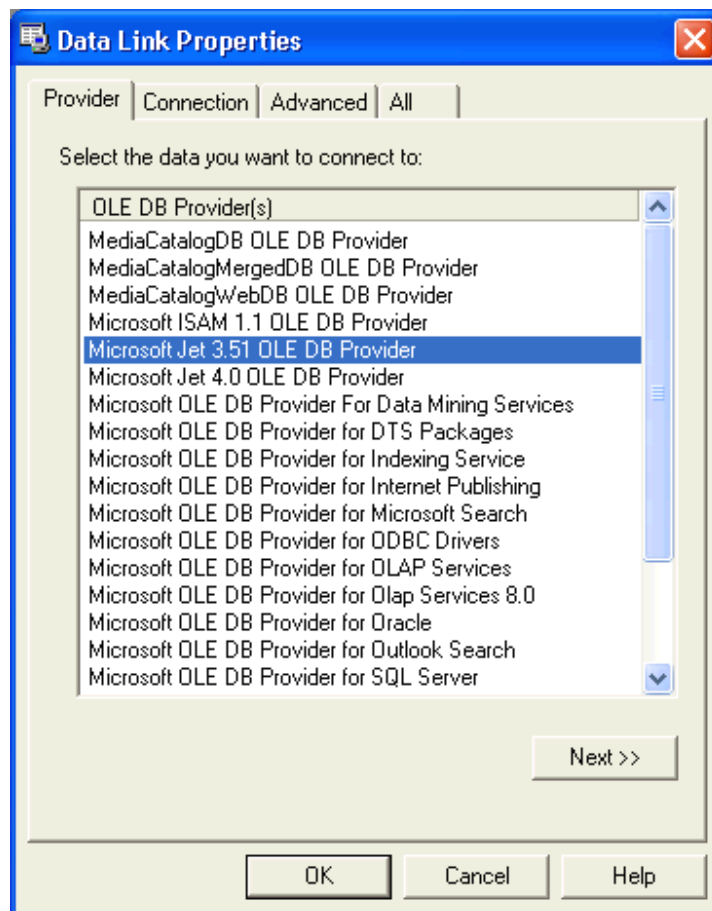
Bạn hãy bắt đầu một dự án VB6 mới, cho nó tên **ADODataControl** bằng cách click tên project trong Project Explorer bên phải rồi edit property Name trong Properties Window. Sửa tên của form chính thành **frmADO**, và đánh câu **ADO DataControl Demo** vào Caption của nó.

DoubleClick lên Icon của Control Data ADO trong Toolbox. Một Control Data ADO tên **Adodc1** sẽ hiện ra trên Form. Muốn cho nó nằm bên dưới Form, giống như một StatusBar, hãy set **property Align** của nó trong Properties Window thành **2 - vbAlignBottom**.

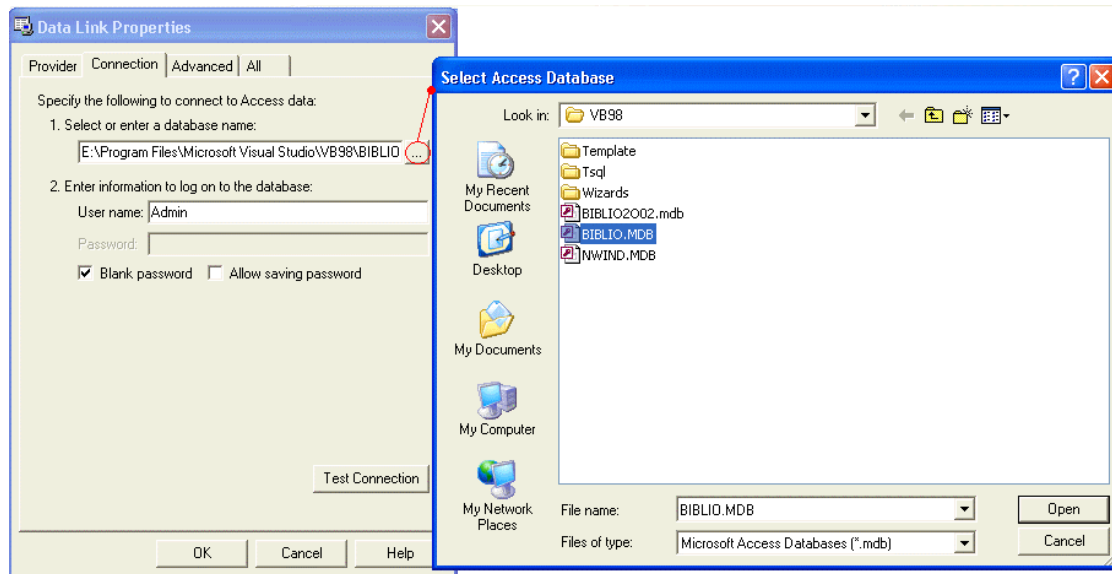
Click bên phải hàng **property (Custom)**, kể đó click lên nút browse có ba chấm để giao thoại **Property Pages** hiện ra. Trong giao thoại này, trên **Tab General** chọn Radio (Option) Button **Use Connection String** rồi click nút **Build....**



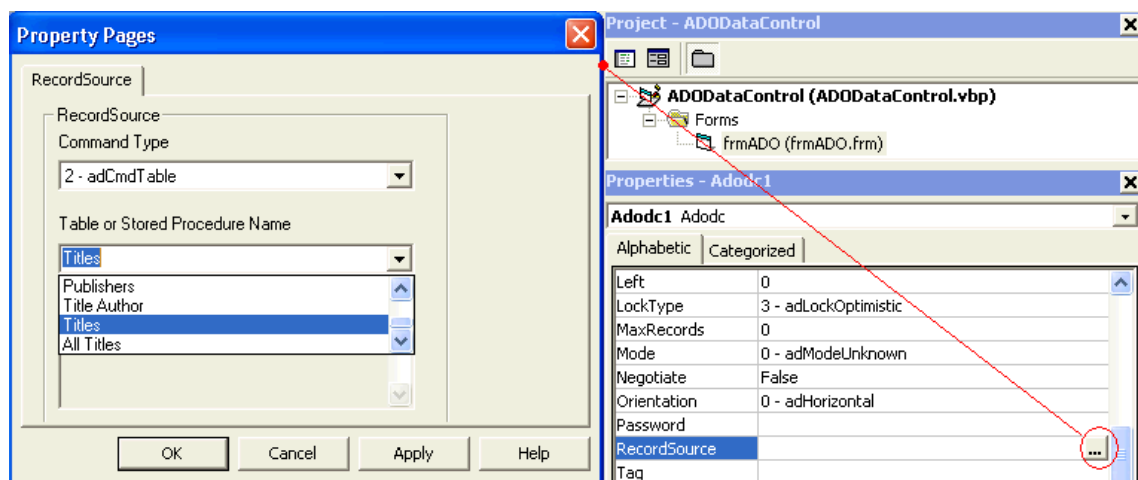
Trong giao thoại **Data Link Properties**, Tab **Provider**, chọn **Microsoft Jet 3.51 OLE DB Provider**, rồi click nút **Next >>** hay Tab **Connection**.



Ở chỗ **Select or enter a database name** ta chọn **E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB**, trong computer của bạn có thể file ấy nằm trên disk **C** hay **D**. Sau đó, bạn có thể click nút **Test Connection** phía dưới để thử xem connection có được thiết lập tốt không.



Lập connection xong rồi, ta chỉ định muốn lấy gì về làm Recordset bằng cách click **property Recordsource** của Adodc1. Trong giao diện Property Pages của nó chọn **2-adCmdTable** làm **Command Type**, kế đó mở Combo box cho **Table or Stored Procedure Name** để chọn **table Titles**.



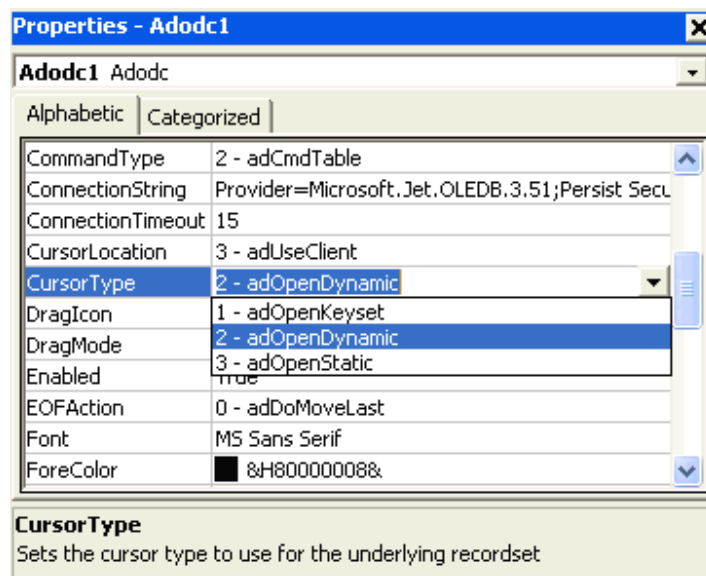
Tùy theo cách ta dùng Recordset trong ADO, nó có ba loại và được gọi là **Cursor Type**. Cursor chẳng qua là một tên khác của Recordset:

- **Static Cursor:** Static Cursor cho bạn một static copy (bản sao cứng nhắc) của các records. Trong lúc bạn dùng Static Cursor, nếu có ai khác sửa đổi hay thêm, bớt gì vào recordset bạn sẽ không thấy.
- **Keyset Cursor:** Keyset Cursor hơn Static Cursor ở chỗ trong lúc bạn dùng nó, nếu có ai sửa đổi record nào bạn sẽ

biết. Nếu ai delete record nào, bạn sẽ không thấy nó nữa. Tuy nhiên bạn sẽ không biết nếu có ai thêm một record nào vào recordset.

- **Dynamic Cursor:** Như chữ **sống động (dynamic)** hàm ý, trong lúc bạn đang dùng một Dynamic Cursor, nếu có ai khác sửa đổi hay thêm, bớt gì vào recordset bạn sẽ thấy hết.

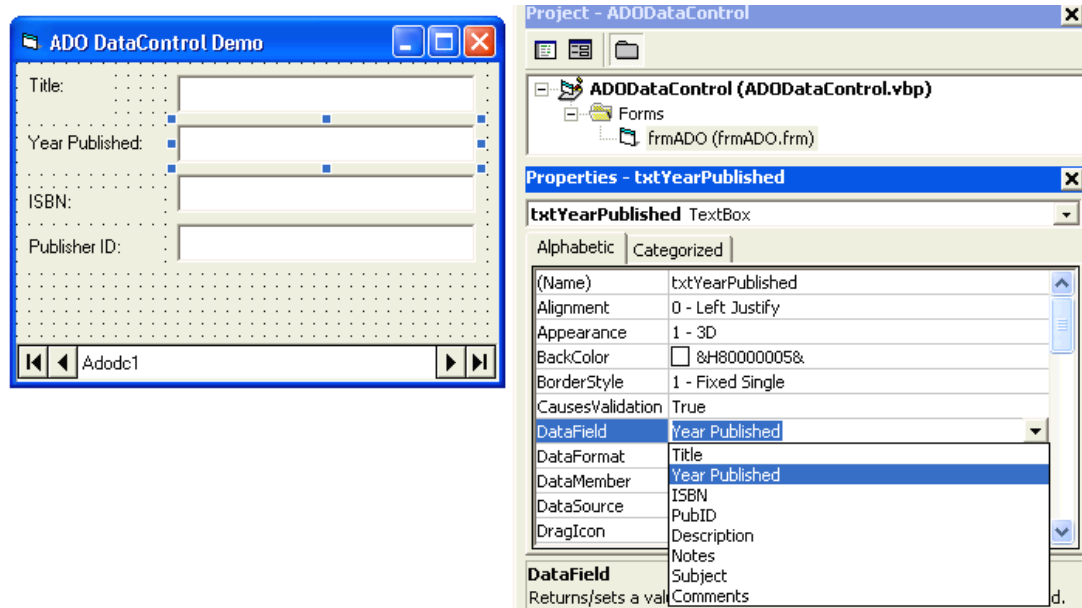
Bạn hãy chọn trị số **2-adOpenDynamic** cho property Cursor Type của Adodc1:



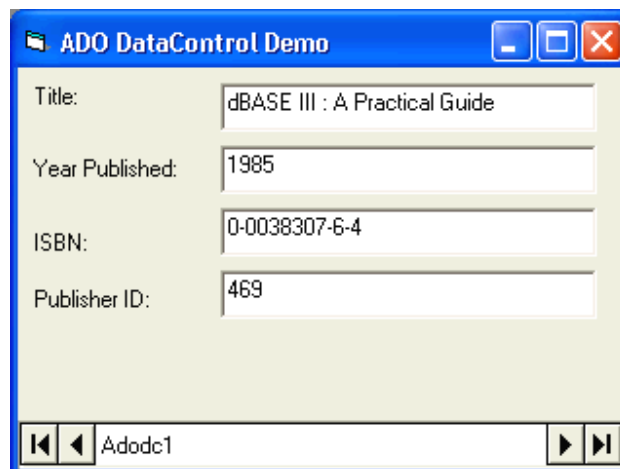
Bây giờ bạn hãy đặt lên Form 4 labels với captions: **Title**, **Year Published**, **ISBN** và **Publisher ID**. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là **txtTitle**, **txtYearPublished**, **txtISBN** và **txtPublisherID**.

Để thực hiện Data Binding, bạn hãy chọn textbox **txtYearPublished** (năm xuất bản), rồi set **property Datasource** của nó trong Properties Window thành **Adodc1**. Khi click lên **property DataField** của txtYearPublished và mở ComboBox ra bạn sẽ thấy liệt kê tên các Fields trong table Titles. Đó là vì Adodc1 được coi như trung gian lấy table Titles từ database. Ở đây ta sẽ chọn cột Year Published.

Lập lại công tác này cho 3 textboxes kia, và chọn các cột Title (Tiêu đề), ISBN (số lý lịch trong thư viện quốc tế), và PubID (số lý lịch nhà xuất bản) làm DataField cho chúng.



Đến đây, mặc dầu chưa viết một hàng code nào, bạn có thể chạy chương trình và nó sẽ hiển thị như dưới đây:

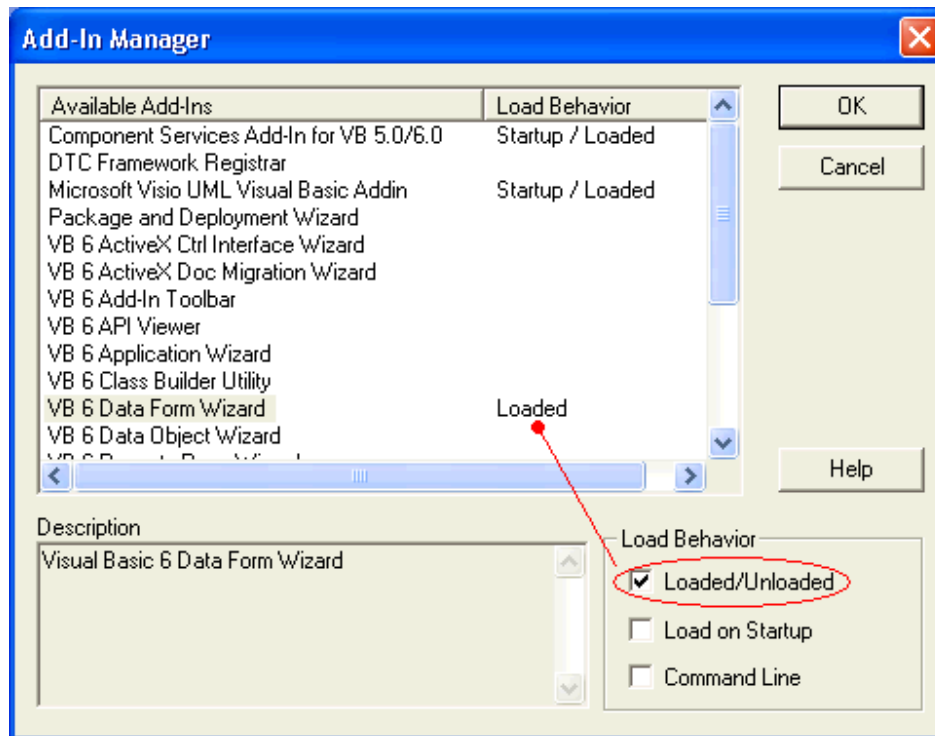


Data Form Wizard

Để giúp lập trình viên thiết kế các data forms nhanh hơn, VB6 cho ta **Data Form Wizard** để generate (phát sinh) ra một form có hỗ trợ Edit, Add và Delete records.

Bây giờ bạn hãy khởi động một standard project VB6 mới, tên **ADOCClass** và copy MS Access file BIBLIO.MDB, tức là database, vào trong cùng folder của dự án mới này.

Muốn dùng Data Form Wizard, trước hết ta phải thêm nó vào môi trường phát triển (IDE) của VB6. Bạn hãy dùng IDE Menu Command **Add-Ins | Add-In Manager....** Chọn **VB6 Data Form Wizard** trong giao thoại, rồi click Checkbox **Loaded/Unloaded** để chữ Loaded hiện bên phải hàng "VB6 Data Form Wizard" như trong hình dưới đây:

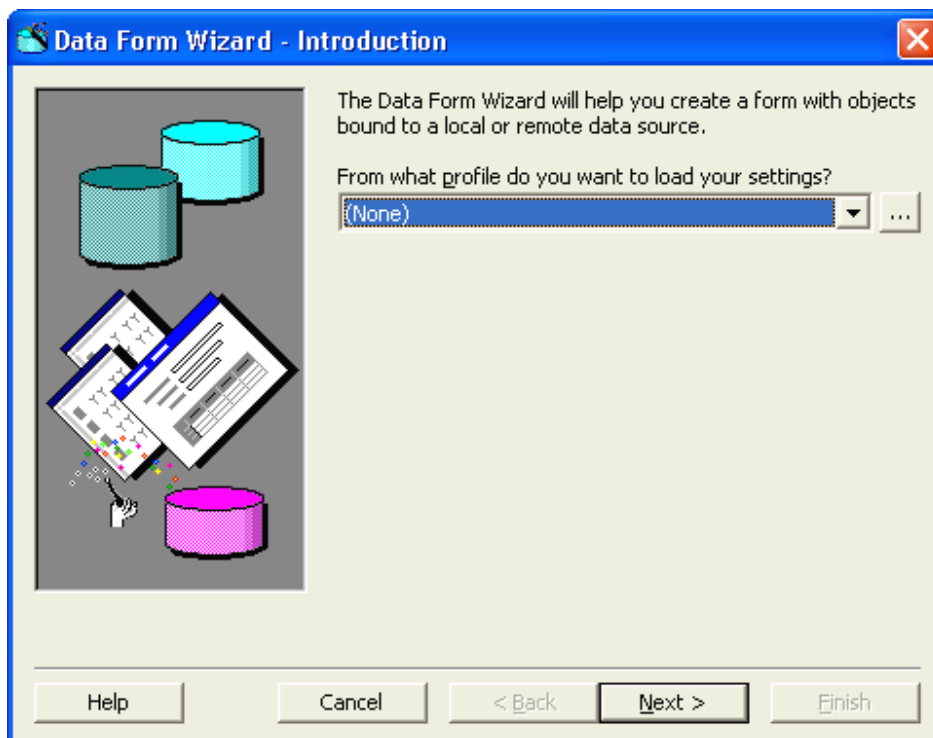


Nếu bạn muốn mỗi lần khởi động VB6 IDE là có sẵn Data Form Wizard trong menu Add-Ins thì ngoài option Loaded, bạn click thêm check box **Load on Startup**.

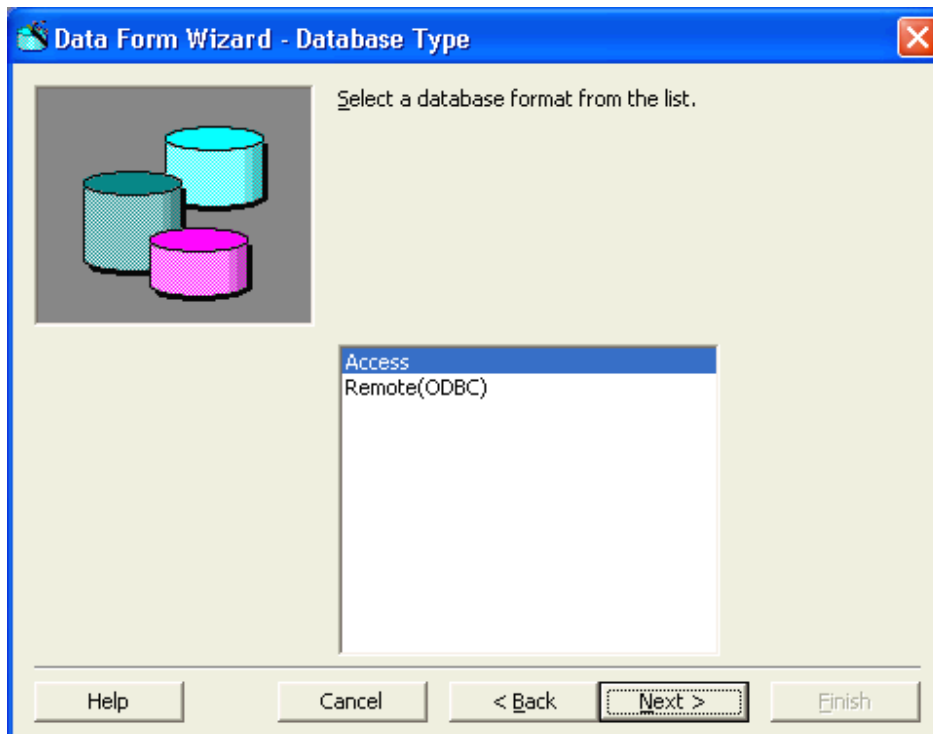
Một **Add-In** là một menu Item mới mà ta có thể thêm vào một chương trình ứng dụng có sẵn. Thường thường, người ta dùng Add-Ins để thêm chức năng cho một chương trình, làm như là chương trình đã có sẵn chức năng ấy từ đầu. Bạn hãy khởi động Data Form Wizard từ IDE Menu Command mới **Add-Ins | Data Form Wizard...**



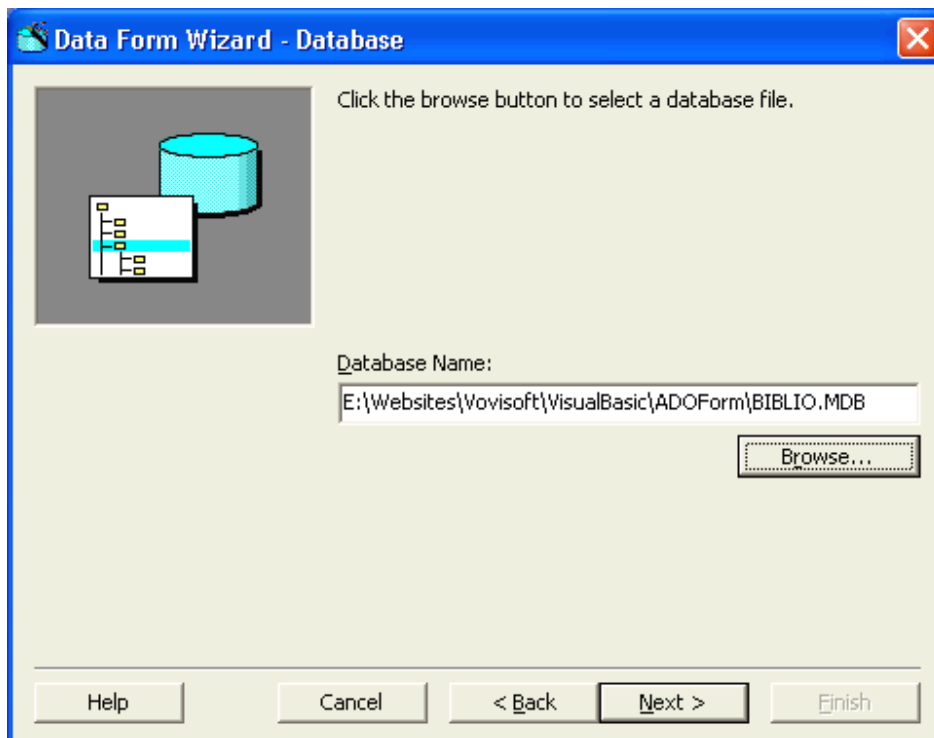
Khi trang **Data Form Wizard - Introduction** hiện ra, click **Next**



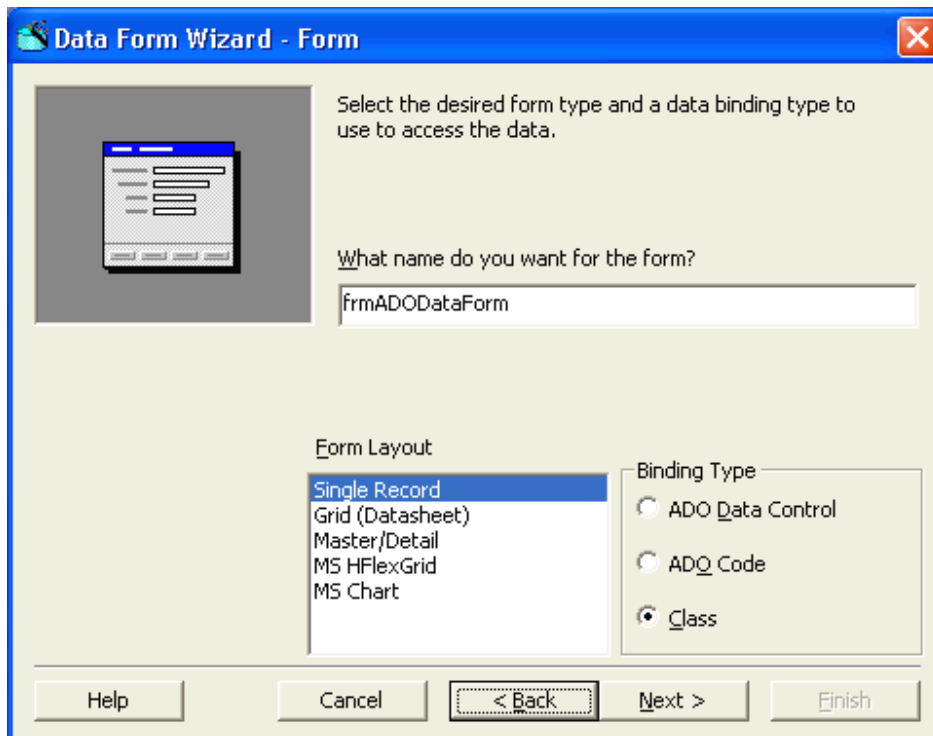
Trong trang kế đó chọn **Access** làm **Database Type**.



Trong trang Database, click **Browse** để chọn một MS Access database file. Ở đây ta chọn file **BIBLIO.MDB** từ chính folder của chương trình này. Đoạn click **Next**.



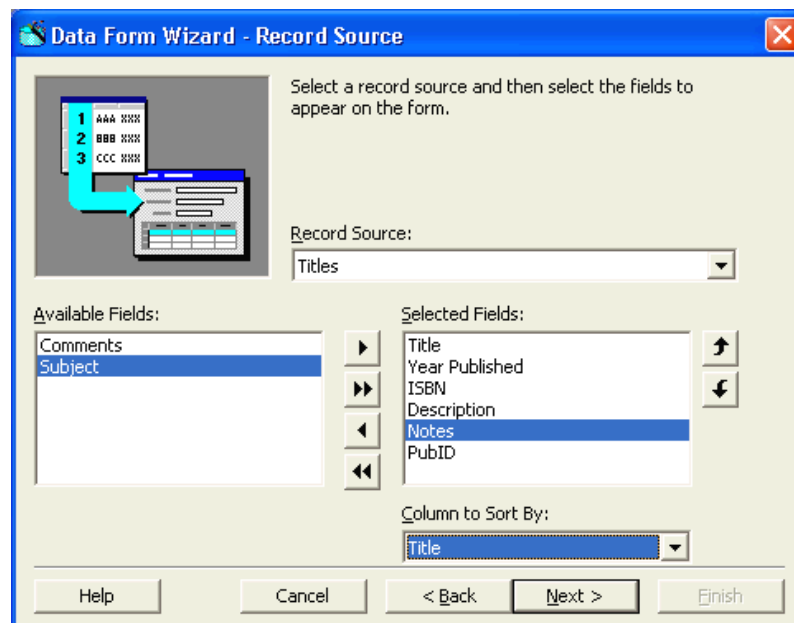
Trong trang Form, ta chọn **Single Record** cho **Form Layout** và **Class** cho **Binding Type**. Đoạn click **Next**. Nếu ta chọn **ADO Data Control** thì kết quả sẽ giống giống như khi ta dùng Control Data DAO như trong một bài trước.



Trong trang Record Source ta chọn **table Titles**. Listbox của **Available Fields** sẽ hiển thị các fields của table Titles. Sau khi chọn một field bằng cách click lên tên field ấy trong Listbox, nếu bạn click hình tam giác chỉ qua phải thì tên field ấy sẽ được dời qua nằm dưới cùng trong Listbox **Selected Fields** bên phải.

Nếu bạn click hình hai tam giác chỉ qua bên phải thì tất cả mọi fields còn lại bên trái sẽ được dời qua bên phải. Bạn cũng có thể sắp đặt vị trí của các selected fields bằng cách click lên tên field ấy rồi click hình mũi tên chỉ lên hay xuống để di chuyển field ấy lên hay xuống trong danh sách các fields.

Ngoài ra, bạn hãy chọn Title làm **Column to Sort By** trong cái Combobox của nó để các records trong Recordset được sắp xếp theo thứ tự ABC (alphabetical order) của field Tiêu đề (Title).



Trong trang Control Selection, ta sẽ để y nguyên để có đủ mọi buttons. Bạn hãy click **Next**.



Khi Data Form Wizard chấm dứt, nó sẽ generate form **frmADODataForm**. Bạn hãy remove Form1 và dùng Menu Command **Project | ADODataControl Properties...** để đổi **Startup Object** thành frmADODataForm. Thế là tạm xong chương trình để Edit các records của table Titles.

Chúng ta hãy quan sát cái Form và phần code được Data Form Wizard generated. Trong frmADODataForm, các textboxes làm thành một array tên **txtFields**. Mọi textbox đều có property **DataField** định sẵn tên field của table Titles. Thí dụ như **txtFields(2)** có DataField là **ISBN**. Form chính không dùng Control Data ADO nhưng dùng một Object của **class clsTitles**.

Phần Initialisation của class **clsTitles** là Open một Connection và lấy về một Dataset có tên **DataMember** là **Primary** như sau:

```
Private Sub Class_Initialize()
    Dim db As Connection
    Set db = New Connection
    db.CursorLocation = adUseClient
    ' Open connection
    db.Open "PROVIDER=Microsoft.Jet.OLEDB.3.51;Data
Source=E:\Websites\Vovisoft\VisualBasic\ADOForm\BIBLIO.MDB;"
    ' Instantiate ADO recordset
    Set adoPrimaryRS = New Recordset
    ' Retrieve data for Recordset
    adoPrimaryRS.Open "select Title,[Year
Published],ISBN,Description,Notes,PubID from Titles Order by Title", _
        db, adOpenStatic, _
        adLockOptimistic
    ' Define the only data member, named Primary
    DataMembers.Add "Primary"
End Sub
```

Về vị trí của database, nếu bạn không muốn nó chết cứng ở một folder nào thì dùng **App.Path** để xác định mối liên hệ giữa vị trí của database và folder của chính chương trình đang chạy, thí dụ như:

```
db.Open "PROVIDER=Microsoft.Jet.OLEDB.3.51;Data Source=" & App.Path &
"\BIBLIO.MDB;"
```

Trong **Sub Form_Load**, ta có thể dùng **For Each** để đi qua hết các textboxes trong array txtFields. Vì property Datasource của textbox là một Object nên ta dùng keyword **Set** để point nó đến Object **PrimaryCLS**. Đồng thời ta cũng phải chỉ định tên của DataMember của mỗi textbox là Primary:

```
Private Sub Form_Load()
    ' Instantiate an Object of class clsTitles
    Set PrimaryCLS = New clsTitles
    Dim oText As TextBox
    ' Iterate through each textbox in the array txtFields
    ' Bind the text boxes to the data source, i.e. PrimaryCLS
    For Each oText In Me.txtFields
        oText.DataMember = "Primary"
        ' Use Set because property Datasource is an Object
        Set oText.DataSource = PrimaryCLS
    Next
End Sub
```

Khi sự di chuyển từ record này đến record khác chấm dứt, chính Recordset có raise **Event MoveComplete**. Event ấy được handled (giải quyết) trong class clsTitles bằng cách lại raise **Event MoveComplete** để nó được handled trong Form.

Muốn handle Event trong clsTitles ta phải declare recordset adoPrimaryRS với WithEvents:

```
Dim WithEvents adoPrimaryRS As Recordset
Private Sub adoPrimaryRS_MoveComplete(ByVal adReason As
ADODB.EventReasonEnum, _
    ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum,
    ByVal pRecordset As ADODB.Recordset)
    ' Raise event to be handled by main form
    RaiseEvent MoveComplete
End Sub
```

Và trong Form ta cũng phải declare (object clsTitles) PrimaryCLS với WithEvents:

```
Private WithEvents PrimaryCLS As clsTitles
```

Trong Form, Event MoveComplete sẽ làm hiển thị vị trí tuyệt đối (Absolute Position) của record bằng code dưới đây:

```
Private Sub PrimaryCLS_MoveComplete()
    ' This will display the current record position for this recordset
    lblStatus.Caption = "Record: " & CStr(PrimaryCLS.AbsolutePosition)
End Sub
```

Khi user clicks **Refresh**, các textboxes sẽ được hiển thị lại với chi tiết mới nhất của record từ trong recordset, nhờ khi có ai khác đã sửa đổi record. **Method Requery** của clsTitles lại gọi method Requery của Recordset như sau:

```
Private Sub cmdRefresh Click()  
    'This is only needed for multi user applications  
    On Error GoTo RefreshErr  
    ' fetch the latest copy of Recordset  
    PrimaryCLS.Requery  
    Exit Sub  
RefreshErr:  
    MsgBox Err.Description  
End Sub  
  
'In Class clsTitles  
Public Sub Requery()  
    ' Fetch latest copy of record  
    adoPrimaryRS.Requery  
    DataMemberChanged "Primary"  
End Sub
```