Chương 7:

LẬP TRÌNH CƠ SỞ DỮ LIỆU

I. TỔNG QUAN VỀ LẬP TRÌNH CƠ SỞ DỮ LIỆU

1. Một số khái niệm cơ bản

Ngôn ngữ lập trình Visual Basic, cho tới thời điểm hiện tại là ngôn ngữ phù hợp nhất cho các ứng dụng truy cập cơ sở dữ liệu. Ta thấy:

- Một chương trình quản lý thông thường sẽ đi kèm với một cơ sở dữ liệu chứa đựng các thông tin về lĩnh vực cần quản lý.
- Công việc của người lập trình là tạo ra các giao diện cho phép cập nhật, tìm kiếm, thống kê báo cáo trên cơ sở dữ liệu đó.

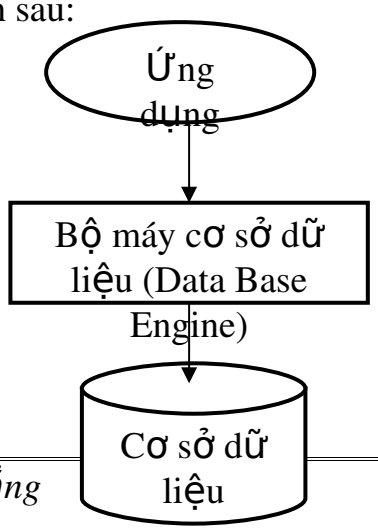
Chương này tập trung giới thiệu các phương pháp để tạo ra các ứng dụng như vậy. Ta làm quen với một số khái niệm

- *Recordset:* là một "kiểu dữ liệu" trong Visual Basic. Trong lập trình cơ sở dữ liệu, ta thường xuyên thực hiện thao tác trên các bảng dữ liệu hoặc trên một tập bản ghi của bảng. Khi đó, đối tượng Recordset được tạo ra dùng để chứa tập hợp các bản ghi mà ta lấy về từ một hay nhiều bảng nào đó trong cơ sở dữ liệu.

Như vậy, Recordset tương tự như một cấu trúc dữ liệu để chứa các bản ghi lấy được và một số phương thức thao tác trên tập bản ghi đó.

Về bản chất, một biến có kiểu Recordset, tại một thời điểm có thể chứa một bảng dữ liệu. Tuy nhiên, đó là sự mở rộng khái niệm bảng dữ liệu bằng cách thêm vào đó các thao tác cơ bản trên bảng dữ liệu đó.

- *Bộ máy cơ sở dữ liệu (DataBase Engine):* Một ứng dụng sẽ kết nối tới cơ sở dữ liệu theo mô hình sau:



Hình 1: Mô hình kết nối cơ sở dữ liệu từ ứng dụng Visual Basic

Vì các cơ sở dữ liệu được tạo ra từ nhiều hệ quản trị cơ sở dữ liệu khác nhau nên một ứng dụng Visual Basic không thể truy cập trực tiếp chúng mà thông qua các bộ máy cơ sở dữ liệu khác nhau. Các bộ máy này đóng vai trò trung gian trong quá trình giao tiếp giữa ứng dụng với cơ sở dữ liệu.

- *Connection:* Một biến đối tượng có kiểu Connection được dùng để kết nối với một cơ sở dữ liệu. Thông thường, quá trình kết nối cần có các thông tin đầu vào như: Tên bộ máy cơ sở dữ liệu dùng để kết nối, tên, đường dẫn tới cơ sở dữ liệu, tên người truy cập, mật khẩu ..v..v…Các thông tin đó được gộp chung trong một chuỗi gọi là chuỗi kết nối (ConnectString). Khi sử dụng biến có kiểu Connection để kết nối tới cơ sở dữ liệu, ta cần xác định ConnectString của biến đó và sử dụng chuỗi kết nối này để kết nối tới cơ sở dữ liệu, gọi là "Mở cơ sở dữ liệu"

- *Các OCX:* Là các điều khiển có thể đưa vào một ứng dụng Visual Basic. Các điều khiển này có thể là nội tại (Build - in: có sẵn trong môi trường lập trình Visual Basic) hoặc do các nhà lập trình tạo nên và đưa vào môi trường lập trình này (còn gọi là Add - in). Ta có thể tìm kiếm các OCX bằng cách chọn: Project\ Components.

- *Các ActiveX:* Là các kiểu dữ liệu đặc biệt có thể dùng trong khi lập trình. Thực chất đây là các lớp đối tượng đã được định nghĩa sẵn và cung cấp cho chúng ta những thuộc tính, phương thức rất hữu dụng khi lập trình. Như vậy, ngoài các kiểu dữ liệu cơ bản như: integer, single, double, string… ta còn có các kiểu dữ liệu khác tiện dụng hơn, chính là các ActiveX. Có thể lựa chọn các kiểu dữ liệu ActiveX bằng cách chọn: Project\ References.

2. Tạo một cơ sở dữ liệu

a. Các kiểu dữ liệu cơ bản:

Cơ sở dữ liệu nội tại của Visual Basic cung cấp rất nhiều kiểu dữ liệu khác nhau. Sau đây là một số kiểu thông dụng có thể dùng để định nghĩa kiểu cho các trường trong bảng dữ liệu:

Tên kiểu	ý nghĩa - Độ rộng
Boolean	Kiểu logic chỉ nhận 2 giá trị True/ False

Byte	Kiểu số nguyên ngắn 1 byte
Integer	Kiểu số nguyên 2 byte
Long	Kiểu số nguyên dài 4 byte
Single	Kiểu số thực đơn 4 byte
Double	Kiểu số thực kép 8 byte
Date/ Time	Kiểu ngày/ giờ
Text	Kiểu văn bản tới 255 ký tự
Memo	Kiểu văn bản ghi chú tới 32000 ký tự

Mét sè kiÓu d÷ liÖu cña Visual Basic t¬ng thÝch víi c¸c kiÓu d÷ liÖu trong Access. Tuy nhiªn mét sè kiÓu kh¸c kh«ng thÓ ®îc ®Þnh nghÜa trong m«i trêng Access.

b. T¹o c¬ së d÷ liÖu tõ Access

Mét øng dông Visual Basic cã thÓ truy cËp c¬ së d÷ liÖu ®îc thiÕt kÕ trong m«i têng Access (hay cã ®Þnh d¹ng Access). ViÖc t¹o c¬ së d÷ liÖu tõ hÖ qu¶n trÞ c¬ së d÷ liÖu Access lµ t¬ng ®èi ®¬n gi¶n vµ quen thuéc. Ở đây chỉ giới thiệu những thao tác cơ bản:

- Vào môi trường Microsoft Access.
- Mở một cơ sở dữ liệu mới để thiết kế.
- Tạo các bảng (table) của cơ sở dữ liệu (Create Table in Design View):
  + Đặt tên trường, kiểu dữ liệu của trường và một số thuộc tính khác của trường dữ liệu đang định nghĩa.
  + Tạo khoá cho bảng quan hệ (nếu cần)
- Tạo các truy vấn trong cơ sở dữ liệu (nếu cần). Các truy vấn có vai trò như một bảng quan hệ trong môi trường lập trình Visual Basic.
- Tạo các quan hệ ràng buộc tính toàn vẹn dữ liệu Relationship (nếu cần).

Chú ý:

Thông thường, ta sử dụng môi trường Access để tạo cơ sở dữ liệu dùng cho ứng dụng đang xây dựng trong môi trường Visual Basic.

c. Tạo cơ sở dữ liệu bằng trình Data Manager của Visual Studio

Việc tạo cơ sở dữ liệu bằng Microsoft Access là tương đối thuận tiện và đơn giản. Tuy nhiên, một khó khăn đặt ra là chúng ta luôn phải cài đặt môi trường Microsoft Access kèm theo bộ Visual Studio. Một bất tiện nảy sinh là: nếu ứng dụng Visual Basic cần truy cập cơ sở dữ liệu thuộc hệ quản trị cơ

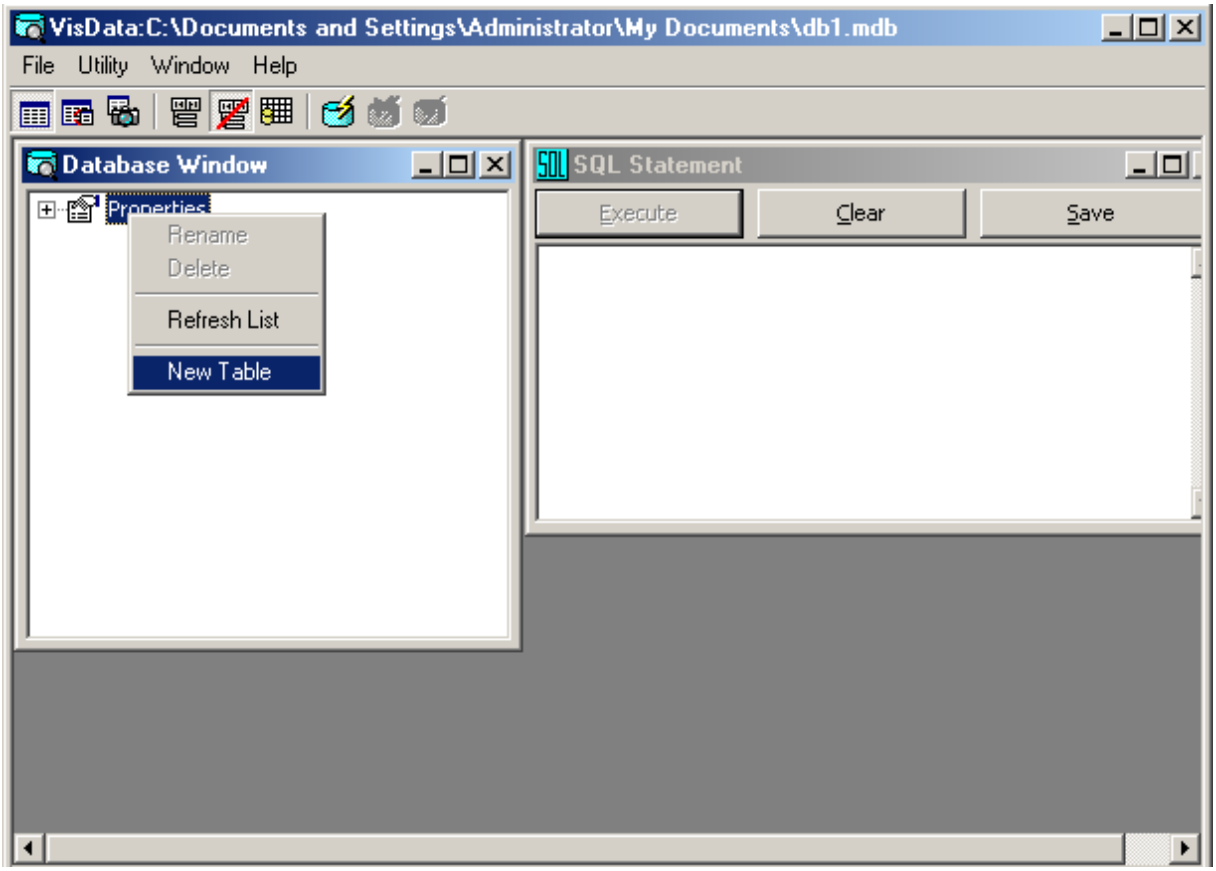
sở dữ liệu nào, ta đều sử dụng hệ quản trị cơ sở dữ liệu đó để tạo dữ liệu thì bắt buộc phải cài đặt hệ quản trị cơ sở dữ liệu đó kèm theo.

Để giải quyết vấn đề trên, Visual Basic đã xây dựng một môi trường giúp ta tạo ra các cơ sở dữ liệu thuộc các hệ quản trị cơ sở dữ liệu khác nhau (hay có định dạng tương thích nhiều hệ quản trị cơ sở dữ liệu khác nhau) như Access, Dbase, Foxpro, Paradox… mà không cần phải cài đặt các hệ quản trị cơ sở dữ liệu này trên máy.

Để tạo cơ sở dữ liệu bằng Visual Data Manager ta làm như sau:

- Từ cửa sổ lập trình Visual Basic, Chọn Add – Ins\ Visual Data Manager. Khi đó môi trường thiết kế cơ sở dữ liệu sẽ xuất hiện.

- Chọn File\ New và chọn một loại cơ sở dữ liệu muốn tạo (có thể chọn MS. Access). Khi đó cửa sổ tạo cơ sở dữ liệu sẽ xuất hiện như hình sau:



Hình 2: Cửa sổ tạo cơ sở dữ liệu của Visual Data Manager

- Trên Data Base Window, chọn Properties, kích phải chọn New Table để tạo bảng quan hệ mới. Khi đó ta cần:

  + Đặt tên bảng quan hệ cần tạo.

  + Chọn Add Field để thêm một trường mới vào bảng quan hệ đang định nghĩa.

  + Để thêm trường mới, ta cần nhập tên trường, kiểu dữ liệu của trường và đặt một số thuộc tính cho trường dữ liệu đó (nếu cần).

- Chọn Build table để lưu bảng quan hệ vừa tạo.

Khi đã có các bảng quan hệ, ta có thể tạo các query tuỳ ý bằng cách kích phải vào một trong các bảng quan hệ đã tạo và chọn New Query.

3. Các hàm thường dùng

Trong quá trình lập trình cơ sở dữ liệu, ta thường dùng các hàm sau để kiểm tra tính hợp lệ và chuẩn hoá dữ liệu:

Tên hàm	Chức năng
Str(Number)	Đổi các số thành chuỗi
Chr()	Đổi số thành ký tự tương ứng (ASCII)
Trim(Chuỗi)	Cắt các ký tự trống hai bên chuỗi
Ltrim(Chuỗi)	Cắt các ký tự trống bên trái chuỗi
Rtrim(chuỗi)	Cắt các ký tự trống bên phải chuỗi
IsDate(Chuỗi)	Kiểm tra xem chuỗi có dạng Date không (True/False)
IsNumeric(Chuỗi)	Kiểm tra xem chuỗi có dạng số không (True/False)
IsString(Giá trị)	Kiểm tra xem giá trị có ở dạng xâu không (True/False)
CDate(Giá trị)	Đổi giá trị có định dạng date sang kiểu Date
CString(Giá trị)	Đổi giá trị sang định dạng string
Val(Giá trị)	Đổi giá trị có định dạng số sang kiểu số

C¸c hµm nµy thêng ®îc dïng ®Ó kiÓm tra xem d÷ liÖu ®· phï hîp cha (tríc khi lu vµo c¬ së d÷ liÖu) hoÆc chuÈn ho¸ d÷ liÖu (nÕu cÇn).

II. CẬP NHẬT DỮ LIỆU QUA CÁC ĐIỀU KHIỂN

1. Giới thiệu chung

Một công việc rất quan trọng, có trong hầu hết các ứng dụng là tạo ra các giao diện dùng cho việc cập nhật dữ liệu. Visual Basic đã cung cấp các điều khiển rất hữu dụng cho người lập trình.

Trong phần này, chúng ta sẽ tìm hiểu về hai cặp điều khiển cơ bản để tạo các form cập nhật dữ liệu cho ứng dụng: Cặp Data và DBGrid, cặp ADODC và DataGrid.

Trong các điều khiển trên thì:

- Điều khiển Data và ADODC: dùng để kết nối tới cơ sở dữ liệu và tới bảng quan hệ cần cập nhật.

- Điều khiển DBGrid và DataGrid: là các lưới làm nhiệm vụ hiển thị dữ liệu trong bảng đã được điều khiển Data và ADODC kết nối tới. Vì lý do đó, chúng thường đi theo cặp.

Với mỗi cặp điều khiển, ta cần:

- Nắm được cách thuộc tính của chúng để sử dụng khi cần

- Nắm được các phương thức của chúng để tạo các thao tác khi truy cập dữ liệu.

2. Cặp Data và DB Grid

Điều khiển Data luôn có sẵn trên thanh công cụ. Để có được điều khiển DBGrid, ta chọn: Chọn Project\ Components\ Microsoft Data Bound Grid Control 5.0 (hoặc cao hơn).

Để hiển thị được dữ liệu qua 2 điều khiển trên, ta làm theo các bước sau:

Bước 1: Vẽ hai điều khiển trên vào form.

Bước 2: Thiết đặt các thuộc tính cho điều khiển Data để kết nối tới cơ sở dữ liệu và bảng quan hệ cần thiết bằng cách:

- Chọn điều khiển Data.
- Đặt thuộc tính DataBaseName bằng cách chọn cơ sở dữ liệu muốn kế nối tới.
- Đặt thuộc tính RecordSource bằng cách chọn tên bảng quan hệ (hoặc Query) muốn thao tác.

Bước 3: Chọn điều khiển DBGrid.

Bước 4: Thiết đặt mối quan hệ giữa hai điều khiển:

- Đặt thuộc tính DataSource của điều khiển DBGrid bằng cách chọn tên của điều khiển Data trên form.

Đến đây, công việc hiển thị dữ liệu đã hoàn tất. Tuy nhiên, ta cần quan tâm tới các thuộc tính khác của các điều khiển trên để thiết đặt khi cần. Danh sách các thuộc tính cơ bản của 2 điều khiển trên được liệt kê trong bảng dưới đây:

[1]. Danh sách một số thuộc tính của điều khiển Data

Stt	Tên thuộc tính	Ý nghÜa
1	Connect	KiÓu c¬ së d÷ liÖu sÏ lµm viÖc (Access, Fox, Oracle…). MÆc ®Þnh lµ Access
2	Caption	Tiªu ®Ò sÏ hiÖn lªn trªn Data
3	DataBase name	Tªn c¬ së d÷ liÖu sÏ kÕt nèi
4	DefaultType	KiÓu bé m¸y c¬ së d÷ liÖu dïng ®Ó kÕt nèi. NÕu lµm viÖc trªn c¬ së d÷ liÖu Access th× chän UseJet.
5	Record Source	Tªn b¶ng trong c¬ së d÷ liÖu võa chän trong Data Base name.

[2]. Một số thuộc tính của DBGrid

(chọn điều khiển, kích phải, chọn Properties)

Stt	Tên thuộc tính	ý nghĩa
1	AllowAdd new	Cho phép không cho phép thêm bản ghi mới ngay trên DB Grid
2	AllowArrow	Cho phép/ không cho phép sử dụng các phím mũi tên khi cập nhật
3	Allow Delete	Cho phép/ không cho phép xoá dữ liệu ngay trên DB Grid
4	Allow Update	Cho phép/ không cho phép sửa đổi dữ liệu ngay trên DB Grid
5	Column Header	Có/ không có phần tiêu đề cho mỗi cột
6	Data Mode	Bound/ Unbound cho phép ràng buộc/ không ràng buộc tới bảng dữ liệu của data.
7	Data source	Nguồn dữ liệu hiển thị, thường chọn tên của điều khiển Data
8	DefColumn Width	Đặt độ rộng mặc định cho các cột. Nếu không đặt thì để 0
9	HeadLine	Đặt số dòng cho phần tiêu đề của lưới
10	Row DividerStyle	Kiểu đường kẻ ngang của lưới

Với các thuộc tính trên, chúng ta có thể thiết kế một lưới tuỳ ý. Tuy nhiên phần sau đây sẽ hướng dẫn một cách thiết đặt một lưới dữ liệu hay dùng nhất.

- Thiết kế lưới tuỳ ý:

[1]. Chọn lưới cần thiết kế, kích phải chọn edit.

[2]. Chọn lưới cần thiết kế, kích phải chọn:

+ Insert: thêm cột.

+ Append: Xoá cột.

+ Split: Chia lưới thành nhiều khung nhìn khác nhau.

+ Remove : Xoá bỏ các khung nhìn vừa thêm vào.

+ Retrieve Filed: thiết kế lưới tự dộng phù hợp với bảng dữ liệu đã liên kết tới lưới.

[3]. Chọn lưới cần thiết kế, kích phải, chọn Properties. Khi đó, có thể đặt các thuộc tính để thiết kế cho lưới. Lưu ý cách đặt Caption và độ rộng cho từng cột:

+ Chọn Columns: Khi đó, chọn một cột tuỳ ý trong danh sách các cột và đặt các thuộc tính cho cột vừa chọn bao gồm:

. Caption: tiêu đề cột.

. Data field: Tên trường mà cột đó sẽ liên kết tới.

. Default value: Giá trị mặc định cho cột, trong trường hợp chưa liên kết dữ liệu hoặc dữ liệu bị trống.

. NumberFormat: Định dạng cho dữ liệu kiểu số.

+ Chọn Layout: Khi đó, chọn từng cột trong danh sách các cột và đặt độ rộng cho các cột trong ô Width, đặt kiểu căn lề trong ô Alignment.

3. Cặp ADODC và Data Grid

Để có được hai điều khiển này, ta cần chọn: Project\ Components\ Microsoft ADO Data Control 6.0 và Microsoft Data Grid Control 6.0.

Để hiển thị được dữ liệu qua 2 điều khiển trên, ta làm theo các bước sau:

Bước 1: Vẽ hai điều khiển trên vào form.

Bước 2: Thiết đặt các thuộc tính cho điều khiển ADODC để kết nối tới cơ sở dữ liệu và bảng quan hệ cần thiết bằng cách:

- Chọn điều khiển ADODC.
- Đặt thuộc tính ConnectString bằng cách kích chuột vào thuộc tính này trên cửa sổ Properties để xây dựng chuỗi kết nối cho điều khiển.

Cách đặt chuỗi kết nối:

[1]. Chọn ConnectString trên cửa sổ Properties, chọn Build.

[2]. Chọn bộ máy liên kết cơ sở dữ liệu. Nếu dùng cơ sở dữ liệu Access ta chọn Microsoft Jet 4.0 OLE...

[3]. Chọn Next để chọn cơ sở dữ liệu sẽ kết nối tới bằng cách trong mục Select or Enter data base name.

[4]. Chọn Test Connected để kiểm tra sự kết nối có thành công không, chọn OK.

- Đặt thuộc tính RecordSource để chỉ ra bảng dữ liệu muốn thao tác. Khi đó cần:

[1]. Chọn Record Source trong cửa sổ Properties.

[2]. Trong Command type, chọn:

+ *adCmdUnknown* hoặc *adCmdText* nếu muốn liên kết tới một truy vấn. Khi đó cần gõ câu truy vấn SQL vào ô Command text.

+ *adCmdTable* nếu muốn liên kết tới một table. Khi đó cần chọn tên bảng trong ô Table or Stored Proceduce name, chọn OK.

Bước 3: Chọn điều khiển DataGrid.

Bước 4: Thiết đặt mối quan hệ giữa hai điều khiển:

- Đặt thuộc tính DataSource bằng cách chọn tên của điều khiển ADODC trên form.

Khi đó, công việc hiển thị dữ liệu qua ADODC và DataGrid đã hoàn tất.

Chú ý:

Khi đặt ConnectString trực tiếp vào điều khiển ADODC thì đường dẫn tới cơ sở dữ liệu trong chuỗi kết nối này được đặt cứng (cố định). Như vậy, khi thay đổi thư mục của cơ sở dữ liệu sẽ gây ra lỗi. Để khắc phục lỗi này, người ta thường không đặt hai thuộc tính ConnectString và RecordSource ngay lúc thiết kế mà đặt khi sự kiện Form Load xảy ra và sử dụng đường dẫn động App.Path. Biến App.Path sẽ cho ta đường dẫn động tới file thực thi của chương trình. Như vậy, chẳng hạn ta viết:

```
Private Sub Form_Load()
```

```
'Đặt ConnectString cho ADODC, thay đường dẫn thư mục bằng biến App.Path.
    Me.Adodc1.ConnectionString  =
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " &
    App.Path & "\db1.mdb;Persist Security Info=False"
'Đặt thuộc tính RecordSource cho ADODC
    Me.Adodc1.CommandType = adCmdText
    Me.Adodc1.RecordSource = "select * from table1"
    …
End Sub
```

Ngài ra, ta cần tham khảo các thuộc tính của hai điều khiển trên để có thể sử dụng khi cần. Các thuộc tính được liệt kê trong bảng sau:

[1]. Các thuộc tính của ADODC

Stt	Tên thuộc tính	Ý nghÜa
1.	Caption	Tiªu ®Ò sÏ hiÓn thÞ trªn ADODC, cã thÓ thay ®æi khi lËp tr×nh
2.	CommandType	KiÓu cña b¶ng d÷ liÖu sÏ liªn kÕt, bao gåm
3.	ConnectString	Chuçi kÕt nèi, cho phÐp kÕt nèi ADODC tíi mét c¬ së d÷ liÖu
4.	Record Source	B¶ng d÷ liÖu trong c¬ së d÷ liÖu mµ ADODC sÏ liªn kÕt tíi
	….	

[2]. Các thuộc tính của Data Grid

(chọn điều khiển DataGrid, kích phải, chọn Properties)

Stt	Tên thuộc tính	ý nghĩa
1.	Allow Add New	Giống như trong Data
2.	Allow Arrow	Giống như trong Data
3.	Allow Update	Giống như trong Data
4.	Column Headers	Giống như trong Data
5.	Data Source	Chọn tên ADODC trên form
6.	DefColumn Width	Giống như trong Data
7.	Headline	Giống như trong Data
8.	Row DividerStyle	Giống như trong Data

9.	RowHeight	Chiều cao của một dòng trên lưới
	…	

Như vậy, ta cũng có thể thiết kế một lưới DataGrid tuỳ ý tương tự như thiết kế DBGrid.

Vì tính ưu việt của cặp ADODC và DataGrid nên người ta thường sử dụng chúng trong lập trình cơ sở dữ liệu. Do vậy, các ví dụ từ đây về sau sẽ tập trung giới thiệu về cặp điều khiển này.

(Lưu ý: Khi sử dụng cặp Data – DBGrid rất dễ xảy ra hiện tượng không kết nối được tới cơ sở dữ liệu do các phiên bản của MS Access không tương thích. Khi đó cần Convert cơ sở dữ liệu sang phiên bản phù hợp. Có thể tránh điều này bằng cách dùng ADODC-DataGrid)

4. Cập nhật dữ liệu qua các điều khiển

a. Mẫu form cập nhật

Có thể có nhiều mẫu form cập nhật khác nhau nhưng nhìn chung, một form cập nhật luôn có các chức năng:

- *Xem:* cho phép người dùng xem dữ liệu. Thông thường dữ liệu được thể hiện trên lưới.

- *Bổ sung:* cho phép người dùng có thể thêm các bản ghi mới vào trong bảng dữ liệu. Có thể bổ sung bản ghi trực tiếp trên lưới. Tuy nhiên cách này không hiệu quả hoặc nếu làm cho có hiệu quả sẽ cần một kỹ thuật lập trình tương đối phức tạp. Vì vậy người ta thường dùng các điều khiển đơn giản như TextBox, ComboBox để người dùng nhập dữ liệu qua đó.

- *Xoá:* Cho phép xoá một bản ghi nào đó trong bảng dữ liệu.

-

-

cập nhậ

phụ, trợ

Ta

Hình 3: Mẫu form cập nhật

Sau đây, ta sẽ xem xét các kiểu form cập nhật khác nhau. Các form cập nhật được giới thiệu ngày càng hoàn thiện và đòi hỏi kỹ thuật cao hơn theo trình tự từ dễ tới khó.

b. Tạo form cập nhật đơn giản

Trong suốt phần lập trình cơ sở dữ liệu này, chúng ta luôn sử dụng một cơ sở dữ liệu duy nhất cho các ví dụ. Ta cần tạo cơ sở dữ liệu trong môi trường Access (hoặc Visual Basic tuỳ ý). Cơ sở dữ liệu có tên DB1. mdb và bao gồm một bảng duy nhất Table1. Bảng này có cấu trúc như sau:

Stt	Tên thuộc tính	Độ rộng	Mô tả
1	ID	Text (8)	Mã nhân viên (Khoá chính)
2	HoTen	Text (50)	Họ và tên
3	NgaySinh	DataTime	Ngày sinh
4	Luong	Long	Lương

Sẽ không khó để tạo một form cập nhật đơn giản. Ta làm theo các bước sau:

- Thiết kế form: bao gồm các điều khiển cần thiết.

- Đặt liên kết từ ADODC tới cơ sở dữ liệu: đặt ConnectString và RecordSource.

- Đặt liên kết DataGrid tới ADODC: đặt thuộc tính DataSourrce.

- Đặt thuộc tính DataSource của các điều khiển TextBox, ComboBox (nếu có) bằng cách chọn tên của ADODC trên form.

- Đặt thuộc tính FieldName của các điều khiển TextBox, ComboBox (nếu có) bằng cách chọn tên trường tương ứng với nó (khi đó sẽ xuất hiện).

Như vậy, một form cập nhật đã hoàn tất. Cần chú ý tới các thuộc tính Allow Update, Allow Delete, Allow Add New.. của điều khiển DataGrid phải được chọn.

Tuy nhiên form cập nhật này có những hạn chế rất lớn là mọi việc kiểm soát trong quá trình cập nhật đều do ADODC thực hiện. Như vậy, chưa triệt để trong khi xây dựng các ứng dụng (chẳng hạn các thông báo lỗi bằng tiếng anh khi người dùng nhập sai dữ liệu …)

Để có thể tạo ra các form linh hoạt hơn, hãy xem phần sau: lập trình cập nhật dùng ADODC.

5. Lập trình cập nhật dùng ADODC

Ta dùng các thuộc tính và phương thức của ADODC và DataGrid để lập trình tạo ra các form cập nhật rất linh hoạt. Danh sách các thuộc tính, phương thức thường dùng được cho trong bảng sau:

Me.ADODC1. Recordset. RecordCount	Cho biết tổng số bản ghi có trong bảng
Me.ADODC1. Recordset. AbsulatePosition	Cho biết số thứ tự của bản ghi hiện hành
Me.ADODC1. Recordset. EOF	Bằng True nếu truy cập tới bản ghi cuối cùng trong tệp và ngược lại.
Me.ADODC1. Recordset. BOF	Bằng True nếu truy cập tới bản ghi đầu tiên trong tệp và ngược lại.
Me. ADODC1. Recordset. Fields("<Tên trường>")	Truy cập tới 1 trường trong bảng
Me.ADODC1. Recordset. Add New	Thêm một bản ghi mới (bản ghi trống)
Me.ADODC1. Recordset. Move <n>	Chuyển đến bản ghi thứ n kể từ bản ghi hiện hành
Me.ADODC1. Recordset. MoveFirst	Chuyển đến bản ghi đầu tiên
Me.ADODC1. Recordset. MoveLast	Chuyển đến bản ghi cuối cùng
Me.ADODC1. Recordset. MoveNext	Chuyển tới bản ghi tiếp theo
Me.ADODC1. Recordset. MovePrevious	Chuyển tới bản ghi trước

Me.ADODC1. Recordset. Update	Cập nhật bản ghi mới vào bảng dữ liệu
Me.ADODC1. Recordset.Requery	Làm tươi bảng dữ liệu recordset
Me.ADODC1. Refresh	Làm tươi lại dữ liệu trên ADODC
Me.ADODC. Delete	Xoá bản ghi hiện hành trên ADODC

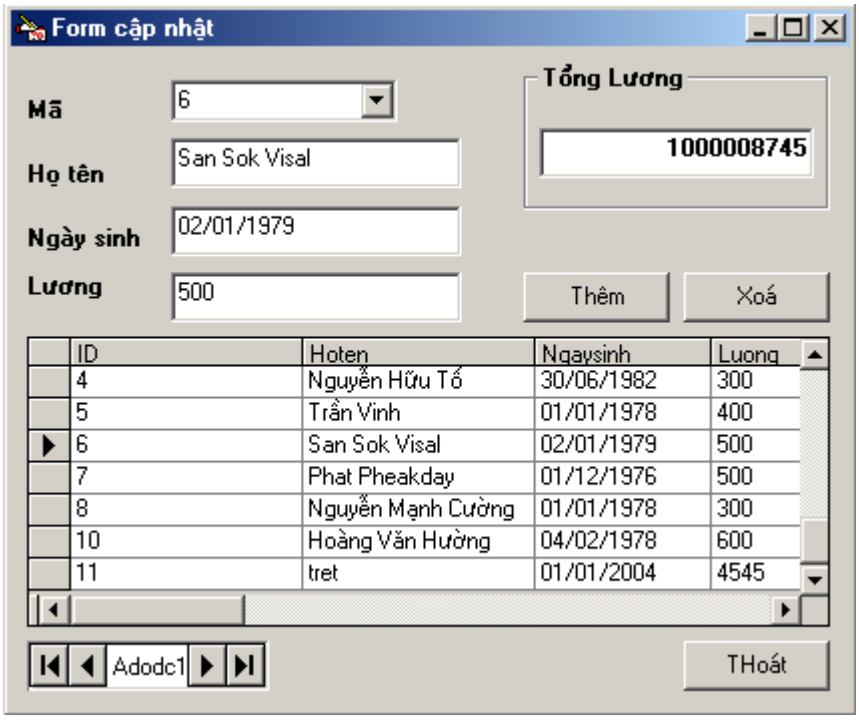
Các thao tác cơ bản thường dùng trong lập trình cập nhật được tóm tắt dưới đây:

[1] . *Thêm một bản ghi mới* (chưa lưu vào bảng): me.ADODC1. Recordset. AddNew

[2]. *Lưu dữ liệu vào bảng* (dữ liệu thêm mới hoặc dữ liệu vừa sửa) : me.ADODC1. Recordset. Update.

[3]. *Xoá bản ghi hiện hành trên lưới:* me.ADODC1. Recordset. Delete

VD 1: Lập trình để tạo form cập nhật vào bảng NhanVien của cơ sở dữ liệu NhanSu theo mẫu sau:



Hình 4: Ví dụ về lập trình cập nhật đơn giản

Các bước chính:

Bước 1: Đảm bảo đã tồn tại một cơ sở dữ liệu DB1.mdb trong đó có một bảng table1 như đã giới thiệu ở phần trên.

Bước 2: Thiết kế form, vẽ các điều khiển và đặt các ràng buộc cần thiết cho ADODC, DataGrid, các TextBox, các ComboBox...các điều khiển được đặt tên lần lượt là: txtID, txtHoten, txtNgaySinh, txtLuong, txtTongLuong…

Bước 3: Tạo một cơ sở dữ liệu DB1.mdb từ MS. Access với duy nhất một bảng Table1 gồm các trường: ID, Hoten, NgaySinh, Luong.

Bước 4: Lập trình, xem các đoạn mã chính dưới đây:

```
‘ Hàm CoutSalary() trả về tổng lương của tất cả các cán bộ
Public Function CoutSalary()
     Dim Total As Double
     With Me.Adodc1.Recordset
          .MoveFirst
          Do While .EOF = False
               Total = Total + .Fields("Luong")
               .MoveNext
          Loop
     End With
     CoutSalary = Total
End Function
‘Mã lệnh cho nút Thêm và Xoá
Private Sub cmdThem_Click()
     Me.Adodc1.Recordset.AddNew
End Sub

Private Sub cmdXoa_Click()
     Me.Adodc1.Recordset.Delete
     Me.Adodc1.Recordset.Update
     Me.txtTongluong.Text = CoutSalary
End Sub
‘Tính lại tổng lương khi thay đổi lương
Private Sub txtLuong_LostFocus()
     Me.txtTongluong.Text = CoutSalary
End Sub
```

Nhận xét:

- Mẫu form cập nhật trên rất đơn giản do sử dụng các phương thức có sẵn của ADODC.
- Rất dễ xảy ra lỗi khi cập nhật, các thông báo lỗi là của ADODC.
- Chưa đáp ứng được các thao tác cập nhật trong thực tế.

- Hàm CoutSalary() phải viết rất dài, ta có thể thực hiện công việc này chỉ bằng 1 câu SQL: Select Sum(Luong) from table1. Tuy nhiên, với phương pháp lập trình dùng ADODC thì việc cần phải duyệt từ đầu đến cuối bảng để tính tổng lương là cần thiết. Đây cũng chính là một hạn chế của ADODC.

Để khắc phục nhược điểm trên, chúng ta sẽ xem xét VD2 với phần lập trình nhiều hơn.

VD2: Lập trình form cập nhật giống như mẫu form cập nhật chuẩn ở phần trên, sử dụng kỹ thuật lập trình dùng ADODC.

Các chú ý:

[1]. Hàm SetStatus: đặt lại trạng thái cho Form như khi form mới hiển thị. Ta cần gọi hàm này khi Form Load hoặc khi Thêm mới, Sửa đã hoàn tất…

[2]. Hàm Test: Kiểm tra xem dữ liệu nhập vào đã phù hợp chưa. Test = True nếu dữ liệu nhập vào đã phù hợp và ngược lại.

[3]. Cơ chế sinh giá trị của trường khoá (trường ID) một cách tự động và các chú ý khi xoá bản ghi.

[4]. Biến Status : để phân biệt chức năng Lưu là lưu của chức năng Thêm mới hay lưu của chức năng Sửa (1 nút lệnh 2 chức năng). Khi người dùng chọn Thêm, Status sẽ được đặt là True và khi người dùng chọn Sửa thì Status được đặt là False.

Bước 1: Đảm bảo đã tồn tại một cơ sở dữ liệu DB1.mdb trong đó có một bảng table1 như đã giới thiệu ở phần trên.

Bước 2: Vào môi trường Visual Basic thiết kế form cập nhật như hình 3. Đặt tên cho các Textbox và nút lệnh lần lượt là txtID, txtTen, txtNgaySinh, txtLuong, cmdThem, cmdSua, cmdLuu, cmdHuy, cmdXoa.

Chú ý: Đặt thuộc tính ConnectString và DataSource cho ADODC. Đặt thuộc tính DataSource cho DataGrid. Không ràng buộc các textbox vào ADODC.

Bước 3: Lập trình. Xem đoạn mã sau:

```
Public Status As Boolean

Public Sub SetStatus()
  'Dat noi dung ban ghi hien hanh len text box
  If Me.Adodc1.Recordset.RecordCount > 0 Then
    Me.txtID.Text = Me.Adodc1.Recordset.Fields("ID")
```

```
    Me.txtTen.Text = Me.Adodc1.Recordset.Fields("HoTen")
    Me.txtNgaySinh.Text = Me.Adodc1.Recordset.Fields("NgaySinh")
    Me.txtLuong.Text = Me.Adodc1.Recordset.Fields("Luong")
  End If
    'Dat trang thai cho cac nut
    Me.cmdThem.Enabled = True
    Me.cmdSua.Enabled = True
    Me.cmdLuu.Enabled = False
    Me.cmdHuy.Enabled = False
    Me.cmdXoa.Enabled = True
End Sub

Public Function Test() As Boolean
Test = True
If Trim(Me.txtTen.Text) = "" Then
    Test = False
       MsgBox "Tên không được trống ! hãy xem lai ", vbOKOnly +
            vbExclamation, "Thông báo"
    Me.txtTen.SetFocus
Else
    If IsDate(Me.txtNgaySinh.Text) = False Then
        Test = False
          MsgBox "Kiểu ngày sinh không đúng. Hãy xem lại !", vbOKOnly +
                vbExclamation, "THông báo"
        Me.txtNgaySinh.SetFocus
        Else
        If IsNumeric(Me.txtLuong) = False Then
            Test = False
             MsgBox "Kiểu Lương không đúng. Hãy xem lại !", vbOKOnly +
                   vbExclamation, "THông báo"
            Me.txtLuong.SetFocus
        End If
    End If
End If
End Function


Private Sub Form_Load()
     'Dat ConnectString cho ADODC – Duong dan dong
     Me.Adodc1.ConnectionString = "Provider= Microsoft. Jet.
     OLEDB. 4.0; Data Source=" & App.Path & "\db1.mdb;Persist
     Security Info=False"
     Me.Adodc1.RecordSource = "select * from table1"
     Me.Adodc1.Refresh
     SetStatus
End Sub

Private Sub DataGrid1_RowColChange(LastRow As Variant, ByVal
                                   LastCol As Integer)
     On Error GoTo Er
```

```
     SetStatus
     Exit Sub
     Er:
End Sub

Private Sub cmdThem_Click()
     'dat trang thai cac nut
     Me.cmdThem.Enabled = False
     Me.cmdSua.Enabled = False
     Me.cmdLuu.Enabled = True
     Me.cmdHuy.Enabled = True
     Me.cmdXoa.Enabled = False
     Status = True
     'Ma cua ban ghi moi se duoc tinh tu dong
     Me.txtID.Text = Me.Adodc1.Recordset.RecordCount + 1
     'Xoa noi dung cac text box
     Me.txtTen.Text = ""
     Me.txtNgaySinh.Text = ""
     Me.txtLuong.Text = ""
     Me.txtTen.SetFocus
End Sub

Private Sub cmdSua_Click()
     'dat trang thai cac nut
     Me.cmdThem.Enabled = False
     Me.cmdSua.Enabled = False
     Me.cmdLuu.Enabled = True
     Me.cmdHuy.Enabled = True
     Me.cmdXoa.Enabled = False
     Status = False
     Me.txtTen.SetFocus
End Sub

Private Sub cmdHuy_Click()
     SetStatus
End Sub

Private Sub cmdLuu_Click()
If Test Then
With Me.Adodc1.Recordset
If Status Then
      'neu luu cua them moi
      .AddNew
      .Fields("ID") = Me.txtID.Text
      .Fields("Hoten") = Me.txtTen.Text
      .Fields("Ngaysinh") = CDate(Trim(Me.txtNgaySinh.Text))
      .Fields("Luong") = Val(Trim(Me.txtLuong.Text))
      .Update
Else
      ' neu luu cua sua
      .Fields("Hoten") = Me.txtTen.Text
```

```
      .Fields("Ngaysinh") = CDate(Trim(Me.txtNgaySinh.Text))
      .Fields("Luong") = Val(Trim(Me.txtLuong.Text))
      .Update
End If
End With
SetStatus
End If
End Sub

Private Sub cmdXoa_Click()
Dim Result As Byte
Dim Position As Long
Result   =   MsgBox("Có thực sự muốn xoá bản ghi " &
             Me.Adodc1.Recordset.Fields("Hoten") & " không ? ",
             vbYesNo, "Thông báo")
If Result = vbYes Then
      With Me.Adodc1.Recordset
      If .EOF And .BOF Then
      Else
            'Luu lai vi tri cua ban ghi se xoa
            Position = Me.Adodc1.Recordset.AbsolutePosition
            ' Xoa ban ghi
            .Delete
            .Update
            .Requery
            'Đẩy ID của các bản ghi sau dồn lên
            .MoveFirst
             Do While .EOF = False
                           If .Fields("ID") > Position Then
                           .Fields("ID") = .Fields("ID") - 1
                           .Update
                           End If
                           .MoveNext
             Loop
        End If
        End With
End If
End Sub

Private Sub cmdThoat_Click()
Unload Me
End Sub
```

Nhận xét:

- Phương pháp lập trình dùng ADODC tương đối đơn giản và đáp ứng được các đòi hỏi của lập trình cập nhật.
- Một số hạn chế vẫn chưa khắc phục được, đó là: mọi thao tác trên bảng quan hệ luôn phụ thuộc vào điều khiển ADODC. Trong nhiều trường hợp, các thao tác trên quan hệ là ẩn (theo nghĩa là không cho người sử dụng

biết) khi đó, việc luôn phải vẽ ADODC trên form là bất tiện. (VD như form đăng nhập).

- Nếu muốn tính tổng lương thì sao? việc duyệt từ đầu tới cuối bảng quan hệ để tính tổng lương vẫn rất cần thiết.
- Để khắc phục trệt để những nhược điểm trên, ta hãy xem xét kỹ thuật lập trình dùng ADODB ở phần sau.

III. LẬP TRÌNH CƠ SỞ DỮ LIỆU DÙNG ADODB

1. Các biến cần thiết

Để lập trình được linh hoạt hơn, người ta thường sử dụng kỹ thuật lập trình dùng ADODB. ADODB là một đối tượng ActiveX được cung cấp sẵn trong môi trường lập trình Visual Basic. Muốn sử dụng nó, ta chọn: Project\ Refrences\ Microsoft ADO library…

Trên giao diện, ta vẫn có thể dùng ADODC và DataGrid để liên kết và hiển thị dữ liệu (nếu cần). Tuy nhiên, các thao tác cập nhật nói riêng và các thao tác với cơ sở dữ liệu ta sẽ sử dụng các biến của ADODB. Các biến này sẽ làm việc trực tiếp trong cơ sở dữ liệu mà kết quả của nó chỉ được hiển thị trên giao diện khi đã hoàn tất một quá trình nào đó.

Để thực hiện được như vậy, ta hay dùng 2 biến toàn cục sau:

- ***Biến để kết nối tới CSDL:*** Thường được đặt tên là DB và có kiểu ADODB. Connection.

- ***Biến để mở bảng dữ liệu:*** Thường được đặt tên là Rs và có kiểu là ADODB. Recordset.

Vì là biến toàn cục nên chúng được khai báo trong module. Khi đó, trong một module nào đó của chương trình, ta khai báo:

Public DB as New ADODB.Connection

Public Rs as New ADODB.Recordset

Chú ý: nếu trong Project chưa có module, hãy chọn Project\ Add Module

a. *Dùng DB để mở cơ sở dữ liệu*

Ta dùng hai biến này để mở cơ sở dữ liệu và bảng dữ liệu cần thao tác. Thông thường ta mở cơ sở dữ liệu ngay trong sự kiện Form_Load của form

đầu tiên bằng cách sử dụng biến DB. Biến này sẽ kết nối tới cơ sở dữ liệu cần làm việc thông qua chuỗi kết nối ConnectString:

DB. Open <Connect String>

Chuỗi kết nối Connect String được lấy từ thuộc tính ConnectString của đối tượng ADODC (đặt ConnectString cho ADODC và sao chép chuỗi đó vào đây). cần chú ý tới khả năng đặt đường dẫn động trong chuỗi kết nối bằng cách sử dụng biến App.Path.

Thông thường, một chương trình ứng dụng cơ sở dữ liệu chỉ làm việc trên một cơ sở dữ liệu duy nhất. Khi đó, ta có thể đặt dòng lệnh mở cơ sở dữ liệu này tại sự kiện Form_Load của form đầu tiên sẽ xuất hiện khi chương trình thực thi. Biến DB sẽ cung cấp một kết nối tới cơ sở dữ liệu trong suốt quá trình chương trình thực thi mà không cần mở lại cơ sở dữ liệu một lần nữa.

b. Dùng Rs để làm việc với một bảng hay query bất kỳ.

Sau khi đã liên kết được tới cơ sở dữ liệu qua biến DB, ta sử dụng biến Rs để mở bảng quan hệ cần thao tác. Cú pháp:

Rs. Open < "Tên bảng/ Câu SQL" > , DB, <Tham số 1>, <Tham số 2>

Trong đó:

- *<"Tên bảng/ Câu SQL">* có thể là một tên bảng quan hệ mà ta muốn thao tác hoặc một query tương ứng với một câu hỏi SQL.

- *DB* là tên biến kết nối tới CSDL mà ta đã khai báo ở trên.

- Các *<Tham số 1>, <Tham số 2>* là các tham số tuỳ chọn. Thông thường ta chọn *adOpenKeyset* và *adLockPessimistic*

VD: ta có thể mở bảng Table1 trong cơ sở dữ liệu đang được kết nối bởi biến DB bằng cách viết

Rs.Open "Select * from Table1", DB, adOpenKeyset, adLockPessimistic

Khi đó, biến Rs chứa toàn bộ dữ liệu của bảng Table1 và ta có thể dùng biến này để cập nhật hoặc thực hiện các thao tác trên bảng Table1.

2. **Các thuộc tính, phương thức của biến Rs**

Để sử dụng biến Rs trong việc lập trình các thao tác trên cơ sở dữ liệu, ta cần biết các thuộc tính và phương thức của đối tượng Rs. Các thành phần này được liệt kê trong bảng dưới đây:

Stt	Thuộc tính, phương thức	Ý nghÜa
1.1	*RS. RecordCount*	Tæng sè b¶n ghi cã trong b¶ng mµ Rs ®ang më
2.2	*RS. AbsulatePosition*	VÞ trÝ cña con trá trong b¶ng mµ Rs ®ang më
3.3	*RS.EOF*	B»ng True nÕu Rs ®ang truy cËp tíi b¶n ghi n»m ë cuèi b¶ng vµ b»ng False nÕu ngîc l¹i
4.4	*RS. BOF*	B»ng True nÕu Rs ®ang truy cËp b¶n ghi ®Çu tiªn cña b¶ng. B»ng False nÕu ngîc l¹i
5.	*RS. Fields("<Tªn trêng")*	Truy cËp tíi d÷ liÖu cña mét trêng trong b¶ng quan hÖ
6.5	*RS.MoveFirst*	§a con trá vÒ b¶n ghi ®Çu tiªn trong b¶ng mµ Rs trá tíi
7.	*RS.MoveLast*	§a con trá vÒ b¶n ghi cuèi cïng trong b¶ng mµ Rs trá tíi.
8.	*RS.MoveNext*	§a con trá vÒ b¶n ghi tiÕp theo trong b¶ng mµ Rs trá tíi.
9.	*RS.MovePrevious*	§a con trá vÒ b¶n ghi tríc trong b¶ng mµ Rs trá tíi
10.	*RS. Move <n>*	§a con trá vÒ b¶n ghi thø n kÓ tõ vÞ trÝ hiÖn hµnh
11.	*RS. Add New*	Thªm mét b¶n ghi míi vµo cuèi b¶ng mµ Rs ®ang më.
12.	*RS. Update*	CËp nhËt l¹i b¶ng mµ Rs ®ang më
13.	*RS. Delete*	Xo¸ b¶n ghi hiÖn hµnh trong b¶ng mµ Rs ®ang më
14.	*RS.Close*	§ãng b¶ng quan hÖ mµ Rs ®ang më

3. Các thao tác trên bảng quan hệ

[1]. Duyệt từ đầu tới cuối bảng:

Ta sử dụng đoạn trình mẫu sau:

```
RS. Open …
If RS.EOF And RS.BOF then 'Neu bang du lieu rong
Else
     RS. MoveFirst
     Do While RS.EOF = False
          ' Truy cập tới bản ghi…
          RS. MoveNext
     Loop
End if
RS.Close
```

[2]. Thêm một bản ghi mới

Ta sử dụng đoạn trình sau:

```
RS. Open …
RS. Add New
RS. Fields(“Tên trường 1”) = <Giá trị tương ứng>
RS. Fields(“Tên trường 2”) = <Giá trị tương ứng>
…
RS. Update
RS. Close
‘ Nếu trên form có sử dụng ADODC và DataGrid thì cần làm tươI lại:
    RS. Recordset. Requery
    RS. Refresh
```

[3]. Sửa bản ghi hiện hành trên Data Grid

```
RS. Open “Select * from <Tên bảng> where <ĐK sửa>”, db, 3, 3
RS. Fields(“Tên trường 1”) = <Giá trị mới tương ứng>
RS. Fields(“Tên trường 2”) = <Giá trị mới tương ứng>
…
RS. Update
RS. Close
‘ Nếu trên form có sử dụng ADODC và DataGrid thì cần làm tươI lại:
RS. Recordset. Requery
RS. Refresh
```

<ĐK sửa> ở đây là: Mã của bản ghi cần sửa trùng với mã của bản ghi hiện hành trên ADODC. VD với bảng dữ liệu trên, ta viết:

“Select * from Table1 where ID = ‘ “ & me.ADODC.Recordset. Fields(“ID”) & “ ‘ “

Khi đó, ta sẽ select được duy nhất 1 bản ghi có mã như vậy, chính là bản ghi cần sửa.

[4]. Xoá bản ghi hiện hành trên Data Grid

Ta dùng đoạn trình sau:

```
RS. Open  Select * from <Tên bảng> where <ĐK xoá>”, db, 3, 3
RS. Delete
RS.Update
RS.Close
‘ Làm tươi trên form
RS. Recordset. Requery
RS. Refresh
```

Để xoá bản ghi hiện hành trên ADODC thì <ĐK xoá > sẽ là: Mã của bản ghi cần xoá trùng với mã của bản ghi hiện hành trên ADODC. VD với bảng dữ liệu trên, ta viết:

“Select * from Table1 where ID = ‘ “ & me.ADODC.Recordset. Fields(“ID”) & “ ‘ “

Chú ý: Cần kiểm soát quá trình xoá: không cho xoá nếu như tệp không còn bản ghi nào; luôn hỏi người dùng trước khi xoá.

VD: Hãy lập trình tạo form cập nhật chuẩn theo mẫu ở phần trên, sử dụng kỹ thuật ADODB (thao tác trên bảng Table1 của cơ sở dữ liệu DB1.mdb ở trên)

Các chú ý:

[1]. Hàm SetStatus: đặt lại trạng thái cho Form như khi mới Load lên. Ta cần gọi hàm này khi Form Load, Thêm mới, hoặc sửa…

[2]. Hàm Test: Kiểm tra xem dữ liệu nhập vào đã đúng kiểu chưa. Test = true nếu dữ liệu nhập vào đã đúng kiểu và ngược lại.

[3]. Cơ chế sinh giá trị của trường khoá (trường ID) một cách tự động và các chú ý khi xoá bản ghi

Trước khi xem đoạn mã dưới đây, cần đảm bảo các biến toàn cục DB, Rs đã được khai báo trong một module nào đó của chương trình. Trong ví dụ này, ta sử dụng kỹ thuật lập trình dùng ADODB. Tuy nhiên, một số thao tác có kết hợp cả ADODC để rút ngắn việc viết mã chương trình.

Các điều khiển trên form được thiết kế và đặt tên lần lượt là: txtID, txtHoten, txtNgaySinh, txtLuong, cmdThem, cmdSua, cmdXoa, cmdLuu, cmdHuy, DataGrid1, ADODC1…

Sau đây là một số đoạn mã chính; (Xem trong bản Demo nếu cần.)

```
Public Sub Setstatus()
     Me.cmdThem.Enabled = True
     Me.cmdSua.Enabled = True
     Me.cmdXoa.Enabled = True
     Me.cmdLuu.Enabled = False
     Me.cmdHuy.Enabled = False
  If Me.ADODC.Recordset. RecordCount > 0 Then
     Me.txtHT.Text = Me.Adodc1.Recordset.Fields("Hoten")
     Me.txtNS.Text = Me.Adodc1.Recordset.Fields("Ngaysinh")
     Me.txtLuong.Text = Me.Adodc1.Recordset.Fields("Luong")
     Me.Adodc1.Caption  =  Me.Adodc1.Recordset.AbsolutePosition
     & "/" & Me.Adodc1.Recordset.RecordCount
     Me.txtID.Text = Me.Adodc1.Recordset.Fields("ID")
  End If
End Sub
Public Function Test() As Boolean
     Test = True
  If Trim(frmUpdate.txtHT.Text) = "" Then
    Test = False
    MsgBox "Tên không được trống! hãy xem lai", vbOKOnly + vbExclamation, "Thông báo"
    frmUpdate.txtHT.SetFocus
  Else
    If IsDate(frmUpdate.txtNS.Text) = False Then
        Test = False
```

```
            MsgBox "Kiểu ngày sinh không đúng. Hãy xem lại !", vbOKOnly +
                  vbExclamation, "Thông báo"
            frmUpdate.txtNS.SetFocus
        Else
            If IsNumeric(frmUpdate.txtLuong) = False Then
                Test = False
                 MsgBox "Kiểu Lương không đúng. Hãy xem lại !", vbOKOnly +
                        vbExclamation, "Thông báo"
                frmUpdate.txtNS.SetFocus
            End If
        End If
    End If
End Function
Private Sub cmdThem_Click()
     Me.txtHT.Text = ""
     Me.txtNS.Text = ""
     Me.txtLuong.Text = ""
     Status = True
     Me.cmdThem.Enabled = False
     Me.cmdSua.Enabled = False
     Me.cmdXoa.Enabled = False
     Me.cmdLuu.Enabled = True
     Me.cmdHuy.Enabled = True
     Me.txtID.Text = Me.Adodc1.Recordset.RecordCount + 1
     Me.txtHT.SetFocus
End Sub
Private Sub cmdSua_Click()
     Status = False
     Me.cmdThem.Enabled = False
     Me.cmdSua.Enabled = False
     Me.cmdXoa.Enabled = False
     Me.cmdLuu.Enabled = True
     Me.cmdHuy.Enabled = True
     Me.txtHT.SetFocus
End Sub
Private Sub cmdXoa_Click()
     Dim Result As Byte
     Dim Position As Integer
     Result = MsgBox("Có thực sự muốn xoá bản ghi " &
     Me.Adodc1.Recordset.Fields("Hoten") & " không?", vbYesNo, "Thông báo")
     If Result = vbYes Then 'Neu dong y xoa thi:
         'Lay vi tri cua ban ghi can xoa
          Position = Me.Adodc1.Recordset.AbsolutePosition
          'Mo Bang quan he nhung chi select ban ghi can xoa
```

```
            RS.Open "Select * from table1 where ID = '" &
          me.ADODC1. Recordset. Fields("ID") & "'", DB,
          adOpenKeyset, adLockPessimistic
          If RS.EOF And RS.BOF Then
          Else ' Neu co ban ghi can xoa thi xoa
             RS.Delete
             RS.Update
          End if
          RS.Close
          'Xoa xong roi thi sua lai ID cho cac ban ghi co ID >
     ID cuar ban ghi vua xoa
          RS.Open      "table1",      DB,      adOpenKeyset,
     adLockPessimistic
             Do While Not RS.EOF
                  If RS.AbsolutePosition >= Position Then
                     RS.Fields("ID") = RS.Fields("ID") - 1
                     RS.Update
                  End If
                  RS.MoveNext
             Loop
          End If
          RS.Close
          RS.Open "table1", DB, adOpenKeyset, adLockPessimistic
          Me.Adodc1.Recordset.Requery
          Me.Adodc1.Refresh
          RS.Close
     End If
End Sub
Private Sub cmdLuu_Click()
If Test Then
If Status Then
    'neu ghi cua them moi
    RS.Open "table1", DB, adOpenKeyset, adLockPessimistic
    RS.AddNew
    RS.Fields("ID") = RS.RecordCount + 1
    RS.Fields("Hoten") = Trim(Me.txtHT.Text)
    RS.Fields("Ngaysinh") = CDate(Trim(Me.txtNS.Text))
    RS.Fields("Luong") = Val(Trim(Me.txtLuong.Text))
    RS.Update
    RS.Close
     'Lam tuoi lai tren form
    RS.Open "table1", DB, adOpenKeyset, adLockPessimistic
    Me.Adodc1.Recordset.Requery
    Me.Adodc1.Refresh
    RS.Close
```

```
Else
    'neu ghi cua sua, mo quan he- chi chon ban ghi can sua
    RS.Open "Select * from table1 where ID  = '" & Me.ADODC1.
Recordset.    Fields("ID")    &    "'",    DB,    adOpenKeyset,
adLockPessimistic
    RS.Fields("Hoten") = Trim(Me.txtHT.Text)
    RS.Fields("Ngaysinh") = CDate(Trim(Me.txtNS.Text))
    RS.Fields("Luong") = Val(Trim(Me.txtLuong.Text))
    RS.Update
    RS.Close
    ' Lam tuoi lai tren form
    RS.Open "table1", DB, adOpenKeyset, adLockPessimistic
    Me.Adodc1.Recordset.Requery
    Me.Adodc1.Refresh
    RS.Close
End If
    Setstatus
End If
End Sub
Private Sub cmdHuy_Click()
    Setstatus
End Sub
Private Sub DataGrid1_RowColChange(LastRow As Variant, ByVal
LastCol As Integer)
    Setstatus
End Sub
```

Chú ý: *Kỹ thuật lập trình dùng ADODB có những ưu điểm hơn so với kỹ thuật dùng ADODC. Tuy nhiên, trong một số trường hợp nhất định, ta vẫn sử dụng kỹ thuật ADODC kết hợp với kỹ thuật ADODB để rút ngắn các đoạn mã chương trình.*

Hai kỹ thuật này là tương đương nhau về mặt sử dụng. Điểm khác nhau là ở chỗ :

- Nếu dùng ADODB thì ta phải dùng biến DB để kết nối tới cơ sở dữ liệu, biến Rs để kết nối tới bảng dữ liệu làm việc. Kết thúc quá trình làm việc cần phải đóng các bảng dữ liệu đang làm việc. Còn nếu dùng ADODC thì chỉ cần kết nối tới cơ sở dữ liệu thông qua ADODC.

- Nếu dùng ADODC thì kết quả của việc thay đổi dữ liệu sẽ hiển thị lập tức trên form, ngược lại, dùng ADODB sẽ can thiệp trực tiếp vào cơ sở dữ liệu nhưng kết quả không hiển thị ngay lập tức trên form (tức trên ADODC và DataGrid) mà ta cần phải làm tươi lại dữ liệu.

Ta so sánh các thuộc tính và phương thức của chúng trong bảng sau

Me.ADODC1. Recordset. RecordCount	Rs. RecordCount
Me.ADODC1. Recordset. AbsulatePosition	Rs. AbsulatePosition
Me.ADODC1. Recordset. EOF	Rs. EOF
Me.ADODC1. Recordset. BOF	Rs. BOF
Me.ADODC1. Recordset. Add New	Rs. Add New
Me.ADODC1. Recordset. Move <n>	Rs. Move <n>
Me.ADODC1. Recordset. MoveFirst	Rs. MoveFirst
Me.ADODC1. Recordset. MoveLast	Rs. MoveLast
Me.ADODC1. Recordset. MoveNext	Rs. MoveNext
Me.ADODC1. Recordset. MovePrevious	Rs. MovePrevious
Me.ADODC1. Recordset. Update	Rs. Update

IV. LẬP TRÌNH TÌM KIẾM

1. Bài toán tìm kiếm

Bài toán tìm kiếm được hình dung như sau:

Hình 6. Bài toán tìm kiếm

Khi giải quyết bài toán tìm kiếm, chúng ta có một tập các bảng dữ liệu đầu vào và một tập các tiêu chí tìm kiếm hay điều kiện tìm kiếm. Chỉ các bản ghi thoả mãn tập các điều kiện tìm kiếm này mới trở thành kết quả tìm kiếm.

Một form tìm kiếm thường có các điều khiển sau:

- *Các điều khiển để người dùng nhập tiêu chí tìm kiếm vào bao gồm:*

  + ComboBox: để lựa chọn giá trị cần tìm kiếm.

  + TextBox: Để nhập giá trị cần tìm kiếm.

  + Option: Để chọn một trong các tiêu chí tìm kiếm.

  + CheckBox: để chọn nhiều tiêu chí tìm kiếm.

  - Các điều khiển trình bày kết quả tìm kiếm.

+ Lưới Data Grid và ADODC hoặc dùng một ListView để trình bày kết quả tìm kiếm.

Ta tạm phân loại bài toán tìm kiếm như sau:

- Tìm kiếm đơn tiêu chí: sử dụng các điều khiển để người dùng nhập một giá trị tìm kiếm vào. Chỉ cho phép tìm kiếm theo một tiêu chí định trước, ví dụ như tìm kiếm theo họ tên, hoặc theo ngày sinh, hoặc theo lương…

- Tìm kiếm đa tiêu chí: Sử dụng nhiều điều khiển để người dùng nhập nhiều tiêu chí tìm kiếm trong một lần tìm kiếm. Dữ liệu tìm được phải thoả mãn đồng thời nhiều tiêu chí, các tiêu chí có thể kết hợp với nhau theo toán tử AND hoặc OR.

- Tìm kiếm động - đơn tiêu chí: Sử dụng các Option để người dùng có thể lựa chọn một trong số rất nhiều tiêu chí có thể. Tìm kiếm theo tiêu chí nào là do người dùng lựa chọn.

- Tìm kiếm động - đa tiêu chí : Sử dụng các CheckBox để người dùng chọn và nhập nhiều tiêu chí tìm kiếm một lúc. Lựa chọn tìm kiếm theo tiêu chí nào là tuỳ ở người dùng. Các tiêu chí được lựa chọn sẽ kết hợp với nhau theo toán tử AND hoặc OR.

2. Phương pháp chung để lập trình trình kiếm

Có nhiều phương pháp tìm kiếm nhưng nói chung, ta hay sử dụng phương pháp sau:

B1: Tạo một câu SQL sao cho kết quả của câu SQL là kết quả mong đợi (cần tìm kiếm). Câu SQL được tạo ra theo những tiêu chí mà người dùng đã nhập vào.

B2: Gán câu SQL đó cho thuộc tính Me.ADODC1. RecordSource và làm tươi lại để kết quả hiện lên trên form.

Như vậy, vấn đề còn lại là làm sao để xây dựng được câu hỏi SQl theo những tiêu chí mà người dùng đã nhập vào.

Thông thường với mỗi form tìm kiếm ta hay tạo một hàm Search() để làm nhiệm vụ tìm kiếm. Hàm này có hai phần (tương ứng với B1 và B2) là:

Phần 1: Phát sinh một câu SQL theo các tiêu chí người dùng nhập vào.

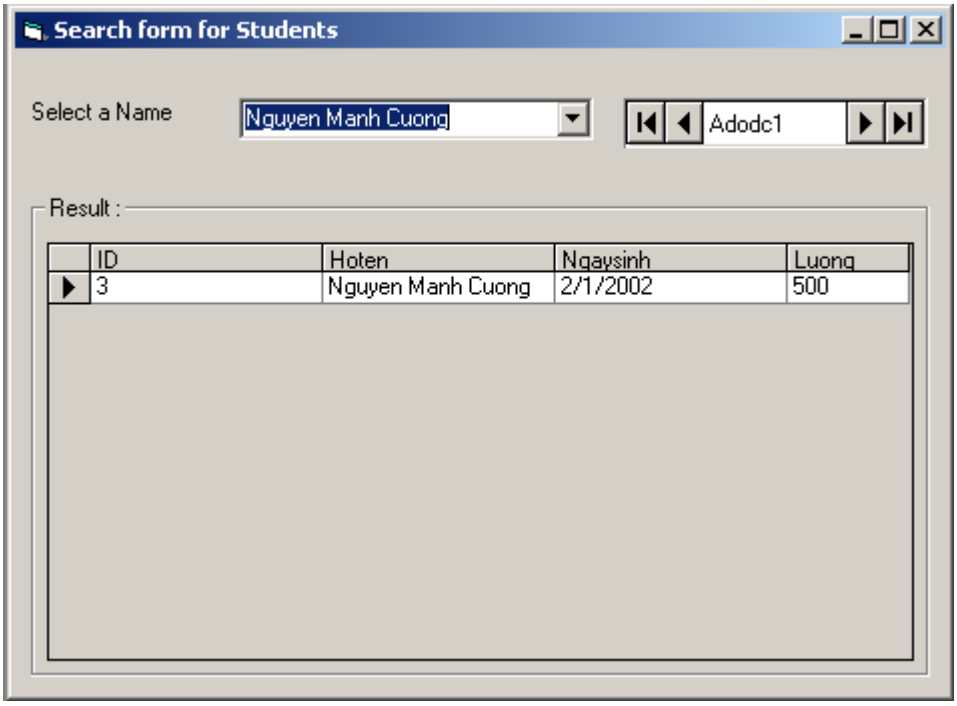
Phần 2: Gán câu SQL vừa phát sinh cho Me.ADODC1. RecordSource và làm tươi lại.

Hàm này được gọi tới khi form Load và mỗi khi người dùng nhập các giá trị tìm kiếm khác nhau.

3. Các ví dụ tìm kiếm

Trong các ví dụ dưới đây, ta áp dụng tìm kiếm trên cơ sở dữ liệu DB1.mdb, trong đó có một bảng Table1 gồm các trường ID, Hoten, NgaySinh, Luong như đã giới thiệu ở phần trên.

[1]. Tìm kiếm đơn tiêu chí dùng ComboBox



Hình 2: Mẫu form tìm kiếm đơn tiêu ch

Hình 6: Mẫu form tìm kiếm đơn tiêu chí

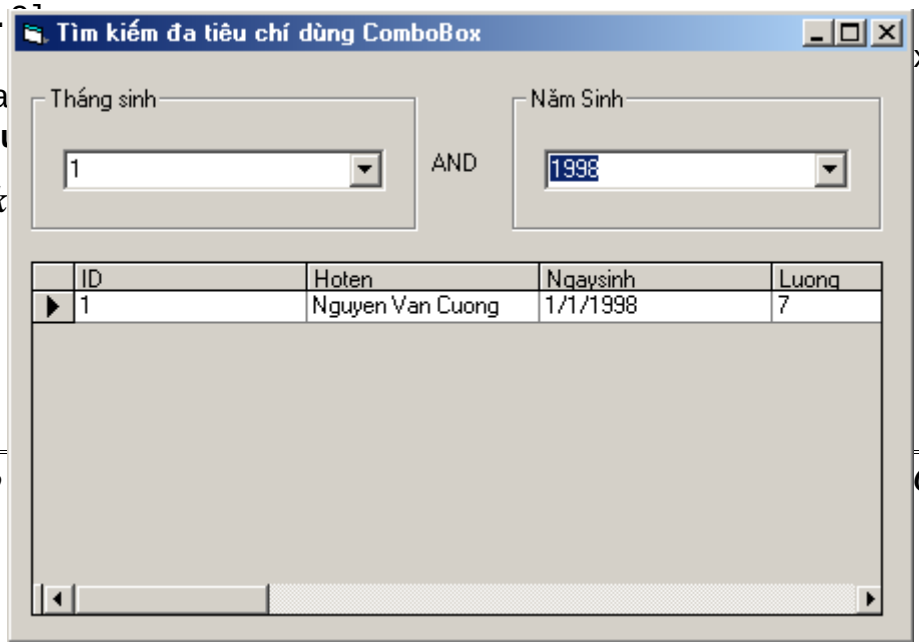
B1: Thiết kế form:

- Vẽ các đối tượng lên form (chú ý là ở đay ta dùng ADODC và Data Grid. Do có ít đối tượng trên form nên ta sử dụng các tên mà Visual Basic tự đặt cho chúng (Combo1, DataGrid1, ADODC1…)

- Đặt Connect String và Record Source cho ADODC1. Chú ý khi đặt Record Source: Đặt commandType là adCmdUnknow hoặc adCmdText và nhập câu hỏi SQL bất kỳ VD: Select * from table1.
- Ràng buộc DataGrid với ADODC1 bằng cách đặt thuộc tính Data Source của DataGrid bằng ADODC1.

B2: Lập trình:

```
Public Sub Search()
   Dim SQL As String
   SQL = "Select * from table1 where Hoten Like '%" &
   Trim(Me.Combo1.Text) & "%'"
   Me.Adodc1.RecordSource = SQL
   Me.Adodc1.Recordset.Requery
   Me.Adodc1.Refresh
End Sub

Private Sub Combo1_Click()
Search
End Sub

Private Sub Form_Load()
   DB.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
   App.Path & "\db1.mdb;Persist Security Info=False"
   'Mo bang dữ liệu va load Ho ten len Combo
   Rs.Open "table1", DB, adOpenKeyset, adLockPessimistic
   If Rs.EOF And Rs.BOF Then
   Else
       Rs.MoveFirst
       Do While Not Rs.EOF
            Me.Combo1.AddItem Rs.Fields("Hoten")
            Rs.MoveNext
       Loop
   End If
   Rs.Close
   DB.
   If                                                    x = 0
   Sea
End Su
```



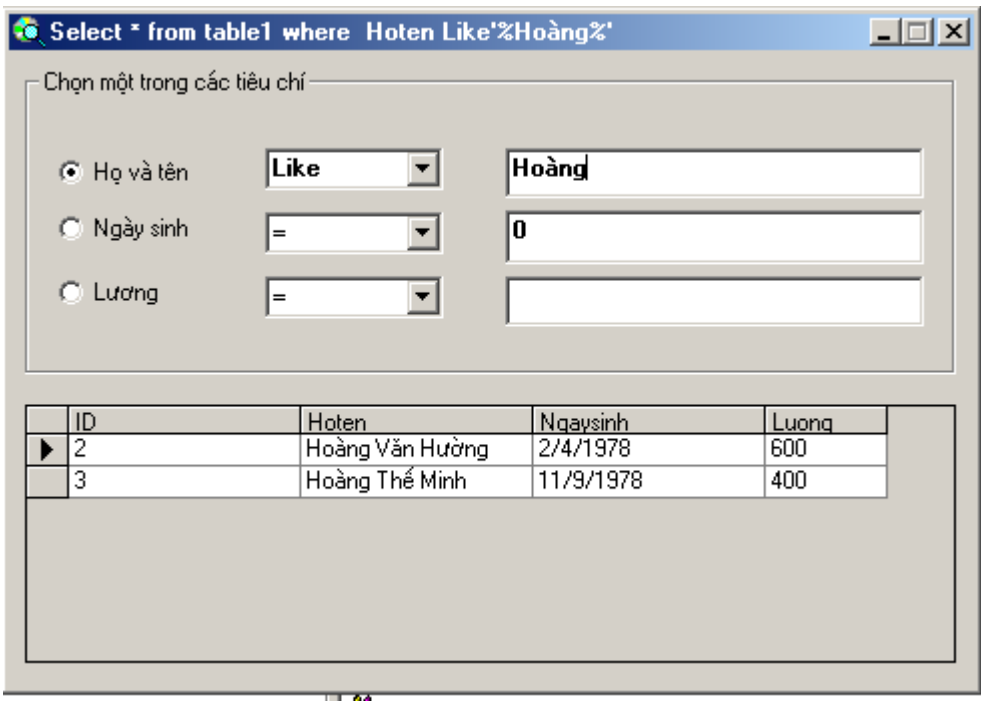
[2]. Tìm k

Hình 7: Tìm kiếm đa tiêu chí

```
Public Sub Search()
     Dim Sql As String  Sql = "Select * from table1 where
     Month(Ngaysinh) = " & Val(Me.Combo1.Text) & " AND
     Year(Ngaysinh) = " & Val(Me.Combo2.Text)
     Me.Adodc1.RecordSource = Sql
     Me.Adodc1.Recordset.Requery
     Me.Adodc1.Refresh
End Sub
'========================
Private Sub Combo1_Click()
     Search
End Sub
'========================
Private Sub Combo2_Click()
     Search
End Sub
'========================
Private Sub Form_Load()
     Dim i As Integer
     For i = 1 To 12 Me.Combo1.AddItem i Next
     For i = 1978 To 2010 Me.Combo2.AddItem i Next
     Me.Combo1.ListIndex = 0
     Me.Combo2.ListIndex = 0
     Search
End Sub
```

[3]. Tìm kiếm động đơn tiêu chí

Các tiêu chí được người dùng lựa chọn tuỳ ý bằng cách chọn các Option. Tuy nhiên, chỉ có thể chọn được 1 tiêu chí tại mỗi thời điểm.



Hình 8: mẫu form tìm kiếm động đơn tiêu chí

Các đối tượng được đặt tên lần lượt là: optHT, optNS, optL, Combo1, Combo2, Combo3, Text1, Text2, Text3…

Sau đây là mẫu code:

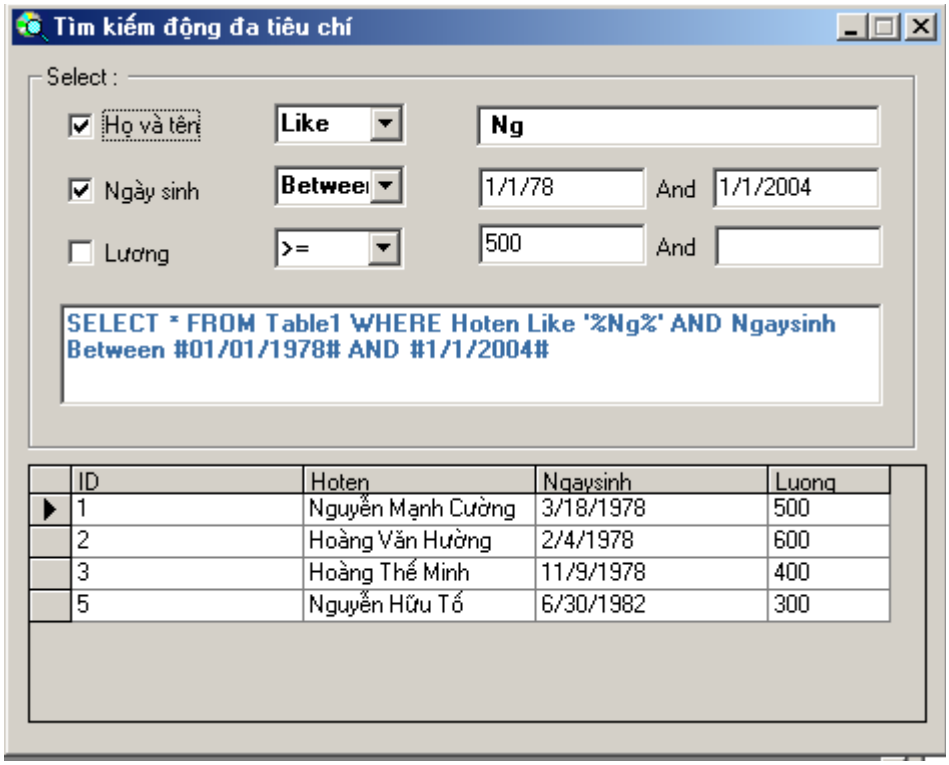
```
Public Sub Search()
      Dim SQL As String
      SQL = "Select * from table1 where "
      If Me.optHT.Value = True Then
          If Trim(Me.Combo1.Text) = "Like" Then
               SQL = SQL & " Hoten " & Me.Combo1.Text & "'%" & Me.Text1.Text & "%'"
          Else
               SQL = SQL & " Hoten " & Me.Combo1.Text & "'" & Me.Text1.Text & "'"
          End If
      End If

      If Me.optNS.Value = True Then
            SQL = SQL & " NgaySinh " & Me.Combo2.Text & "#" &
                  Format(CDate(Me.Text2.Text), "mm/dd/yyyy") & "#"
      End If

      If Me.OptL.Value = True Then
          SQL = SQL & " Luong " & Me.Combo3.Text & Val(Me.Text3.Text)
      End If
        On Error GoTo Er
             Me.Adodc1.RecordSource = SQL
             Me.Adodc1.Recordset.Requery
             Me.Adodc1.Refresh
```

```
                Exit Sub
           Er: ‘ thong bao loi dat o day
End Sub
‘================================
Private Sub Combo1_Click()
      Search
End Sub
‘================================
Private Sub Combo2_Click()
      Search
End Sub
‘================================
Private Sub Combo3_Click()
      Search
End Sub
‘================================
Private Sub Form_Load()
      Me.Combo1.ListIndex = 0
      Me.Combo2.ListIndex = 0
      Me.Combo3.ListIndex = 0
      Search
End Sub
‘================================
Private Sub optHT_Click()
      Search
End Sub
‘================================
Private Sub OptL_Click()
      Search
End Sub
‘================================
Private Sub optNS_Click()
      Search
End Sub
‘================================
Private Sub Text1_Change()
      Search
End Sub
‘================================
Private Sub Text2_KeyPress(KeyAscii As Integer)
      If KeyAscii = 13 Then
      Search
      End If
End Sub
‘================================
Private Sub Text3_Change()
      Search
End Sub
```

[4]. Tìm kiếm động đa tiêu chí

- Các tiêu chí được người dùng chọn tuỳ ý.

- Có thể chọn nhiều tiêu chí một lúc. Thông tin tìm được phải thoả mãn đồng thời các tiêu chí.



Hình 9: Mẫu form tìm kiếm động đa tiêu chí

Đến đây, ta đã quen với phương pháp tạo form tìm kiếm. Điều quan trọng còn lại là tạo cho được hàm Search() để đáp ứng yêu cầu tìm kiếm. Ta hãy xem xét hàm Search() cho phương pháp tìm kiếm động đa tiêu chí này. (Các phần còn lại, sinh viên tự hoàn thiện)

‘==========

Public Sub Search()

'Xay dung cau SQL

SQL = "SELECT * FROM Table1 "

'Kiem tra xem nguoi dung co chon tieu chi nao khong (cau SQL co where khong)

If Me.chkHT.Value = 1 Or Me.chkNS.Value = 1 Or Me.chkT.Value = 1 Then

SQL = SQL & "WHERE "

```
End If
'kiem tra xem co tim kiem theo ho ten khong
If Me.chkHT.Value = 1 Then
   If Trim(Me.cmbHT.Text) = "Like" Then
   SQL = SQL & "Hoten " & Me.cmbHT.Text & " '%" & Trim(Me.txtHT.Text) & "%' "
   Else
   SQL = SQL & "Hoten " & Me.cmbHT.Text & " '" & Trim(Me.txtHT.Text) & "' "
   End If
End If
'Kiem tra xem co tim kiem theo Ho ten ket hop voi Ngay sinh khong
'Neu HT ket hop voi ngay sinh thi....
If Me.chkHT = 1 And Me.chkNS.Value = 1 Then
   If Trim(Me.cmbNS.Text) = "Between" Then
                SQL = SQL & "AND Ngaysinh " & Me.cmbNS.Text & " #" &
Format(CDate(Trim(Me.txtNS1.Text)), "dd/mm/yyyy") & "# "
      SQL = SQL & "AND #" & Format(CDate(Trim(Me.txtNS2.Text)), "dd/mm/yyyy") & "#
"
   Else
                SQL = SQL & "AND Ngaysinh " & Me.cmbNS.Text & " #" &
Format(CDate(Trim(Me.txtNS1.Text)), "dd/mm/yyyy") & "# "
   End If
Else
'neu chi tim theo ngay sinh (HT khong ket hop voi NS thi...
If Me.chkNS.Value = 1 Then
   If Trim(Me.cmbNS.Text) = "Between" Then
                SQL = SQL & "Ngaysinh " & Me.cmbNS.Text & " #" &
Format(CDate(Trim(Me.txtNS1.Text)), "dd/mm/yyyy") & "# "
      SQL = SQL & "AND #" & Format(CDate(Trim(Me.txtNS2.Text)), "dd/mm/yyyy") & "#
"
   Else
                SQL = SQL & "Ngaysinh " & Me.cmbNS.Text & " #" &
Format(CDate(Trim(Me.txtNS1.Text)), "dd/mm/yyyy") & "# "
   End If
```

```
End If
End If
'Neu chi tim kiem theo Luong thi... khong them AND
If (Me.chkHT.Value = 0 And Me.chkNS.Value = 0) And Me.chkT.Value = 1 Then
SQL = SQL & "Luong " & Me.cmbT.Text & " " & Trim(Me.txtT1.Text) & " "
   If Me.cmbT.Text = "Between" Then
      SQL = SQL & " AND " & Trim(Me.txtT2.Text) & " "
   End If
Else
'Truong hop Luong ket hop voi HT hoac/va NS thi them AND thoi.
If Me.chkT.Value = 1 Then
SQL = SQL & "AND Luong " & Me.cmbT.Text & " " & Me.txtT1.Text
   If Me.cmbT.Text = "Between" Then
      SQL = SQL & " AND " & Trim(Me.txtT2.Text) & " "
   End If
Else
End If
End If
Me.txtSQL.Text = SQL
On Error GoTo Er
'Bat dau cho ket qua len luoi
Me.Adodc1.RecordSource = SQL
Me.Adodc1.Recordset.Requery
Me.Adodc1.Refresh
Exit Sub
'Thong bao Loi (neu co)
Er: MsgBox "SQL wrong or Find no record in the table ! Please try again !", vbOKOnly + vbInformation, "Message"
Me.Adodc1.RecordSource = "select * from table1"
Me.Adodc1.Refresh
Me.chkHT.Value = 0
Me.chkNS.Value = 0
```

```
Me.chkT.Value = 0
Me.txtNS1.Text = 0
Me.txtNS2.Text = 0
End Sub
'==============================
```

V. TẠO BÁO CÁO BẰNG VB

1. Phương pháp chung

Nhìn chung, phương pháp tạo báo cáo cũng tương tự như phương pháp lập trình tìm kiếm. Điểm khác cơ bản là kết quả tìm kiếm được trình bày trên mẫu báo cáo để có thể xem và in ra được, thay vì hiển thị trên các lưới như trong bài toán tìm kiếm.

Có rất nhiều phần mềm hỗ trợ làm báo cáo, có thể điểm qua:

- Tạo báo cáo bằng Data Report của Visual Basic
- Tạo báo cáo bằng MS. Access, sau đó gọi báo cáo từ form Visual Basic
- Tạo báo cáo bằng Crystal Report, sau đó gọi báo cáo từ Visual Basic

….

Một báo cáo động là báo cáo đảm bảo các điều kiện sau:

- Các tiêu đề báo cáo có thể thay đổi bởi người dùng tại lúc xem – in báo cáo. Trên thực tế, một phần mềm có thể được chuyển giao cho nhiều khách hàng khác nhau, để thay đổi phần tiêu đề (chẳng hạn Tên công ty) mà không phải thiết kế lại báo cáo thì tiêu đề báo cáo phải động.

- Nội dung dữ liệu trong báo cáo thay đổi tuỳ theo các tiêu chí mà người dùng muốn khi xem – in báo cáo. Như vậy, trước khi xem – in báo cáo, người dùng có thể lựa chọn các tiêu chí để in báo cáo. Chẳng hạn chỉ in ra thông tin của những nhân viên có Lương cao hơn 500 thay vì in thông tin của tất cả các nhân viên…

Sau đây, ta xem xét một phương pháp tạo báo cáo trong Visual Basic mà có thể đáp ứng phần nào các yêu cầu của báo cáo động. Phương pháp sử dụng DataReport. Bảng dữ liệu để báo cáo cũng là bảng table1 của cơ sở dữ liệu DB1.mdb ở trên.

B1. Chọn Project\ Add Data Report để thêm một form báo cáo vào đề án.

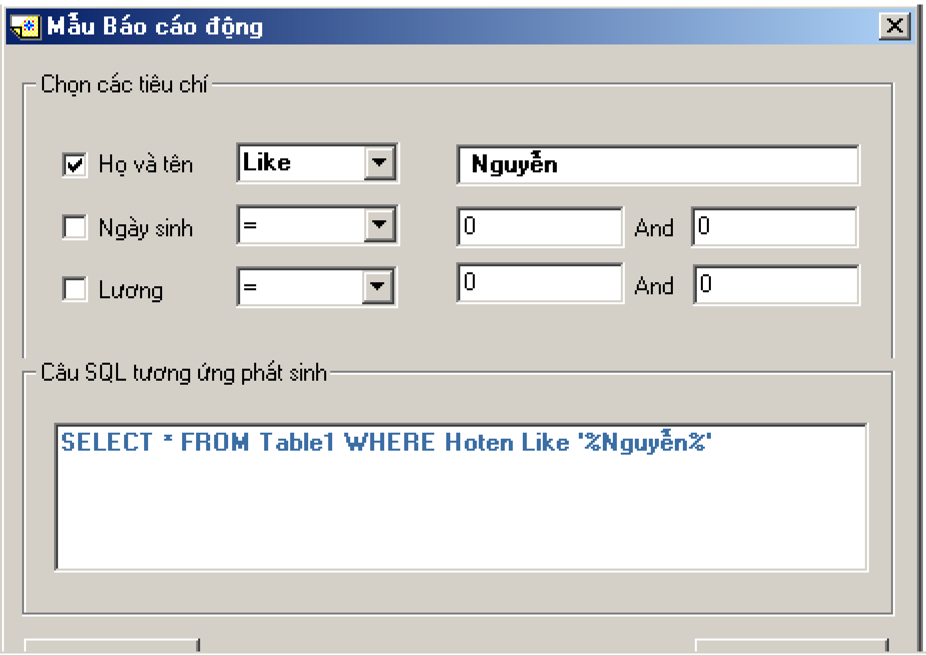
B2. Thiết kế báo cáo (giống như trong Access) trong đó bao gồm các Trường dữ liệu muốn xuất hiện trên báo cáo. Khi đó, cần chú ý:

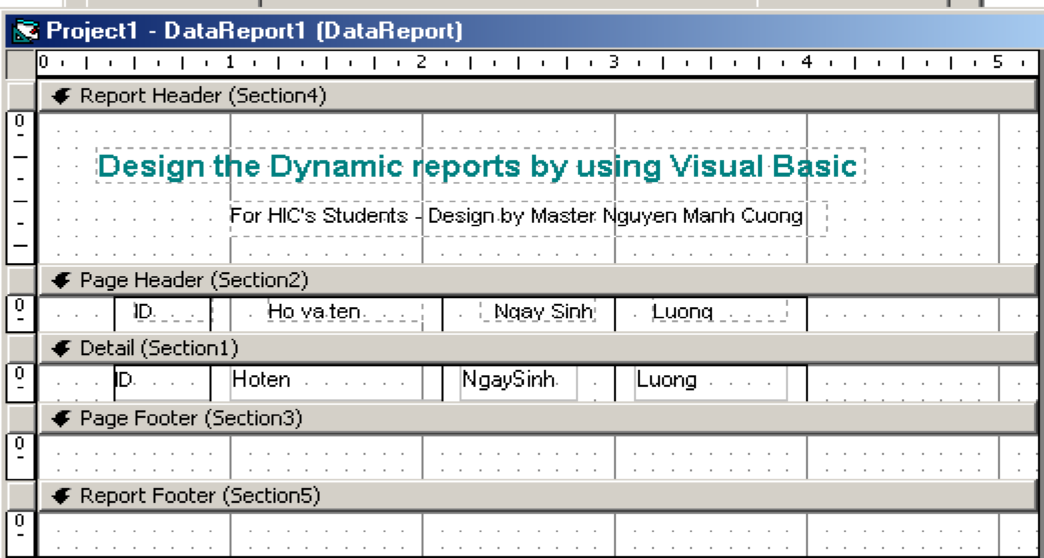
- o Mỗi trường dữ liệu là một TextBox, mỗi tiêu đề là một Label
- o Chưa liên kết báo cáo tới một bảng cụ thể nào.
- o Gõ tên trường tương ứng vào ô Field Name cho từng Text Box

B3. Gọi báo cáo: Trên form, người dùng kích chọn nút gọi Xem – In báo cáo, khi đó cần:

- Xây dựng câu SQL sao cho kết quả của nó là kết quả sẽ hiện trên báo cáo. Câu hỏi SQL được xây dựng từ những tiêu chí do người dùng nhập vào. (tương tự form tìm kiếm động).
- Dùng biến Rs để mở bảng tương ứng với câu SQL vừa tìm được.
- Gán RS cho thuộc tính Me.DataReport1. DataSource.
- Hiển thị form báo cáo.

2. **Ví dụ về báo cáo động**





Hình 11: Mẫu báo cáo thiết kế bằng DataReport

Các thành phần trong Page Header là các tiêu đề (Label)

Các thành phần trong Detail là các TextBox, tương ứng với các trường dữ liệu muốn hiển thị trên báo cáo. ***Mỗi TextBox đều được đặt thuộc tính FieldName bằng tên trường tương ứng trong bảng dữ liệu Table1.***

Khung báo cáo được vẽ bởi các đối tượng Line

Bước 2: tạo form gọi báo cáo như trong Hình 10.

Bước 3: Lập trình

Trước tiên ta tạo hàm MakeSQL. Hàm này trả về câu SQL sao cho kết quả của nó là những dữ liệu muốn hiển thị trên báo cáo. Câu SQL được phát sinh tương tự như trong phần lập trình tìm kiếm.

```
Public Function MakeSQL() As String
     Dim SQL As String
     SQL = "SELECT * FROM Table1 "
     If  Me.chkHT.Value  =  1  Or  Me.chkNS.Value  =  1  Or
         Me.chkT.Value = 1 Then
         SQL = SQL & "WHERE "
     End If

     If Me.chkHT.Value = 1 Then
         If Trim(Me.cmbHT.Text) = "Like" Then
           SQL = SQL & "Hoten " & Me.cmbHT.Text & " '%" &
                 Trim(Me.txtHT.Text) & "%' "
         Else
```

```
            SQL = SQL & "Hoten " & Me.cmbHT.Text & " '" &
                Trim(Me.txtHT.Text) & "' "
        End If
    End If
    If Me.chkHT = 1 And Me.chkNS.Value = 1 Then
        If Trim(Me.cmbNS.Text) = "Between" Then
            SQL = SQL & "AND Ngaysinh " & Me.cmbNS.Text & " #" &
              Trim(Me.txtNS1.Text) & "# AND #" & Me.txtNS2.Text &
              "# "
        Else
            SQL = SQL & "AND Ngaysinh " & Me.cmbNS.Text & " "
                 & Trim(Me.txtNS1.Text) & " "
        End If
    Else
    If Me.chkNS.Value = 1 Then
        If Trim(Me.cmbNS.Text) = "Between" Then
              SQL = SQL & "Ngaysinh " & Me.cmbNS.Text & " #" &
                Format(CDate(Trim(Me.txtNS1.Text)), "dd/mm/yyyy")
                & "# AND #" & Format(CDate(Trim(Me.txtNS2.Text)),
                "dd/mm/yyyy")  & "# "
        Else
             SQL = SQL & "Ngaysinh " & Me.cmbNS.Text & " " &
                Trim(Me.txtNS1.Text) & " "
        End If
    End If
    End If

    If  (Me.chkHT.Value  =  0  And  Me.chkNS.Value  =  0)  And
         Me.chkT.Value = 1 Then
         SQL  =  SQL  &  "  Luong  "  &  Me.cmbT.Text  &  "  "  &
                Trim(Me.txtT1.Text) & " "
        If Me.cmbT.Text = "Between" Then
            SQL = SQL & " AND " & Trim(Me.txtT2.Text) & " "
        End If
    Else
    If Me.chkT.Value = 1 Then
             SQL = SQL & " AND Luong " & Me.cmbT.Text & " " &
                   Me.txtT1.Text
        If Me.cmbT.Text = "Between" Then
            SQL = SQL & " AND " & Me.txtT2.Text & " "
        End If
    Else
    End If
    End If
MakeSQL = SQL
End Function
```

Tại nút Xem – in báo cáo, ta lập trình sử dụng biến Rs để mở bảng dữ liệu từ câu SQL trả về từ hàm MakeSQL, sau đó gán biến Rs cho thuộc tính DataSource của DataReport. Tuy nhiên có thể phát sinh lỗi do câu SQL sinh ra bị lỗi, tốt nhất, hãy chú ý bẫy lỗi.

```
Private Sub CmdXemIn_Click()
    On Error GoTo Err
    If ME.ADODC1.State = 1 Then ME.ADODC1.Close
    ME.ADODC1.Open  MakeSQL, DB, adOpenKeyset, adLockPessimistic
    Set DataReport1.DataSource = RS
    DataReport1.Show
    Exit Sub
    Err:  MsgBox "Sai câu SQL  !", vbInformation + vbOKOnly, "Message"
End sub
```

Một báo cáo động hoàn chỉnh được giới thiệu trong kỹ thuật tạo báo cáo bằng CrystalReport, có sẵn trong phần Help của bản demo kèm theo.

Mục lục

Đọc thêm:

Thiết lập môi trường cần thiết để làm việc

Để sử dụng các kỹ thuật nêu trên trong đợt thực tập và làm đồ án này, Xa hơn nữa là trở thành một lập trình viên chuyên nghiệp. Sinh viên cần các công cụ hỗ trợ sau:

[1]. Bộ OCX chuyên dụng: cung cấp các OCX cần thiết giúp sinh viên có các điều khiển mới cần thiết trong việc lập trình. Bộ này bao gồm 7 bộ OCX với kích thước 50 MB được cung cấp bởi tác giả.

Bộ OCX đã được Crack với mỗi file crack đi kèm. Sinh viên cần tìm hiểu cách Crack. Tuy nhiên có rất nhiều OCX khác nữa, sinh viên có thể tìm kiếm trên mạng hoặc từ đồng nghiệp.

[2]. Phần mền Winhelp: giúp sinh viên tạo các bộ Help cho ứng dụng. Được cung cấp bởi tác giả hoặc có thể Download trên mạng với kích thước 2.8MB. Sinh viên hoàn toàn có thể không dùng Winhelp để viết Help mà có thể dùng các phần mềm khác như RoboHelp, HTMLHelp... cũng rất hữu dụng.

[3]. Bộ Seagate Crystal report: Phiên bản 8.0; 8.5; 9.0 hoặc 9.5 được ghi thành 01 đĩa CD hiện có bán tại các đại lý đĩa phần mềm (có thể mua tại số 2 Tạ Quang Bửu Hà Nội). Bộ này cũng có thể lấy từ tác giả với kích thước 97 MB (7.5).

[4]. Bộ tạo bộ cài SetupFactory: Giúp sinh viên đóng gói sản phẩm để tạo ra các bộ cài rất tốt. Bộ này được cung cấp từ tác giả với kích thước 5MB. Sinh viên có thể DownLoad trên mạng hoặc tìm mua tại các đại lý đĩa CD phần mềm. Sinh viên hoàn toàn có thể dùng các chương trình tạo bộ cài khác như: InstallShield, bộ đóng gói của VB (nếu không dùng các OCX)

[5]. Bộ ảnh chuyên dụng: Để giúp phần mềm chuyên nghiệp hơn, sinh viên có thể sử dụng bộ ảnh, biểu tượng chuyên dụng để đưa vào phần mềm của mình. Bộ ảnh được cung cấp bởi tác giả có kích thước 30MB. Sinh viên có thể downLoad trên mạng những ảnh khác cung rất đa dạng.

[6]. Bộ Help của MicroSoft: Là bộ MSDN nổi tiếng trợ giúp bằng tiếng Anh hầu hết các kỹ thuật lập trình trên Windows. Với dung lương trên 1 GB, sinh viên có thể mua trọn bộ 3 đĩa CD tại các đại lý CD có tín nhiệm (có thể mua tại 2- Tạ Quang Bửu Hà Nội).

[7]. Ngoài các phần mềm trợ giúp trên, sinh viên có thể sử dụng các công cụ mạnh hơn trợ giúp thiết kế và lập trình như:

+ Rational Rose (để thiết kế phần mềm hướng đối tượng)

+ Visio (để thiết kế các biểu đồ - nói chung là tất cả công việc của thiết kế phần mềm - rất tốt) được đóng gói trên 1 đĩa CD, có bán tại các đại lý đĩa CD phần mềm.

+ SQL Server (để tạo các database có thể hỗ trợ môi trường mạng tốt hơn - thường được dùng với các phần mềm hỗ trợ đa truy cập và mô hình Client/ Server)

Về trang chủ