

VỤ GIÁO DỤC CHUYÊN NGHIỆP

GIÁO TRÌNH **ACCESS** và ỨNG DỤNG

SÁCH DÙNG CHO CÁC TRƯỜNG ĐÀO TẠO HỆ TRUNG HỌC CHUYÊN NGHIỆP



NHÀ XUẤT BẢN GIÁO DỤC

TS. HUỖNH QUYẾT THẮNG

Giáo trình
ACCESS VÀ ỨNG DỤNG

(Sách dùng cho các trường đào tạo hệ Trung học chuyên nghiệp)

NHÀ XUẤT BẢN GIÁO DỤC

Năm 2002, Vụ Giáo dục Chuyên nghiệp – Bộ Giáo dục và Đào tạo đã phối hợp với Nhà xuất bản Giáo dục xuất bản 21 giáo trình phục vụ cho đào tạo hệ THCN. Các giáo trình trên đã được nhiều trường sử dụng và hoan nghênh. Để tiếp tục bổ sung nguồn giáo trình đang còn thiếu, Vụ Giáo dục Chuyên nghiệp phối hợp cùng Nhà xuất bản Giáo dục tiếp tục biên soạn một số giáo trình, sách tham khảo phục vụ cho đào tạo ở các ngành : Điện – Điện tử, Tin học, Khai thác cơ khí. Những giáo trình này trước khi biên soạn, Vụ Giáo dục Chuyên nghiệp đã gửi đề cương về trên 20 trường và tổ chức hội thảo, lấy ý kiến đóng góp về nội dung đề cương các giáo trình nói trên. Trên cơ sở nghiên cứu ý kiến đóng góp của các trường, nhóm tác giả đã điều chỉnh nội dung các giáo trình cho phù hợp với yêu cầu thực tiễn hơn.

Với kinh nghiệm giảng dạy, kiến thức tích lũy qua nhiều năm, các tác giả đã cố gắng để những nội dung được trình bày là những kiến thức cơ bản nhất nhưng vẫn cập nhật được với những tiến bộ của khoa học kỹ thuật, với thực tế sản xuất. Nội dung của giáo trình còn tạo sự liên thông từ Dạy nghề lên THCN.

Các giáo trình được biên soạn theo hướng mở, kiến thức rộng và cố gắng chỉ ra tính ứng dụng của nội dung được trình bày. Trên cơ sở đó tạo điều kiện để các trường sử dụng một cách phù hợp với điều kiện cơ sở vật chất phục vụ thực hành, thực tập và đặc điểm của các ngành, chuyên ngành đào tạo.

Để việc đổi mới phương pháp dạy và học theo chỉ đạo của Bộ Giáo dục và Đào tạo nhằm nâng cao chất lượng dạy và học, các trường cần trang bị đủ sách cho thư viện và tạo điều kiện để giáo viên và học sinh có đủ sách theo ngành đào tạo. Những giáo trình này cũng là tài liệu tham khảo tốt cho học sinh đã tốt nghiệp cần đào tạo lại, nhân viên kỹ thuật đang trực tiếp sản xuất.

Các giáo trình đã xuất bản không thể tránh khỏi những sai sót. Rất mong các thầy, cô giáo, bạn đọc góp ý để lần xuất bản sau được tốt hơn. Mọi góp ý xin gửi về : Công ty Cổ phần sách Đại học – Dạy nghề, 25 Hàn Thuyên – Hà Nội.

VỤ GIÁO DỤC CHUYÊN NGHIỆP - NXB GIÁO DỤC

Mở đầu

Trong thời đại ngày nay, máy tính đã có mặt trong mọi lĩnh vực xã hội, đặc biệt nó là một công cụ hỗ trợ không thể thiếu trong công tác quản lý.

Môn học Hệ quản trị cơ sở dữ liệu đã được đưa vào giảng dạy của các Khoa CNTT và Toán tin tại các Trường Đại học, Cao đẳng. Với mong muốn có một giáo trình hỗ trợ cho các em sinh viên, những bạn trẻ muốn học sử dụng và lập trình với Hệ quản trị cơ sở dữ liệu, chúng tôi viết Giáo trình này nhằm trang bị cho độc giả một số kiến thức cơ bản về khái niệm cơ sở dữ liệu quan hệ và một số kỹ năng trong Microsoft Access để giúp các bạn độc giả có thể xây dựng nhanh các ứng dụng nhỏ, nhằm phục vụ công tác quản lý hoặc học tập của các bạn được tốt hơn.

Giáo trình được chia làm 6 chương với nội dung xuyên suốt :

Chương I - Giới thiệu hệ quản trị cơ sở dữ liệu Access

Chương II - Tạo lập cơ sở dữ liệu

Chương III - Thiết kế biểu mẫu và báo cáo

Chương IV - Lập trình trên ACCESS

Chương V - Sử dụng đối tượng trong Visual Basic for Access

Chương VI - Sử dụng cơ sở dữ liệu trong Visual Basic

Cuối mỗi chương đều có các câu hỏi ôn tập và bài tập. Các bạn độc giả khi đọc từng chương có thể cài đặt và sử dụng các ví dụ trình bày trong các chương để thực hành. Những bài tập tự làm cuối mỗi chương là những bài kiểm tra nhỏ giúp bạn củng cố kiến thức của mình, các bạn không nên bỏ qua những bài tập này.

Chúc các bạn thành công.

Rất mong nhận được ý kiến đóng góp của các bạn để lần tái bản sau được hoàn chỉnh hơn. Mọi góp ý xin được gửi về : Nhà XBGD - 81 Trần Hưng Đạo, Hà Nội.

TÁC GIẢ

Chương I

GIỚI THIỆU HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ACCESS

1.1. KHÁI NIỆM VÀ CÁC TÍNH NĂNG CỦA HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU QUAN HỆ

Hệ quản trị Cơ sở dữ liệu (CSDL) quan hệ – DBMS là hệ cho phép lưu trữ và lọc thông tin có cấu trúc từ thiết bị lưu trữ. Ví dụ về các hệ quản trị CSDL mạnh :

- Oracle
- Microsoft SQL Server
- IBM DB2
- Informix

Ví dụ về DBMS để bàn có :

- Microsoft Access
- Microsoft Foxpro
- Borland dBase

1.2. GIỚI THIỆU HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ACCESS

Access là một hệ quản trị CSDL có nhiều các tính năng cần thiết. Ví dụ, Access chứa các thành phần sau :

- Một hệ quản trị CSDL hỗ trợ hai ngôn ngữ văn tin chuẩn công nghiệp là SQL (Structured Query Language) và QBE (Query By Example).
- Ngôn ngữ lập trình thủ tục – về cơ bản là một tập con của Visual Basic.
- Môi trường triển khai ứng dụng nhanh có các công cụ phát triển báo biểu và báo cáo trực quan.
- Hỗ trợ lập trình hướng đối tượng.
- Các thuật sỹ (wizard và builder) trợ giúp thiết kế.

Microsoft Access được sử dụng để tập hợp, lưu trữ và xử lý nhiều kiểu thông tin. Sự đa dạng này nhiều khi gây khó khăn cho người mới sử dụng, vì vậy khi làm việc với Access cần xem xét dưới những góc độ khác nhau, ví dụ :

- Dưới góc độ CSDL quan hệ, xem ứng dụng gồm các bộ dữ liệu.
- Dưới góc độ lập trình thủ tục, xem ứng dụng là những lệnh thực hiện tuần tự.
- Dưới góc độ hướng đối tượng, xem ứng dụng gồm các đối tượng có phương thức, thuộc tính và tương tác qua lại.

Việc lựa chọn phương pháp phát triển ứng dụng nào là tùy thuộc vào người sử dụng nhưng tốt nhất là hiểu hết các tính năng của Access để áp dụng vào từng việc sao cho phù hợp nhất.

Lợi ích lớn nhất khi dùng Access là người sử dụng làm quen được với các khái niệm của các hệ thống khác nhau :

- Oracle cho CSDL quan hệ.
- PowerBuilder để triển khai ứng dụng nhanh.
- SmallTalk cho lập trình hướng đối tượng.

Tập CSDL của Access

- Khái niệm CSDL thường chỉ một tập các bảng liên quan đến nhau nhưng một tập CSDL của Access bao gồm nhiều thành phần hơn :

- + Các query để vấn tin.
- + Các form để tương tác.
- + Các report để in báo cáo.
- + Macro và đơn thể VBA để lập trình.

- Khái niệm “cơ sở dữ liệu” bao hàm nhiều nghĩa khác nhau tùy thuộc vào hệ quản trị CSDL sử dụng. Ví dụ trong dBase IV, một CSDL là một tệp (<tên tệp >.dbf) chứa một bảng duy nhất, các biểu mẫu (Form) và báo biểu (Report) được lưu trên từng tệp riêng lẻ. Ngược lại, trong Oracle một CSDL có thể chứa nhiều bảng từ nhiều dự án/ ứng dụng khác nhau và bản thân CSDL có thể chia làm nhiều tệp (thậm chí có thể nằm trên nhiều máy).

Access nằm ở vị trí cân bằng ở giữa – tất cả các đối tượng (bảng, truy vấn, biểu mẫu, báo biểu, ...) của một dự án/ ứng dụng được lưu trong một tệp. Tuy nhiên, cũng có thể tách các đối tượng thành nhiều tệp, ví dụ, các bảng vào một tệp, chương trình nằm ở tệp khác.

– Dồn tệp CSDL : Một tệp CSDL của Access có thể phân mảnh và trở nên lớn hơn rất nhiều so với thực tế, có thể vài megabyte cho một tệp chỉ có vài bản ghi. Dồn tệp sẽ loại bỏ phân mảnh và giảm đáng kể kích thước tệp.

– Đổi tên CSDL : Access không có lệnh Save As cho một CSDL, để đổi tên ta sử dụng Explorer của Windows hoặc khi compact một tệp đang đóng, Access cho khả năng lưu tệp đã dồn dưới một tên khác.

– Sử dụng bảng liên kết : Nhiều nhà thiết kế ứng dụng Access đặt các bảng trong một tệp CSDL và các đối tượng còn lại như truy vấn, báo biểu, chương trình trên một tệp khác nhằm để tách biệt giữa dữ liệu và chương trình.

Access cho phép sử dụng tính năng “linked table” để liên kết bảng từ một tệp này sang một tệp khác. Tệp chứa tất cả các bảng là tệp dữ liệu (data) còn tệp chứa các chương trình là tệp giao diện (interface).

Nhờ cách tách riêng dữ liệu ta có thể dễ dàng nâng cấp ứng dụng mà không ảnh hưởng đến dữ liệu. *Chú ý : bảng link phụ thuộc địa chỉ thư mục tuyệt đối.*

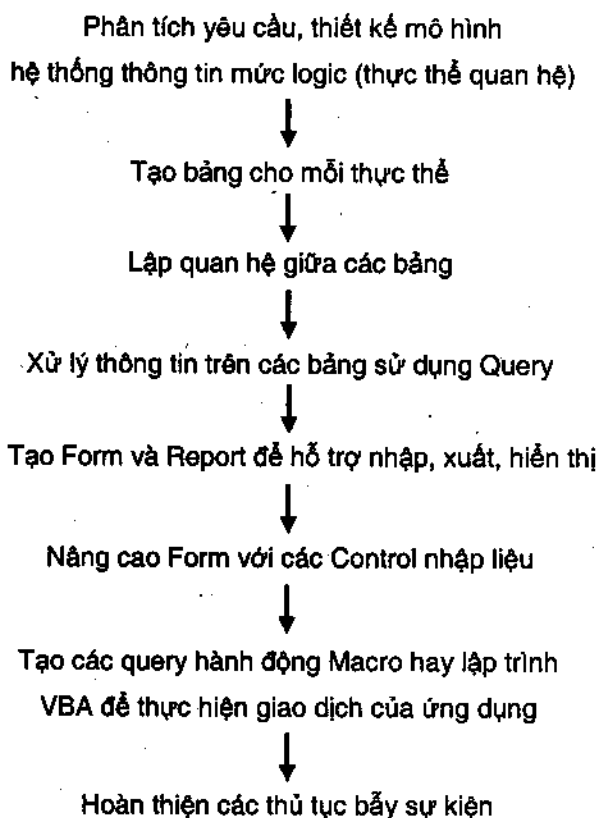
– Viết một ứng dụng với Access : có hai cách cơ bản để xây dựng một hệ thống thông tin :

+ Phân tích, thiết kế và cài đặt hệ thống một cách kỹ lưỡng.

+ Lập trình thử nghiệm nhanh (trong đó các khâu phân tích, thiết kế và cài đặt được thực hiện lặp đi lặp lại).

Access cung cấp nhiều tính năng như công cụ thiết kế trực quan, các Wizard, Macro thuận lợi cho việc thiết kế nhanh. Access thường được sử dụng để thiết kế hệ thống vừa phải và nhanh nên cách tiếp cận thứ hai phù hợp hơn.

Các bước để lập trình thử nghiệm một ứng dụng Access được thể hiện trong sơ đồ sau :



1.3. HỆ THỐNG MENU CHÍNH CỦA ACCESS (hình 1.1)

Để khởi động Access, tìm biểu tượng của Access rồi nhấn kép chuột hoặc gọi Microsoft Access từ nhóm Microsoft Office trên cửa sổ Start.

– File : Get external Data : lấy dữ liệu từ nguồn khác ví dụ từ Foxpro, Excel Text.

Database Properties : xem thuộc tính của CSDL.

Export : xuất các đối tượng ra ngoài.

– Edit/ Create shortcut.

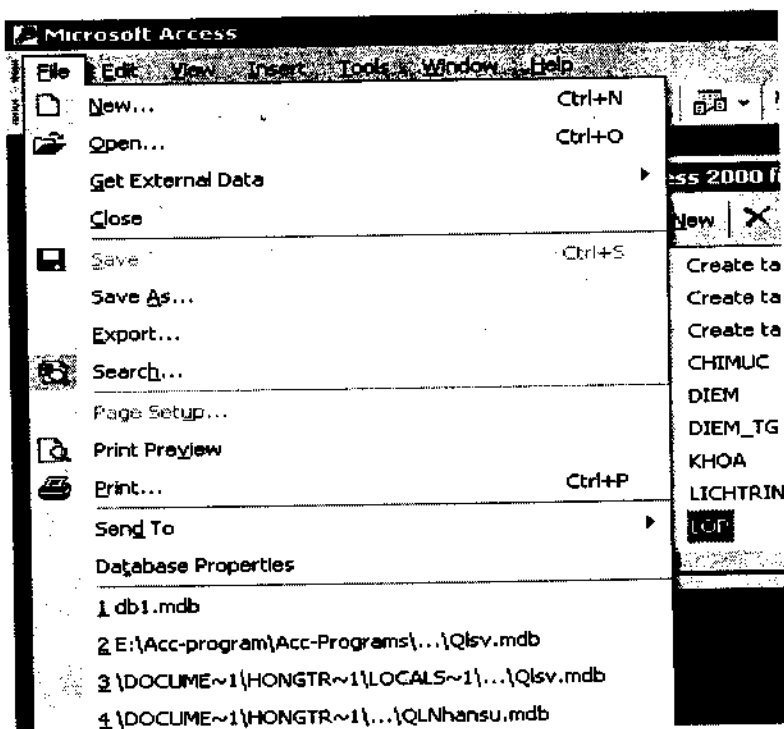
– Tools : Relationship : thể hiện mối quan hệ giữa các bảng

Database Utilities : các tiện ích của CSDL

– View

Database Object : xem các đối tượng như : bảng, truy vấn, form, report ...

- Insert
- Windows
- Help : có thể nhấn F1



Hình 1.1. Hệ thống menu chính của Access 2000.

1.4. CÁCH TỔ CHỨC DỮ LIỆU TRONG ACCESS

Tất cả các CSDL được thiết kế bằng Microsoft Access đều sử dụng các đối tượng CSDL cơ bản (hình 1.2). Đối tượng của Microsoft Access bao gồm :

Bảng (table) : Nơi trực tiếp chứa dữ liệu. Mỗi bảng chứa thông tin về một kiểu dữ liệu riêng, ví dụ như các khách hàng, các loại sản phẩm, các nhà cung cấp hay các nhân công. Thông thường việc tìm kiếm các thông tin đó được thực hiện trên nhiều bảng có liên quan nhau. Một CSDL thực sự có quan hệ sẽ có nhiều bảng và số bảng đó tùy thuộc vào chức năng mà CSDL đó được thiết kế. Bên trong mỗi bảng, thông tin được lưu trữ theo những bản ghi (record).

Truy vấn (query) : Tạo nguồn dữ liệu cho các giao diện nhập liệu, các báo cáo của người sử dụng trực tuyến.

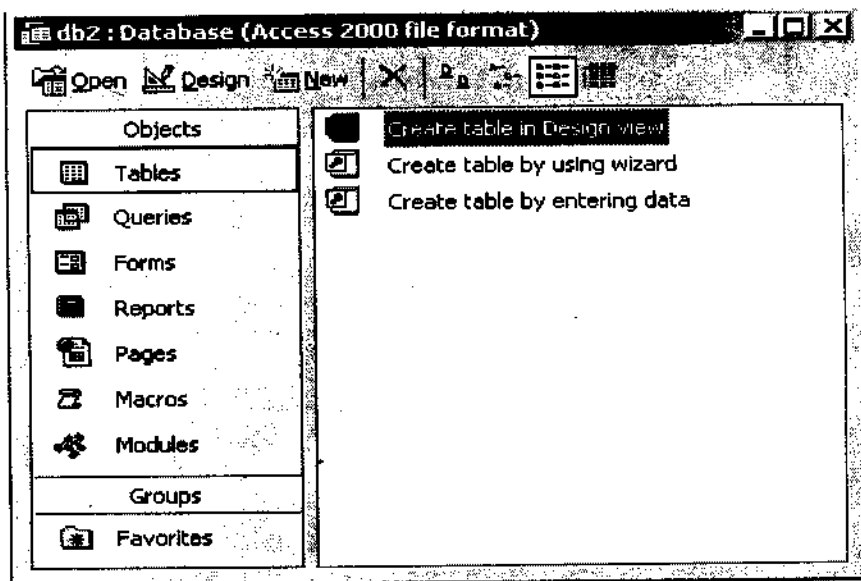
Biểu mẫu (form) : Để xây dựng giao diện giữa người sử dụng và máy, nhập dữ liệu vào CSDL, xây dựng menu cho người sử dụng...

Báo cáo (report) : Dùng đưa thông tin ra giấy.

Macro : Là một hình thức lập trình đơn giản được sử dụng để gắn kết các đối tượng chính trong chương trình như liên hệ giữa các form, tạo menu...

Module : Là chương trình viết bằng Visual Basic for Applications (VBA).

Trang WEB : Chứa các trang web xây dựng từ các đối tượng của CSDL.



Hình 1.2. Các đối tượng CSDL cơ bản của Access 2000.

1.5. CÔNG CỤ WIZARD VÀ BUILDER

– Wizard có thể sử dụng ở nhiều nơi khác nhau :

+ Tạo lập toàn bộ một CSDL (Database Wizard).

+ Trên bảng, truy vấn, biểu mẫu, báo cáo. Ví dụ, sử dụng Tables Wizards để trợ giúp tạo ra một bảng bằng cách chọn một trong các cấu trúc table ví dụ, rồi chọn các trường mà ta muốn chứa trong bảng đó sẽ

nhANH chóng tạo ra một table hoàn chỉnh. Nếu table không hoàn toàn chính xác như ý muốn, có thể thay đổi nó trong chế độ Design View.

+ Trên các điều khiển.

+ Tạo trang web.

– *Expresion Builder* là bộ công cụ thuận tiện nhất để tạo các biểu thức, các điều khiển liên quan tới form hoặc report.

1.6. CÁC VÍ DỤ

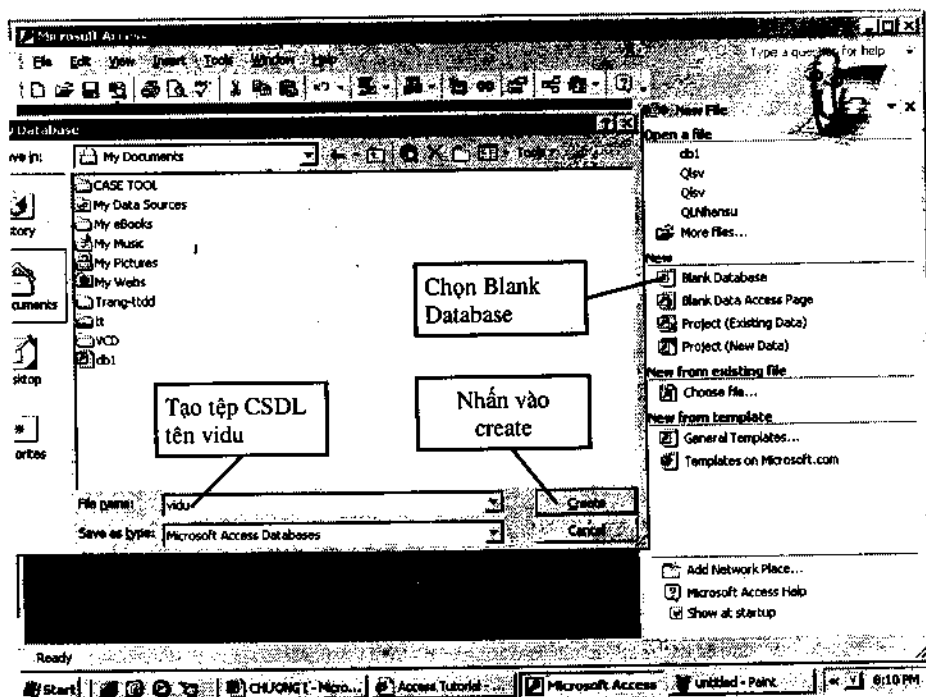
Ví dụ 1.1 :

Tạo một CSDL vidu.mdb (ví dụ sử dụng Access 2000) (hình 1.3).

1. Chạy Access, xuất hiện cửa sổ Microsoft Access.

2. Chọn Blank Database, nhấn OK → xuất hiện cửa sổ File New Database.

3. Nhập tên vào mục File Name, ví dụ vidu sau đó nhấn Create. Tập vidu được tạo và mở ra.



Hình 1.3. Tạo một tập CSDL mới trong Access 2000.

Ví dụ 1.2 :

Mở một CSDL cũ

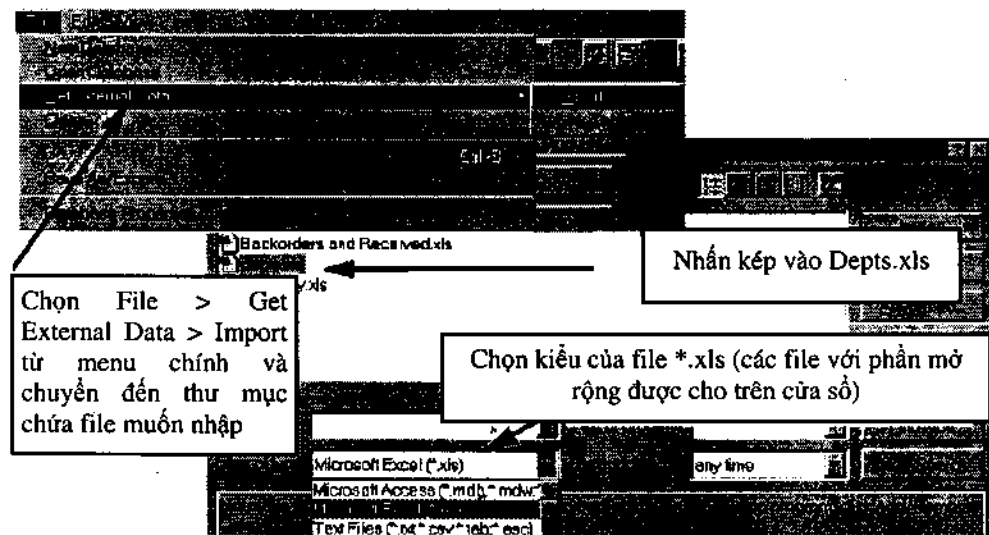
1. Chạy Access, xuất hiện cửa sổ Microsoft Access.
2. Chọn File > Open.
3. Chọn tên tệp cần mở.

Ví dụ 1.3 :

Nhập dữ liệu từ bảng tính

Việc nhập dữ liệu từ các ứng dụng khác sang bảng của Access rất đơn giản. Ví dụ, nhập một bảng từ Excel (hình 1.4, 1.5).

1. Gọi File > Get External Data > Import.
2. Trong hộp File type chọn Microsoft Excel, chọn tên tệp cần nhập.
3. Chọn Show Worksheets, chọn tên bảng.
4. Chọn First Row Contains Column Heading.
5. Chọn In New Table.
6. Chọn No Primary Key.



Hình 1.4. Nhập dữ liệu từ bảng tính. Chọn tên tệp và kiểu tệp.

Chương II

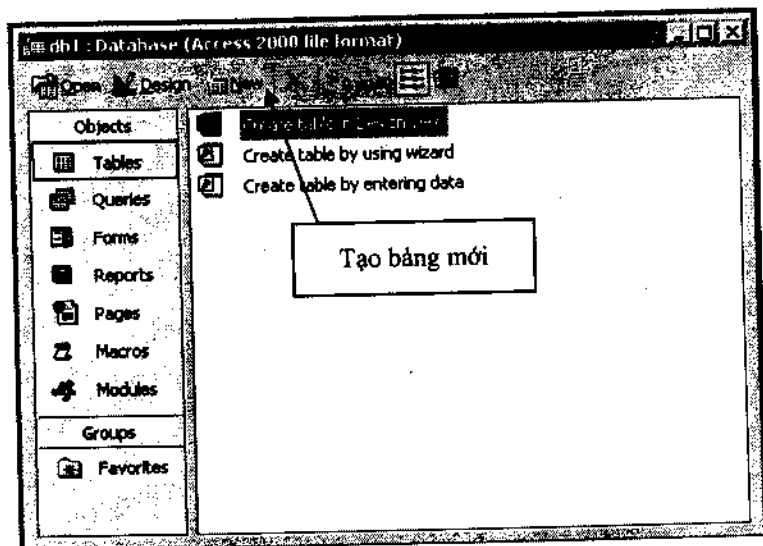
TẠO LẬP CƠ SỞ DỮ LIỆU

2.1. TẠO BẢNG

Bảng là nơi chứa dữ liệu của CSDL, vì vậy bảng là nền tảng của mọi ứng dụng CSDL. Bên cạnh dữ liệu, Access còn cho phép lưu những thuộc tính dữ liệu kèm theo bảng như tiêu đề cột, giá trị ngầm định, định dạng hiển thị hay nhập...

Nghiên cứu kỹ việc thiết kế bảng sẽ cho phép người sử dụng đơn giản hoá được nhiều công việc trong những giai đoạn kế tiếp. Nếu cấu trúc bảng hay quan hệ giữa các bảng bị thay đổi sẽ dẫn đến nhiều thay đổi khác trong ứng dụng, điều đó sẽ làm cho công việc trở nên rất phức tạp.

Màn hình tạo bảng mới của Access 2000 sẽ có dạng như hình 2.1



Hình 2.1. Màn hình tạo bảng mới của Access 2000.

Ví dụ : Tạo bảng tên là SINHVIEN để theo dõi sinh viên trong trường
Bảng gồm các thông tin sau :

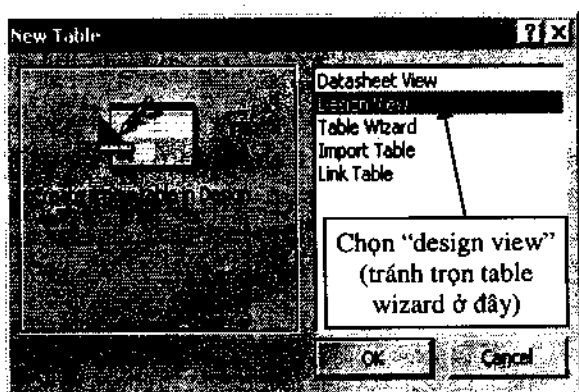
TÊN TRƯỜNG	KIỂU DỮ LIỆU	MÔ TẢ (không bắt buộc)
MASV	TEXT	Mã sinh viên
HO	TEXT	Họ của sinh viên
DEM	TEXT	Phần tên đệm của sinh viên
TEN	TEXT	Tên của sinh viên
NAM	YES/NO	Nam hay nữ
NGSINH	DATE/TIME	Ngày sinh của sinh viên
QUEQUAN	TEXT	Quê quán của sinh viên
NOIO	TEXT	Nơi ở hiện tại
MALOP	TEXT	Mã lớp

Các bước tiến hành tạo bảng :

1. Chọn New từ menu File, chọn Blank Database, đặt tên cho CSDL là quanlysinhvien

2. Chọn phiếu Table, nhấn New.

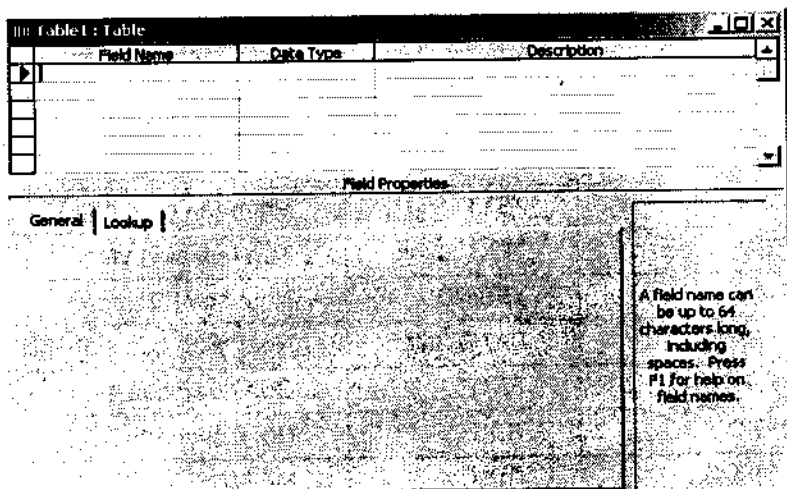
3. Chọn Design View trong cửa sổ New Table (hình 2.2). Màn hình nhập tham số của bảng sẽ hiện ra (hình 2.3).



Hình 2.2. Màn hình chọn kiểu thiết kế cho bảng.

Các kiểu dữ liệu :

- Text : xâu ký tự ≤ 255 ký tự.
- Memo : văn bản ≤ 64000 ký tự.
- Number : số (số nguyên và số thực).
- AutoNumber : tự động tăng khi thêm bản ghi mới.
- Currency : tiền tệ.
- Yes/No : logic
- Date/Time : ngày/giờ.
- OLE Object : lưu ảnh biểu đồ văn bản lớn.
- Hyperlink : địa chỉ trang web.



Hình 2.3. Màn hình nhập các tham số của bảng.

4. Trong cửa sổ Table Design, thiết kế bảng theo thông tin sau (các thông tin này là cơ bản, ngoài ra có thể thêm một số trường phụ).

Lần lượt thực hiện các thao tác nhập tên trường, chọn kiểu dữ liệu và nhập mô tả tên trường cho 9 trường của bảng, chọn File > Save từ menu hay nhấn tổ hợp phím CTRL S để lưu bảng dưới tên SINHVIEN (hình 2.4, 2.5).

Gõ vào tên trường và kiểu dữ liệu

Cột mô tả cho phép gõ vào một đoạn chú thích ngắn về trường (thông tin này không được xử lý bởi Access)

Field Name	Data Type	Description
ID	Text	Mã sinh viên
HO	Text	Họ
DEM	Text	Dem
TEN	Text	Tên
NAM	Yes/No	Nam hay Nu
NGAYSINH	Date/Time	Ngày sinh
QUEQUAN	Text	Que quan
NOIO	Text	Nơi ở hiện tại
LOAI	Text	

Field Properties

General | Lookup

Field Size: 8

Format:

Input Mask: *SY0000000

Caption:

Default Value: *SY0000000

Validation Rule:

Validation Text:

Required: Yes

Allow Zero Length: No

Indexed: Yes (No Duplicates)

Unicode Compression: Yes

IME Mode: No Control

IME Sentence Mode: None

Các thuộc tính trường cho phép ghi thông tin về trường và các ràng buộc về các giá trị cho trường.

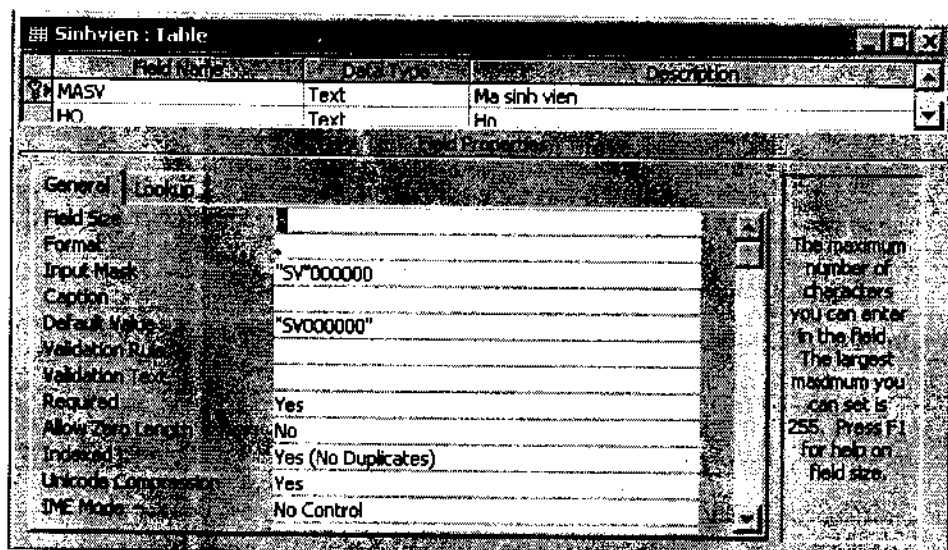
Hình 2.4. Màn hình tạo các trường của bảng.

KÝ TỰ	Ý NGHĨA
0	Các chữ số. Bắt buộc nhập
9	Các chữ số hoặc dấu cách. Không bắt buộc nhập
#	Các chữ số, dấu +, - hoặc dấu cách. Không bắt buộc nhập
L	Chữ cái. Bắt buộc nhập
?	Chữ cái. Không bắt buộc nhập
A	Chữ cái hoặc chữ số. Bắt buộc nhập
a	Chữ cái hoặc chữ số. Không bắt buộc nhập
&	Ký tự bất kỳ hoặc khoảng trống. Bắt buộc nhập
.,- /	Các dấu phân cách cho kiểu dữ liệu số và ngày giờ
<	Đổi các ký tự bên phải ký tự này thành chữ thường
Ký tự khác	Hiện nguyên dạng ký tự đó. Không được lưu trong bảng. Tương tự thuộc tính format. Dùng cách này người sử dụng dễ hiểu hơn.
Password	Che giấu thông tin khi nhập

Nếu sử dụng chưa thành thạo, có thể gọi Wizard bằng cách nhấn ô vuông bên phải của Input Mask.

Ví dụ 2.1 :

Thuộc tính Input Mask của trường MASV của bảng SINHVIEN (hình 2.6).



Hình 2.6. Ví dụ về thuộc tính Input Mask của trường MASV.

- *Default Value* : Giá trị ngầm định, thường là một biểu thức có xuất hiện hằng, các hàm mẫu và các phép toán.
- *Required* : (Yes/No – Ngầm định No) không cho phép đưa giá trị Null vào trường.
- *AllowZeroLength* : (Yes/No – Ngầm định No) chỉ dùng cho các trường kiểu Text hay Memo : cho phép nhận giá trị là chuỗi rỗng.
- *Caption (phụ đề)* : Tên cột khi hiện bảng ở chế độ Datasheet.
- *Validation Rule* : Điều kiện mà dữ liệu nhập vào trường phải thoả mãn. Điều kiện dài tối đa 2000 ký tự.

Có thể sử dụng các ký tự sau :

- + Các hằng : Cặp nháy kép cho hằng chuỗi ký tự, cặp # cho hằng ngày giờ (khi sử dụng các hằng phải dùng dấu cách).
- + Các phép toán so sánh : =, <>, >, <, >=, <=
- + Các phép toán logic : Not, And, Or, Xor, Imp, Like, In ...
- + Các toán tử đặc biệt khi thao tác dữ liệu: Between, Like, In ...

Ví dụ 2.2 : trường NGSINH

- *Validation Rule* : Điều kiện mà dữ liệu nhập vào trường phải thoả mãn. Khi cần đặt điều kiện tuổi của sinh viên tính tới thời điểm hiện tại nằm trong khoảng 18 đến 70, ta viết như sau :

Year(Date()) – Year([NGSINH]) between 18 and 70

- *Validation Text* : Dòng thông báo được hiển thị khi dữ liệu nhập vào trường vi phạm điều kiện được nêu trong Validation Rule.
- *Lookup* : Để kiểm soát dữ liệu đảm bảo dữ liệu nhập vào một trường chỉ nhận một trong số các giá trị hoặc lấy từ một trường nào đó ở bảng khác cần chuyển thành dạng trường lookup.
- + Display Control : Listbox/ Combobox
- + Row Source Type : Bảng/ Truy vấn danh sách giá trị
- + Row Source : Tên bảng/ truy vấn lấy giá trị. Dãy giá trị phân cách bằng dấu ; . Ví dụ "A" ; "B" ; "C".
- + Bound Column : số thứ tự của cột cho giá trị
- + Column Count : số cột được hiển thị trong hộp Combo

+ Column Width : độ rộng các cột (phân cách bằng dấu ;)

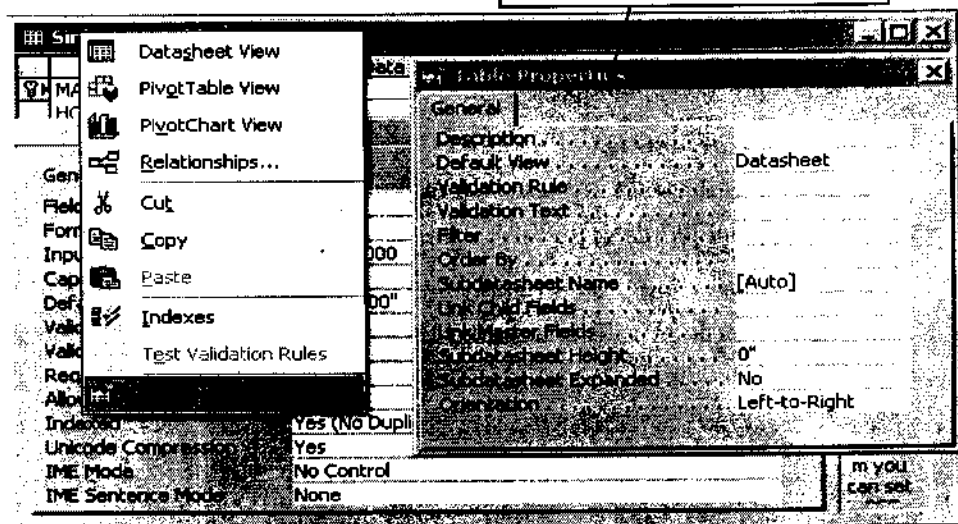
+ Limit To List : là Yes thì không nhập thêm giá trị mới, là No thì nhập thêm được giá trị mới.

2.2.2. Thuộc tính bảng

Việc kiểm soát dữ liệu liên quan đến nhiều trường khác nhau được xác định tại Table Properties.

1. Nhấn vào thanh trên cùng chứa tên bảng.
2. Mở thuộc tính của bảng bằng cách nhấn chuột phải và chọn Properties (hình 2.7).

Thuộc tính của bảng SINHVIEN



Hình 2.7. Mở và xem thuộc tính của bảng.

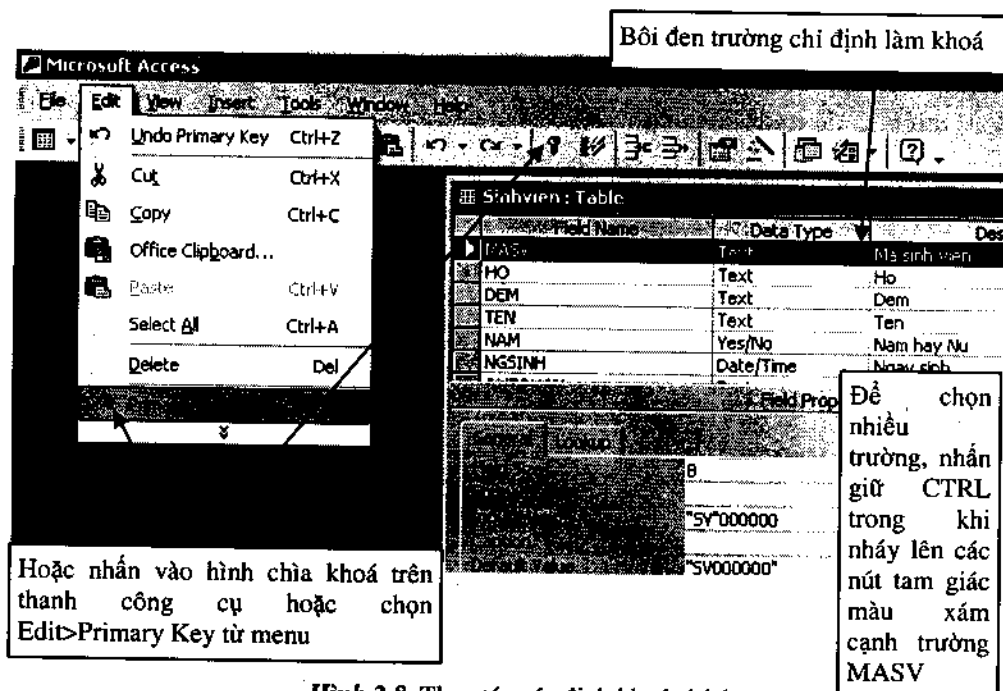
2.2.3. Xác định khoá chính

Mỗi bảng có một khoá chính, khoá này xác định duy nhất bản ghi trong bảng. Khi chỉ định một trường làm khoá chính, Access không cho phép nhập các trị giống nhau vào trường khoá. Để chỉ định một hay nhiều trường làm khoá chính, ta làm theo các bước sau :

1. Bôi đen trường chỉ định làm khoá (Để chọn nhiều trường, nhấn giữ CTRL trong khi nháy lên các nút tam giác màu xám cạnh trường MASV).

2. Nhấn vào hình chìa khoá trên thanh công cụ hoặc chọn Edit > Primary Key từ thanh Menu.

Thao tác xác định khoá chính được thể hiện trên hình 2.8.



Hình 2.8. Thao tác xác định khoá chính.

Thảo luận về khoá

Khoá là một hay nhiều trường xác định duy nhất một đối tượng trong thế giới thực được thể hiện qua một bản ghi trong bảng. Ví dụ một bản ghi SINHVIEN chứa đựng thông tin về sinh viên cụ thể. Để đảm bảo bản ghi đó là duy nhất, nó phải chứa đựng một trường là MASV (mã sinh viên), mã này được đảm bảo là duy nhất (phải sử dụng MASV làm khoá thay cho tên sinh viên vì có thể có hai sinh viên trùng tên).

Các khái niệm chính về khoá được tóm tắt như sau :

– **Khoá chính (primary key)** : khái niệm khoá và khoá chính được sử dụng lẫn lộn. Một bảng có thể có nhiều trường được chọn làm khoá, các trường này gọi là khoá dự tuyển (candidate key). Người thiết kế chọn khoá chính từ một trong các khoá dự tuyển.

– **Khoá ghép (concatenated key)** : khoá được tạo từ nhiều trường. Ví dụ kết hợp mã khoa với số khoá học tạo nên khoá.

– *Khoá ngoại* (foreign key) : trong quan hệ một nhiều, một khoá ngoại là trường (hay các trường) trong bảng bằng nhiều (“parent”) liên kết với khoá chính trong bảng bằng một (“child”).

2.2.4. Nhập dữ liệu

Chọn Datasheet View trên thanh công cụ hoặc mở bảng ở chế độ Open. Bảng hiện ra để nhập dữ liệu.

Đổi cấu trúc bảng : Việc đổi cấu trúc của bảng liên quan đến những công việc sau :

- Thêm trường : mở lại chế độ Design View, thêm trường vào cuối hay vào một vị trí bất kỳ.
- Xóa trường : chọn trường cần xóa và ấn chuột phải, chọn delete.
- Đổi thứ tự trường.
- Đổi kiểu trường.
- Đổi khoá chính.

2.3. QUAN HỆ GIỮA CÁC BẢNG

Cần xây dựng mối quan hệ giữa hai bảng LOP và SINHVIEN vì sinh viên có mã số sinh viên là duy nhất nhưng có thể có cùng mã lớp do đó quan hệ giữa bảng SINHVIEN và LOP là quan hệ một – nhiều (hình 2.9).

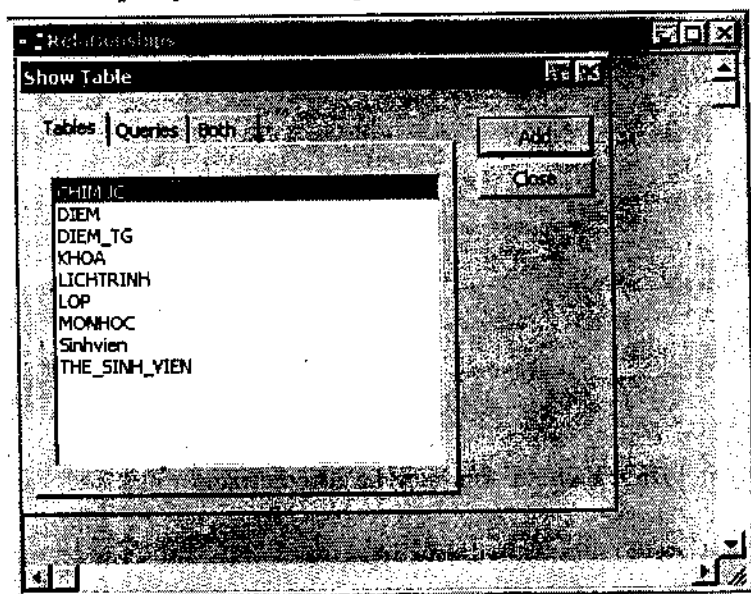
Mã lớp “001”
nằm trong nhiều
bản ghi

	MASV	HO	DEM	TEN	MALOP
▶	SV000001	Đào	Trần	Minh	001
	SV000002	Đào	Trần	Quang	001
	SV000003	Nguyễn	Anh	Tuân	001
	SV000004	Phạm	Thị Thu	Huyền	002
	SV000005	Lương	Thị Thu	Trang	002
	SV000006	Phạm	Xuân	Sang	004
	SV000007	Bùi	Văn	Quang	003
	SV000008	Hoàng	Minh	Son	003
	SV000009	Nguyễn	Xuân	Bình	004
	SV000010	Nguyễn	Tuân	Anh	001
*					

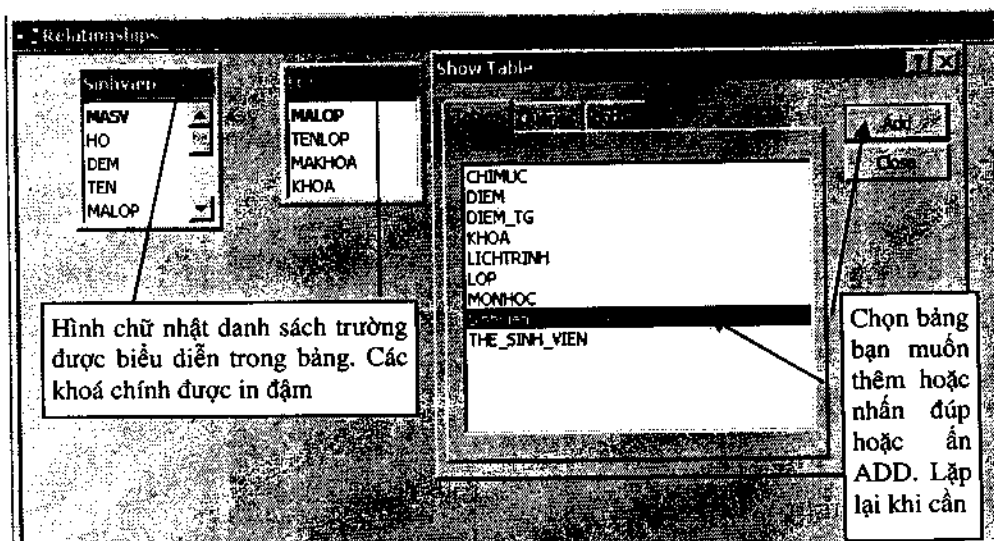
Hình 2.9. Quan hệ một – nhiều giữa MASV và MALOP.

Mở cửa sổ Relationship bằng cách chọn Tools -> Relationship | kích hoạt biểu tượng

1. Chọn kiểu đối tượng. Như chọn đối tượng là các bảng hay truy vấn hoặc cả hai (hình 2.10).
2. Chọn bảng hay truy vấn mà bạn muốn thêm (hình 2.11).
3. Nhấn đúp hay ấn ADD. Lập lại khi cần (hình 2.11).

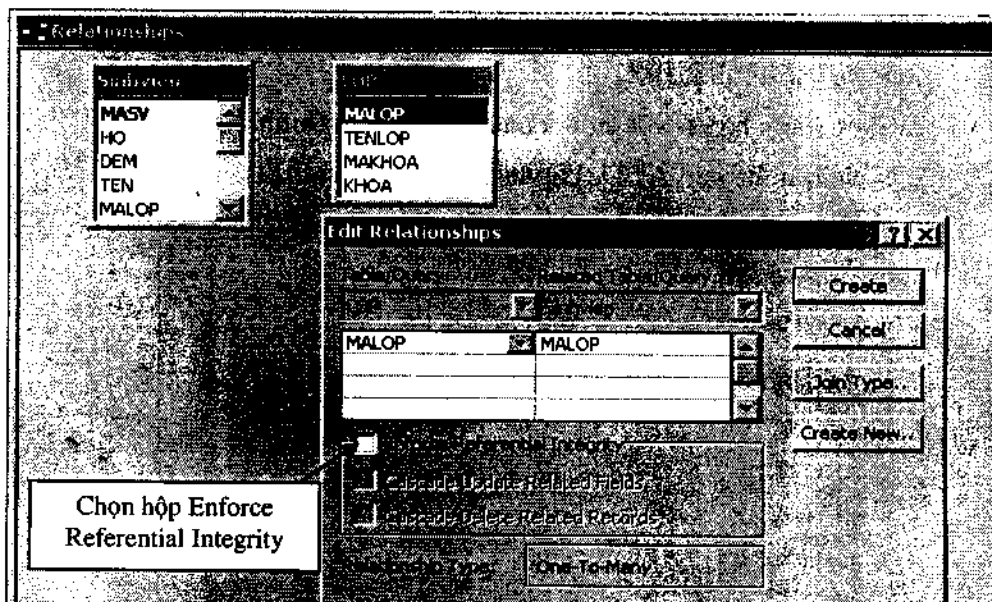


Hình 2.10. Màn hình Show Table.



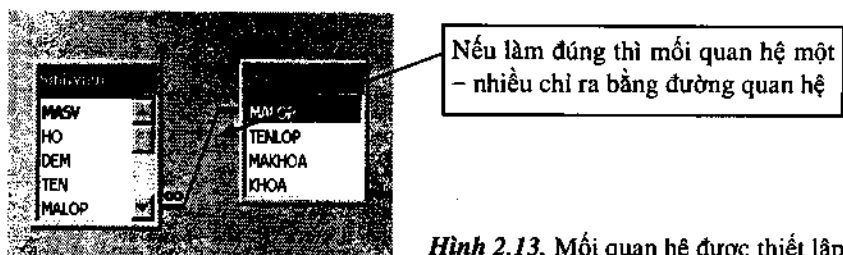
Hình 2.11. Chọn bảng hay truy vấn để tạo mối quan hệ.

2.3.1. Xác định quan hệ giữa các bảng (hình 2.12)



Hình 2.12. Xác định quan hệ giữa các bảng.

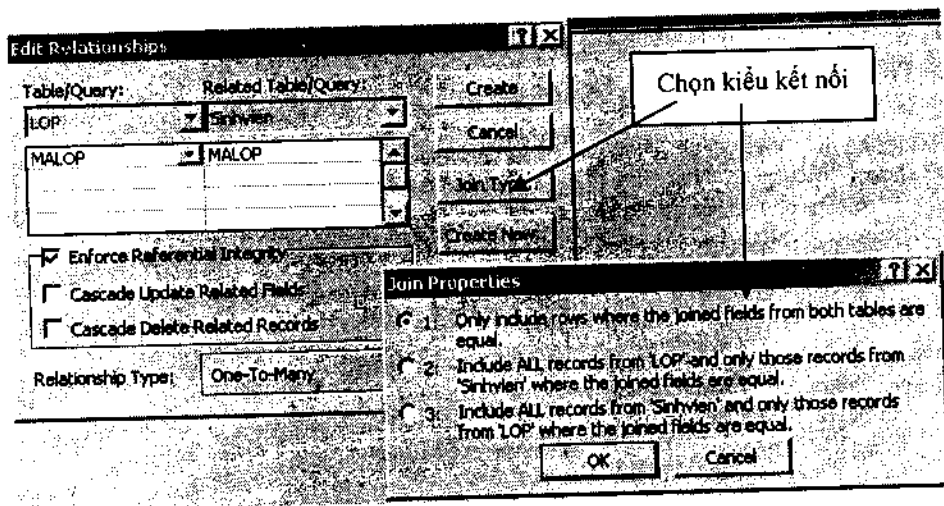
Kéo trường đặt quan hệ từ bên một sang bên nhiều (hình 2.13)



Hình 2.13. Mỗi quan hệ được thiết lập.

2.3.2. Toàn vẹn tham chiếu

- Trong cửa sổ Relationship, chọn kiểu kết nối Join Type (hình 2.14).
- Cửa sổ Join Properties xuất hiện, chọn một trong ba kiểu kết nối :
 - + Kết nối tự nhiên.
 - + Kết nối bên trái.
 - + Kết nối bên phải.



Hình 2.14. Chọn kiểu kết nối Join Type.

*** Enforce Referential Integrity**

– Không cho nhập vào bên nhiều những giá trị không tồn tại trong bên một.

– Không cho xoá giá trị bên một nếu bên nhiều còn giá trị tương ứng.

*** Cascade Delete Related Records** : những bản ghi liên quan bên một xoá thì bên nhiều xoá theo.

*** Cascade Update Related Fields** : bên một cập nhập thì bên nhiều cập nhập theo.

2.3.3. Chỉnh sửa và xoá các quan hệ (hình 2.15)

– Lý do để chỉnh sửa và xoá các quan hệ :

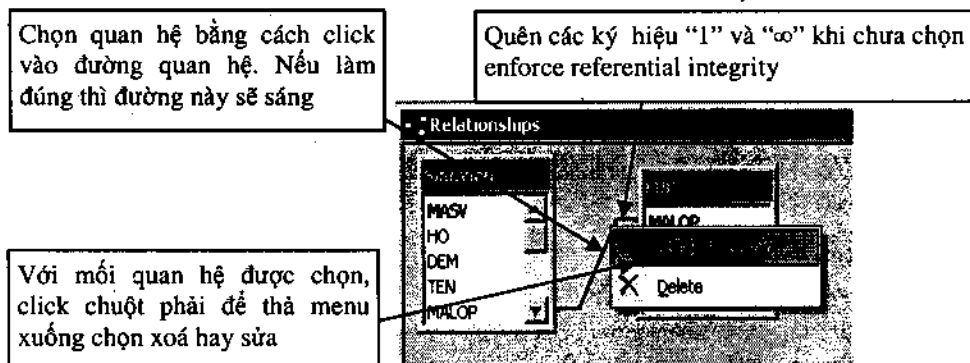
1. Khi muốn thay đổi kiểu dữ liệu của một trong các trường trong mối quan hệ. Access sẽ không cho phép làm điều này khi chưa xoá mối quan hệ (sau khi thay đổi kiểu dữ liệu cần phải tạo lại mối quan hệ).

2. Quên mô tả referential integrity – nếu 1 và ∞ không xuất hiện trên dòng quan hệ thì ta đã không chọn hộp enforce referential integrity

– Các bước chỉnh sửa hoặc xoá các quan hệ :

1. Chọn quan hệ bằng cách click vào đường quan hệ (nếu làm đúng thì đường này sẽ sáng).

2. Với mỗi quan hệ được chọn, click chuột phải để thả menu xuống, chọn xoá hay sửa.



Hình 2.15. Các thao tác chỉnh sửa hay xoá mỗi quan hệ.

2.3.4. Nhập, xuất dữ liệu

- Nhập : File → Get External Data → Import hoặc Link Tables
- Xuất : File → Save as/Export
- Lưu bảng dưới dạng HTML : File → Save as HTML.

2.4. TẠO TRUY VẤN BẰNG NGÔN NGỮ QBE

Các truy vấn (query) cho phép kết nối dữ liệu từ nhiều bảng, sắp xếp dữ liệu theo các tiêu chí khác nhau, tính toán trường mới và chỉ định điều kiện để lọc lấy những bản ghi cần thiết.

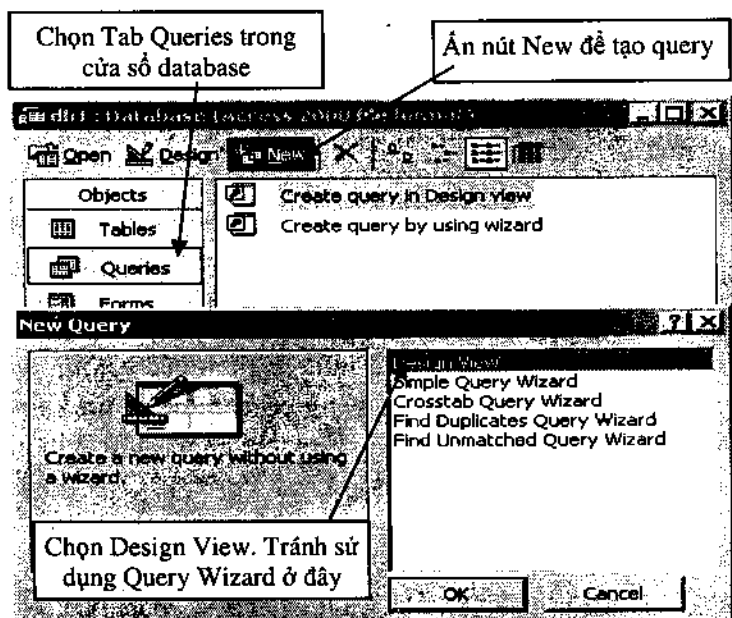
Bản thân query không chứa dữ liệu mà chỉ sắp xếp lại dữ liệu trên bảng (các bảng) mà query thao tác trên đó và không làm thay đổi các bảng đó.

Khi đã có query, ta có thể sử dụng chúng giống như bảng. Có thể coi các query là các "bảng ảo". Trong một số các hệ quản trị CSDL khác, query có tên là "view", vì chúng cho phép người sử dụng có các quan sát khác nhau trên cùng một dữ liệu.

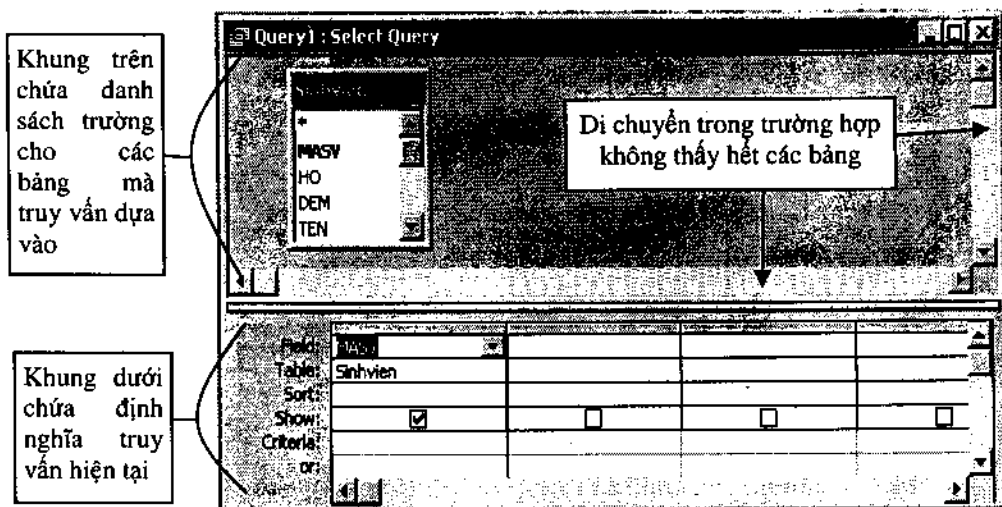
2.4.1. Tạo query (hình 2.16)

1. Sử dụng nút New trong mục Queries của cửa sổ Database để tạo một query mới. Chọn Design View.

2. Chọn bảng để đưa vào cửa sổ Query (ví dụ trên hình 2.17 chọn bảng SINHVIEN)
3. Khảo sát các thành phần của cửa sổ thiết kế query.
4. Lưu query (CTRL S).



Hình 2.16. Các thao tác tạo một query mới.



Hình 2.17. Màn hình truy vấn.

- Dòng *Field* chỉ ra tên của trường trong truy vấn, được kéo trực tiếp từ bảng xuống.
- Hàng *Table* chỉ ra tên của bảng mà trường đó thuộc vào.
- Dòng *Sort* xác định thứ tự các bản ghi được hiển thị.
- Các hộp *Show* xác định các trường trong truy vấn có được hiển thị hay không.
- Dòng *Criteria* chỉ rõ tiêu chuẩn để bao gồm hoặc loại trừ các bản ghi từ các tập kết quả.

2.4.2. Truy vấn chọn các bản ghi theo điều kiện (Select Query)

1. Phép chiếu

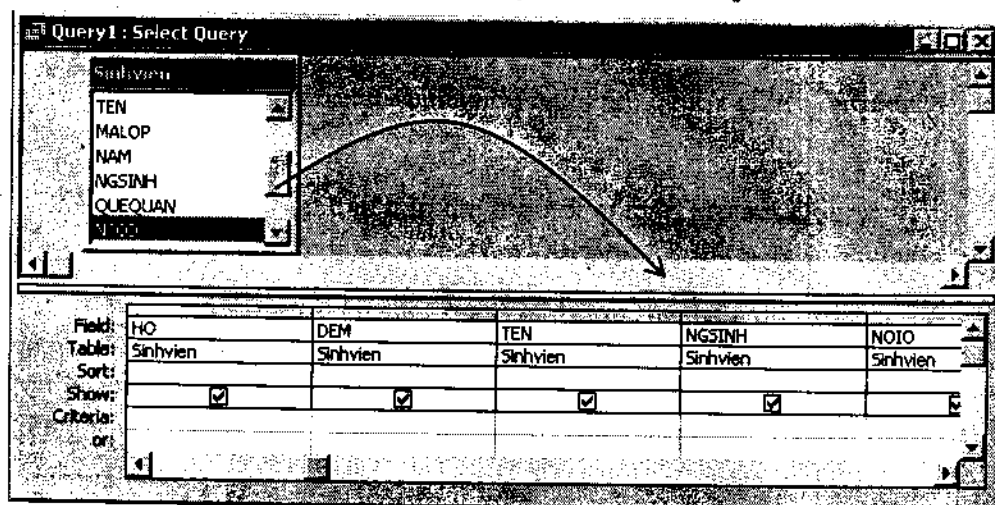
Chiếu một trường lên một query chỉ đơn thuần là đưa trường đó vào query. Khả năng tạo query dựa trên một tập con các trường của bảng (hay các bảng) rất tiện lợi khi xử lý các bảng có các thông tin không bí mật và thông tin bí mật. Trong một số trường hợp, chỉ chọn một số thông tin.

Ví dụ 2.3 :

Lấy một vài thông tin về sinh viên từ bảng SINHVIEN (hình 2.18)

1. Kéo các trường MASV, HO, DEM, TEN, NGSINH, NOIO lên lưới tạo query (nửa dưới của cửa sổ query).

2. Nhấn nút  trên thanh công cụ để xem kết quả



Hình 2.18. Tạo truy vấn chọn lấy một số trường của bảng SINHVIEN.

2. Sắp xếp

Sắp xếp trên query không ảnh hưởng đến trật tự vật lý các bản ghi trong bảng liên quan (tức là không sắp xếp bảng). Vì vậy có thể tạo các query sắp xếp theo các cột khác nhau cho cùng một bảng.

Ví dụ 2.4 :

Sắp xếp MASV theo chiều tăng (hình 2.19).

THU1 : Select Query

Sinhvien

TEN
MALOP
NAM
NGSINH
QUEQUAN
NOIO

Chọn sắp xếp theo chiều tăng cho trường MASV

Field:	MASV	HO
Table:	Sinhvien	Sinhvien
Sort:	Ascending	
Show:	☒	☒
Criteria:		

Hình 2.19. Sắp xếp MASV theo chiều tăng.

3. Phép chọn

Các bản ghi được chọn bằng cách chỉ định điều kiện mỗi bản ghi phải thoả mãn để đưa vào bộ kết quả. Trong lưới QBE (query by example), ta nhập điều kiện vào hàng criteria.

Ví dụ 2.5 :

Chọn các bản ghi có HO = "Nguyễn". Nhập "Nguyễn" vào hàng criteria (hình 2.20). Ta có thể nhập = "Nguyễn" nhưng toán tử = là ngầm định nên không cần nhập.

Field:	MASV	HO
Table:	Sinhvien	Sinhvien
Sort:	Ascending	
Show:	☒	☒
Criteria:		"Nguyễn"

Hình 2.20. Ví dụ về phép chọn.

4. Điều kiện chọn phức hợp

Có thể tạo các điều kiện chọn phức hợp sử dụng phép AND, OR hay NOT.

- Đưa thêm trường MALOP vào truy vấn.
- Tạo truy vấn xem tất cả các bản ghi có mã lớp là 001 hoặc 003 (hình 2.21).

Query: T1001: Select Query

Criteria:

Field	Criteria
NOI0	Sinhvien
MALOP	Sinhvien
	"001"
	"003"

Results:

ID	Họ và Tên	Ngày Sinh	Ngày Nhập Học	Mã Lớp
SV000001	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000002	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000003	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000004	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000005	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000006	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000007	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000008	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000009	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001
SV000010	Nguyễn Văn Tuấn	11/11/1981	11/11/1981	001

Các kết quả. Chỉ có các bản ghi thỏa mãn Criteria mới

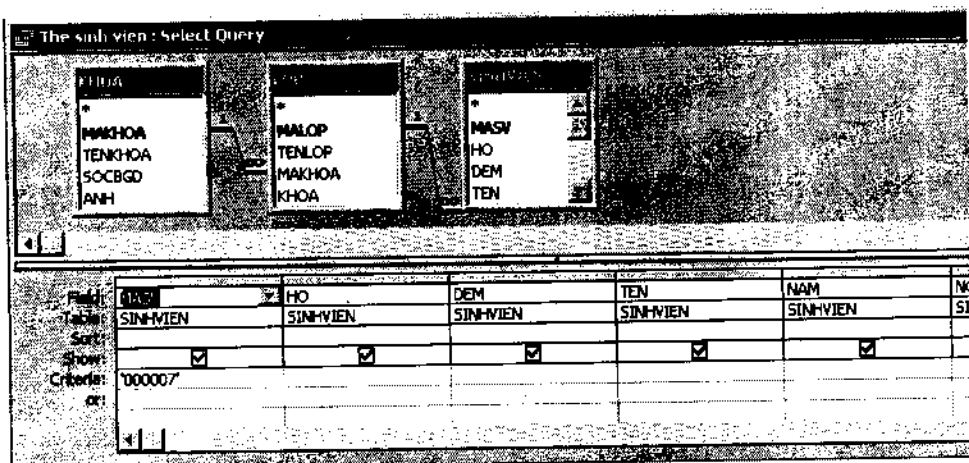
Hình 2.21. Điều kiện chọn phức hợp và các kết quả thu được.

5. Phép nối

Tạo một query dựa trên nhiều bảng để có nhiều thông tin trên nhiều bảng (hình 2.22).

Ví dụ 2.6 :

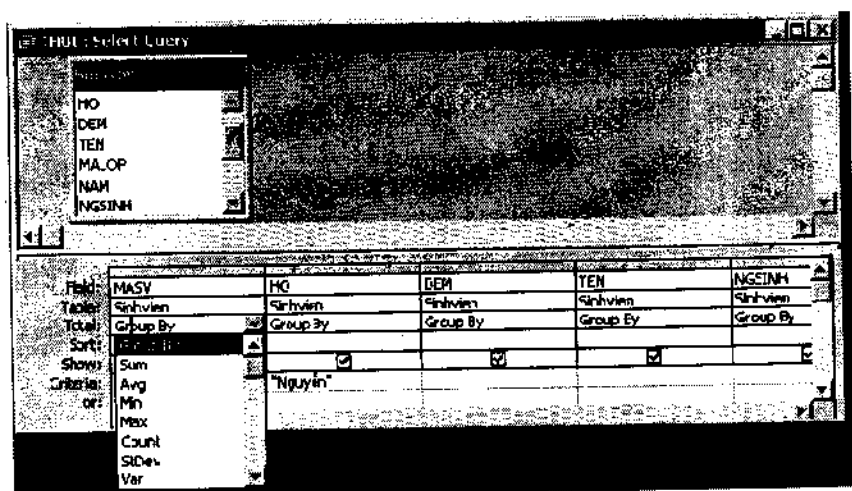
- Tạo một query mới tên thesinhvien dựa trên ba bảng SINHVIEN, LOP, KHOA.
- Access tự tìm thông tin trên bảng “một” dựa trên khoá ngoại trên bảng “nhiều”.



Hình 2.22. Tạo một query dựa trên nhiều bảng.

6. Tạo truy vấn tính toán (hình 2.23)

- Tạo một truy vấn mới và nguồn dữ liệu.
- Chuyển từ truy vấn chọn thông thường sang truy vấn tính toán bằng cách chọn biểu tượng totals trên thanh công cụ hoặc View → totals



Hình 2.23. Màn hình thiết kế truy vấn tính toán.

Trên dòng total có thể chọn các giá trị :

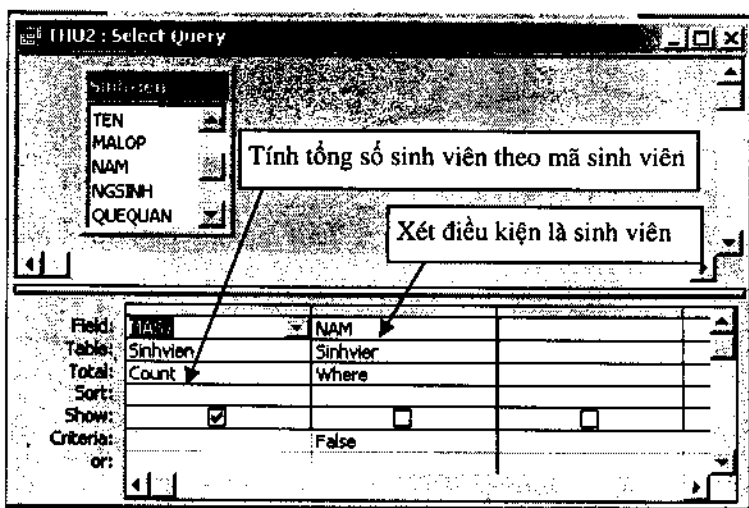
- + *Group by* : dùng để chỉ ra trường nào là trường phân nhóm.
- + *Sum* : Tính tổng.
- + *Avg* : Tính trung bình cộng.

- + *Min* : Tìm giá trị nhỏ nhất.
- + *Max* : Tìm giá trị lớn nhất.
- + *Count* : Đếm.
- + *StDev* : Tìm độ lệch chuẩn.
- + *Var* : Tìm phương sai.
- + *First* : Tìm bản ghi đầu tiên.
- + *Last* : Tìm bản ghi cuối cùng.
- + *Expression* : Một biểu thức trong tính toán đi kèm với một tính toán bằng hàm thư viện khác.
- + *Where* : Điều kiện lọc các bản ghi tham gia vào tính toán.

Ví dụ 2.7 :

Ví dụ về các truy vấn tính toán : Tìm tổng số sinh viên nữ trong trường (hình 2.24).

1. Xét trường NAM với điều kiện là sinh viên nữ.
2. Xét trường MASV để tính tổng số sinh viên theo mã sinh viên.



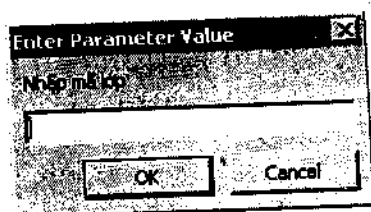
Hình 2.24. Ví dụ về tìm tổng số sinh viên nữ trong trường.

7. Truy vấn có tham số

Khi thực hiện truy vấn, màn hình nhập giá trị tham số sẽ xuất hiện (hình 2.25) :

Truy vấn sẽ thực hiện với giá trị tham số được nhập từ bàn phím cho kết quả hoặc điều kiện.

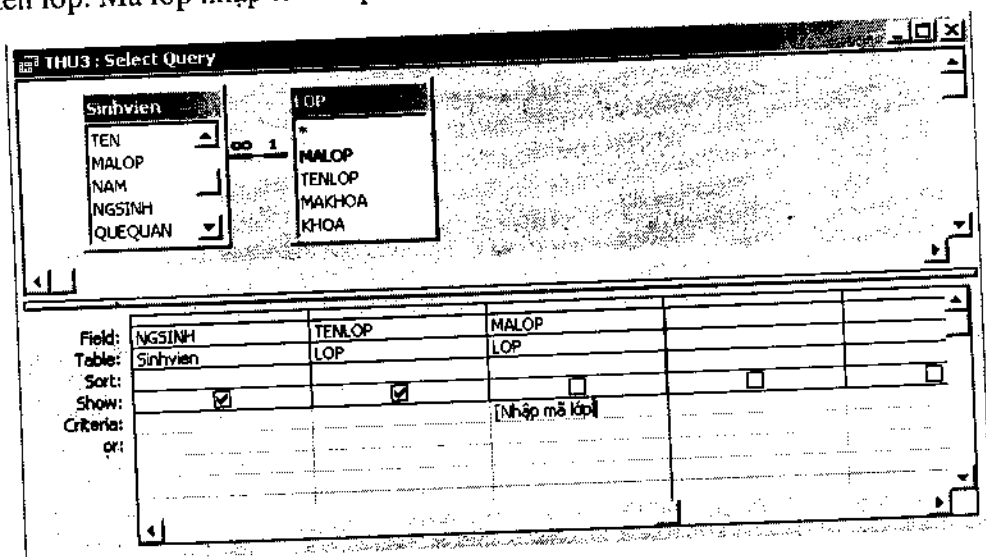
Để biến một đối tượng nào thành truy vấn, chỉ cần bao nó bởi cặp ngoặc vuông.



Hình.2.25. Màn hình nhập giá trị tham số.

Ví dụ 2.8 : (hình 2.26)

Đưa ra danh sách sinh viên một lớp, gồm mã sinh viên, họ tên, ngày sinh, tên lớp. Mã lớp nhập từ bàn phím.



Hình 2.26. Ví dụ về truy vấn có tham số.

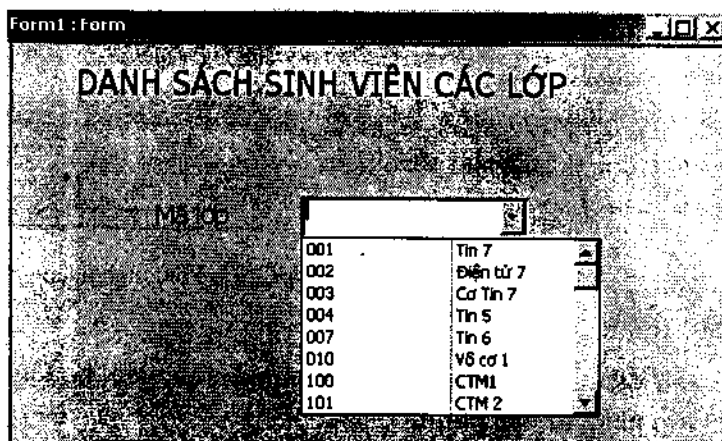
8. Truy vấn với điều kiện đặc biệt

Trong điều kiện của truy vấn có thể xuất hiện những giá trị của các điều kiện xuất hiện trong form nào đó. Thay cho việc nhập thông tin qua tham số, có thể chọn thông tin đó trên một hộp combo của form.

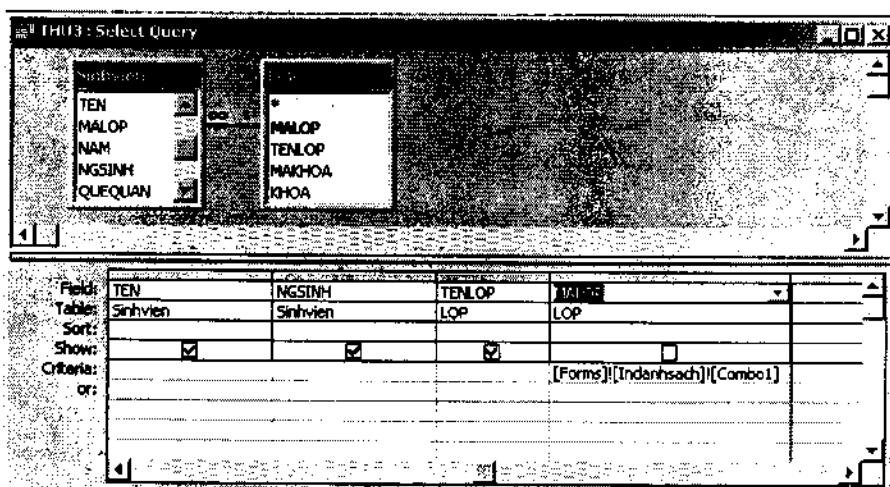
Ví dụ 2.9 :

Đưa ra danh sách sinh viên một lớp, gồm mã sinh viên, họ tên, ngày sinh, tên lớp. Mã lớp được chọn trên form (hình 2.27).

Truy vấn được thiết kế (hình 2.28)



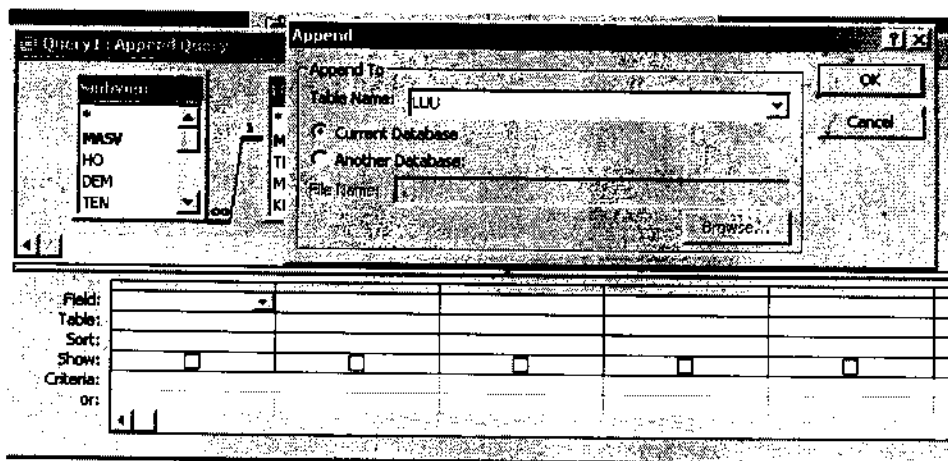
Hình 2.27. Form để lựa chọn mã lớp.



Hình 2.28. Truy vấn được thiết kế theo ví dụ.

2.4.3. Truy vấn bổ sung (Append Query) (hình 2.29)

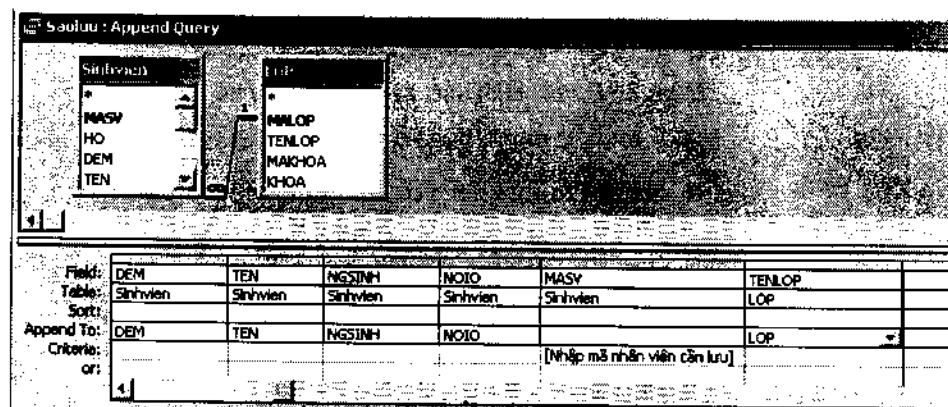
- Khi cần bổ sung dữ liệu vào CSDL, có thể thực hiện theo các cách sau :
 1. Bổ sung vào bản ghi đơn lẻ : Mở bảng hoặc form, thêm một số dòng.
 2. Bổ sung hàng loạt bản ghi theo một điều kiện nào đó (chẳng hạn sao lưu hay nhập dữ liệu từ bảng trung gian) : Lập truy vấn.
- Để tạo truy vấn bổ sung mới bằng QBE phải tuân theo trình tự sau :
 1. Tạo truy vấn mới : tại hộp thoại Show Table chọn bảng cho dữ liệu.
 2. Trên màn hình thiết kế truy vấn chọn mới, chuyển sang truy vấn bổ sung bằng cách chọn menu Query → Append Query.



Hình 2.29. Màn hình thiết kế truy vấn bổ sung.

Ví dụ 2.10 :

Đưa vào bảng LUU thông tin về sinh viên với mã sinh viên nhập từ bàn phím (hình 2.30).



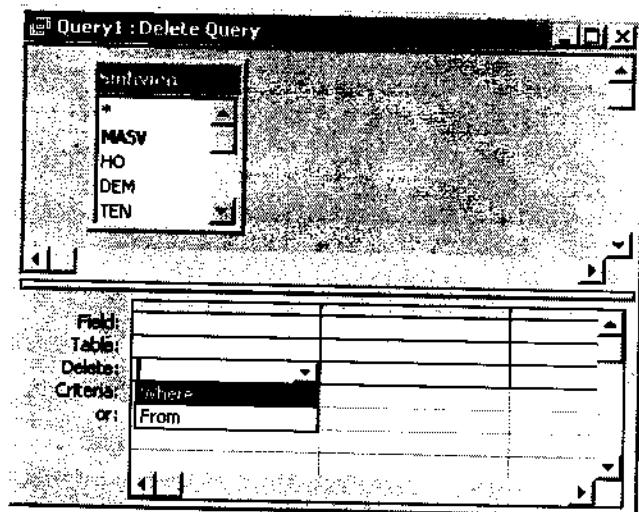
Hình 2.30. Ví dụ về truy vấn bổ sung.

2.4.4. Truy vấn loại bỏ (Delete Query) (hình 2.31)

Nếu loại bỏ một số bản ghi đơn lẻ, có thể loại bỏ trực tiếp trên bảng hoặc form. Truy vấn loại bỏ được sử dụng khi loại bỏ các bản ghi thoả mãn điều kiện nào đó, có thể được thiết kế bằng QBE hoặc bằng SQL để chạy tự động trong chương trình.

Các bước thực hiện :

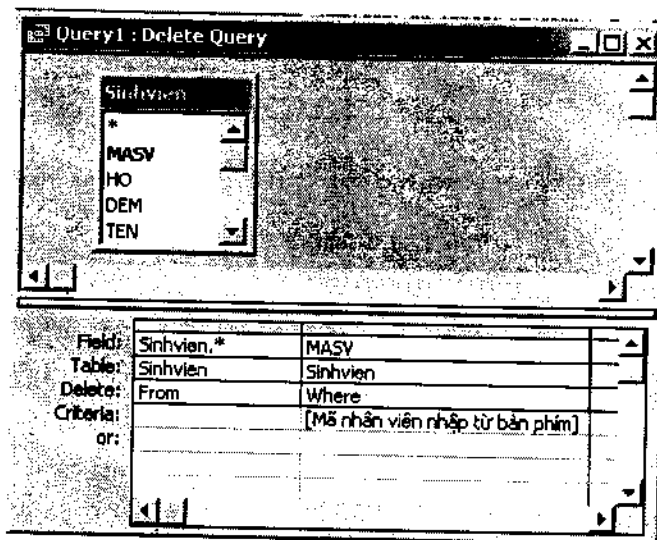
1. Tạo truy vấn mới → Trên màn hình Show Table chọn các bảng liên quan đến truy vấn. Đó là bảng cần loại bỏ và các bảng liên quan đến điều kiện.
2. Trên màn hình thiết kế truy vấn chọn ngầm định → menu Query → Delete Query.



Hình 2.31. Màn hình thiết kế truy vấn loại bỏ.

Ví dụ 2.11 :

Xoá bỏ thông tin về một sinh viên với mã sinh viên nhập từ bàn phím (hình 2.32).



Hình 2.32. Ví dụ về truy vấn loại bỏ.

2.4.5. Truy vấn thay đổi (Update Query) (hình 2.33)

Việc thay đổi các giá trị đơn lẻ được thực hiện trực tiếp trên bảng hoặc form.

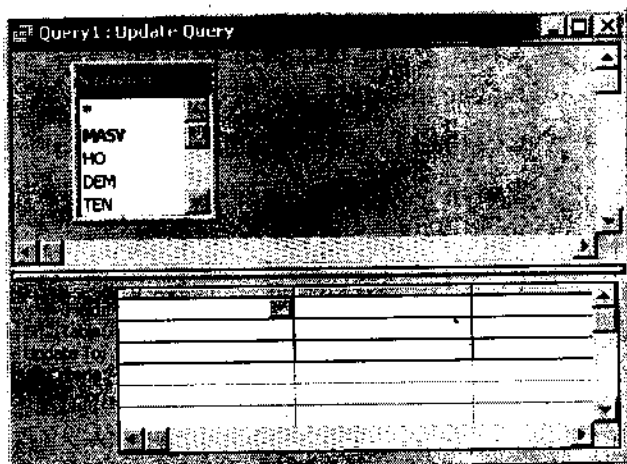
Khi thay đổi nhiều bản ghi thoả mãn điều kiện nào đó cần tạo truy vấn thay đổi.

Các bước thiết kế truy vấn thay đổi :

1. Tạo truy vấn mới.
2. Chọn các bảng liên quan : Một bảng cần thay đổi và các bảng xuất hiện trong điều kiện.

Trước khi thực hiện truy vấn thay đổi cần xem trước những bản ghi cần thay đổi xem sự thay đổi đó có phù hợp với yêu cầu không.

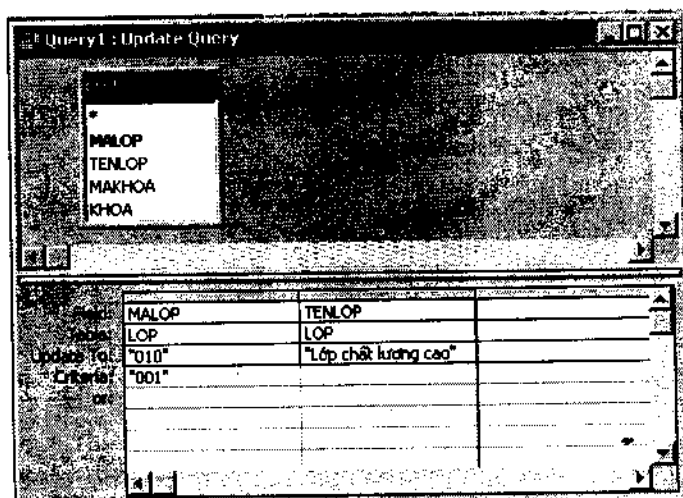
Chú ý đến việc xác lập tùy chọn Cascade Update để tránh những thao tác thay đổi không cần thiết.



Hình 2.33. Màn hình thiết kế truy vấn thay đổi.

Ví dụ 2.12 :

Đổi mã lớp 001 thành 010 và thay tên lớp thành tên "Lớp chất lượng cao" (hình 2.34).



Hình 2.34. Ví dụ về truy vấn thay đổi.

2.4.6. Truy vấn Crosstab

– Truy vấn Crosstab được dùng trong các biểu mẫu thống kê. Đặc trưng quan trọng của truy vấn này là có thể chuyển dữ liệu trong bảng thành tiêu đề cột.

– Để tạo truy vấn Crosstab có thể dùng Wizard hoặc tự thiết kế.

* Nếu dùng Wizard, phải thực hiện các bước sau :

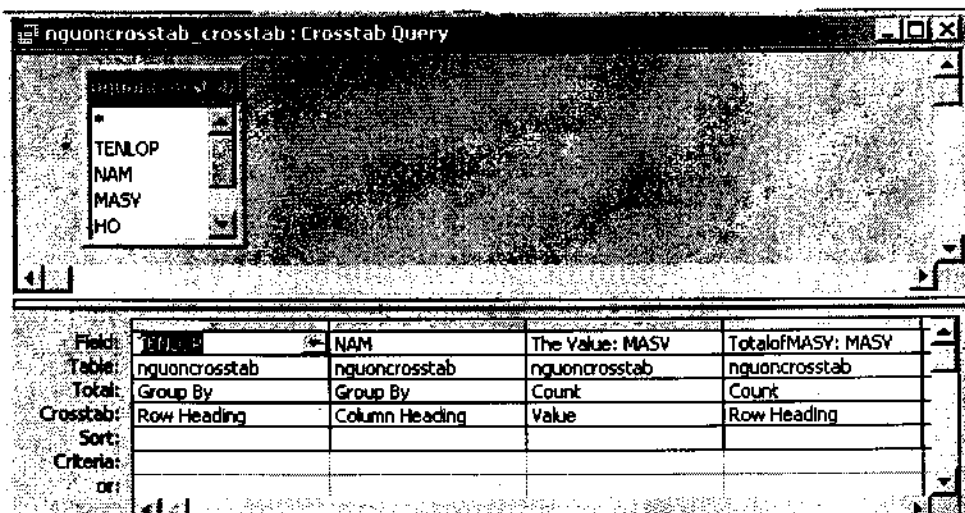
1. Chọn bảng hoặc truy vấn nguồn dữ liệu.
2. Chọn các trường để nhóm theo hàng.
3. Chọn các trường để nhóm theo cột.
4. Chọn giá trị để lưu ở giao của một hàng và một cột, ở đó có thể dùng các hàm thư viện để tính toán.

* Nếu tư thiết kế phải theo các bước sau :

1. Tạo truy vấn mới.
2. Chọn bảng hoặc truy vấn nguồn.
3. Chuyển sang truy vấn Crosstab (Menu Query).

Ví dụ 2.13 :

Một truy vấn để tính tổng nam nữ trong từng lớp, chuẩn bị nguồn dữ liệu cho báo cáo về tỷ lệ nam nữ trong từng lớp. Màn hình thiết kế truy vấn cho kết quả trên hình 2.35.



Hình 2.35. Ví dụ về truy vấn Crosstab.

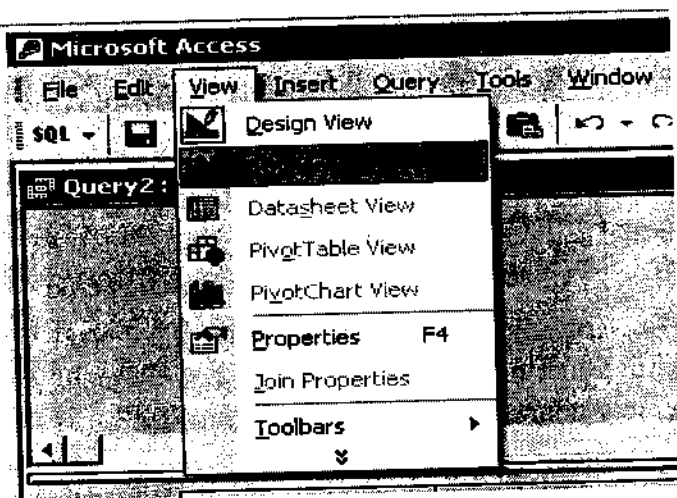
2.5. TẠO TRUY VẤN BẰNG NGÔN NGỮ SQL

Query – By – Example (QBE) và Structured Query Language (SQL) là hai ngôn ngữ chuẩn công nghiệp nổi tiếng để vấn tin trên CSDL quan hệ. Ưu điểm của QBE là dễ sử dụng nhờ vào giao diện đồ họa còn ưu điểm của SQL là tính phổ biến của nó trong thế giới CSDL quan hệ.

QBE và SQL hoàn toàn có thể trao đổi qua lại chỉ trừ một số trường hợp ngoại lệ. Vì vậy, nắm vững được các phép toán trên một ngôn ngữ (chiều, chọn, sắp xếp, kết nối và trường tính toán) thì trên ngôn ngữ còn lại cũng tương tự. Trong thực tế Access cho phép chuyển một query qua lại giữa QBE và câu lệnh SQL.

– Cách tạo một query sử dụng SQL.

1. Tạo một query mới nhưng đóng cửa sổ “show table” mà không chọn bảng nào.
2. Chọn View > SQL để chuyển sang SQL editor.



Hình 2.36. Cách tạo một câu lệnh SQL.

– Một câu lệnh SQL phổ biến có dạng :

SELECT MASV, Ho, Dem, Ten **FROM** SINHVIEN **WHERE** Ho = "Nguyễn"

Bao gồm bốn phần :

1. **SELECT** <field1, field2,...,fieldn> chỉ ra trường nào được chọn (phép chiếu)

2. ...**FROM** <table>... Chỉ ra bảng (các bảng) là nguồn của query.

3. **WHERE** <đk1> AND/OR <đk2>, ..., AND/OR <đkn> Chỉ ra điều kiện hay các điều kiện bản ghi phải thoả mãn

4. **Dấu chấm phẩy (;)** Tất cả các câu lệnh SQL được kết thúc bằng dấu chấm phẩy. Nếu quên, Access sẽ tự điền vào.

– *Mệnh lệnh WHERE phức hợp*

Các lệnh AND, OR và NOT được sử dụng trong mệnh đề WHERE để tạo các điều kiện phức hợp.

Ví dụ 2.14 :

Tìm họ tên, Ngày sinh của các sinh viên nữ, tên bắt đầu bằng chữ cái “A”, “D”, “C”

```
SELECT DISTINCTROW ho, dem, ten, ngsinh  
FROM sinhvien
```

```
WHERE Not Nam AND Ten like "[A,D,C]*";
```

Chú ý : các trường văn bản cần để trong dấu nháy kép.

Ngôn ngữ thao tác dữ liệu

1. Bỏ sung

```
INSERT INTO <Tên bảng> [(<danh sách tên cột>)]  
VALUES (<Danh sách giá trị>)
```

Ví dụ :

```
INSERT INTO LOP  
VALUES ("220","TAI NANG","F20","K44")
```

2. Loại bỏ

```
DELETE [(<danh sách các trường>)] FROM <Tên bảng>  
WHERE <điều kiện>
```

2.6. CÂU HỎI ÔN TẬP – BÀI TẬP

Sử dụng tệp Quanlysinhvien.mdb vừa tạo trong chương 1

1. *Tạo bảng*

– Tạo các bảng DIEM, LOP, KHOA theo như các hình 1, 2, 3 :

DIEM : Table

	Field Name	Data Type	Description
🔍	MASV	Text	Ma sinh vien
🔍	MAMH	Text	Ma mon hoc
	DIEM	Number	Diem thi
🔍	LAN	Text	Lan thi

Field Properties

General | Lookup

Field Size: 8

Format:

Input Mask: "SV"000000

Caption:

Default Value: "SV000000"

Validation Rule:

Validation Text:

Required: Yes

Allow Zero Length: No

Indexed: No

Unicode Compression: Yes

IME Mode: No Control

IME Sentence Mode: None

Hình 1

LOP : Table

	Field Name	Data Type	Description
🔍	MALOP	Text	Ma lop
	TENLOP	Text	Ten lop
	MAKHOA	Text	Ma khoa
	KHOA	Text	Khoa hoc

Field Properties

General | Lookup

Field Size: 3

Format:

Input Mask: 000

Caption:

Default Value: "000"

Validation Rule:

Validation Text:

Required: Yes

Allow Zero Length: No

Indexed: Yes (No Duplicates)

Unicode Compression: Yes

IME Mode: No Control

IME Sentence Mode: None

Hình 2

KHOA : Table

Field Name	Data Type	Description
MAKHOA	Text	Ma khoa
TENKHOA	Text	Ten khoa
SOCBGD	Number	So can bo g
ANH	OLE Object	Logo Khoa

Field Properties

General | Lookup

Field Size: 3

Format:

Input Mask: "F"00

Caption:

Default Value: "F00"

Validation Rule:

Validation Text:

Required: Yes

Allow Zero Length: No

Indexed: Yes (No Duplicates)

Unicode Compression: Yes

IME Mode: No Control

IME Sentence Mode: None

Hình 3

- Thực hành gán khoá chính, sử dụng autonumber
- Dùng các thuộc tính khác như caption, lookup

2. Tạo QBE

- Tạo một query sắp xếp bảng SINHVIEN đã tạo trong nội dung phần lý thuyết theo MASV như hình 4 :

THUAT : Select Query

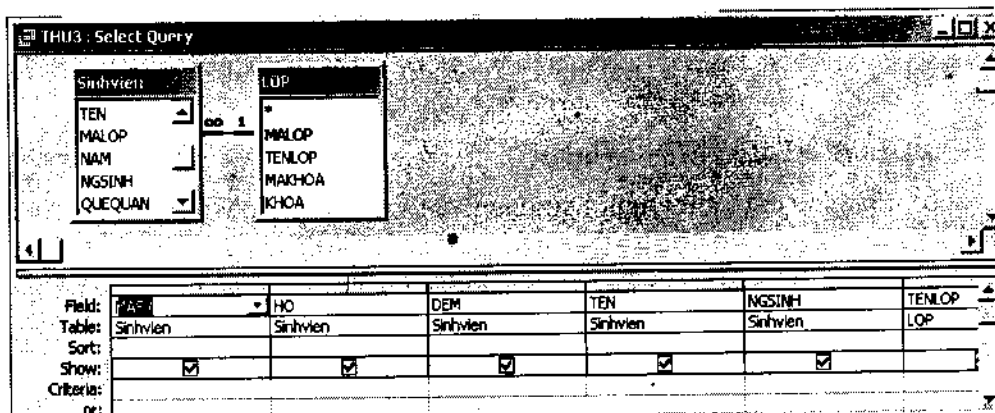
Sinhvien

HO
DEM
TEN
MALOP
NAM
NGSINH

Field:	MASV	HO	DEM	TEN	NGSINH
Table:	Sinhvien	Sinhvien	Sinhvien	Sinhvien	Sinhvier
Total:	Group By	Group By	Group By	Group By	Group B
Sort:	Ascending				
Show:	☒	☒	☒	☒	
Criteria:					
or:					

Hình 4

– Tạo một query kết nối bảng SINHVIEN và LOP. Khi nhập một MASV, mọi thông tin liên quan như tên sinh viên, ngày sinh, tên lớp, mã lớp .. sẽ tự xuất hiện. Query có dạng như hình 5 :



Hình 5

– Nhập một số dữ liệu vào bảng SINHVIEN và bảng LOP

(Yêu cầu nhập một số dữ liệu vào bảng nhằm mục đích để cho người sử dụng thấy rõ được sự phức tạp của công việc nhập dữ liệu trực tiếp. Công việc này sẽ đơn giản khi sử dụng form để nhập).

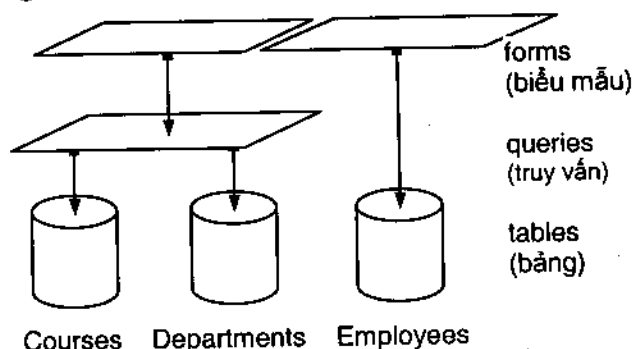
Chương III

THIẾT KẾ BIỂU MẪU VÀ BÁO CÁO

3.1. GIỚI THIỆU BIỂU MẪU (Form)

Form là giao diện giữa người sử dụng và dữ liệu ; cho phép người sử dụng viết chương trình điều khiển cách hiển thị dữ liệu trong bảng/query, đồng thời kiểm soát người sử dụng trong các thao tác nhập và hiệu chỉnh dữ liệu.

Cũng giống như query, form không chứa dữ liệu mà chỉ là “cửa sổ” thông qua đó người sử dụng nhìn thấy dữ liệu trên các table hay query. quan hệ giữa form, table và query được minh hoạ trong hình 3.1.



Hình 3.1. Quan hệ giữa Form, Table và Query.

Form thường dùng trong các trường hợp sau :

- Thiết kế màn hình nhập dữ liệu.
- Thiết kế menu.
- Thiết kế các màn hình tra cứu thông tin.
- Tạo các màn hình giới thiệu, trợ giúp.

Mọi đối tượng xuất hiện trên form được gọi là điều khiển. Điều khiển được chia thành 3 loại chính :

- Điều khiển bị buộc : Chỉ xuất hiện trên form có nguồn dữ liệu.
- Điều khiển không bị buộc : Không gắn với nguồn dữ liệu.
- Điều khiển tính toán được : Là điều khiển không bị buộc.

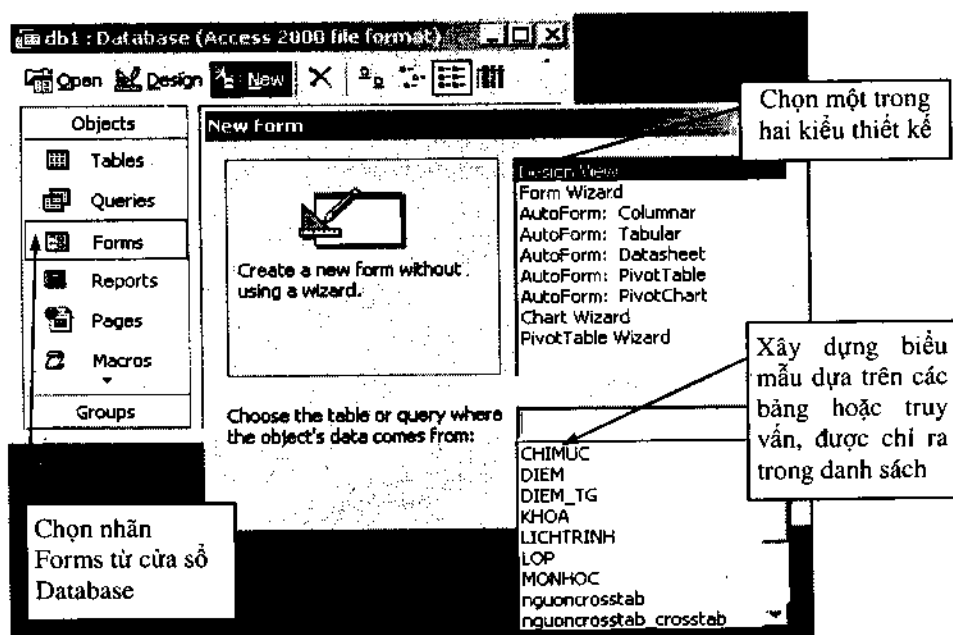
Các bước tạo form mới (hình 3.2) :

1. Để thiết kế một form mới, từ cửa sổ Database chọn Forms.

2. Chọn New để mở hộp thoại New Form. Trong hộp thoại New Form :

+ Chọn một trong hai kiểu thiết kế : Design View hay Form Wizard.

+ Chọn các bảng hay truy vấn mà biểu mẫu dựa trên.



Hình 3.2. Trình tự tạo Form mới.

Trên hộp thoại này có thể chọn một trong hai cách :

– Dùng Wizard : chọn 1 trong 6 lựa chọn cuối trong đó :

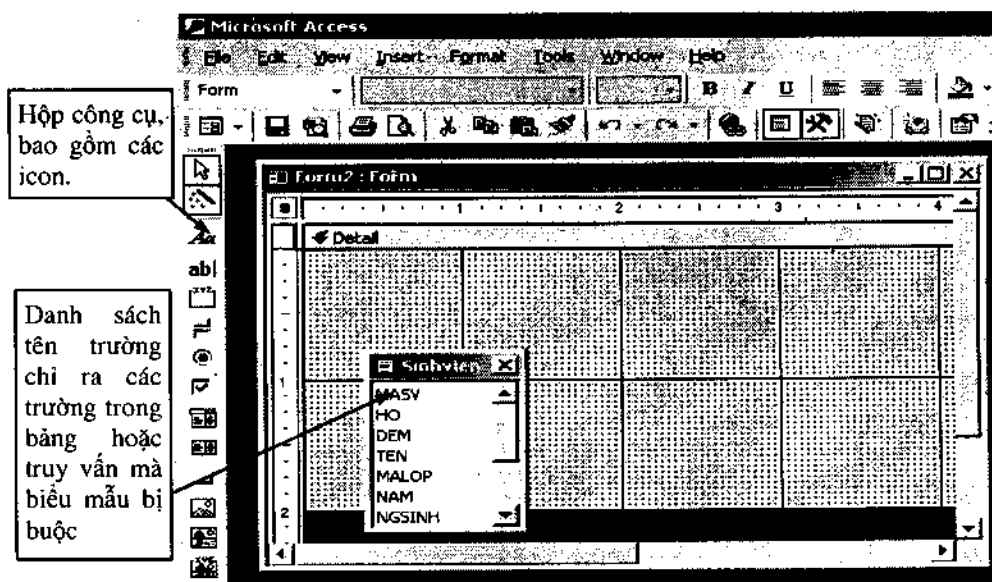
+ Form wizard : thiết kế form đơn một cách tự động. Người sử dụng có thể quyết định một số đặc trưng của form : chọn bảng, chọn trường, chọn cách hiển thị trường, hình thức trình bày form...

+ Auto form : người sử dụng chỉ được chọn bảng hay truy vấn nguồn. Access sẽ tự động thực hiện mọi thao tác tiếp theo để cuối cùng tạo được form dạng Datasheet (bảng), Columnar (phiếu nhập dữ liệu) hay Tabular (Bảng không kẻ nhưng có trang trí như form dạng Columnar).

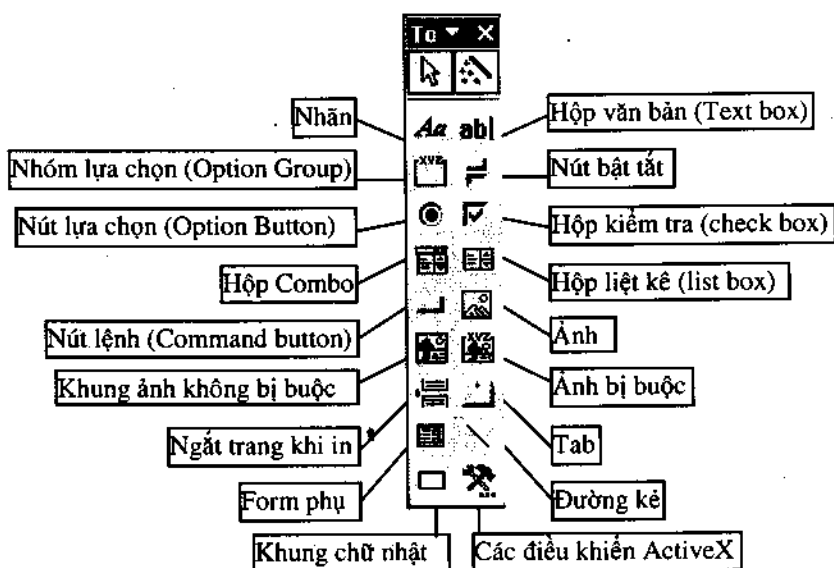
+ Chart Wizard : tự động tạo biểu đồ so sánh dữ liệu trong CSDL. Được phép quyết định một số tham số : thông tin ở trục tung, trục hoành, hình thức của biểu đồ.

+ Pivot Table Wizard : Wizard cho phép đưa bảng tính Excel, qua đó có thể tính toán ngay khi làm việc trên form.

– *Không dùng Wizard* : Chọn mục Design View. Màn hình thiết kế form có dạng như hình 3.3, trong đó xuất hiện hai công cụ luôn được dùng trong thiết kế form là toolbox (hình 3.4) và fieldlist. Ví dụ từ cửa sổ New Form, chọn Design View sau đó chọn bảng SINHVIEN.



Hình 3.3. Màn hình thiết kế Form.



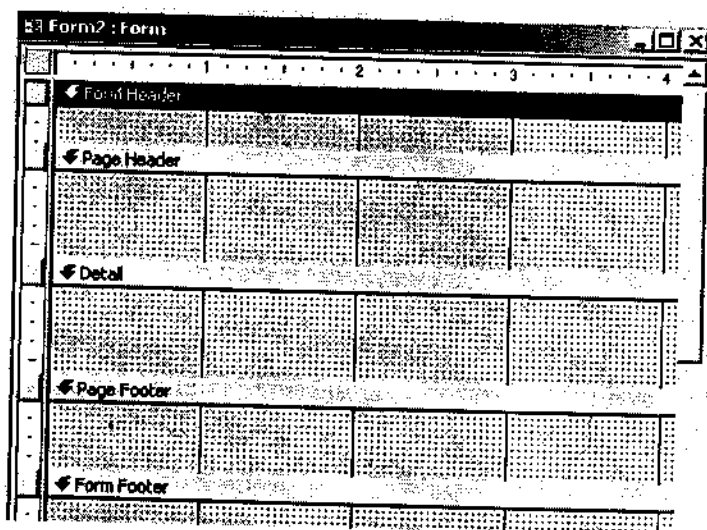
Hình 3.4. Thanh công cụ thiết kế Form.

3. Lưu form với một tên thể hiện ý nghĩa của form.

3.2. THIẾT KẾ CÁC ĐIỀU KHIỂN TRÊN FORM

3.2.1. Vị trí của các điều khiển

Khi thiết kế các điều khiển cần quan tâm đến vị trí của chúng trên form. Màn hình thiết kế form có dạng như hình 3.5.



Hình 3.5. Màn hình thiết kế Form.

Form Header/Footer : header chứa tiêu đề chính, footer chứa nút lệnh.

Page Header/Footer : header chứa tiêu đề của trang.

Detail : chứa nội dung biểu mẫu.

3.2.2. Căn chỉnh các điều khiển

– Sao chép, cắt dán, xóa các điều khiển (tuy nhiên chỉ với một số loại nào đó).

– Đánh dấu các điều khiển

+ Đánh dấu theo cột

+ Đánh dấu theo hàng

+ Đánh dấu theo khung

+ Đánh dấu các điều khiển rời rạc

– Căn chỉnh các điều khiển :

+ *Giống thẳng* : Từ menu Format chọn Align (có 1 trong 4 lựa chọn : Top, Bottom, Left, Right)

+ *Kích cỡ* : Từ menu Format chọn Size

+ **Cách đều** : Từ menu Format chọn :

Vertical Spacing : khoảng cách theo chiều dọc

Horizontal Spacing : Khoảng cách theo chiều ngang

+ **Nhân bội** : Từ menu Edit chọn Duplicate.

3.2.3. Điều khiển hộp văn bản (Text box)

– Hộp văn bản là điều khiển quan trọng nhất trên form.

– Đây là loại điều khiển có thể thuộc cả 3 loại : bị buộc, không bị buộc và tính toán được.

+ Điều khiển không bị buộc : là điều khiển nằm trong form không có nguồn dữ liệu. Như vậy nó dùng để nhập dữ liệu vào một biến nhớ sử dụng trong tính toán hoặc để làm trung gian khi nhập dữ liệu vào bảng. Để tạo điều khiển dạng này kéo từ Toolbox vào.

+ Điều khiển bị buộc : nếu điều khiển loại này thì form phải có nguồn dữ liệu. Hiện Fieldlist để hiển thị danh sách các trường. Kéo trường từ Fieldlist vào form, ta có Textbox bị buộc với trường được chọn.

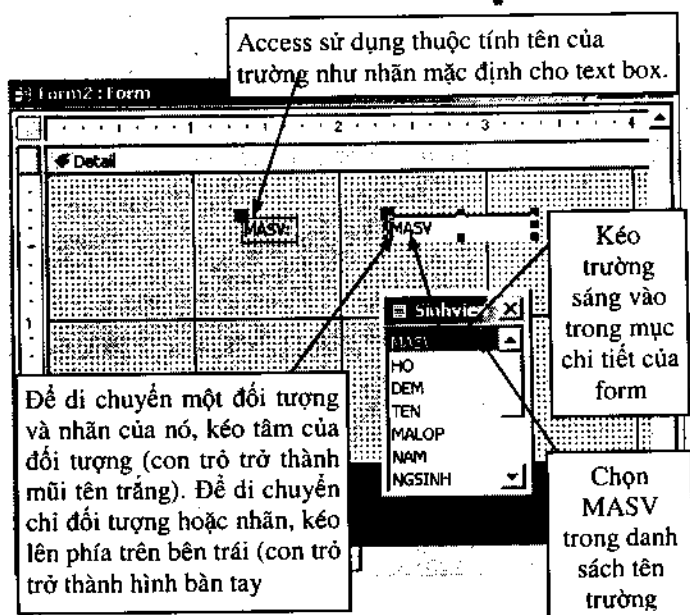
+ Điều khiển tính toán được : Nếu là điều khiển tính toán thì trước hết tạo như một điều khiển không bị buộc sau đó nhập các biểu thức vào :

- Chính ô điều khiển

- Thuộc tính Control Source của điều khiển, ở đây có thể gọi Expression Builder để tự động xây dựng biểu thức.

– Các thao tác :

- + Với điều khiển bị buộc. Các bước thực hiện như sau (hình 3.6) :



Hình 3.6. Các bước thực hiện để tạo ra điều khiển hộp văn bản textbox.

• Chọn MASV trong danh sách tên trường. Access sử dụng thuộc tính tên của trường như nhãn mặc định cho Textbox.

• Kéo trường sáng vào trong mục chi tiết của form.

+ Điều khiển không bị buộc (“unbound”) : chỉ cần chọn điều khiển Textbox trên thanh công cụ sau đó kéo nó lên form.

– Thuộc tính của điều khiển Textbox (hình 3.7) (chọn hộp văn bản rồi nhấn chuột phải mở hộp thuộc tính văn bản Properties) :

• Decimal Places : định dạng số của dữ liệu.

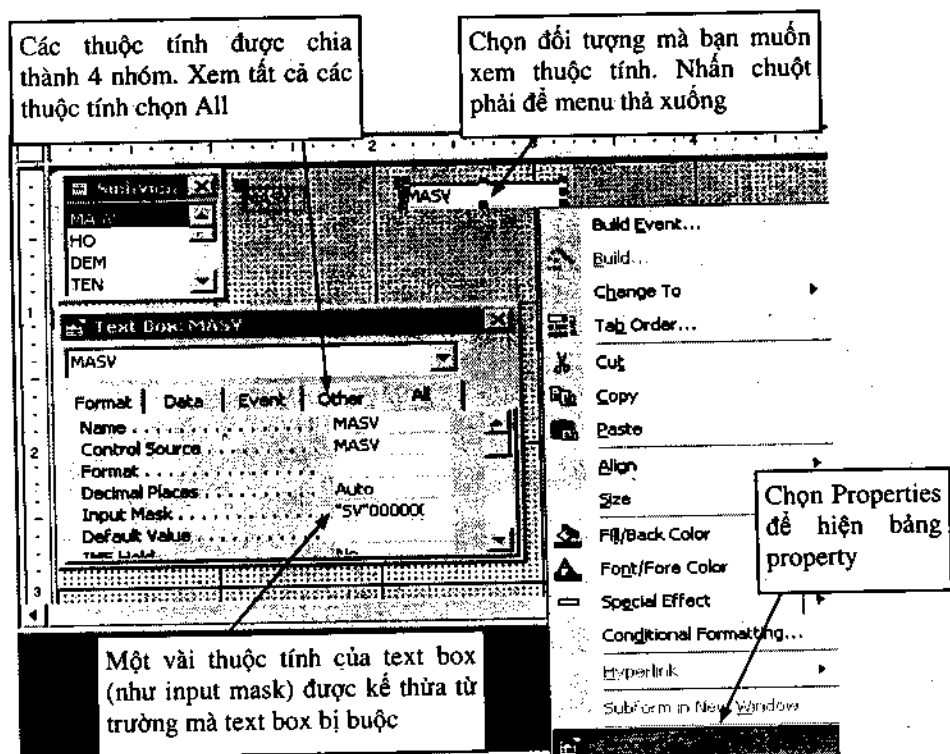
• Default Value : giá trị mặc định.

• Input Mask : định dạng nhập liệu.

• Validation Rule : quy tắc quy định kiểm soát dữ liệu nhập vào.

• Validation Text : thông báo.

• Các thuộc tính khác quy định khuôn dạng của điều khiển như màu nền, font chữ, cách căn lề (trái, phải,...), thanh cuộn (ScrollBar).

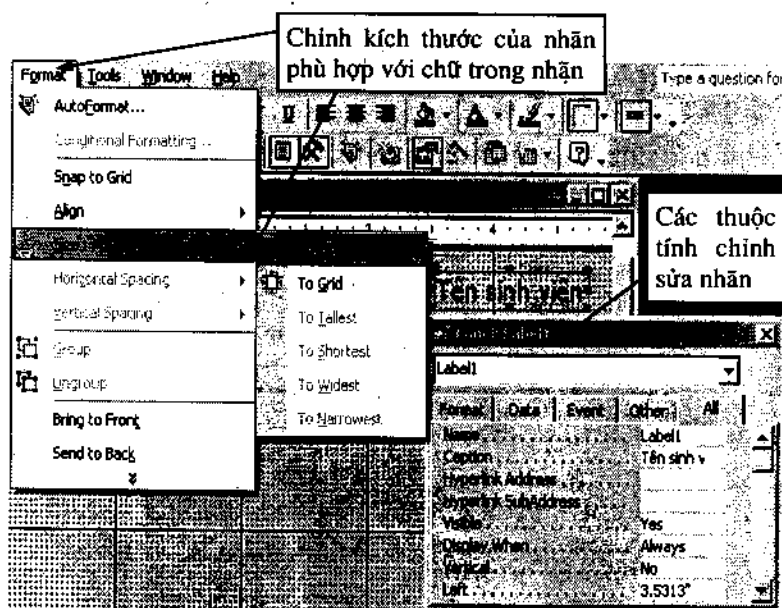


Hình 3.7. Bảng thuộc tính của Textbox.

Điểm khác biệt giữa một hộp văn bản không bị buộc và bị buộc đó là thuộc tính Control Source của hộp văn bản bị buộc được gán cho tên trường mà nó ràng buộc.

3.2.4. Nhãn (Label)

- Nhãn được sử dụng làm tiêu đề, chú giải... Gõ nội dung của nhãn vào trong hộp. Khi soạn thảo nhiều dòng trên nhãn sử dụng tổ hợp phím Ctrl Enter để xuống dòng giữa chừng.
- Hình thức trình bày trên nhãn : Font, màu sắc...
- Việc tạo và chỉnh sửa nhãn trên biểu mẫu được thể hiện trên hình 3.8.



Hình 3.8. Tạo và chỉnh sửa nhãn trên form.

3.2.5. Điều khiển ảnh (Image)

- Trong Access, việc điều khiển ảnh có thể là một trong ba loại sau :
 - + *Image* : kích cỡ nhỏ, không đổi khi xem ở chế độ Form View, dùng để trang trí.
 - + *Bound Object Frame, OLE Object* : dùng để nhập ảnh, ở chế độ form view thay đổi được.
 - + *Unbound Object Frame* : có thể thay đổi ở chế độ Form View, dùng để nhập ảnh khi có nút nhập ảnh.

– Khi dùng điều khiển ảnh cần chú ý các thuộc tính sau :

+ *Clip* : cho phần ảnh vừa với khung.

+ *Size Mode* : chế độ đặt kích thước ảnh.

+ *Special Effect* : các hiệu ứng ảnh đặc biệt.

3.2.6. Hộp kiểm tra (Check Box), nút lựa chọn (Option Button), nút bật tắt (Toggle Button)

– Nếu điều khiển bị buộc, nó luôn luôn được gắn với trường kiểu Yes/No. Tùy yêu cầu thực tế mà chọn kiểu nút cho thích hợp. Sau khi đã chọn kiểu nút thích hợp, kéo trường kiểu Yes/No từ Field List vào form.



– Nếu điều khiển không bị buộc : quan tâm đến thuộc tính value –1/0 để lập trình xử lý sự kiện.

3.2.7. Nhóm lựa chọn (Option Group)

– Nhóm lựa chọn là tập hợp gồm nhiều điều khiển thuộc một trong ba dạng : Option Button, Check Box, Toggle Button.

– Nếu là điều khiển bị buộc, nó dùng để kiểm soát sao cho dữ liệu nhập vào một trường chỉ có thể nhận một trong một số ít giá trị.

– Để tạo nhóm lựa chọn có thể chọn một trong hai cách : dùng wizard và không dùng wizard

+ Dùng Wizard, cần xác định các thông tin sau :

• Chọn các nhãn

• Giá trị ngầm định

• Giá trị ứng với từng mục chọn – chú ý giá trị ngầm định là 1,2,3

• Kiểu nút

• Kiểu khung

• Tên của nhóm

+ Không dùng Wizard

• Tắt Control Wizard

• Chọn Option Group



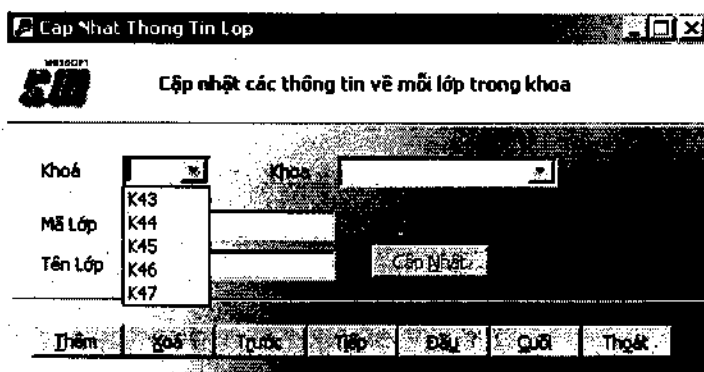
- Chọn trường tên Fieldlist đưa vào form – Một khung xuất hiện
- Chọn kiểu nút – đưa vào khung. Khi đó phần bên trong khung sẽ đổi màu
- Nếu giá trị nhập vào trường được chọn là khác 1, 2, 3... thì phải đổi các giá trị đó tại thuộc tính Option Value.

– Nếu điều khiển là không bị buộc thì làm như cách tạo nhóm lựa chọn không dùng wizard, chỉ có điều là không kéo trường vào (thường form cũng không có nguồn dữ liệu). Sau khi tạo điều khiển xong, căn cứ vào giá trị của các nút lựa chọn mà lập trình xử lý sự kiện.

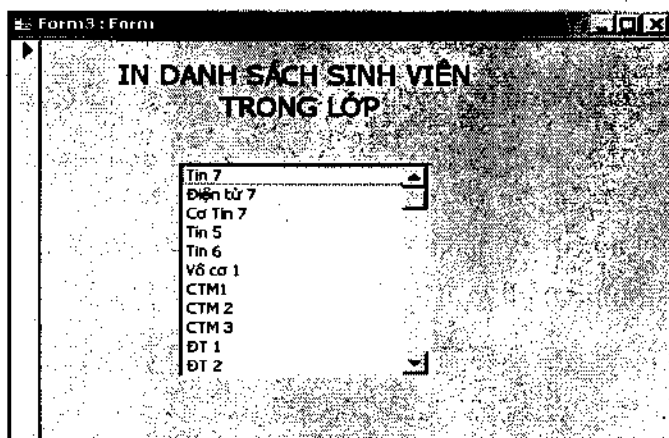
3.2.8. Hộp liệt kê (List Box) và hộp kết hợp (Combo Box)

Hộp kết hợp (hình 3.9) và hộp liệt kê (hình 3.10) thường được sử dụng trong các trường hợp sau :

- Chọn một trong nhiều giá trị để nhập vào trường (trên 7 giá trị).
- Chọn trong một danh sách được hiển thị trên form.



Hình 3.9. Ví dụ về hộp kết hợp.



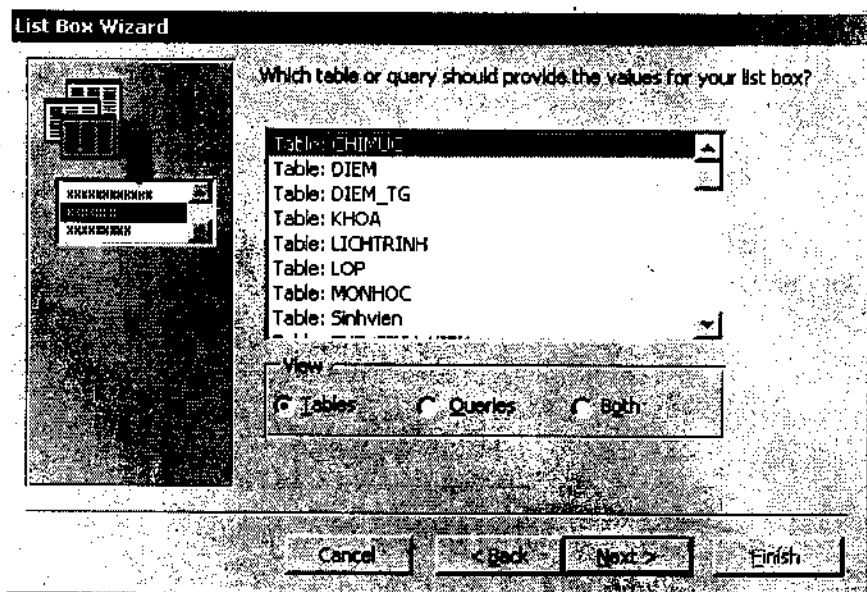
Hình 3.10
Ví dụ về hộp liệt kê.

Sự khác nhau giữa hộp liệt kê và hộp kết hợp : có thể thêm giá trị khác (không có trong danh sách giá trị) vào hộp kết hợp trong khi không thể thêm vào hộp liệt kê.

Tạo List box hoặc Combo box

– Dùng Wizard

1. Bật Control wizard.
2. Chọn điều khiển trên Tool Box (List hoặc Combo).
3. Kéo vào form, Wizard khởi động.
4. Chọn nguồn dữ liệu : lấy từ bảng hay danh sách giá trị (ví dụ hình 3.11).



Hình 3.11. Chọn nguồn dữ liệu cho List box sử dụng Wizard.

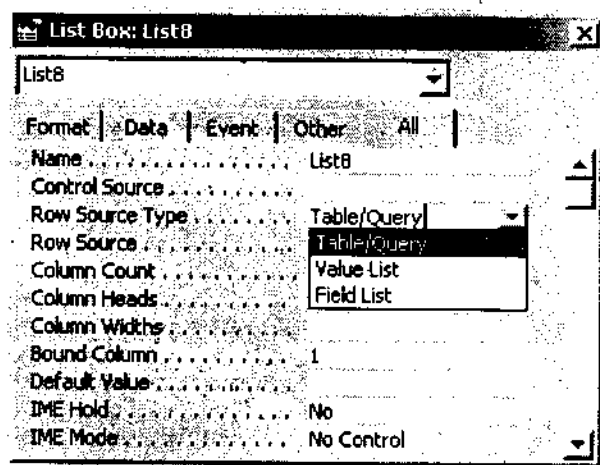
5. Chọn bảng hoặc danh sách giá trị nguồn.
6. Chọn các trường hiện trong hộp.
7. Quyết định có hiện trường khoá không?
8. Chọn trường nhập dữ liệu (Form phải có nguồn dữ liệu. Trường nhận dữ liệu là một trong các trường của bảng hoặc truy vấn nguồn dữ liệu).
9. Tên hộp.

- Không dùng wizard

1. Tắt Control Wizard.
2. Chọn biểu tượng, kéo trường vào form (nếu là điều khiển bị buộc).
3. Mở thuộc tính của List/Combo (hình 3.12).
4. Xác định các thuộc tính sau :

- *Row Source Type* : Chọn Table/Query hoặc Value List

- *Row Source* : Tên bảng hoặc truy vấn nếu Row Source Type là Table/Query. Nếu Row Source Type là Value List thì để danh sách giá trị phân cách bằng dấu ; (có thể là dấu khác nếu đặt lại ở Regional Setting. Cũng có thể là một lệnh Select của SQL nếu muốn hiển thị dữ liệu trong hộp hay nhập vào trường giá trị của những cột bất kỳ.



Hình 3.12. Bảng thuộc tính của List Box.

- *Bound Column* : cột cho giá trị
- *Column Count* : Số các cột sẽ hiển thị dữ liệu trong hộp (tính từ cột 1).
- *Limit to list (yes/no)* : Dùng riêng cho Combo. Khi thêm mới giá trị vào, giá trị sẽ được cập nhật vào nguồn nếu thuộc tính này đặt là No. Tuy nhiên không thêm giá trị này vào danh sách nguồn của hộp Combo.

3.2.9. Nút lệnh (Command Button)

Khi ta kích hoạt nút lệnh, hệ quản trị CSDL sẽ thực hiện một công việc nào đó. Công việc đó có thể đơn giản là mở 1 form, cũng có thể là tính điểm trung bình của sinh viên cả một trường đại học.

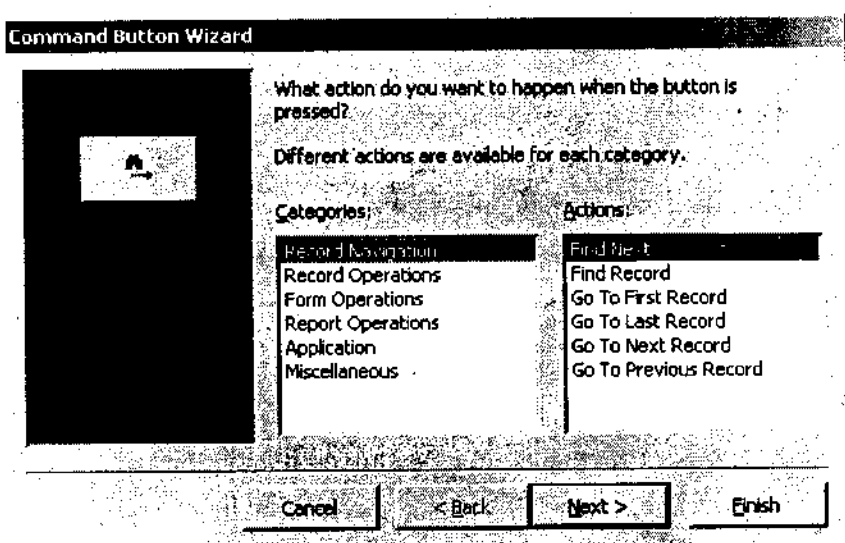
- Tạo nút lệnh bằng Wizard (hình 3.13)

1. Chọn lĩnh vực của nút lệnh

- Di chuyển con trỏ bản ghi
- Thao tác trên bản ghi : thêm xoá, ghi...
- Thao tác trên form
- Thao tác trên báo cáo
- Gọi 1 ứng dụng/ Thoát khỏi ứng dụng.
- Khác : Truy vấn, bảng

2. Chọn hành động và tham số của hành động.

3. Hình thức : chọn chữ; hình ảnh trên nút lệnh.



Hình 3.13. Màn hình tạo Command Button Wizard.

- Không dùng Wizard

1. Tải Control Wizard

2. Chọn nút lệnh, đưa vào form

3. Lập trình xử lý sự kiện trên nút lệnh : sự kiện quan trọng nhất là Click (Macro hoặc Code).

3.2.10. Điều khiển Tab

Điều khiển Tab được sử dụng khi cần nhập dữ liệu cho nhiều bảng độc lập khác nhau trên cùng một form hoặc khi form nhập dữ liệu quá dài.

3.2.11. Thuộc tính của form

Ví dụ : Một form quản lí sinh viên (hình 3.14)

The screenshot shows a web-based form for finding students. The title bar says 'Tìm kiếm sinh viên'. The main heading is 'Tìm kiếm sinh viên theo các tiêu chí tùy chọn'. Below this, there's a section 'Thông tin phục vụ tìm kiếm' with input fields for 'Họ Tên', 'Lớp', 'Khoa', and 'MSSV'. To the right of these fields are buttons: 'Tìm kiếm', 'Dừng', 'Nhập lại', and 'Thoát'. Below the search fields is a table titled 'Kết quả tìm kiếm' with columns: 'MSSV', 'Ten Sinh Viên', 'Lớp', 'Khoa', and 'Khoa Học'. The table body is currently empty. At the bottom, there's a section 'In thẻ sinh viên' with buttons: 'Chọn lọc', 'Tất cả', 'Xem trước', and 'In'.

Hình 3.14. Ví dụ về form quản lý sinh viên.

Các thuộc tính của form gồm :

– Thuộc tính về hình thức trình bày của form

1. *Default View* : Dạng hiển thị ngầm định của form.
2. *Scroll Bars* : Các thanh cuộn để chuyển điều khiển lên xuống.
3. *Record Selector* : Mũi tên chỉ ra bản ghi hiện hành.
4. *Navigation Button* : Các nút để dịch chuyển bản ghi.
5. *Dividing Lines* : Đường phân chia các bộ phận của form.
6. *Picture* : Hình ảnh trên toàn form.
7. *MinMax Buttons* : Nút cho phép thu nhỏ, phóng to kích cỡ của form.

– Thuộc tính liên quan đến dữ liệu của form

8. *Records Source* : Tên truy vấn, bảng nguồn dữ liệu hoặc văn bản SQL của nguồn dữ liệu.
9. *Allow Filters* : Cho phép lọc dữ liệu hiện trên form.
10. *Allow Delections* : Cho phép xoá các bản ghi trên form.

12. *Allow Additions* : Cho phép bổ sung bản ghi trên form.

→ *Các thuộc tính khác*

14. Shortcut Menu : Menu hiện khi nhấn chuột phải.

Có nhiều cách để tạo Main Form/SubForm, cách phổ biến nhất được giới thiệu dưới đây:

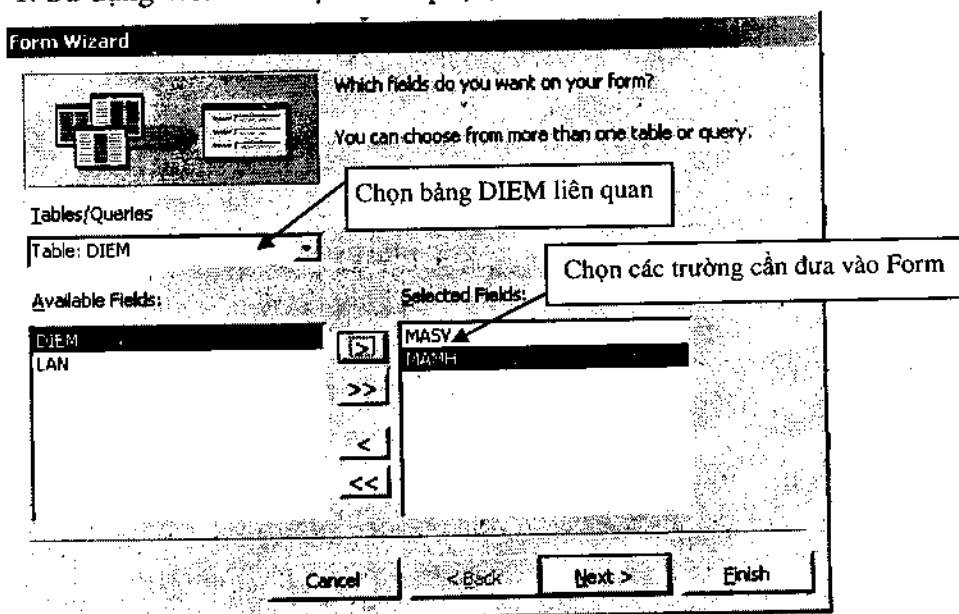
1. Tạo Form chính dạng cột (columnar) và Form phụ dạng hàng (tabular) riêng biệt ;
2. Kéo Form phụ lên Form chính ;
3. Kiểm tra mối liên kết giữa hai Form:

3.3.1. Tạo Form chính

1. Sử dụng Wizard để tạo một Form dạng columnar dựa trên bảng Courses
2. Sắp xếp lại các trường sao cho hợp lý
3. Lưu lại Form dưới tên frmDIEM

3.3.2. Tạo Form phụ

1. Sử dụng Wizard để tạo Form phụ (hình 3.16).



Hình 3.16. Tạo Sub Form dùng Wizard.

2. Các Subform tạo dùng Wizard thường phải được tinh chỉnh để giảm không gian chiếm dụng của các trường.

+ Chọn dạng Form là tabular.

- + Không cần đặt tiêu đề cho Form vì nó sẽ được nhúng vào Form chính
 - + Chọn Modify the form's design để sang chế độ thiết kế.
3. Lưu Form dưới tên sfrmDIEM và đóng lại.

3.3.3. Liên kết hai Form

Trong phần này, chúng ta sẽ quay lại Mainform và kéo Subform từ cửa sổ Database lên vị trí thích hợp trên Mainform.

1. Mở lại Form chính trong chế độ thiết kế.
2. Kéo Form phụ từ cửa sổ Database lên Form chính. Chiều rộng của điều khiển Subform (cửa sổ trắng) tự động giãn để chứa hết Subform.

3.4. GIỚI THIỆU REPORT WIZARD

Báo cáo dùng để in ra giấy những thông tin kết xuất từ CSDL theo khuôn dạng, yêu cầu của từng cơ quan

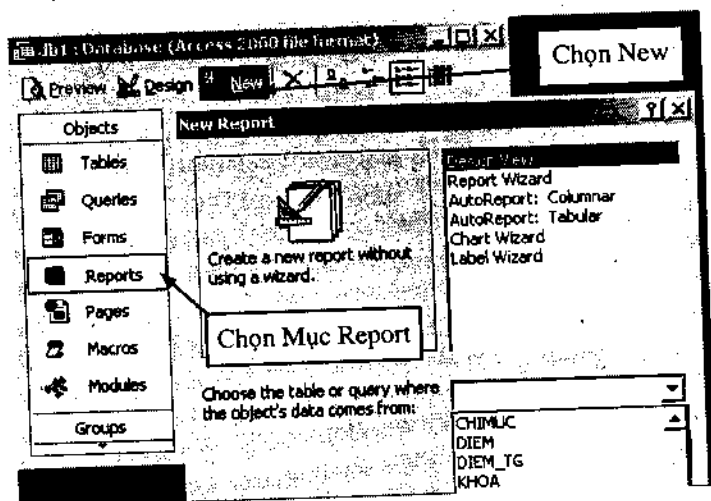
Các báo cáo có thể dựa trên cả table hoặc query.

Báo cáo có nhiều dạng nhưng thông dụng nhất là các dạng sau :

- Báo cáo dạng văn bản
- Báo cáo dạng bảng đơn giản
- Báo cáo thực hiện một số tính toán, thống kê
- Báo cáo dạng biểu đồ, nhãn thư...

Tạo khuôn dạng của báo cáo ban đầu

1. Chọn mục Report trên cửa sổ Database.
2. Chọn New → Xuất hiện màn hình tạo báo cáo mới (hình 3.17).



Hình 3.17. Màn hình tạo báo cáo mới.

3.4.1. Tạo báo cáo bằng wizard

Có thể chọn một trong các lựa chọn sau :

* *Auto report*

Tự động đưa ra báo cáo khi đã chọn nguồn dữ liệu. Không được chọn thông tin nào.

* *Report Wizard :*

Người sử dụng phải lựa chọn các mục sau :

1. Chọn nguồn dữ liệu (bảng hoặc truy vấn).
2. Chọn trường dùng trong báo cáo.
3. Nếu muốn tạo báo cáo phân nhóm thì xác định trường cần phân nhóm.
4. Sắp thứ tự (nếu cần).
5. Dạng và kiểu trình bày của báo cáo.

* *Báo cáo dạng nhãn*

– Dùng wizard để xác định kích cỡ, các văn bản chủ yếu lấy thông tin từ trong CSDL

– Các bước cơ bản :

1. Xác định kích cỡ và cách bố trí các nhãn trên giấy.
2. Font chữ ngầm định.
3. Các trường cần lấy thông tin.
4. Trường cần sắp xếp.
5. Mở thiết kế của báo cáo dạng nhãn, điều chỉnh lại một số điều khiển cho phù hợp yêu cầu thực tế.

* *Báo cáo dạng biểu đồ*

1. Tạo báo cáo mới và xác định nguồn dữ liệu.
2. Chọn Chart Wizard.
4. Chọn các trường xuất hiện trong biểu đồ.
5. Chọn dạng trình bày của biểu đồ.
6. Sắp đặt các thông tin trên các trục tọa độ.
7. Tiêu đề của biểu đồ.

3.4.2. Các vấn đề khác liên quan đến thiết kế báo cáo

1. Các thuộc tính chung của báo cáo

Báo cáo có các thuộc tính giống biểu mẫu như các thuộc tính liên quan đến kích cỡ, ảnh.

Ngoài ra báo cáo còn có một số thuộc tính đặc biệt : Page Header| Footer: trên mọi trang không có ở trang có Report Header, không có ở trang có Report Footer, không có cả ở hai trang đó.

2. Các sự kiện trên báo cáo

- *On Open* : khi mở Report
- *On Close* : khi đóng Report
- *On Activate* : khi cửa sổ chứa Report trở thành cửa sổ hiện hành
- *On Deactivate* : khi cửa sổ chứa Report không còn là cửa sổ hiện hành
- *On No Data* : nguồn dữ liệu không có bản ghi nào, tốt nhất là không nên in báo cáo đó.

3. In báo cáo

- Kích hoạt biểu tượng máy in hoặc dùng menu File → Print
- Nếu dùng hộp thoại Print có thể in ra tệp dạng .PRN và có thể đem in tại máy tính bất kỳ có nối với máy in dù trên máy đó không cài Access.

3.5. THIẾT KẾ CÁC THÀNH PHẦN CỦA BÁO CÁO

Chọn mục Design View. Màn hình thiết kế báo cáo hiện ra với 3 chế độ song song tồn tại, đó là :

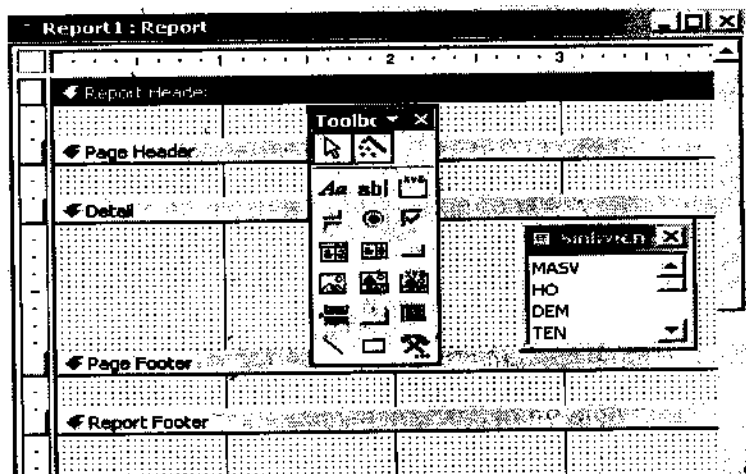
1. Design View : thiết kế báo cáo.
2. Print Preview : trường hợp xem bản in báo cáo.
3. Layout Preview : Xem hình thức của báo cáo khi in ra.

Trên màn hình thiết kế báo cáo cũng luôn luôn để hai công cụ là toolbox và fieldlist.

Nhìn vào màn hình thiết kế báo cáo (hình 3.18) ta thấy có các phần chính :

- + *Detail* : Một bản ghi một lần, dùng để chứa dữ liệu của báo cáo.

- + *Page Header* : tiêu đề đầu của trang, lặp một trang/ một lần.
- + *Page Footer*: số trang, dùng hàm Page, trang hiện thời.
dùng hàm Pages tổng số trang = "Trang"& [page]&"/"& [pages]
ngày lập, tiêu đề dưới
- + *Report Header* : Chứa tiêu đề chung của toàn báo cáo.
- + *Report Footer* : Chứa dòng tổng cộng.
- + *Group Header/ Footer* : Dùng cho các báo cáo được phân nhóm.



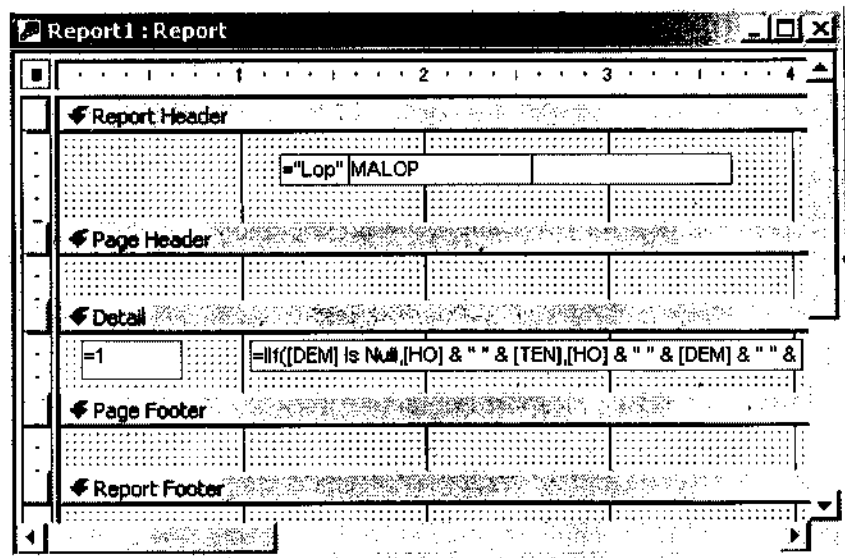
Hình 3.18. Màn hình thiết kế Report.

3.5.1. Báo cáo dạng văn bản

- Đặc trưng : Văn bản trộn lẫn với thông tin trong CSDL, vì vậy toàn bộ thiết kế nằm trong phần Detail
- Trong báo cáo thường bao gồm các điều khiển :
 - + *Nhãn* : chứa các đoạn văn bản với font chữ theo yêu cầu thực tế.
 - + *Textbox* : thường bao gồm 2 loại : bị buộc và tính toán được.
 - + *Ảnh* : Có thể bị buộc hay không bị buộc.

Ví dụ 3.2 :

In danh sách sinh viên trong một lớp. Màn hình thiết kế báo cáo có dạng như hình 3.19 :



Hình 3.19. Màn hình thiết kế báo cáo "In danh sách sinh viên trong một lớp".

3.5.2. Báo cáo đơn giản dạng bảng

1. Tiêu đề :

- + Nếu chỉ xuất hiện ở trang đầu : Report Footer.
- + Nếu xuất hiện trên mọi trang : Page Footer.

2. Đầu cột : thường để ở Page Footer (Chỉ trừ trường hợp dán nổi các trang thì để ở Report Footer)

Textbox chứa dữ liệu nguyên dạng : kéo các dữ liệu xuất hiện trong bảng từ Field List vào phần Detail. Xóa nhãn đi kèm điều khiển. Việc căn chỉnh giống như căn chỉnh điều khiển trên form.

- Các giá trị trùng nhau chỉ xuất hiện một lần
- Để có các Textbox có độ cao giống nhau, nên chọn cách sao chép hoặc tác động vào thuộc tính của điều khiển.

3. Tạo các Textbox (điều khiển tính toán được) chứa dữ liệu kết xuất dạng :

DCount

DSum (<Biểu thức>,<Miền cần tính>,<điều kiện>)

DAvg

DMax

...

Ví dụ 3.3 :

Tính điểm Trung bình của sinh viên

$$\text{DSum}("[\text{diem}] * [\text{sotrinh}], "[\text{Bang Diem Ca Nhan}], "[\text{lan}] = '1'") / \text{DSum}("[\text{sotrinh}], "[\text{Bang Diem Ca Nhan}], "[\text{lan}] = '1'")$$

4. Textbox được dùng làm trung gian để tính giá trị cho những Textbox khác.

5. Vị trí dòng tổng cộng : Report Footer.

6. Số trang hoặc ngày lập báo cáo

3.5.3. Báo cáo thống kê

Nói chung nguồn dữ liệu của các báo cáo này thường là các truy vấn dạng crosstab hoặc tập trung dữ liệu vào bảng trắng trong quá trình lập trình.

Cũng có thể dùng các điều khiển dạng tính toán được chứa các hàm thư viện hoặc hàm thư viện có điều kiện, đó là :

= Sum|Avg|StDev|StDevP|Var|VarP(<Biểu thức số>)

= Count|Min|Max(<Biểu thức số>)

= DSum|DAvg|DStDev|DStDevP|DVar|DVarP(<Biểu thức số>; <Tên nguồn dữ liệu>; <Điều kiện>)

3.6. BÁO CÁO CHÍNH - PHỤ

Cũng như trên form, khi tạo báo cáo với nguồn dữ liệu lấy từ nhiều bảng yêu cầu đảm bảo đúng khuôn dạng thực tế, người ta phải dùng báo cáo chính - phụ, ví dụ phiếu xuất vật tư, lý lịch nhân viên ...

Cách tạo báo cáo chính - phụ cũng giống như trên form, bao gồm các bước sau :

1. Tạo báo cáo chính

2. Tạo báo cáo phụ như một báo cáo độc lập

3. Mở đồng thời cửa sổ thiết kế báo cáo chính và cửa sổ Database. Kéo báo cáo phụ từ cửa sổ Database vào.

4. Mở thuộc tính của điều khiển SubReport trên báo cáo chính. Kiểm tra sự kết nối dữ liệu qua các thuộc tính *LinkMasterFields* và *LinkChildFields*

Chú ý : các thuộc tính *LinkMasterFields* và *LinkChildFields* còn xuất hiện cả trong biểu đồ. (Không cần tạo báo cáo phụ dạng biểu đồ mà nên dùng Insert → Chart). Do vậy có thể liên kết dữ liệu ở báo cáo chính và dữ liệu ở biểu đồ trong đó.

3.7. SẮP XẾP, PHÂN NHÓM TRÊN BÁO CÁO

– Mở View

– Chọn Sorting and Grouping để mở hộp thoại Sorting and Grouping (hình 3.20)

Ví dụ 3.4 : Sắp xếp trường MASV theo chiều tăng

Field/Expression	Sort Order
MASV	Ascending

Group Properties	
Group Header	No
Group Footer	No
Group On	Each Value
Group Interval	1
Keep Together	No

Select a field or type an expression to sort or group on

Hình 3.20. Sắp xếp phân nhóm trên báo cáo.

1. Chọn trường/biểu thức.
2. Chọn thứ tự sắp xếp.
3. Các thuộc tính (Có cần Group Header/Footer hiển thị không).
4. Group On: giá trị để phân nhóm.

Chữ cái đầu để phân nhóm (phải quan tâm đến thuộc tính tiếp theo Group Interval).

5. Group Interval (khoảng phân nhóm) : Số ký tự cần để phân nhóm.

Ví dụ 3.5 :

ANH Group Interval 1 → ở cùng nhóm 1

AN Group Interval 2 → ở cùng nhóm 2

AT Group Interval 3 → ở cùng nhóm 3

6. Keep together

Ví dụ 3.6 :

In 5 người một nhóm nhưng chỉ in 2 người đã hết trang thì chuyển toàn bộ sang trang mới (Yes) hoặc để nguyên (No).

3.8. CÂU HỎI ÔN TẬP – BÀI TẬP

1. Sử dụng Wizard để tạo Form dạng columnar cho tất cả các bảng đã tạo.

Tạo mainform và subform cho sinh viên và điểm thi. Form này được dùng để theo dõi điểm thi, số lần thi của sinh viên theo lớp, khoa và khoá học của sinh viên (hình 1).

Hình 1

Các form được tạo theo các chỉ dẫn sau :

– Form chính dựa trên query tạo theo bảng LOP và bảng MONHOC lấy vào các trường như trên form

– Form phụ dựa trên bảng DIEM_TG được tạo bởi các trường được cho trong hình 2 :

DIEM_TG : Table			
	Field Name	Data Type	Description
?	MASSV	Text	Mã sinh viên
	TEN	Text	Tên đầy đủ của sinh viên
	DIEM	Text	Điểm thi
?	LAN	Number	Lan Thi

Field Properties	
General	Lookup
Field Size	8
Format	
Input Mask	"SV"000000
Caption	
Default Value	"SV000000"
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None

Hình 2

2. Tạo Form danh sách lớp, chỉ nhập các thông tin tạo danh sách lớp (hình 3)

Tạo Danh Sách Lớp	
 Nhập thông tin tạo danh sách lớp	
Khoá	
Khoa	
Lớp	
Xem trước In Thoát	

Hình 3

3. Tạo các báo cáo sau :

– Báo cáo : bảng điểm cá nhân. Thống kê điểm của từng sinh viên (hình 4)

BẢNG ĐIỂM CÁ NHÂN									
Họ tên sinh viên: =If([DEM] Is Null,[-0] & "" & [TEN],[HO]									
Lớp: [TENLOP]									

Hình 4

- Báo cáo để in danh sách sinh viên (hình 5)

- Báo cáo về in danh sách sinh viên				
Report Header				
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI				
=IF([TENKHOA] Is Null,"", "Khoa " & [TENKHOA])				
		DANH SÁCH SINH VIÊN		
= "Lớp: " & [TENLOP]				
Page Header				
STT	MÃ SINH VIÊN	HỌ TÊN	QUÊ QUÁN	GHI CHÚ
Detail				
=1	=[MASV] & "	=IF([DEM] Is Null," " & [HO] & "	QUEQUAN	Unbound
Page Footer				
= "Trang " & [Page] & "/" & [Pages]				
Report Footer				
= "Tổng số " & Count([MASV]) & "				

Hình 5

Chương IV

LẬP TRÌNH TRÊN ACCESS

4.1. LẬP TRÌNH ĐƠN GIẢN BẰNG MACRO – ỨNG DỤNG CỦA MACRO

Macro là một trong 6 đối tượng cơ bản của CSDL (Database) dùng để liên kết các đối tượng của CSDL làm cho CSDL trở nên linh hoạt hơn.

Macro là tập các thao tác thường được dùng trong các xử lý dữ liệu, được chuyển thành các hành động và có thể sử dụng nó thay vì phải viết một đoạn chương trình xử lý.

Macro thường được gắn với các sự kiện của đối tượng CSDL.

Các trường hợp sử dụng Macro :

- Khi khởi động CSDL.
- Khi kích hoạt nút lệnh hay chọn một mục trong Menu Pull Down.
- Thực hiện khi kích hoạt các sự kiện trên form/report.

Khi ra lệnh thực hiện Macro, các hành động được thực hiện từ trên xuống dưới.

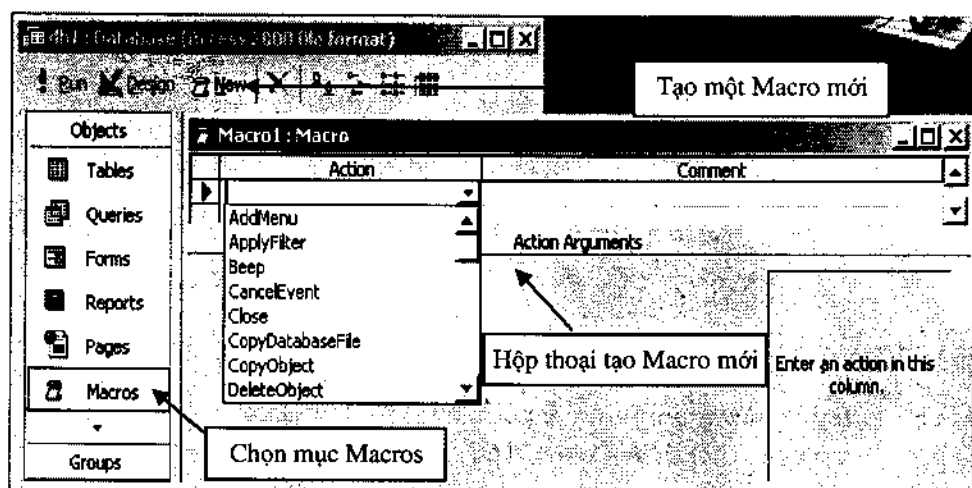
4.1.1. Tạo Macro mới

Tại cửa sổ Database – chọn Macro – chọn New, hộp thoại tạo Macro mới được mở ra (hình 4.1).

4.1.2. Các hành động trong Macro

1. *Addmenu* : Thêm một cột vào Menu. Các tham số :

- + Menu Name : Tên cột hiện trên màn hình.
- + Menu Macro Name : Tên Macro ứng với cột đó
- + Status Bar Text



Hình 4.1. Hộp thoại tạo Macro mới.

2. *Close* : Đóng một đối tượng

+ Object Type (Kiểu đối tượng) : Form/ Record/...

+ Object Name (Tên đối tượng)

+ Save : Yes/No/Prompt

3. *Apply Filter* : Lọc trên Form → Truy vấn/ Điều kiện.

4. *Run Command* : Thực hiện một mục chọn trên menu chính.

5. *Run Application* : Chạy một ứng dụng (phải chỉ ra đường dẫn).

6. *Goto Control* : Chuyển con trỏ đến điều khiển nào đó.

7. *Find Next, Find Record* : Tìm bản ghi thoả mãn điều kiện.

8. *GotoRecord* : chuyển con trỏ tới một bản ghi nào đó, có thể là bản ghi đầu, cuối, trước, sau, số cụ thể.

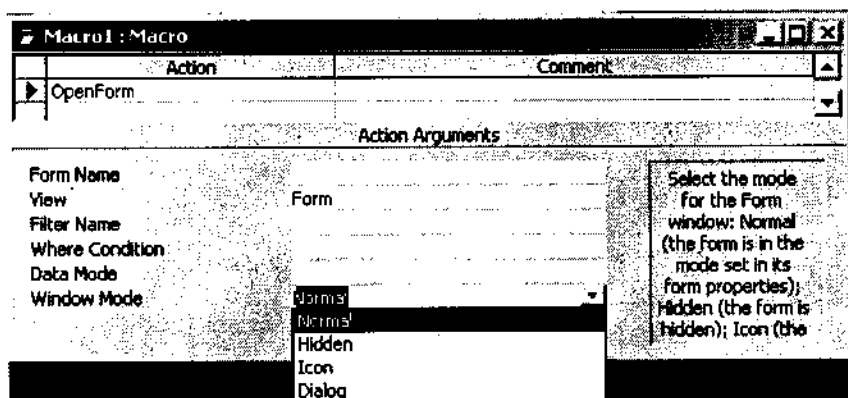
9. *MsgBox* : Hiển thị thông báo.

10. *Open Form* : Cho phép mở form (hình 4.2). Gồm 6 tham số :

+ Form Name : tên của form cần mở. Chọn trong list box.

+ View : Mở ở 1 trong 3 chế độ Design View, Form View (ngắm định), Datasheet View.

- + Filter Name : Tên của bộ lọc – Tên của query.
- + Where Condition : Điều kiện lọc.
- + Data Mode : Phương thức hiện dữ liệu: có thể sửa thêm ... hay không.
- + Window mode : Kích cỡ của cửa sổ.

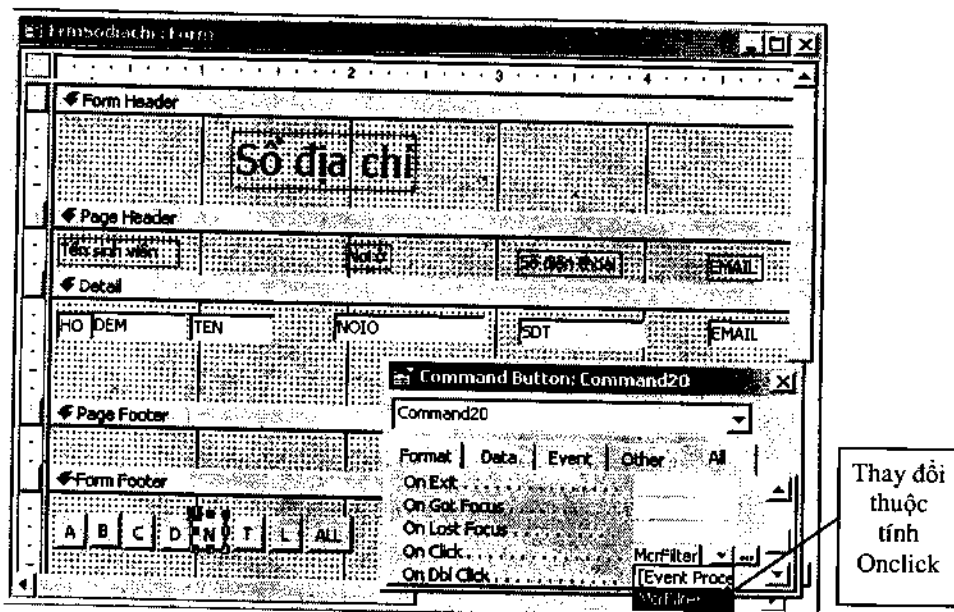


Hình 4.2. Ví dụ Macro Open Form.

11. *OpenReport* : Mở report. Các tham số : Report Name – View – Filter Name – Where Condition.
12. *OpenQuery*, *OpenTable*, *OpenModule*.
13. *SetValue* : Đặt giá trị cho một trường hoặc điều khiển nào đó.
14. *Requery* : Thực hiện lại truy vấn nguồn của form (điều khiển chứa dữ liệu lấy trên form).
15. *Repaint Object* : Tính toán lại giá trị của các điều khiển tính toán được.
16. *RunSQL*.
17. *TransferDatabase* : Chuyển đổi dữ liệu trong bảng sang .xls(Excel) hay Text.
18. *TransferText* : Chuyển đổi dữ liệu trong bảng sang .xls(Excel) hay Text.

4.1.3. Macro có điều kiện

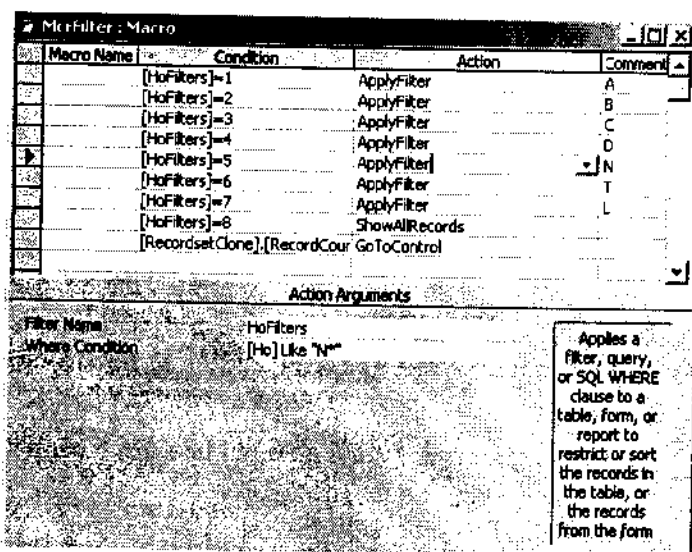
Ví dụ 4.1 : Thiết kế form sổ địa chỉ. Màn hình thiết kế form sổ địa chỉ có dạng như hình 4.3.



Hình 4.3. Màn hình thiết kế form số địa chỉ.

1. Dùng truy vấn qrySodiachi gồm họ tên, nơi ở, số điện thoại, Email của sinh viên để xây dựng form số địa chỉ.

2. Tạo các nút lệnh và thay đổi thuộc tính OnClick của các nút thành mcrFilters. McrFilters có nhiệm vụ lọc các tên sinh viên có chữ cái đầu trong HO thỏa mãn điều kiện của nút lệnh. Macro có điều kiện được tạo ra (hình 4.4).

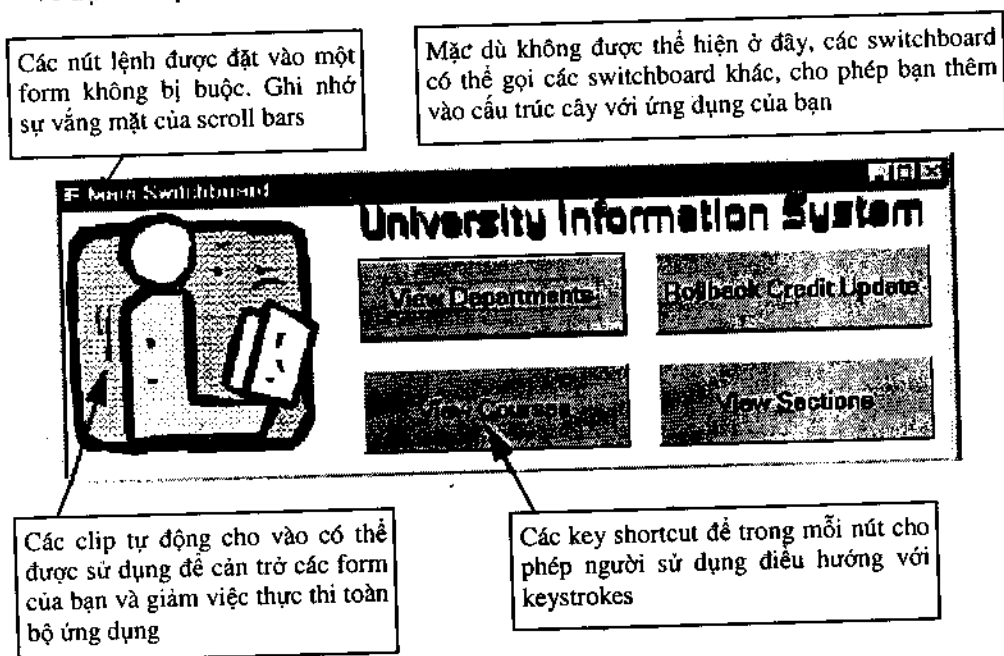


Hình 4.4. Macro có điều kiện.

4.1.4. Ứng dụng của Macro

1. Xử lý các sự kiện
2. Tạo menu
3. Tạo shortcut Menu
4. Menu Switchboard : Một trong những ứng dụng đơn giản nhưng hiệu quả của bảy sự kiện là gán lệnh OpenForm với một nút trên Form để tạo bảng menu mở các form khác.

Ví dụ về một menu Switchboard được minh hoạ trên hình 4.5.



Hình 4.5. Ví dụ một menu Switchboard.

- Tạo một Form không ràng buộc dữ liệu.
- Bỏ thanh trượt (scroll bars), nút duyệt (Navigation Buttons) và nút bôi đen bản ghi (record selectors) sử dụng bảng thuộc tính.
- Cài form dưới tên swbMain.

4.2. GIỚI THIỆU NGÔN NGỮ VBA (Visual Basic For Application) VÀ PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG SỰ KIỆN

Để lập trình ACCESS có thể viết các loại sau :

1. Macro
2. Hàm (Function)
3. Thủ tục (Subroutine)

Các thủ tục và hàm được thực hiện khi có sự kiện được kích hoạt

VBA là một tập con của Visual Basic, các khái niệm lập trình ở đây là chung cho các ngôn ngữ lập trình.

VBA được dùng trong ACCESS để :

- Sử dụng các cấu trúc điều khiển phức tạp
- Sử dụng hằng và biến
- Sử dụng các hàm có sẵn
- Xử lý giao dịch
- Tạo và xử lý các đối tượng database chương trình
- Cài đặt xử lý lỗi
- Tạo thư viện hàm người dùng
- Gọi các hàm API của Windows
- Thực hiện các lệnh DDE và OLE

Cửa sổ debug

Cửa sổ Debug là một công cụ giúp chạy thử, gỡ lỗi chương trình, in kết quả trung gian.

1. Nhấn vào module trong cửa sổ và nhấn New. Chúng ta phải mở một module thì mới mở được cửa sổ Debug
2. Chọn menu Debug

Trong cửa sổ Debug Window, có thể nhập các lệnh trực tiếp, ví dụ nhập các lệnh sau :

Print "hello world!"

s = "hello"

? s & "world"

? "s" & "world"

pi = 3.14159

? cos(2*pi)

s = "basic or cobol"

? UCase(s)

?mid(s,5,6)

4.2.1. Cấu trúc của một module (đơn thể)

* Module sử dụng chung

Module sử dụng chung bao gồm các bộ phận sau :

– Tuỳ chọn :

+ Option Explicit : khai báo mọi biến.

+ Option Base

+ Option Compare : quy định các so sánh chuỗi ký tự

Binary : phân biệt chữ (ký tự) hoa/thường.

Text : Không phân biệt chữ hoa/thường.

Database : quy định thứ tự riêng cho CSDL

– Khai báo : Biến hằng dùng chung cho toàn module

– Các thủ tục, hàm của module

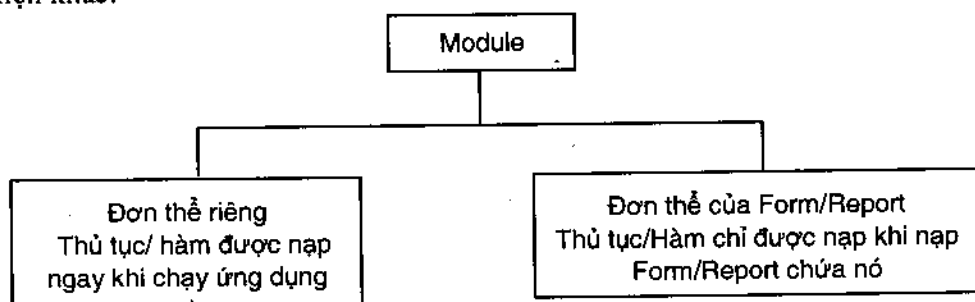
* Module lớp, module xử lý sự kiện trên form, report

Module lớp, module xử lý sự kiện trên form, report chỉ dùng cho các sự kiện chứa nó và bao gồm các bộ phận sau :

– Các tuỳ chọn.

– Khai báo biến xác định hằng dùng trong module.

- Các thủ tục xử lý sự kiện.
- Các thủ tục được thủ tục xử lý sự kiện gọi. Không dùng chung với sự kiện khác.



** Cấu trúc một module*

[General Declarations] ‘Phần khai báo chung

Option Explicit ‘Nếu có mục này thì khai báo biến là bắt buộc

Option Base 0 ‘Để khai báo chỉ mục cho mảng (0 hay 1)

Sub Thu_tuc1(...)

‘Các lệnh

End Sub

...

Sub Thu_tucn(...)

‘Các lệnh

End Sub

Function HamXYZ()

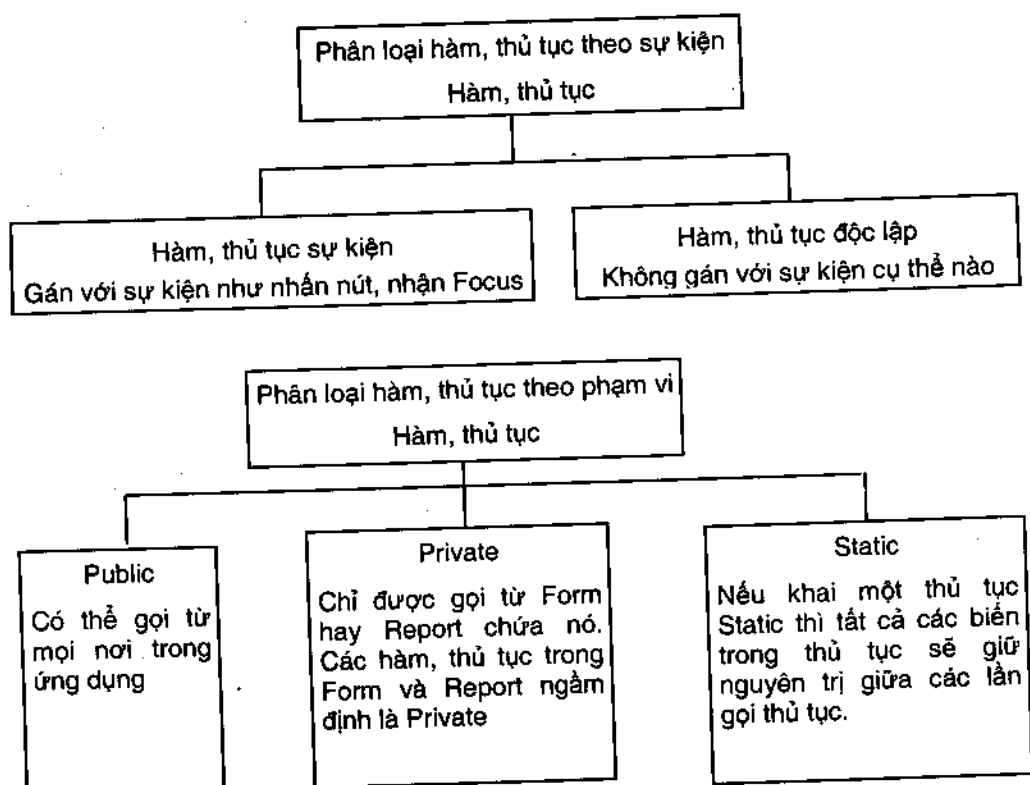
‘Các lệnh

HamXYZ=...

End Function

<!> Nếu muốn đặt khai báo biến luôn là bắt buộc, vào menu tools chọn Options, chọn Editor chọn Require Variable Declaration.

4.2.2. Cấu trúc của thủ tục và hàm



Ví dụ 4.2 :

Về thủ tục static. Nhập trong module và chạy trong cửa sổ Debug, các biến không lập lại 0 sau mỗi lần gọi

```
Static Sub IncremenThem()
```

```
Dim intCounter1 As Integer
```

```
Dim intCounter2 As Integer
```

```
Dim intCounter3 As Integer
```

```
intCounter1 = intCounter1 + 1
```

```
intCounter2 = intCounter2 + 1
```

```
intCounter3 = intCounter3 + 1
```

```
Debug.Print intCounter1 & ""&intCounter2 & ""&intCounter3  
End Sub
```

** Cấu trúc của một hàm*

[Static] [Public| Private] Function <Tên hàm> (<Danh sách tham số hình thức và kiểu>) As <kiểu>

<Các lệnh>

<Tên hàm> = <Biểu thức>

<Các lệnh>

End Function

<Danh sách tham số hình thức và kiểu> có dạng

[By Val] <Tên> As <Kiểu>|, [By Val] <Tên> As <kiểu> ...|

<Các lệnh> bao gồm cả lệnh khai báo

** Cấu trúc của một thủ tục*

– Thủ tục thường

[Static] [Public|Private] Sub <Tên thủ tục> (<Danh sách tham số hình thức và kiểu>)

<Các lệnh>

End Sub

– Thủ tục xử lý sự kiện

+ Sự kiện trên form

Private Sub Form_<tên sự kiện>(<Danh sách tham số hình thức và kiểu>)

<Các lệnh>

End Sub

+ Sự kiện trên báo cáo

Private Sub Report_<tên sự kiện>(<Danh sách tham số hình thức và kiểu>)

<Các lệnh>

End Sub

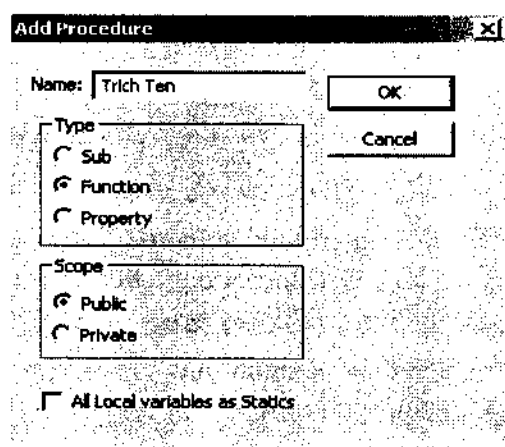
+ Sự kiện trên điều khiển của Form hoặc Report

Private Sub <Tên điều khiển>_<Tên sự kiện>(<Danh sách tham số hình thức và kiểu>)

<Các lệnh>

End Sub

Màn hình tạo thủ tục mới có dạng như hình 4.6.



Hình 4.6. Màn hình tạo thủ tục mới.

4.2.3. Các quy định khi viết chương trình VBA

- Mỗi lệnh thường được viết trên một dòng. Nếu muốn xuống dòng dùng dấu _
- Mỗi dòng được phép viết nhiều lệnh. Các lệnh phân cách bằng dấu :
- Không phân biệt chữ hoa, chữ thường
- Chú thích có thể viết như sau :
 - + Chú thích trên toàn bộ dòng : Mở đầu dòng bằng dấu nháy đơn (có thể dùng từ Rem thay cho dấu nháy đơn như trong các phiên bản trước)
 - + Chú thích trên phần còn lại của dòng : Phần chú thích bắt đầu bằng dấu nháy đơn.

4.2.4. Dịch và chạy chương trình

Cửa sổ trực tiếp : Thủ tục

<Tên thủ tục><Danh sách tham số thực sự>

Hàm

?<Tên hàm>(<Danh sách tham số thực sự>)

Debug.Print (In ra trên cửa sổ Debug)

4.2.5. Cách gọi chương trình con trong VBA

1. Gọi thủ tục

<Tên thủ tục> <Danh sách tham số thực sự>

Call <Tên thủ tục> (<Danh sách tham số thực sự>)

Các tham số thực sự có thể không đặt đúng trình tự mà tham số hình thức tương ứng xuất hiện trong chương trình con. Khi đó viết danh sách tham số thực sự dưới dạng:

<Tên tham số hình thức> := <tham số thực sự> [, <Tên tham số hình thức> := <Tham số thực sự> ...]

2. Gọi hàm

<Tên hàm> (<Danh sách tham số thực sự>)

Khi không quan tâm đến giá trị mà hàm trả lại, có thể gọi hàm như thủ tục

<Tên hàm><Danh sách tham số thực sự>

4.3. CÁC YẾU TỐ CƠ BẢN TRONG CHƯƠNG TRÌNH VBA

4.3.1. Các kiểu dữ liệu dùng trong chương trình

- Số nguyên : Integer, Long, Byte
- Số thực : Single, Double
- Logic: Boolean
- Tiền tệ : Currency
- Xâu ký tự : String, String*n
- Ngày/ giờ : Date
- Variant : Biến thuộc kiểu dữ liệu này có thể nhận giá trị là số, xâu ký tự, ngày/giờ hoặc giá trị Null.

4.3.2. Tên

- Tên gồm tối đa 255 ký tự và được bắt đầu bằng chữ cái.

– Tên có thể xây dựng từ mọi ký hiệu trừ !@&\$#% (Đây là những ký hiệu phân cách và ký hiệu định kiểu của Access).

4.3.3. Biến

– Khai báo biến : Phải khai báo mọi biến nếu dùng Option Explicit. Nếu không dùng, các biến không khai báo sẽ được ngầm định kiểu là Variant.

– Khai báo biến có tác dụng định kiểu cho các biến được khai báo, ngoài ra còn khởi tạo (gán giá trị đầu) cho biến.

1. Khai báo biến bằng lệnh Dim

Kiểu khai báo thông dụng nhất là dùng lệnh Dim

Dạng lệnh :

Dim<tên biến> As <kiểu>|<tên biến> As <kiểu>|

<kiểu> là tất cả các kiểu dữ liệu đã nhắc đến ở trên

Ví dụ 4.3 :

Dim a,b,c As Integer

Thì a,b → variant

c → integer

* Khai báo biến cục bộ (local)

Ví dụ 4.4 :

Private Sub cmdOkey-Click

Dim strAnimal As String

strAnimal = "Dog"

Call ChangeAnimal

Debug.Print strAnimal ' vẫn là Dog

End Sub

Private Sub ChangeAnimal

strAnimal = "Cat"

End Sub

** Biến tĩnh (Static) :* là một dạng biến địa phương đặc biệt

Ví dụ 4.5 :

```
Private Sub cmdLocal_Click()  
    Dim intCounter As Integer  
    intCounter = intCounter + 1  
    Debug.Print intCounter  
End Sub
```

Trong ví dụ này, biến cục bộ được khởi gán lại sau mỗi lần chạy

Ví dụ 4.6 :

```
Private Sub cmdStatic_Click()  
    Static sintCounter As Integer  
    sintCounter = sintCounter + 1  
    Debug.Print sintCounter  
End Sub
```

Trong ví dụ này, biến tĩnh vẫn lưu lại trị sau mỗi lần chạy.

** Biến Private*

```
[General Declarations]  
Option Explicit  
Private Sub cmdModule_Click()  
    mintCounter = 5  
    Debug.Print mintCounter  
End Sub
```

2. Khai báo trong Form, Report hay Module

** Biến Public*

```
[General Declarations]  
Option Explicit  
Public pintCounter as Integer
```

Ví dụ 4.7 :

```
Private Sub cmdPublic_Click()
```

```
    PintCounter = 50
```

```
    Debug.Print pintCounter
```

```
End Sub
```

Biến Public được khai báo trong Form, Report hay Module. Các biến Login-IDs, biến thiết lập môi trường thường được khai báo Public.

Ví dụ 4.8 :

Form đăng nhập hệ thống (hình 4.7). Chỉ cho phép nhập mật khẩu 3 lần. Nếu cả 3 lần đều sai, sẽ tự động thoát khỏi Access.

Hình 4.7. Form đăng nhập hệ thống.

4.3.4. Hằng

1. Cách biểu diễn các hằng thuộc các kiểu dữ liệu khác nhau

- Số → Toán học
- Xâu ký tự → đặt trong cặp dấu “ ”
- Ngày giờ → phân cách bằng dấu # #
- Logic → True, False.

2. Một số hằng đặc biệt

- Các hằng của VB: bắt đầu bằng vb. Ví dụ: vbYesNo, vbMonday ...
- Các hằng của Access: bắt đầu bằng ac. Ví dụ: acForm ...
- Các hằng của DAO: bắt đầu bằng db. Ví dụ: dbReadOnly, dbOpenDynaset
- Các hằng của ADO: bắt đầu bằng ad. Ví dụ : adModeRead, adLockReadOnly

3. Khai báo hằng

Const <Tên hằng> = <Biểu thức hằng>

4.3.5. Mảng

Dim <Tên mảng>([<kích cỡ>])[As<kiểu>][,<tên mảng>([<kích cỡ>])
[As<kiểu>]...]

(Public|Private|Static)

Kích cỡ:

[<cận dưới> To] <cận trên> [[<cận dưới> To] <cận trên> ...]

Mảng ≤ 60 chiều

Ví dụ 4.9 :

Dim a(10), b(5,-2 to 6) As Integer

a: mảng variant có tối đa 10 hoặc 11 phần tử thuộc OptionBased là (1 hay 0)

b: mảng các số nguyên

1. Mảng động không khai báo kích cỡ

Mảng động không khai báo kích cỡ, khi sử dụng mới chỉ ra kích cỡ bằng lệnh Redim.

Redim[Preserve]<tên mảng>(<kích cỡ>) [As<kiểu>][,<tên mảng>(<kích cỡ>)[As<kiểu>]...]

Ví dụ 4.10 :

Dim a(10) As Integer

Dim a() As Integer

Redim a(5) As Integer

a(1) = 1

a(2) = 2

a(3) = 3

a(4) = 4

a(5) = 5

Redim a(6) As Integer

a(6) = 6

thì a(i) (i= 1..5) = 0, a(6) = 6

2. Hàm Array

Hàm array thường gán cho các biến Variant

Ví dụ 4.11 :

Dim a

a = Array ("một", "hai", "ba", "bốn")

a(1) = "một"

4.3.6. Các phép toán

1. Số học

- Đổi dấu (-), lũy thừa (^)
- Nhân (*), chia (/), chia lấy phần nguyên (\), mod (dư)
- Cộng (+), trừ (-)

2. Xâu ký tự

Ghép xâu và ghép các kiểu dữ liệu khác nhau

3. Ngày

Ngày - Ngày → Số

Ngày + Số $\rightarrow$ Ngày

Ngày - Số $\rightarrow$ Ngày

4. So sánh

Logic : Not, And, Or, Xor, Imp (kéo theo), Eqv (tương đương)

Cho biểu thức chứa nhiều loại phép toán mà không có dấu ngoặc thì thứ tự ưu tiên là :

- Số học - xâu - ngày
- So sánh
- Logic (NOT_AND_OR)

4.3.7. Các hàm mẫu

1. Hàm xâu ký tự

Space(n), Left(s,n), Right(s,n), Mid(s,i,n), Len(s), Instr(n,s,t, [c]),
Lease(s), Ucase(s), Rtrim(s), Trim(s), Str(x), Val(s)

Ví dụ 4.12 :

b = Val ("2457")

c = Val ("2 45 7")

d = Val ("2457 And 1396")

$\rightarrow b = c = d = 2457$

2. Hàm số học

Abs(x), Fix(x), Int(x), Rnd(n)

Ví dụ 4.13 :

Số ngẫu nhiên từ 1 $\rightarrow$ m: $\text{Int}(m * \text{Rnd} + 1)$

3. Hàm ngày giờ

Date, Time, Now, DateAdd(s,n,d), DateDiff(s,d1,d2), DateValue(s)
Day(d), Month(d), Year(d), Weekday(d)

Ví dụ 4.14 :

DatePart("yyyy",Date()) ' hàm DatePart : trả lại các phần của kiểu ngày

DateDiff("yyyy", [ngayBC], [ngaysinh]) ≥ 17 'Hàm DateDiff: cộng, trừ số với ngày

4. Hàm cho kiểu dữ liệu Variant

IsDate(x), IsNumeric, IsNull

5. Hàm liên quan nhiều kiểu dữ liệu

if(<điều kiện>, <giá trị 1>, <giá trị 2>)

Choose (<chỉ số>, <giá trị 1>, ... <giá trị n>)

Format (<biểu thức>, <khuôn dạng>)

6. Hàm InputBox và MsgBox

InputBox (<lời nhắc>[, <tiêu đề>][, <giá trị ngầm định>][, <toa độ x>, <toa độ y>])

Ví dụ 4.15: Dim mk

mk = InputBox("Cho biết" & Chr(10) & "Mat khau", "quan ly sinh vien")

Ví dụ 4.16:

```
Private Sub Form_Before DelConfirm(Cancel as Integer, Response As Integer)
```

```
If MsgBox("Dong y xoa", vbOkCancel + vbQuestion) = vbCancel then
```

```
Cancel = True
```

```
Else
```

```
Response = acDataErrContinue
```

```
EndIf
```

```
End Sub
```

4.4. CÁC CẤU TRÚC LẬP TRÌNH

4.4.1. Lệnh If

Kiểm tra điều kiện để rẽ nhánh ; ví dụ kiểm tra trị nhập vào hộp văn bản txtValue :

1. Viết trên một dòng

If <điều kiện> **Then** <nhóm lệnh 1> | [**Else** <nhóm lệnh 2>]

Theo ví dụ trên

If IsNull(Me!txtValue) Then MsgBox "You must enter a value" End If

2. Viết trên nhiều dòng

If <điều kiện> **Then**

<nhóm lệnh 1>

Else

[<nhóm lệnh 2>]

End If

Theo ví dụ trên

Private Sub cmdIf_Click()

If IsNull(Me!txtValue) Then

MsgBox "You must Enter a Value"

Else

MsgBox "You entered"&Me!txtValue

End If

End Sub

Ví dụ 4.17 : về If nhiều nhánh

Sub MultipleIfs(intNumber as Integer)

If intNumber = 1 Then

MsgBox "You entered a One"

ElseIf intNumber = 2

MsgBox "You entered a Two"

ElseIf intNumber >= 3 And intNumber <= 10 Then

MsgBox "You entered a Number Between 3 and 10"

ElseIf

MsgBox "You entered some other number"

End If

End Sub

<!/> Các điều kiện trong lệnh If được kiểm tra theo trật tự xuất hiện. Nên dùng Case thay vì If nhiều nhánh, trừ phi dùng TypeOf để kiểm tra kiểu một đối tượng khi đó bắt buộc phải dùng If

3. Hàm Immediate If (IIF)

Một biến thể của lệnh If. Ví dụ hàm đánh giá doanh thu

Function EvalSales(curSales As Currency) As String

EvalSales = IIf(curSales >= 100000, "Great Job", "keep Plugging")

End Function

Ví dụ 4.18 :

Hiển thị 0 cho một trị Null trên form

= IIf (IsNull(Forms!frmOrders!Freight),0,Forms!frmOrders!Freight)

4.4.2. Lệnh Select Case

Thay vì dùng If...Then...Else nhiều nhánh người ta dùng Case

Select Case <biểu thức>

[Case <danh sách giá trị 1>]

[<nhóm lệnh 1>]

....

[Case <danh sách giá trị n>]

[<nhóm lệnh n>]

[Case Else

[<nhóm lệnh n + 1>]]

End Select

Ví dụ 4.19 :

Hàm những ngày trong tuần

```
Public Function Thu(ngay As Date) As String
```

```
    Select Case Weekday (ngay)
```

```
        Case 1
```

```
            Thu = "Chu nhật"
```

```
        Case 2
```

```
            Thu = "Thu hai"
```

```
        Case
```

```
            Thu = "Thu ba"
```

```
        ...
```

```
    End Select
```

```
End Function
```

4.4.3. Lệnh For ... Next

```
For <biến> = <giá trị đầu> To <giá trị cuối> [Step<bước nhảy>]
```

```
<Các lệnh>
```

```
Next [<danh sách biến>]
```

Ví dụ 4.20 :

```
Dim b(5,5,5) As Integer, i As Integer, k As Integer, j As Integer
```

```
For i = 1 to 5
```

```
    For j = 1 to 5
```

```
        For k = 1 to 5
```

```
            a(i,j,k) = i+j+k
```

```
        Next k
```

```
    Next j
```

```
Next i
```

4.4.4. Lệnh Do ... Loop

* Dạng 1

```
Do[{ While|Until } <điều kiện>]  
[<Các lệnh>]  
Loop
```

* Dạng 2

```
Do  
[<các lệnh>]  
Loop [{ While|Until } <điều kiện>]
```

Ví dụ 4.21 :

* Do While ..Loop

```
Sub DoWhileLoop()  
Dim intCounter As Integer  
intCounter = 1  
Do While intCounter < 5  
    MsgBox intCounter  
    intCounter = intCounter + 1  
Loop  
End Sub
```

<!> Lặp không biết số lần lặp và thân vòng lặp có thể không được thực hiện lần nào.

* Do Loop ... While

```
Sub DoLoopWhile()  
Dim iCounter As Integer  
iCounter = 5  
Do  
    MsgBox iCounter  
    iCounter = iCounter + 1
```

Loop While iCounter < 5

End Sub

<!--> Lặp không biết số lần lặp và thân vòng lặp được thực hiện ít nhất một lần.

** Do Until ...Loop*

Sub DoUntilLoop ()

Dim intCounter As Integer

intCounter = 1

Do Until intCounter = 5

MsgBox intCounter

intCounter = intCounter + 1

Loop

End Sub

** Do ... LoopUntil*

Sub DoLoopUntil ()

Dim intCounter As Integer

intCounter = 1

Do

MsgBox intCounter

intCounter = intCounter + 1

Loop Until intCounter = 5

End Sub

4.4.5. Lệnh While

While <điều kiện>

[<các lệnh>]

Wend

4.4.6. Lệnh Goto

Thường dùng để nhảy đến nhãn thông báo lỗi

Ví dụ 4.22 :

Click nút chuyển về bản ghi trước

```
Private Sub Command17_click()
```

```
On Error Goto Err_Command17_Click
```

```
DoCmd.GotoRecord, acPrevious
```

```
Err_Command17_Click
```

```
MsgBox Err.Description
```

```
Exit Sub
```

```
End Sub
```

Đây là lỗi do đã ở bản ghi đầu mà vẫn click

On Error Resume Next (bỏ qua lỗi tiếp tục chạy lệnh tiếp)

4.4.7. Thực hiện hành động Macro trong chương trình – Đối tượng DoCmd

DoCmd.<phương thức><Danh sách tham số>

Để chạy các Macro từ VBA

Ví dụ: DoCmd.OpenReport strReportName, acPreview

Các Macro không có lệnh DoCmd tương ứng là: Addmenu, MsgBox, RunCode, Sendkeys, SetValue, StopAllMacros, StopMacro.

Ví dụ về một số phương thức của đối tượng DoCmd

* Close

DoCmd.Close [<kiểu đối tượng>],[<Tên đối tượng>],[<cách ghi>]

Chọn
↓
actable
|
acForm

Xâu ký tự
↓

Chọn
↓
acSaveYes
|
acSavePrompt

VD : DoCmd.Close acForm,"cập nhập", acSaveYes

* OpenForm

DoCmd.OpenForm . [<tên form>],[view], [<tên bộ lọc>],[<điều kiện Where>],[<Data Mode>],[<Window Mode>]

4.5. CÁC THAO TÁC TRÊN TỆP

Các thao tác trên tệp cho phép chuyển dữ liệu ra lưu trữ ở ngoài tệp mdb

– Mở tệp

Open <tên> For <cách thức> As [#]<số hiệu tệp>

– Đóng tệp

Close [#] <số hiệu vùng>

Reset

– Ghi lên tệp tuần tự

Write # <số hiệu vùng>, [<danh sách biểu thức>]

– Đọc từ tệp tuần tự

Input #<số hiệu tệp>,[<danh sách biến>]

* Các hàm dùng trong thao tác tệp

– Eof (<số hiệu vùng>)

– Lof (<số hiệu vùng>)

– Seck (<số hiệu vùng>)

4.6. CÂU HỎI ÔN TẬP – BÀI TẬP

1. Viết hàm để chuẩn hoá xâu tên nhập vào, nghĩa các chữ cái đầu viết hoa, giữa các từ chỉ cách nhau một dấu cách. Dùng hàm đó trong Form.

2. Tạo bảng, tạo trường nhờ đối tượng.

– Tạo bảng : CreateTabledef, ví dụ tạo bảng KHOA.

– Tạo trường : CreateField.

– Thêm trường vào bảng.

– Thêm bảng vào CSDL.

3. Xử lý sự kiện : Khi người sử dụng cần phải nhập MASV vào textbox. Các trường hợp xảy ra :

- Nếu để trống, hay không nhập mã thì cần đưa ra thông báo.
- Nhập mã không đúng cũng cần đưa ra thông báo cho người sử dụng.
- Nếu đúng mã thì hiển thị để người sử dụng quyết định
 - + Nếu có xử lý thì chuyển sang sao lưu rồi xoá.
 - + Nếu không xử lý thì bỏ qua.

Chương V

SỬ DỤNG ĐỐI TƯỢNG TRONG VBA

5.1. KHÁI NIỆM ĐỐI TƯỢNG – TẬP HỢP ĐỐI TƯỢNG

Access 2000 hỗ trợ VBA giúp cho việc phát triển ứng dụng theo hướng đối tượng được đơn giản hơn. Trong phần sau đây sẽ giới thiệu về phát triển ứng dụng theo hướng đối tượng trong khuôn khổ của VBA và Access 2000.

5.1.1. Tập hợp và đối tượng (Collections và Objects)

Access 2000 là môi trường phát triển ứng dụng hướng đối tượng. Cửa sổ Database của ACCESS cho phép người sử dụng truy cập tới các tables, queries, forms, reports, modules và các macros. VBA cũng cho phép điều này thông qua các đối tượng như recordsets và tableDef. Để tận dụng được các thế mạnh của VBA trong Access, cần phải hiểu về đối tượng cũng như các khái niệm liên quan.

Một đối tượng là một thực thể như ô tô, điện thoại, băng video... Các đối tượng đều có thuộc tính. Ví dụ, ô tô có các thuộc tính như màu sắc, cửa, các thông số kỹ thuật của động cơ và một số thuộc tính khác. Thuộc tính có thể xác định bản chất của đối tượng tạo thành. Hành vi hướng đối tượng này cho phép xác định một thực thể riêng biệt của đối tượng dựa trên các thuộc tính của chúng. Ví dụ một chiếc xe ô tô màu đỏ và một chiếc ô tô màu đen là hai thực thể riêng biệt của đối tượng ô tô.

Các thuộc tính của đối tượng phụ thuộc vào lớp đối tượng mà nó thuộc về. Một chiếc ô tô có một tập hợp các thuộc tính khác với một máy điện thoại. Cả hai đều có thuộc tính màu sắc, nhưng máy điện thoại có thuộc tính âm thanh. Mặt khác, xe ô tô lại có thuộc tính động cơ với công suất khác nhau. Một vài đối tượng đóng vai trò là kho chứa các đối tượng khác (containers). Các đối tượng kho cũng có các thuộc tính.

Bên cạnh các thuộc tính, đối tượng còn có các phương thức (methods). Phương thức là các hành vi mà đối tượng có thể tạo ra. Máy điện thoại có thể kết nối, xe ô tô có thể chạy... Nhiều đối tượng có hàng loạt các phương

thức khác nhau. Ví dụ điện thoại có thể cho phép bạn gọi nội hạt hoặc gọi đường dài.

Access không làm việc với các đối tượng vật lý. Chúng làm việc với các cấu trúc lập trình như forms, tables, queries và đại diện cho các đối tượng với các hành vi và thuộc tính. Cửa sổ Database window hiện các lớp đối tượng dữ liệu, nhấn chuột vào nút Forms để mở một khung nhìn về các đối tượng của form. Đối tượng Form có thể chứa các đối tượng khác gọi là các điều khiển (controls).

Tập hợp (Collections) cũng giống như đối tượng. Mọi tập hợp trong Access đều có thuộc tính *Count*, nó xác định số các thực thể (phần tử) trong collection. Collections cũng có thể có thuộc tính *Item*. Ta có thể sử dụng thuộc tính chỉ đọc *Item* để trả về một form cụ thể trong tập hợp *AllForms*. Vì một thành viên trong tập hợp là một đối tượng cụ thể nên nó không có thuộc tính *Count*.

5.1.2. Thuộc tính và phương thức (Properties và Methods)

Thuộc tính và phương thức xác định hình thái, hành vi của các đối tượng. Để truy cập đến phương thức hay thuộc tính của đối tượng ta dùng cú pháp : *object.property* hoặc *object.method*. Thuật ngữ *object* có thể hiểu là một đối tượng hay một tập hợp các đối tượng. Ví dụ, *txtInput1.BackColor* xác định màu nền của một hộp văn bản (text box) trên form, *AllForms.Item(0)* tham chiếu đến form đầu tiên trong tập hợp các forms. Nếu form này có tên là *frmSample1*, có thể dùng *AllForms.Item("frmSample1")*.

Để xem các thuộc tính của đối tượng, ta chọn đối tượng trong cửa sổ thiết kế rồi nhấn-chuột vào nút Properties trên Toolbar.

Đối tượng *DoCmd* là đối tượng có nhiều phương thức nhất của Access vì vậy nó đặc biệt hữu dụng để ta bắt đầu với các phương thức. *DoCmd* có các phương thức như *Close*, *OpenForm*, *GoToControl*, *FindRecord* và *RunCommand* (đã học trong chương trước). Nhiều phương thức của *DoCmd* đòi hỏi có tham số để xác định cách thức hoạt động của phương thức. Một số phương thức khác đòi hỏi các tham số tùy chọn (optional arguments). Nếu không xác định tham số cụ thể cho một optional argument, phương thức sẽ dùng các giá trị ngầm định.

Để đóng một form trong Access ta sử dụng phương thức *close* của đối tượng *DoCmd*. Phương thức này đòi hỏi 2 tham số bắt buộc và một tham số

tùy chọn. Tham số thứ nhất xác định kiểu đối tượng cần đóng. Nếu muốn đóng một form, sử dụng *acForm* (*acForm* là một hằng chứa sẵn của Access). Tham số thứ hai là tên form. Tham số tùy chọn báo cho Access có lưu những thay đổi trên form hay không (ngầm định là *acSaveYes*, ngược lại chọn *acSaveNo*).

Ví dụ 5.1 :

`DoCmd.Close acForm, "formname", acSaveNo`

Nhiều phương thức của *DoCmd* được áp dụng trực tiếp cho từng đối tượng cụ thể. Ví dụ, phương thức *GoToControl* gán điều khiển (focus) cho một control cụ thể trên form, ta cũng có thể dùng phương thức *SetFocus* để làm việc đó.

5.1.3. Sự kiện (Events)

Sự kiện là khái niệm vô cùng quan trọng trong lập trình VBA. Ta có thể dùng các sự kiện để làm cho ứng dụng trở nên linh hoạt. Các đối tượng và các tập hợp có những sự kiện được dùng để đưa mã lệnh của lập trình viên vào ứng dụng. Khi làm việc với các forms, có thể dùng các events cho những việc như : kiểm tra tính đúng đắn của dữ liệu, bật hay tắt các controls, thay đổi control đang có focus, mở form, và đóng form...

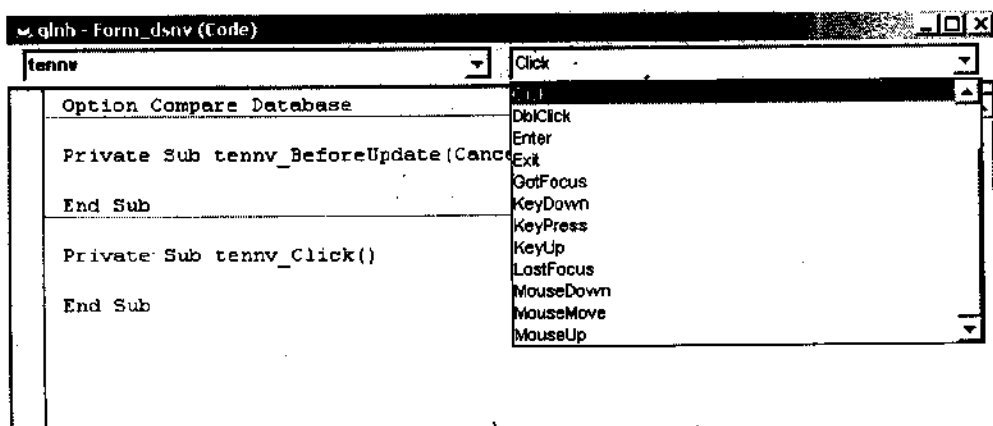
Người sử dụng cần phải biết khi nào xảy ra sự kiện cũng như trình tự xảy ra của các sự kiện. Khi mở một form, nó sẽ gây ra hàng loạt sự kiện : *Open*, *Load*, *Resize* và *Current*. Sự kiện *Open* xảy ra khi form bắt đầu mở nhưng trước khi các bản ghi được hiện ra. Sự kiện *Load* phát sinh sau sự kiện *Open*. Sau sự kiện *Load*, các bản ghi trên form sẽ hiện ra. Bất cứ đoạn mã lệnh nào làm thay đổi kích thước hay vị trí của form như các phương thức *MoveSize*, *Minimize*, *Maximize*, hay *Restore* của đối tượng *DoCmd* sẽ làm xảy ra sự kiện *Resize*. Sự kiện *Current* là sự kiện cuối cùng xảy ra sau khi một form được mở. Nó cũng xảy ra khi người sử dụng định vị tới bản ghi mới hoặc làm tươi lại form.

Cách tiếp cận các sự kiện của forms và các controls của chúng :

1. Chọn form hay control trong cửa sổ thiết kế (Design view).
2. Nhấn phím phải chuột – chọn Build Event từ menu ngữ cảnh để mở hộp thoại Build Event.

3. Chọn tiếp Code Builder để mở một thủ tục sự kiện trong VBA. Có thể lựa chọn các sự kiện cần xử lý trên trong hộp combo ở góc phải trên cửa sổ code.

Ví dụ về một sự kiện trên form được trình bày trên hình 5.1.



Hình 5.1. Ví dụ về một sự kiện trên form.

5.2. CÁC ĐỐI TƯỢNG CỦA ACCESS

Các đối tượng của một ứng dụng Access

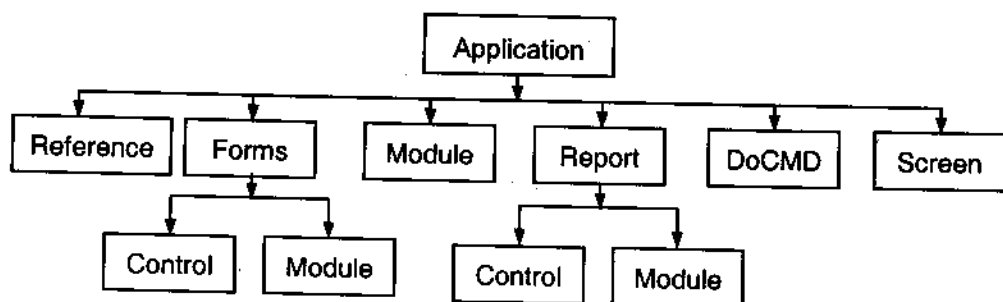
Một ứng dụng MS.Access gồm các đối tượng (Objects) và các tập hợp sau :

KIỂU ĐỐI TƯỢNG	Ý NGHĨA
APPLICATION	Ứng dụng
CONTROL	Đối tượng điều khiển trong Form hay Report
FORM	Form hoặc SubForm
REPORT	Báo cáo
FORMS COLLECTION	Tập hợp các đối tượng Forms đang mở
REPORT COLLECTION	Tập hợp các Reports hiện hành
CONTROLS COLLECTION	Tập hợp các Controls trên Form hay Report hiện hành
MODUL	Modul chuẩn hoặc một lớp chuẩn
MODUL COLLECTION	Tập hợp các moduls đang mở

REFERENCE	Tham chiếu đến một thư viện đối tượng
REFERENCE COLLECTION	Tập hợp các tham chiếu hiện đã được thiết lập
DOCMD	Đối tượng DoCMD
SCREEN	Mô tả cách sắp xếp của các đối tượng hiện có trên màn hình

Ba kiểu đối tượng Table, Dynaset và Snapshot còn có tên chung là đối tượng Recordset.

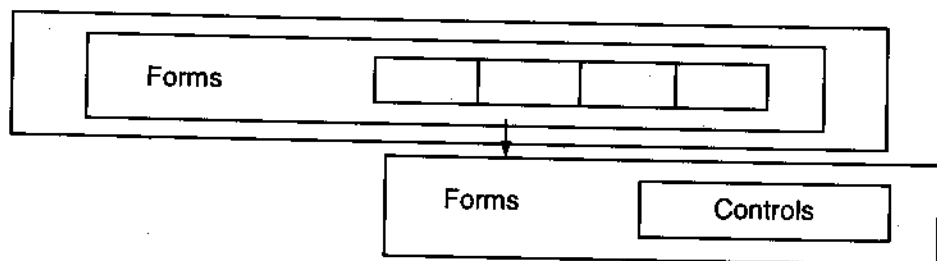
Các đối tượng trên được tổ chức theo mô hình phân cấp (hình 5.2) :



Hình 5.2. Mô hình phân cấp các đối tượng.

5.2.1. Đối tượng Form và quan hệ phân cấp

Một đối tượng Form tham chiếu tới một biểu mẫu cụ thể trong ACCESS. Quan hệ phân cấp của đối tượng Form được mô tả như hình 5.3.



Hình 5.3. Mô hình phân cấp của đối tượng Form.

Một đối tượng Form là thành phần của một tập hợp các Forms. Tập hợp này do Ms.Access tự tạo ra để quản lý tất cả các biểu mẫu. Các thành phần trong tập hợp được đánh số từ chỉ số 0 theo thứ tự được tạo.

Có thể tham chiếu tới một biểu mẫu cụ thể trong tập hợp các biểu mẫu (Forms Collection) bằng tên biểu mẫu hoặc bằng chỉ số của nó trong danh sách. Tuy nhiên nên dùng tên Form sẽ đảm bảo hơn, vì chỉ số có thể thay đổi. Nếu tên Form chứa ký tự trống (blank) thì phải đặt tên trong dấu ngoặc vuông ([]).

Cú pháp

Ví dụ 5.2 :

Forms!formname

Forms!Capnhat

Forms![form name]

Forms![Cap nhât]

Forms("formname")

Forms("Capnhat")

Forms(Index) .

Forms(0)

Mỗi đối tượng Form quản lý một danh sách chứa tất cả các Controls trong form.

Có thể tham chiếu đến các thành phần (controls) trong Form bằng cách tham chiếu tường minh hoặc ngầm định : Ví dụ tham chiếu tới điều khiển Hoten trong form Capnhat

Forms!Capnhat.Controls!Hoten

Hoặc : Forms!Capnhat!Hoten

Ví dụ 5.3 :

Cập nhật đến điều khiển có tên TenVattu trong một SubForm có tên SubCtietCtu thuộc Form Hoadon

Forms!Hoadon.SubCtietCtu.Form.Controls!TenVattu

Forms!Hoadon.SubCtietCtu.Form!TenVattu

Mỗi đối tượng đều có các thuộc tính và phương thức của mình. Chúng bao gồm tất cả các thuộc tính liệt kê trong bảng thuộc tính của đối tượng.

Ví dụ 5.4 :

Sử dụng thuộc tính Count của tập hợp Forms để tính số forms hiện thời đang mở và in tên của từng Forms :

Sub AllOpenForms()

Dim X as integer, N as integer

N = Forms.Count

Debug.Print " Có ";" N;" Form đang mở"

For X = 0 to N - 1

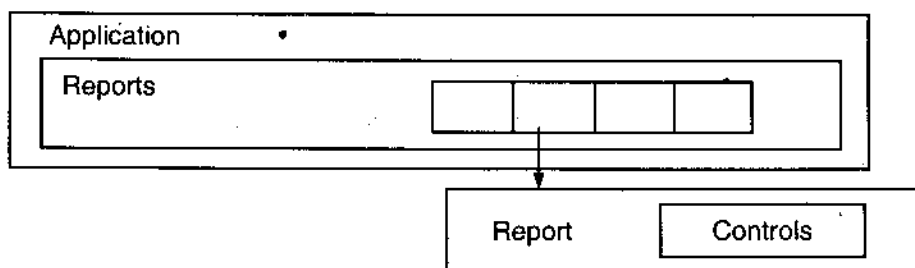
 Debug.Print Forms(X).Name

Next X

End Sub

5.2.2. Báo cáo (Reports) và mối quan hệ phân cấp

Tương tự Form, Report cũng là một đối tượng với quan hệ phân cấp được mô tả như hình 5.4 :



Hình 5.4. Quan hệ phân cấp được mô tả của Report.

Một đối tượng Report là thành phần của một tập hợp các Reports. Tập hợp này do Ms.Access tự tạo ra để quản lý tất cả các biểu mẫu. Các thành phần trong tập hợp được đánh số từ chỉ số 0 theo thứ tự được tạo.

Có thể tham chiếu tới một biểu mẫu cụ thể trong tập hợp các biểu mẫu (Reports Collection) hoặc bằng tên biểu mẫu, hoặc bằng chỉ số của nó trong danh sách. Tuy nhiên nên dùng tên Report sẽ đảm bảo hơn, vì chỉ số có thể thay đổi. Nếu tên Report chứa ký tự trống (blank) thì phải đặt tên trong dấu ngoặc vuông ([]).

Cú pháp

Reports!Reportname

Reports![Report name]

Reports("Reportname")

Reports(Index)

Ví dụ 5.5 :

Reports!Danhsach

Reports![Danh sach]

Reports("danhsach")

Reports(0)

Mỗi đối tượng Report quản lý một danh sách chứa tất cả các Controls trong Report. Có thể tham chiếu đến các thành phần (controls) trong Report bằng cách tham chiếu tương minh hoặc ngầm định.

Ví dụ 5.6 :

Tham chiếu tới điều khiển Hoten trong Report Capnhat

Reports!Capnhat.Controls!Hoten

Hoặc : Reports!Capnhat!Hoten

5.3. ĐỐI TƯỢNG DỮ LIỆU DAO (Data Access Object)

DAO (Data Access Objects) : DAO là kỹ thuật được bảo mật của Microsoft chỉ để sử dụng với Jet Database Engine. DAO rất dễ dùng, hiệu năng và tiện dụng nhưng bị giới hạn trong phạm vi MS Access. Mặc dù vậy, DAO vẫn được sử dụng phổ biến vì có những lợi ích thực tiễn. ACCESS 2000 hỗ trợ DAO 3.6 Jet Engine (Với Access 97 là DAO 3.5).

Bên cạnh kỹ thuật truy cập dữ liệu DAO còn có các kỹ thuật khác như ODBC, RDO, ADO...

ODBC (Open Database Connectivity) : ODBC cho phép người sử dụng nối với các databases mà chỉ dùng một phương pháp duy nhất. Điều này sẽ làm giảm bớt gánh nặng cho lập trình viên, vì khi đó chỉ cần học một kỹ thuật lập trình duy nhất là có thể làm việc với bất cứ loại database nào. Đặc biệt là khi cần phải thay đổi loại database, như nâng cấp từ Access lên SQLServer chẳng hạn, thì sự sửa đổi về mã rất ít. Khi dùng ODBC chung với DAO, ta có thể cho Access Database nối với các databases khác. Tuy nhiên nhược điểm của ODBC là rất rắc rối, phức tạp.

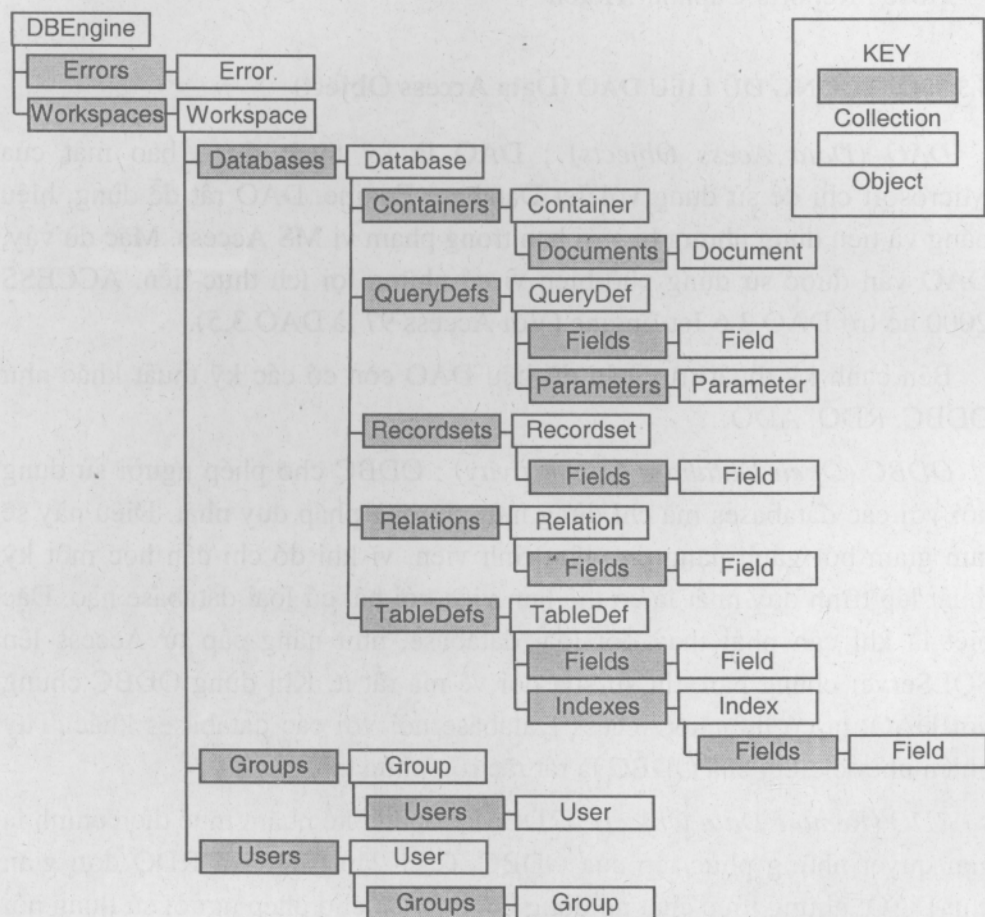
RDO (Remote Data Object) : RDO được thiết kế nhằm mục đích chính là giải quyết những phức tạp của ODBC. Cách lập trình với RDO đơn giản như DAO, nhưng thực chất nó dùng ODBC nên cho phép người sử dụng nối với nhiều databases. Tuy nhiên, RDO không được thịnh hành.

Hiện nay Microsoft vẫn tiếp tục hỗ trợ các kỹ thuật nói trên, ngoài ra còn đưa thêm một kỹ thuật truy cập database mới rất quan trọng, đó là *ADO (ActiveX Data Objects)* (kỹ thuật ADO sẽ được trình bày chi tiết ở phần sau).

5.3.1. Kỹ thuật truy cập Database DAO

Trọng tâm của MS Access và một phần quan trọng của VB là bộ máy cơ sở dữ liệu Microsoft Jet. Các tính năng quản trị cơ sở dữ liệu của Access

xuất phát từ bộ máy Jet Engine. Trong mô hình truy cập Database DAO đối tượng DBEngine là đối tượng ở mức cao nhất trong mô hình các đối tượng truy xuất dữ liệu. Đối tượng DBEngine quản lý tất cả các đối tượng truy xuất dữ liệu khác trong quan hệ phân cấp (hình 5.5).



Hình 5.5. Quan hệ phân cấp.

Đối tượng DBEngine do Ms.Access tự tạo và người sử dụng không thể tạo thêm các đối tượng DBEngine. Nó không là thành phần của bất kỳ tập hợp nào.

Tất cả các đối tượng thuộc loại tập hợp quản lý các phần tử của mình dưới dạng mảng một chiều với phần tử đầu tiên bắt đầu bằng chỉ số 0. Việc gán chỉ số cho phần tử của tập hợp phụ thuộc thứ tự xuất hiện lần đầu của chúng trong cơ sở dữ liệu.

Một đối tượng Workspace xác định một vùng làm việc trong một khoảng thời gian xác định (thường là một phiên làm việc login – logout của người sử dụng).

Một đối tượng Database xác định một cơ sở dữ liệu đang mở.

Một đối tượng RecordSet xác định các bản ghi của một bảng hoặc của một câu truy vấn. RecordSet quản lý một danh sách các trường (Fields) của nó.

Đối tượng TableDef định nghĩa việc cài đặt một bảng. Mỗi đối tượng TableDef quản lý danh sách các trường và danh sách các chỉ mục.

Các đối tượng Container (Kho chứa) ghi lại các thông tin về một cơ sở dữ liệu và mỗi loại đối tượng mà nó chứa : Tất cả các bảng, truy vấn, quan hệ, các biểu mẫu, báo cáo, các macro cũng như các modules chương trình.

QueryDef định nghĩa việc cài đặt các truy vấn trong một cơ sở dữ liệu. Một QueryDef quản lý một danh sách các tham số và một danh sách các trường của nó

Đối tượng Relation định nghĩa quan hệ giữa hai trường trong các bảng hoặc các truy vấn. Quan hệ có thể là một – một hoặc một – nhiều.

Khi người sử dụng mở một cơ sở dữ liệu đầu tiên thì Access tự động tạo đối tượng Workspaces (0) cho pha làm việc đầu tiên của người sử dụng và đối tượng Databases (0) cho cơ sở dữ liệu vừa mở.

5.3.2. Biểu thị đối tượng và các thành phần của đối tượng

Đặt tên cho đối tượng : Tên của đối tượng có thể dài đến 64 ký tự, bắt đầu bằng một chữ cái (A – Z) hoặc một chữ số (0 – 9). Trong tên có thể có dấu trắng (Space), nhưng không được chứa dấu chấm (.). Để tránh nhầm lẫn, lập trình viên thường đặt tên đối tượng trong dấu ngoặc vuông ([]).

Chỉ định một đối tượng và các thành phần của nó : Để tránh không trùng tên với các đối tượng khác trong cơ sở dữ liệu, một đối tượng cũng như các thành phần của nó cần được xác định chính xác theo quy ước sau :

Đối tượng và thành phần của nó phải được chỉ ra trong mỗi quan hệ phân cấp.

Sử dụng dấu chấm than (!) đặt trước đối tượng do người sử dụng tạo ra. Đặt dấu chấm (.) trước đối tượng do MS.Access tạo ra. Có thể dùng

biểu thức chuỗi chứa tên đối tượng đặt bên trong cặp dấu ngoặc tròn () thay cho dấu chấm than và tên đối tượng. Ví dụ :

ĐỀ CHỈ ĐỊNH	CÁCH DÙNG
Thuộc tính của Form	Forms!FormName.PropertyName <i>Ví dụ 5.7 : Forms!FrmMain.Caption</i>
Thuộc tính của một đối tượng trong Form	Forms!FormName!ObjName.PropertyName <i>Ví dụ 5.8 : Forms!FrmMain!TxtHoten.Text</i>
CSDL đầu tiên Phương thức cho Databases (0)	DBEngine.Workspaces(0).Databases(0) <i>Ví dụ 5.9 :</i> DBEngine.Workspaces(0).Databases(0).OpenRecordset ("Hoadon", DB_Open_Table)

Trong thực tế do DBEngine, workspaces(0) và Databases(0) là ngầm định cho CSDL đầu tiên, nên thường được viết tắt :

OpenRecordset("Hoadon",DB_open_Table)

5.3.3. Biến đối tượng

– Biến đối tượng cần 4 bytes (32 bit) bộ nhớ để tham chiếu tới một đối tượng cụ thể nào đó. Để khai báo đối tượng ta dùng lệnh DIM như sau :

Dim Tênbiến AS TênKiểuĐốiTượng

Ví dụ 5.10 :

Dim DB1 as Database

Dim RS1 As RecordSet

– Để gán tham chiếu đối tượng cho một biến ta phải dùng lệnh SET với cú pháp như sau : Set Obj_Var = {Obj_expression | Nothing }

+ Với Obj_Var : Tên của biến đối tượng

+ Obj_Expression : Biểu thức chứa tên của đối tượng hoặc một hàm hay một phương thức trả về một đối tượng cùng kiểu với biến đối tượng

+ Nothing : Giải phóng biến đối tượng

5.4. ĐỐI TƯỢNG DỮ LIỆU ADO

Access 2000 cung cấp ADO version 2.1, trong đó chứa 3 mô hình truy cập dữ liệu ADO : Thư viện ADODB, thư viện ADOX và thư viện JRO. Bằng cách chia thành 3 thư viện, Access cho phép lựa chọn thư viện phù hợp nhất với ứng dụng của người sử dụng. Một thành phần khác cực kỳ quan trọng trong chiến lược truy cập dữ liệu của Access 2000 là mối quan hệ với các nhà cung cấp CSDL : OLE DB providers, nó cho phép làm việc với ADO để truy cập tới các nguồn dữ liệu truyền thống cũng như các nguồn dữ liệu mới, ví dụ thư viện thư điện tử e-mail. Điều này làm tăng rất nhiều sức mạnh của lập trình CSDL.

Thư viện ADODB là thư viện gọn nhẹ nhất, trong đó chứa đựng các đối tượng cơ bản và cung cấp các công cụ cơ bản để kết nối CSDL, tạo các lệnh và truy tìm các bản ghi, đồng thời cho phép định vị các bản ghi. Có thể sử dụng ADODB để tạo những chức năng căn bản như cập nhật, sửa, thêm, xóa bản ghi. Mô hình phi phân cấp của thư viện này làm cho người mới học cảm thấy không phức tạp như mô hình DAO.

Thư viện ADOX chứa đựng ngôn ngữ định nghĩa dữ liệu DDL và hệ thống bảo mật, an toàn thông tin. Nó cung cấp các objects cho phép người sử dụng tiếp cận một góc cạnh của CSDL, ví dụ, cho phép tạo các bảng và tạo quan hệ giữa chúng. Mô hình này cũng cho phép ràng buộc toàn vẹn dữ liệu và cung cấp thủ tục, khung nhìn cũng như tập hợp *Users* và *Groups* để dùng chung dữ liệu và bảo mật.

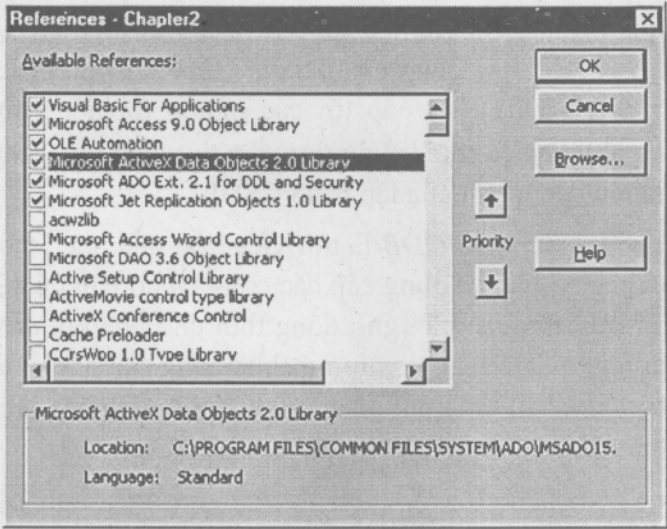
Thư viện JRO cho phép làm việc với cả Jet Engine và cơ sở dữ liệu SQL Server.

Ưu điểm chính của ADO là hỗ trợ mô hình sự kiện ; ODBCDirect cho phép các phép toán đồng bộ (asynchronous operations) còn ADO cung cấp các sự kiện và người sử dụng chỉ đơn thuần tạo các thủ tục để trả lời cho các sự kiện xảy ra.

OLE DB providers làm cho ADO trở lên mạnh hơn. Chúng cung cấp một phương pháp mới để truy cập CSDL từ xa và cho phép truy cập cả CSDL quan hệ lẫn CSDL phi quan hệ thông qua một giao diện ADO duy nhất. Access 2000 có nhiều OLE DB providers, bao gồm Jet, SQL Server, Oracle, các nguồn dữ liệu ODBC chính thống cũng như các nguồn không chính thống như Microsoft Active Directory Service và Microsoft Index

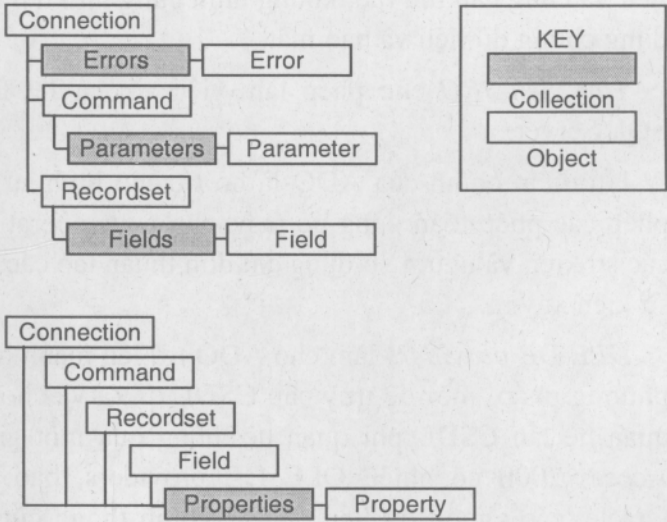
Server. Cùng với thời gian, hy vọng rằng sẽ xuất hiện thêm nhiều nhà cung cấp CSDL khác.

Trước khi dùng một thư viện ADO, cần phải tạo tham chiếu tới ít nhất một trong số các thư viện đó. Ta có thể thực hiện việc này từ cửa sổ Visual Basic Editor (VBE) dùng lệnh Tools – References. Hình 5.6 thể hiện hộp thoại tham chiếu (References dialog box) chọn cả 3 thư viện. Việc chọn cả 3 thư viện làm hao tốn tài nguyên, vì vậy ta chỉ cần chọn chính xác thư viện cần sử dụng. Tuy nhiên khi xây dựng một ứng dụng chạy trên nhiều loại máy khác nhau thì cần chú ý đến nguồn tài nguyên này.



Hình 5.6. Thư viện ADODB.

Thư viện đối tượng ADODB có 7 đối tượng chính. Bốn trong số đó có tập hợp. Đối tượng *Connection* (hình 5.7) nằm ở mức cao nhất của hệ phân cấp, nhưng ta có thể tạo liên kết ẩn trong các đối tượng khác. Các đối tượng *Connection*, *Command*, *Recordset* và *Field* có tập hợp các thuộc tính.



Hình 5.7. Đối tượng Connection

5.4.1. Đối tượng Connection

Đối tượng connection dùng để xác lập kết nối với CSDL. Có thể sử dụng đối tượng *Connection* hoặc là tường minh hoặc là ẩn (không tường minh) khi làm việc với CSDL. Khi tạo kết nối tường minh, có thể xử lý một cách hiệu quả một hoặc nhiều kết nối và gán lại vai trò mà chúng phục vụ trong ứng dụng của người sử dụng. Bằng cách kết nối không tường minh, ta có thể làm cho mã lệnh ngắn hơn. Việc khai báo không tường minh đòi hỏi nhiều tài nguyên hơn, nhưng với ứng dụng chỉ dùng một hoặc hai kết nối thì kiểu kết nối này là lựa chọn tốt nhất.

Khác với DAO, ADO là ngôn ngữ truy cập CSDL tổng hợp, vì vậy không phải mọi thuộc tính và phương thức của nó phù hợp với Jet engine. Tuy vậy trong Access 2000 có một OLE DB provider đặc biệt cho Jet 4, đó là version mới nhất của Jet đi kèm Access 2000. Do đó đối tượng *Connection* phụ thuộc hoàn toàn vào các yêu cầu của nhà cung cấp nên khả năng các tham số của *Connection* tham chiếu vào Jet 4 provider là khác nhau. Khi tham chiếu đến CSDL thuộc loại khác, có thể phải thêm tham số *Data Source* để chỉ tới một vị trí vật lý của CSDL không nằm trong dự án hiện tại.

Ví dụ 5.11 :

Đoạn code sau đây mở CSDL mẫu Northwind có trong MS Office. Chú ý rằng lệnh *Dim* khai báo và tạo một tham chiếu đến *cnnNorthwind* như một đối tượng *Connection*. Dùng phương thức *Open* với *cnnNorthwind* để mở CSDL. Cũng cần chú ý là các tham số *Provider* và *Data Source* được đặt trong cặp “ ”. Tham số *Provider* trỏ tới Jet 4 OLE DB provider và tham số *Data Source* trỏ tới vị trí vật lý của CSDL Northwind.

```
Sub OpenMyDB()
```

```
Dim cnnNorthwind As New Connection
```

```
Dim rsCustomers As Recordset
```

```
'Create the connection.
```

```
cnnNorthwind.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
    "Data Source=C:\Program Files\Microsoft Office\Office\" & _  
    "Samples\Northwind.mdb;"
```

'Create recordset reference and set its properties.

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.CursorType = adOpenKeyset
```

```
rsCustomers.LockType = adLockOptimistic
```

'Open recordset and print a test record.

```
rsCustomers.Open "Customers", cnnNorthwind, , , adCmdTable
```

```
Debug.Print rsCustomers.Fields(0).Value, rsCustomers.Fields(1).Value
```

```
rsCustomers.Close
```

```
cnnNorthwind.Close
```

End Sub

Sau khi tạo tham chiếu để kết nối, chương trình tạo đối tượng *Recordset*. Nó xác lập tham chiếu đến biến đối tượng gắn với recordset rồi gán các thuộc tính cho recordset. Khối lệnh cuối cùng mở recordset và in các trường của bản ghi đầu tiên. Phương thức *Open* cho đối tượng *Recordset* có thể tham chiếu một kết nối đến CSDL và một số bản ghi của CSDL. Đoạn lệnh trên chọn tất cả các bản ghi của bảng *Customers* trong CSDL Northwind. Phương thức *Open* làm cho bản ghi đầu tiên có tiếp cận với ứng dụng.

Hai dòng cuối cùng đóng recordset rồi đóng kết nối và làm cho các đối tượng mà nó tham chiếu không hoạt động được nữa, nếu tiếp tục thao tác trên chúng sẽ gây lỗi thực hiện run – time error.

Đoạn lệnh sau cũng mở một recordset từ bảng *Customers* trong CSDL Northwind và in bản ghi đầu tiên, nhưng ngắn gọn và đơn giản hơn vì nó tạo một kết nối không tường minh và chấp nhận các tham số ngầm định.

```
Sub OpenFast()
```

```
Dim rsCustomers As Recordset
```

```
Set rsCustomers = New ADODB.Recordset
```

'Less code, but potentially greater resource consumption

```
rsCustomers.Open "customers", "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Program Files\Microsoft Office\Office\" & _
    "Samples\Northwind.mdb;"
Debug.Print rsCustomers.Fields(0), rsCustomers.Fields(1)
rsCustomers.Close
```

End Sub

Vì không có các kết nối tường minh, thủ tục *OpenFast* không cần khai báo đối tượng *connection* (và vì vậy không cần đóng hay mở bằng *open* và *close*). Phương thức *Open* một đối tượng *recordset* có thể thêm các thông tin cần thiết về nhà cung cấp và nguồn CSDL cho việc kết nối. Đoạn lệnh trên chỉ dùng một tham số khác là nguồn *recordset* (bảng *Customers*). Phương thức *Open* gán giá trị ngầm định về kiểu bản ghi và kiểu khoá (*CursorType* và *LockType*) là *chỉ đi tới* (forward – only) và *chỉ đọc* (read – only). Những thiết lập này cho phép xử lý dữ liệu rất nhanh nhưng bị hạn chế về chức năng. Nếu chỉ quan tâm đến việc xử lý dữ liệu thì đây là lựa chọn tối ưu nhất.

1. Thuộc tính Mode

Thuộc tính Mode theo ngầm định phương thức *Open* của đối tượng *Connection* tạo một CSDL để dùng chung. Nhưng có thể xác lập thuộc tính này là 1 trong 7 giá trị, cho phép các kiểu truy nhập khác nhau. Đoạn code sau chỉ ra hiệu ứng của việc xác lập thuộc tính *mode* là chỉ đọc với khả năng cập nhật các bản ghi (sẽ có lỗi Run – time).

```
Sub OpenLookOnly()
```

```
Dim cnn1 As New Connection
```

```
Dim rsCustomers As Recordset
```

```
'cnn1.Mode = adModeRead
```

```
cnn1.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office\Office\" & _
```

```
"Samples\Northwind.mdb;"
```

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.Open "Customers", cnn1, adOpenKeyset, _  
    adLockPessimistic
```

'An adModeRead setting for cnn1.Mode causes an error in this procedure.

'Remove the comment from the cnn1.Mode line to see an error here.

```
rsCustomers.Fields("CustomerID") = "xxxxxx"
```

```
rsCustomers.Update
```

```
Debug.Print rsCustomers.Fields("CustomerID")
```

```
rsCustomers.Close
```

End Sub

Sau đây là bảng các giá trị của thuộc tính *Mode* của đối tượng *connection*.

HÀNG SỐ	GIÁ TRỊ	HÀNH VI
<i>adModeUnknown</i>	0	Không xác định
<i>adModeRead</i>	1	Chỉ đọc (Read only)
<i>adModeWrite</i>	2	Chỉ ghi (Write only)
<i>adModeReadWrite</i>	3	Cho phép đọc/ghi (Read/write)
<i>adModeShareDenyRead</i>	4	Khoá không cho người khác mở để đọc
<i>adModeShareDenyWrite</i>	8	Khoá không cho người khác mở để ghi
<i>adModeShareExclusive</i>	12	Không cho người khác mở
<i>adModeShareDenyNone</i>	16	Dùng chung Shared access (default)

2. Phương thức *OpenSchema*

Phương thức *OpenSchema* của đối tượng *Connection* cho phép liệt kê (browse) các đối tượng trong kết nối, cung cấp các thông tin về các bảng, các truy vấn, thủ tục, chỉ mục... của CSDL được kết nối. Ví dụ sau dùng

phương thức *OpenSchema* để liệt kê danh sách các bảng truy vấn (View) của CSDL đang kết nối.

```
Public Sub OpenSchemaX()
```

```
Dim cnn1 As New ADODB.Connection
```

```
Dim rstSchema As ADODB.Recordset
```

```
cnn1.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office\Office\" & _
```

```
"Samples\Northwind.mdb;"
```

```
Set rstSchema = cnn1.OpenSchema(adSchemaTables)
```

```
'Print just views; other selection criteria include
```

```
'TABLE, ACCESS TABLE, and SYSTEM TABLE.
```

```
Do Until rstSchema.EOF
```

```
    If rstSchema.Fields("TABLE_TYPE") = "VIEW" Then
```

```
        Debug.Print "View name: " & _
```

```
        RstSchema.Fields("TABLE_NAME") & vbCrLf
```

```
    End If
```

```
    rstSchema.MoveNext
```

```
Loop
```

```
rstSchema.Close
```

```
cnn1.Close
```

```
End Sub
```

5.4.2. Đối tượng Recordset

Tập các bản ghi recordset là kiến trúc cơ bản để làm việc với các bản ghi của CSDL. Ta có thể dựa vào các bảng hay truy vấn trong dự án hay trong một tệp khác, một câu lệnh SQL, hay một lệnh trả về các bản ghi. Các

thao tác trên recordset phụ thuộc vào nhà cung cấp OLE DB provider và bản chất của nguồn dữ liệu.

Có thể tiến hành nhiều thao tác trên recordsets : Định vị bản ghi, in một phần hay toàn bộ nội dung của chúng ; thêm, sửa, xoá các bản ghi ; tìm kiếm, lọc các bản ghi ...

1. Thuộc tính *ActiveConnection*

Thuộc tính *ActiveConnection* để tạo một kết nối tới một tập các bản ghi. Ta có thể thiết lập thuộc tính này tại bất kỳ thời gian nào sau khi thiết lập đối tượng là một recordset. Sử dụng thuộc tính này sẽ làm đơn giản hoá câu lệnh sử dụng phương thức *Open* của đối tượng recordset vì nó không cần dùng đến thông tin kết nối. Khi thiết lập sẵn thuộc tính này, không cần tham chiếu tới một kết nối đã có trong phương thức *Open*.

2. Phương thức *Open*

Phương thức *Open* của đối tượng recordset là một cách phổ biến để mở ra một recordset dùng trong thủ tục. Đối số nguồn là yêu cầu chủ yếu cho phương thức này. Nó chỉ định nguồn dữ liệu trên đó mà phương thức tác động lên. Thông thường, tùy chọn cho đối số nguồn có thể là một bảng, một câu lệnh SQL, một tệp recordset, hay một stored procedure. Sử dụng đối số *Options* của phương thức *Open* để chỉ ra kiểu nguồn khi mở một recordset.

3. Kiểu *cursor*

Kiểu *cursor* là một trong những các đặc điểm cơ bản nhất của một recordset. Nó quyết định kiểu định hướng thông qua recordset và các kiểu khoá mà ta có thể tác động lên. ADO hỗ trợ 4 kiểu con trỏ :

– *Dynamic* : Sử dụng kiểu con trỏ này người sử dụng có thể xem sự thay đổi diễn ra đối với nguồn dữ liệu tạo bởi người sử dụng khác. Tức là có thể thực hiện các chức năng thêm, thay đổi hay xoá các bản ghi đối với recordset và cũng cho phép định hướng hai chiều trong cơ sở dữ liệu mà không cần dựa vào chỉ dẫn (*bookmarks*).

– *Keyset* : Kiểu con trỏ này giống kiểu *dynamic cursor* ngoại trừ việc không thể truy cập vào sự thay đổi nguồn dữ liệu được thực hiện bởi người sử dụng khác. Một cách để xem sự thay đổi này là gọi phương thức *Requery* của đối tượng recordset.

– *Static* : Kiểu con trỏ này là một ảnh chụp (snapshot) của một recordset tại một thời điểm. Nó cho phép định hướng theo hai chiều. Sự thay đổi cơ sở dữ liệu của những người sử dụng khác là không thể nhìn thấy được.

– *Forward – only* : Con trỏ này chỉ định hướng tiến trong cơ sở dữ liệu nhưng có tốc độ thực hiện nhanh hơn các con trỏ khác. Đây là con trỏ mặc định của ADO. Nếu cần một kiểu con trỏ khác, phải thiết lập thuộc tính *CursorType* trước khi mở một Recordset.

Chú ý : Việc thiết lập kiểu con trỏ và thiết lập kiểu khóa có sự tương tác qua lại với nhau. Nếu để kiểu con trỏ là *forward – only* và kiểu khóa là *read – only* (*adLockReadOnly*), ADO sẽ bỏ qua việc thiết lập kiểu con trỏ của người sử dụng. Ví dụ, ADO sẽ tự động chuyển kiểu con trỏ từ *forward – only* thành *keyset* nếu để kiểu khóa là *optimistic*.

4. Thuộc tính LockType

Thuộc tính LockType tương tác với loại con trỏ vì nó điều khiển khả năng thao tác của người sử dụng đối với recordset. Một kiểu thiết lập khóa *adLockReadOnly* sẽ xác định cụ thể kiểu con trỏ tương ứng là *forward – only*. Đây là kiểu khóa mặc định. Bảng dưới đây chỉ ra bốn cách thiết lập có thể sử dụng cho thuộc tính LockType. Thiết lập *adLockBatchOptimistic* thích hợp cho cơ sở dữ liệu từ xa như SQL Server hay Oracle hoặc ngược lại với cơ sở dữ liệu cục bộ Jet. Nội dung này sẽ được trình bày cụ thể hơn trong chương 12.

Các hàng được sử dụng để thiết lập thuộc tính LockType của đối tượng Connection

HÀNG	GIÁ TRỊ	Ý NGHĨA
<i>adLockReadOnly</i>	1	Quyền chỉ đọc (mặc định).
<i>adLockPessimistic</i>	2	Khóa một bản ghi ngay khi người sử dụng chọn nó để làm.
<i>adLockOptimistic</i>	3	Khóa bản ghi chỉ khi người sử dụng yêu cầu hiệu chỉnh nó trên cơ sở dữ liệu.
<i>adLockBatchOptimistic</i>	4	Cho phép cập nhật một lô các bản ghi trước khi cố gắng cập nhật chúng vào cơ sở dữ liệu.

Chú ý : Để quyết định khi nào sử dụng một chức năng cụ thể hoặc đặt một hằng biểu diễn các chức năng trong từng phần ta sử dụng phương thức *Support*. Giá trị trả về là *True* cho biết recordset cung cấp chính xác. Các tài liệu trực tuyến về phương thức *Supports* mô tả tên của các hằng. Thông qua lựa chọn Object Browser tìm *CursorOptionEnum* để xem danh sách các hằng hỗ trợ cho kết quả *True* hoặc *False*.

5. Định vị các bản ghi

Có bốn phương thức cho phép định vị các bản ghi bằng cách thay đổi vị trí của bản ghi hiện tại.

– *MoveFirst* : Phương thức *movefirst* di chuyển vị trí của bản ghi hiện tại đến vị trí bản ghi đầu tiên của một recordset. Thứ tự của các bản ghi phụ thuộc vào chỉ số hiện tại, hoặc nếu không có chỉ số thì phụ thuộc vào thứ tự các điểm vào. Phương thức này làm việc với tất cả các kiểu con trỏ. Nó sử dụng kiểu con trỏ là *forward – only* để yêu cầu làm lại lệnh đã tạo ra recordset.

– *MoveLast* : Phương thức này thiết lập vị trí đến vị trí của bản ghi cuối cùng trong recordset. Nó yêu cầu kiểu con trỏ có hỗ trợ sự di chuyển quay lui hoặc ít nhất là sự di chuyển dựa trên chỉ dẫn. Sử dụng phương thức này với con trỏ *forward – only* sẽ báo lỗi.

– *MoveNext* : Phương thức *movenext* định vị lại bản ghi hiện tại vào vị trí của bản ghi kế tiếp nó (theo hướng về phía bản ghi cuối cùng của recordset). Nếu vị trí hiện tại là vị trí của bản ghi cuối cùng, thuộc tính EOF của recordset sẽ được thiết lập là *True*. Nếu phương thức này được gọi khi thuộc tính EOF đã là *True* sẽ đưa lại kết quả là một lỗi thời gian chạy.

– *MovePrevious* : Phương thức này sẽ chuyển vị trí của bản ghi hiện tại về bản ghi liền trước nó. Nếu bản ghi hiện tại là bản ghi đầu tiên của recordset, thuộc tính BOF của recordset sẽ được thiết lập là *True*. Nếu phương thức *MovePrevious* được gọi khi thuộc tính BOF đã là *True* sẽ báo lỗi thời gian chạy. Phương thức cũng tạo ra lỗi thời gian chạy nếu sử dụng kiểu con trỏ là *forward – only*.

Phương thức *Move* làm việc hơi khác so với bốn phương thức định vị recordset trước bởi vì nó có thể di chuyển vị trí của bản ghi hiện tại tới vị trí của một bản ghi bất kỳ theo mọi hướng. Ta sử dụng một số dương để chỉ ra

sự di chuyển về phía trước, theo hướng về phía bản ghi cuối cùng, và một số âm để chỉ ra một sự di chuyển quay lui, theo hướng về phía bản ghi đầu tiên. Nếu sự di chuyển vượt quá vị trí của bản ghi đầu tiên hay bản ghi sau cùng, phương thức *Move* sẽ thiết lập hoặc là thuộc tính *BOF* hoặc là thuộc tính *EOF* bằng *True*. Nếu thuộc tính tương ứng đã là *True*, phương thức *Move* sẽ tạo ra lỗi thời gian chạy. Sự di chuyển là tương đối so với vị trí của bản ghi hiện tại trừ khi đã chỉ rõ tham số *Start*, cho phép di chuyển từ vị trí bản ghi đầu tiên hoặc bản ghi sau cùng.

Có thể nâng cao hiệu năng của phương thức *Move* lên gấp đôi bằng cách sử dụng cùng với thuộc tính *CacheSize* của recordset để thiết lập nó lớn hơn mặc định (bằng 1). Thiết lập *CacheSize* làm cho ADO lưu trữ tĩnh một số các bản ghi vào trong bộ nhớ của máy trạm làm việc. Ta có thể tăng tốc sự định vị các bản ghi với phương thức *Move* bằng cách sử dụng nhiều *CacheSize* hơn. Với kiểu con trỏ *forward - only* và *CacheSize* lớn hơn sẽ cho phép di chuyển tiến cũng như lui. Nếu bộ đệm được thiết lập bằng đúng số lượng bản ghi của recordset, ta có thể di chuyển tiến hay lui một cách đầy đủ. Thuộc tính *CacheSize* không cho phép di chuyển quay lui bằng phương thức *MovePrevious* (nhưng có thể quay lui bằng phương thức *Move* với một tham số âm).

6. Phương thức *Find*

Phương thức *Find* của recordset tìm đến bản ghi đầu tiên thích hợp với một điều kiện được chọn. Ở các phiên bản trước của Access, phương thức này tìm kiếm tương đồng trong một tập hợp còn ở phiên bản Access 2000, chúng có cú pháp và ý nghĩa hơi khác (người sử dụng nên tìm hiểu cú pháp và ý nghĩa của nó trong phiên bản mới thay vì ánh xạ theo phiên bản cũ).

Phương thức *Find* mới cần bốn tham số. Tham số đầu tiên là bắt buộc và là điều kiện cho việc tìm kiếm. Cú pháp của nó giống như trong mệnh đề *Where* của câu lệnh SQL. Nếu không xác định các tham số khác, phương thức này sẽ tìm kiếm từ vị trí bản ghi hiện tại cho tới vị trí của bản ghi cuối cùng để tìm một bản ghi phù hợp với điều kiện chọn và ta phải rời khỏi vị trí của bản ghi tìm được để tìm bản ghi kế tiếp có cùng điều kiện. Nếu không có bản ghi phù hợp, phương thức này sẽ thiết lập thuộc tính *EOF* bằng *True*. Tham khảo tài liệu trực tuyến cho các tham số còn lại.

7. Thuộc tính Sort

Thuộc tính *Sort* của một Recordset có ảnh hưởng tới cả hai phương thức *Find* và *Move*. Nó thiết kế cho một hay nhiều hơn các trường và quyết định thứ tự các hàng khi hiển thị. Thuộc tính *Sort* cho phép hiển thị theo thứ tự tăng dần hay giảm dần của bất cứ một trường nào (thứ tự mặc định là tăng dần (ascending)). Thiết lập thuộc tính *Sort* không sắp xếp vật lý các hàng mà chúng xác định thứ tự ở đó các recordset có các hàng hiện hữu.

8. Thuộc tính Filtered

Thuộc tính *Filtered* cho phép một recordset dịch nghĩa một recordset mới là một phiên bản được chọn lọc từ recordset nguyên bản. Ứng dụng đặc biệt của thuộc tính này giúp cho việc đồng bộ cơ sở dữ liệu và cập nhật theo lô vào một nguồn dữ liệu ở xa, nó là một ví dụ đơn giản thay thế cho việc định nghĩa một recordset mới dựa trên một câu lệnh SQL. Nếu đã có một recordset và ta cần một phần của nó cho mục đích khác, thuộc tính này có thể phục vụ cho công việc đó.

9. Phương thức Addnew

Phương thức *Addnew* cho phép thêm một bản ghi mới vào một recordset. Sau khi gọi phương thức này, cần thiết lập các giá trị cho các trường trong hàng ta muốn thêm. Chúng ta có thể sử dụng phương thức *Update* thay đổi sau khi cập nhật các trường bằng cách gán cho chúng những giá trị mới, và sau đó, ta di chuyển khỏi bản ghi đó. Bản ghi chính thức được ghi vào CSDL khi chúng ta rời khỏi vị trí duyệt bản ghi này.

10. In các giá trị trong trường

Ví dụ 5.12 :

Đoạn mã đơn giản sau mở một nguồn dữ liệu và sau đó in kết quả từng hàng trong cơ sở dữ liệu. Một vòng lặp sẽ duyệt qua tất cả các bản ghi và in ra hai trường đầu tiên của mỗi bản ghi đó.

```
Sub EasyLoop()
```

```
Dim rsCustomers As Recordset
```

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.Open "customers", & _
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _  
"Data Source=C:\Program Files\Microsoft Office\Office\" & _  
"Samples\Northwind.mdb;"
```

'Loop through recordset.

```
Do Until rsCustomers.EOF  
    Debug.Print rsCustomers.Fields(0), rsCustomers.Fields(1)  
    rsCustomers.MoveNext  
Loop  
  
rsCustomers.Close
```

End Sub

Nhược điểm của thủ tục *EasyLoop* là chỉ in ra giá trị các trường đã được chỉ định trước. Thủ tục *EasyLoop2* dưới đây không quan tâm đến số lượng trường bên trong dữ liệu nguồn của một recordset mà tự động in ra tất cả các trường.

```
Sub EasyLoop2()  
    Dim rsCustomers As Recordset  
    Dim fldMyField As Field  
    Dim strForRow As String  
    Set rsCustomers = New ADODB.Recordset  
  
    rsCustomers.Open "customers", & _  
        "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
        "Data Source=C:\Program Files\Microsoft Office\Office\" & _  
        "Samples\Northwind.mdb;"
```

'Loop through recordset and fields with rows.

```
Do Until rsCustomers.EOF
```

```

strForRow = ""
For Each fldMyField In rsCustomers.Fields
    strForRow = strForRow & fldMyField & "; "
Next fldMyField
Debug.Print strForRow
rsCustomers.MoveNext
Loop

rsCustomers.Close

```

End Sub

Một vài dòng đầu tiên và một vài dòng cuối cùng trong mỗi thủ tục là xác định. Thủ tục *EasyLoop2* có một vòng lặp *For* lồng trong vòng lặp *Do*. Vòng lặp *For* bên trong sẽ tổ hợp các trường trong một hàng và tạo ra một chuỗi với tất cả các giá trị của từng trường trên mỗi dòng (chuỗi này được xoá sạch đầu mỗi vòng lặp và bắt đầu được xử lý lại tương tự cho dòng khác).

Lặp là một phương pháp đơn giản để thực thi một hành động trên mỗi dòng trong một recordset. Tuy nhiên, đó không phải là cách hiệu quả nhất để đem lại giá trị. Thủ tục *NoEasyLoop* được trình bày sau đây dùng phương thức *GetString* để thu hồi và in ra tất cả các dòng của một recordset từng bước một. Phương thức *GetString* trả về một recordset như là một chuỗi. Nó cần truyền 5 tham số : đoạn mã sau sẽ truyền 3 trong 5 tham số cần thiết. Ta cần tạo ra hằng số *adClipString* như là một tham số đầu tiên và đây cũng là lựa chọn duy nhất trong trường hợp này. Nó chỉ ra định dạng biểu diễn dữ liệu cho recordset là một chuỗi. Tham số thứ hai cho biết số lượng hàng của recordset sẽ được trả về. Đoạn mã này trả về năm hàng. Nếu đặt tham số này trống tức là ta sẽ thu hồi tất cả các hàng của recordset. Tham số thứ ba đặt một dấu chấm phẩy để phân cách các cột bên trong một hàng. Dấu phân cách mặc định là dấu tab. Các tham số thứ tư và tham số thứ năm không xuất hiện trong thủ tục dưới đây, điều đó chỉ ra các dấu phân cách các cột và các biểu thức biểu diễn giá trị null và giá trị mặc định cho các tham số đó là các dấu xuống dòng và một xâu rỗng.

```

Sub NoEasyLoop()
Dim rsCustomers As Recordset

Set rsCustomers = New ADODB.Recordset

rsCustomers.Open "customers", _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Program Files\Microsoft Office\Office\" & _
    "Samples\Northwind.mdb;"

'Print records without a loop.
Debug.Print rsCustomers.GetString(adClipString, 5, "; ")

rsCustomers.Close

End Sub

```

Phương thức *GetString* thay thế cho cả hai vòng lặp lồng nhau. Nếu các giá trị mặc định là chấp nhận được, ta có thể sử dụng phương thức này mà không cần một tham số nào cả. Điều này tạo ra một cách đơn giản để trích các giá trị ra từ một recordset. Mặc dù các vòng lặp lồng nhau là một cách thích hợp để thu hồi thông tin từ một recordset, phương thức *GetString* có thể đem lại các kết quả tương tự chỉ trong một dòng.

11. Thêm một bản ghi

Ví dụ 5.13 :

Đoạn mã sau thực hiện thêm một bản ghi vào dữ liệu nguồn.

```

Sub AddARecord()
Dim rsMyTable As Recordset

```

```

'Set your cursor so that it is not read-only to delete.

```

```

Set rsMyTable = New ADODB.Recordset
rsMyTable.ActiveConnection = CurrentProject.Connection
rsMyTable.Open "MyTable", , adOpenKeyset, adLockOptimistic, _
    adCmdTable

```

'Invoke the AddNew method.

```

rsMyTable.AddNew
rsMyTable.Fields("Column1").Value = 16
rsMyTable.Fields("Column2").Value = 17
rsMyTable.Fields("Column3").Value = 18
rsMyTable.Update

```

End Sub

Trong khi *Easyloop*, *Easyloop2* và *NoEasyloop* đều chấp nhận phương thức con trỏ và loại khoá mặc định của phương thức *Open* thì thủ tục *AddARecord* không chấp nhận. Ta cần gọi lại các giá trị mặc định kiểu con trỏ là *forward – only* và kiểu khoá là *Read – Only*. Những thiết lập này được chấp nhận cho mục đích thêm vào để in nội dung của một recordset. Tuy nhiên, ta cần kiểu con trỏ cũng như kiểu khoá cho phép cập nhật tới một recordset khi công việc yêu cầu như hiệu chỉnh hay xoá một bản ghi. Tham số *adOpenKeyset* và tham số *adLockOptimistic* để *Open* cho phép thêm một hàng mới vào một recordset. Cần chú ý rằng thiết lập *ActiveConnection* trong đoạn mã trên không giống như trong ví dụ Northwind mà thay vào đó, nó chỉ vào dự án hiện tại. Khi cần tham chiếu đến dữ liệu nguồn trong dự án này, ta sử dụng cú pháp sau : Câu lệnh thiết lập kết nối phải chỉ ra một bảng tường minh trong dự án như là dữ liệu nguồn cho recordset. Có vài nguồn thay thế khác nhau, bao gồm đoạn văn bản cho một câu lệnh SQL, một thủ tục có sẵn hay một file bên ngoài với một định dạng đặc biệt hoặc có thể nhiều hơn nữa.

Để thêm vào một record, ta cần sử dụng phương thức *AddNew* và sử dụng các câu lệnh gán để đưa vào các giá trị cho record mới đó, sau đó gọi phương thức *Update*. Việc gọi phương thức *Update* không mang tính

bắt buộc ; đơn giản, ta có thể di chuyển khỏi bản ghi mới hay hiện tại. Ví dụ, ta có thể gọi phương thức *MoveFirst* hoặc bất cứ một phương thức tương tự khác để định vị một bản ghi mới.

12. Hiệu chỉnh hay xoá một bản ghi

Ví dụ 5.14 :

Đoạn mã sau đây hiệu chỉnh hoặc xoá một bản ghi. Nó không sử dụng phương thức *Edit* và phương thức *Update* để ghi lại bản ghi đã được thay đổi. Thay vào đó, nó di chuyển các bản ghi. Nếu di chuyển một bản ghi thực tế hoặc ứng dụng của người sử dụng cần phải thay đổi trước khi di chuyển thì sử dụng phương thức *Update* của recordset để thay thế.

```
Sub DeleteOrUpdateARecord()
```

```
Dim rsMyTable As Recordset
```

```
'Use a non-read-only lock type to be able to delete records.
```

```
Set rsMyTable = New ADODB.Recordset
```

```
rsMyTable.ActiveConnection = CurrentProject.Connection
```

```
rsMyTable.Open "MyTable", , adOpenKeyset, adLockOptimistic, _  
adCmdTable
```

```
'Loop through recordset.
```

```
Do Until rsMyTable.EOF
```

```
    If rsMyTable.Fields("Column1") = 16 Then
```

```
        rsMyTable.Fields("Column1") = 88
```

```
        rsMyTable.Delete
```

```
    End If
```

```
    rsMyTable.MoveNext
```

```
Loop
```

```
rsMyTable.Close
```

```
End Sub
```

Một vòng lặp giống như trong thủ tục *DeleteOrUpdateARecord* giúp người sử dụng lựa chọn các record để xoá hay thay đổi. Thủ tục kiểm tra

từng giá trị trong trường *Coloumn1* một recordset tìm kiếm một record mà trường đó có giá trị bằng 16. Khi tìm thấy, nó sẽ xoá hàng tương ứng. Nhớ rằng vòng lặp có chứa một dòng chú thích. Để chuyển từ thủ tục xoá sang thủ tục cập nhật, chỉ cần bỏ dấu chú thích trong dòng lệnh gán và thay cho dòng lệnh có phương thức *Delete*.

13. Tìm kiếm bản ghi

Một chức năng thông dụng được sử dụng cho một tập bản ghi là tìm một hay nhiều bản ghi thoả mãn yêu cầu được chỉ ra. Access 2000 đưa ra một vài cách tiếp cận chức năng này. Ở những phiên bản Access trước, nhiều nhà phát triển đã sử dụng một hay nhiều biến thể của phương thức *Find*. Đối với Access 2000 thì đưa ra một phương thức *Find* đơn lẻ khác với phương thức *Find* trước đây. Tuy nhiên nếu muốn sử dụng chức năng *Find* trước đó thì ta cũng có thể sử dụng phương thức *Find* với chức năng tương tự.

Ví dụ 5.15 :

Đoạn mã sau chỉ ra một ứng dụng đơn giản của phương thức *Find* để tìm một bản ghi có ID khách hàng bắt đầu bằng chữ cái D. Khi tìm được bản ghi phù hợp, nó sẽ chuyển bản ghi hiện thời tới bản ghi đó. Mã in trường *CustomerID* và *ContactName* để xác nhận đúng trường đó đã thoả mãn yêu cầu.

```
Sub FindAMatch()
```

```
Dim rsCustomers As Recordset
```

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.ActiveConnection = _
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office\Office\" & _
```

```
"Samples\Northwind.mdb;"
```

```
rsCustomers.Open "Customers", , adOpenKeyset, adLockPessimistic, _  
adCmdTable
```

```
rsCustomers.Find ("CustomerID Like 'D*'" )
```

```
Debug.Print rsCustomers.Fields("CustomerID"), _  
rsCustomers.Fields("ContactName")
```

End Sub

Một nhược điểm trong đoạn mã trên là chỉ tìm kiếm một bản ghi duy nhất phù hợp sau đó sẽ ngừng tìm kiếm. Đoạn mã dưới sẽ cho ra mọi bản ghi phù hợp yêu cầu. Ứng dụng này thể hiện sự tiện lợi của phương thức Find.

```
Sub FindAMatch2()
```

```
Dim rsCustomers As Recordset
```

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.ActiveConnection = & _
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office\Office\" & _
```

```
"Samples\Northwind.mdb;"
```

```
rsCustomers.Open "Customers", , adOpenKeyset, _adLockPessimistic,  
adCmdTable
```

```
Do
```

```
rsCustomers.Find ("CustomerID Like 'D*'" )
```

```
If rsCustomers.EOF Then
```

```
Exit Sub
```

```
End If
```

```
Debug.Print rsCustomers.Fields("CustomerID")
```

```
rsCustomers.MoveNext
```

```
Loop
```

End Sub

Thủ thuật để tìm kiếm mọi bản ghi phù hợp yêu cầu là đưa phương thức *Find* vào trong vòng lặp *Do*. Khi phương thức *Find* đặt thuộc tính *EOF* của tập bản ghi là *True*, không có bản ghi nào phù hợp. Trong trường hợp này,

đoạn mã thực thi lệnh *Exit Sub* tới cuối thủ tục con. Chỉ cần *Find* tiếp tục tìm kiếm bản ghi mới, thủ tục in ra ID của khách hàng trong cửa sổ hiện hành. Sau khi in xong, thủ tục chuyển tới bản ghi kế tiếp. Nếu không làm điều này, phương thức *Find* sẽ lặp lại và đưa ra các bản ghi phù hợp khác.

Phương thức *Find* sẽ tìm các bản ghi và kiểm tra xem có phù hợp không. Nó không tạo ra một tập bản ghi khác gồm các bản ghi phù hợp yêu cầu. Nhưng nếu cần điều đó, ứng dụng của người sử dụng phải thay đổi khác đi. Thuộc tính *Filter* sẽ đáp ứng được yêu cầu này. Thuộc tính này cho phép chỉ định tiêu chuẩn cho một trường và sẽ trả lại một tập bản ghi với các bản ghi phù hợp yêu cầu. Bằng cách cấu hình cho thuộc tính *Filter* ngang bằng với bất kỳ một chuỗi hằng số nào đó, ta có thể thu được hiệu quả đặc biệt cho sự lặp lại của dữ liệu hay là cập nhật nguồn dữ liệu từ xa. Một hằng filter là *adFilterNone* xoá bỏ cấu hình filter từ tập bản ghi và khôi phục lại giá trị ban đầu.

14. Lọc bản ghi (filtering records)

Ví dụ 5.16 :

Hai thủ tục sau đây lọc một tập bản ghi dựa trên bảng Customers trong CSDL Northwind. Thủ tục *FilterRecordset* quản lý toàn bộ thuộc tính *Filter*, in ra tập kết quả, xoá filter và in ra tập kết quả một lần nữa. Thủ tục *FilterRecordset* thông qua chức năng *FilterLikeField* để quản lý cấu hình thuộc tính *Filter* dựa trên các tham số chuyển cho nó qua thủ tục *FilterRecordset*.

```
Sub FilterRecordset()
```

```
Dim rsCustomers As Recordset
```

```
'Create recordset variable.
```

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.ActiveConnection = & _
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office\Office\" & _
```

```
"Samples\Northwind.mdb;"
```

```
'Open recordset.
```

```

rsCustomers.Open "Customers", , , , adCmdTable

'Filter recordset.
Set rsCustomers = _
    FilterLikeField(rsCustomers, "CustomerID", "D*")
Debug.Print rsCustomers.GetString

'Restore recordset.
rsCustomers.Filter = adFilterNone
Debug.Print rsCustomers.GetString

rsCustomers.Close

End Sub

Function FilterLikeField(rstTemp As ADODB.Recordset, _
    strField As String, strFilter As String) As ADODB.Recordset
'Set a filter on the specified Recordset object and then
'open a new Recordset object.
    rstTemp.Filter = strField & " LIKE " & strFilter & ""
    Set FilterLikeField = rstTemp

End Function

```

Thủ tục *FilterRecordset* bắt đầu bằng việc tạo và mở tập bản ghi *rsCustomers*, sau đó áp dụng filter bằng cách gọi hàm *FilterLikeField* với ba tham số và trả về một tập bản ghi đã được lọc. *FilterRecordset* trả về tập bản ghi *rsCustomers* và in ra kết quả để xác thực.

Các tham số cho *FilterLikeField* bao gồm cả *rsCustomers*, là tên trường để điền các bản ghi. *FilterRecordset* dò D* để tìm ra những bản ghi có CustomerID bắt đầu với D. Thuộc tính Filter không giới hạn lọc với phép toán Like. Các phép toán khác có thể chấp nhận được như <, >, <=, >=, <> và =. Ta có thể dùng các phép AND và OR để kết hợp hai hay nhiều biểu thức lại với nhau.

Thuộc tính Filter không giới hạn các phép toán tiêu chuẩn tới các form `FieldName – Operator – Value`. Tuy nhiên một vài hằng Filter cho phép sử dụng thuộc tính đặc biệt. Thủ tục `FilterRecordset` sử dụng thuộc tính `adFilterNone` để khôi phục một tập bản ghi bằng cách bỏ đi các bộ lọc của nó.

15. Sử dụng SQL để tạo tập bản ghi

Các câu lệnh SQL thường chỉ là “`SELECT * FROM TABLE NAME`”, nhưng ta có thể dùng hàm đầy đủ của SQL để tạo ra các tập bản ghi hoặc cũng có thể sử dụng những câu lệnh `SELECT` phức tạp kết hợp với các lệnh *inner* hay *outer joins* và các ràng buộc như `WHERE`, `GROUP BY`, `ORDER BY`. Cách đơn giản nhất để tạo một tập bản ghi dựa trên câu lệnh SQL là sử dụng mệnh đề `WHERE`. Ta có thể trích chọn các bản ghi từ nguồn có sẵn bằng cách sử dụng các biểu thức phức tạp hơn so với khi sử dụng thuộc tính Filter.

Ví dụ 5.17 :

Đoạn mã sau sử dụng phương thức `Open` với câu lệnh SQL. Khi lấy một tập bản ghi trên cơ sở câu lệnh SQL, thay vì sử dụng một bảng có sẵn, ta sử dụng các câu lệnh SQL và các tham biến `adCmdTable` thay vì `adCmdText`. Chỉ dựa trên bảng riêng lẻ, ta có thể sử dụng tập bản ghi để xây dựng tập bản ghi đơn giản bất kỳ. Các câu lệnh SQL phức tạp hơn không cho phép khai báo và sử dụng các tập bản ghi trong ADO.

```
Sub SQLRecordset()
```

```
Dim rsCustomers As Recordset
```

```
'Create recordset variable.
```

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.ActiveConnection = & _
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office\Office\" & _
```

```
"Samples\Northwind.mdb;"
```

```
'Open the recordset.
```

```
rsCustomers.Open "SELECT* FROM Customers", , adOpenForwardOnly, _
```

```
adLockReadOnly, adCmdText
```

Debug.Print rsCustomers.GetString

rsCustomers.Close

End Sub

5.4.3. Đối tượng Field

Một trường là một cột dữ liệu bao gồm các kiểu dữ liệu giống nhau. Trong thư viện ADODB, tập các Field thuộc các tập bản ghi và thành viên của nó là các đối tượng Field. Thuộc tính và phương thức của đối tượng Field là để lưu trữ và phục hồi dữ liệu.

Tập bản ghi sử dụng thuộc tính Value của đối tượng Field để hiển thị nội dung của một cột trong bản ghi hiện hành. Khi thay đổi bản ghi, giá trị này có thể thay đổi tương ứng với bản ghi mới. Nhiều thuộc tính Field khác chứa đựng siêu dữ liệu về dữ liệu trong bản ghi. Thuộc tính Name được nắm bắt khi ứng dụng gọi đến một trường. Thuộc tính DefinedSize mô tả kích thước lớn nhất của một trường (trả về số ký tự trong một trường văn bản). Thuộc tính ActualSize là độ dài thực tế (tính theo bytes) của một trường. Thuộc tính Attributes lưu giữ mảng thông tin miêu tả về một trường và chỉ ra xem giá trị của trường có thể cập nhật được hay có chứa giá trị Null hay không.

Chú ý : Thuộc tính DefinedSize và ActualSize sử dụng độ đo khác nhau cho các trường văn bản. DefinedSize là số lớn nhất các ký tự trong trường và ActualSize là số bytes trong trường. Vì trường Text với Jet 4 thể hiện ký tự với 2 bytes, giá trị ActualSize có thể gấp hai lần giá trị của DefinedSize. Với các trường số và trường Text trong CSDL thể hiện ký tự bằng 1 bytes (ví dụ Jet 3.51), sự khác biệt này không tồn tại.

Phương thức Field GetChunk và AppendChunk thuận tiện trong xử lý đoạn văn bản lớn hay các trường dữ liệu dạng nhị phân. Phương thức GetChunk được sử dụng để đưa vào bộ nhớ một mảng dữ liệu lớn. Tham biến Size miêu tả số bytes nhận trong một lần gọi phương thức GetChunk. Mỗi lần gọi phương thức thành công thì bắt đầu đọc dữ liệu mới từ nơi phần trước kết thúc. Phương thức AppendChunk cho phép xây dựng một đoạn văn bản lớn hay trường dữ liệu nhị phân từ bộ nhớ. Giống như phương thức

GetChunk, nó ghi dữ liệu mới vào trường từ chỗ phương thức *AppendChunk* trước kết thúc. Để sử dụng một trong hai phương thức một cách đúng đắn, bit *adFldLong* của đối tượng *Field* trong thuộc tính *Attributes* phải đặt là *True*.

1. Thuộc tính *Name* và *Value*

Thủ tục được trình bày trong ví dụ 5.18 chỉ ra cách sử dụng thông thường cho thuộc tính *Name* và *Value*. Thủ tục này liệt kê các tên trường với những giá trị tương ứng và tạo ra các tập bản ghi dựa trên các câu lệnh SQL.

Ví dụ 5.18 :

```
Sub FieldNameValue()
```

```
Dim cnn1 As ADODB.Connection
```

```
Dim rsCustomers As ADODB.Recordset
```

```
Dim fldLoop As ADODB.Field
```

```
'Open connection and recordset.
```

```
strCnn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office" & _
```

```
"\Office\Samples\Northwind.mdb;"
```

```
Set cnn1 = New ADODB.Connection
```

```
cnn1.Open strCnn
```

```
Set rsCustomers = New ADODB.Recordset
```

```
rsCustomers.ActiveConnection = cnn1
```

```
rsCustomers.Open "SELECT * FROM Customers " & _
```

```
"WHERE CustomerID='BONAP'", , , adCmdText
```

```
'Report field names and values for record.
```

```
For Each fldLoop In rsCustomers.Fields
```

```
Debug.Print fldLoop.Name, fldLoop.Value
```

```
Next fldLoop
```

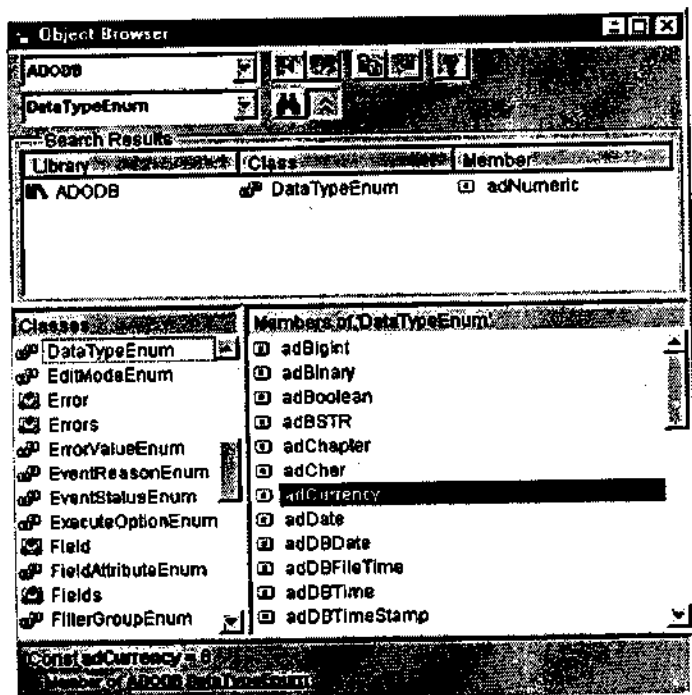
```
End Sub
```

Thủ tục bắt đầu bằng việc mở một kết nối sau đó tạo tập bản ghi dựa trên kết nối đó. Câu lệnh SQL trích bản ghi cho khách hàng với trường CustomerID bằng BONAP. Vòng lặp Do sẽ lặp để tạo ra tập bản ghi qua các trường của tập bản ghi. In thuộc tính Name cùng với thuộc tính Value giúp cho việc đọc dễ dàng.

2. Thuộc tính Type

Thuộc tính *Type* của một đối tượng *Field* chỉ ra loại dữ liệu mà đối tượng có thể lưu trữ.

Thuộc tính này trả về một trong các hằng được định nghĩa trong dải *DataTypeEnum*. Có thể quan sát các lựa chọn này trong cửa sổ *Object Browser* của thư viện ADODB. Hình 5.8 trình bày các hằng đó trong cửa sổ *Object Browser*. Bằng cách chọn kiểu cho một trường, ta có thể quyết định các giá trị tương ứng trong thuộc tính *Value* của nó.



Hình 5.8. Cửa sổ Object Browser biểu diễn tập hợp các hằng định kiểu.

3. In các kiểu dữ liệu của các trường

Hai thủ tục sau đây sẽ cùng làm việc để xử lý các hằng định kiểu với ADO. Thủ tục *FieldNameType* mở một recordset dựa trên bảng *Orders* trong cơ sở dữ liệu Northwind. Bảng này có nhiều kiểu dữ liệu khác nhau, điều này là cơ sở để kiểm tra các kiểu dữ liệu. Sau khi mở một recordset, thủ tục này duyệt qua tất cả các trường trong recordset đó và in ra kiểu và tên của trường. Hàm *FieldType* dịch các hằng số sang các chuỗi biểu diễn tương ứng. Hằng *adCurrency* có giá trị bằng 6. Hàm *FieldType* giải mã giá trị 6

sang chuỗi "adCurrency". Thủ tục *FieldNameType* sau đó in ra tên và kiểu của các trường.

```
Sub FieldNameType()
```

```
Dim cnn1 As ADODB.Connection
```

```
Dim rsOrders As ADODB.Recordset
```

```
Dim fldLoop As ADODB.Field
```

```
'Open connection and recordset.
```

```
strCnn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
    "Data Source=C:\Program Files\Microsoft Office" & _
```

```
    "\Office\Samples\Northwind.mdb;"
```

```
Set cnn1 = New ADODB.Connection
```

```
cnn1.Open strCnn
```

```
Set rsOrders = New ADODB.Recordset
```

```
rsOrders.ActiveConnection = cnn1
```

```
rsOrders.Open "orders", , , , adCmdTable
```

```
'Report field names and types for record.
```

```
For Each fldLoop In rsOrders.Fields
```

```
    Debug.Print " Name: " & fldLoop.Name & vbCr & _
```

```
        " Type: " & FieldType(fldLoop.Type) & vbCr
```

```
Next fldLoop
```

```
End Sub
```

```
Public Function FieldType(intType As Integer) As String
```

```
    Select Case intType
```

```
        Case adVarChar
```

```
            FieldType = "adVarChar"
```

```
        Case adCurrency
```

```
            FieldType = "adCurrency"
```

```
        Case adInteger
```

```
FieldType = "adInteger"
```

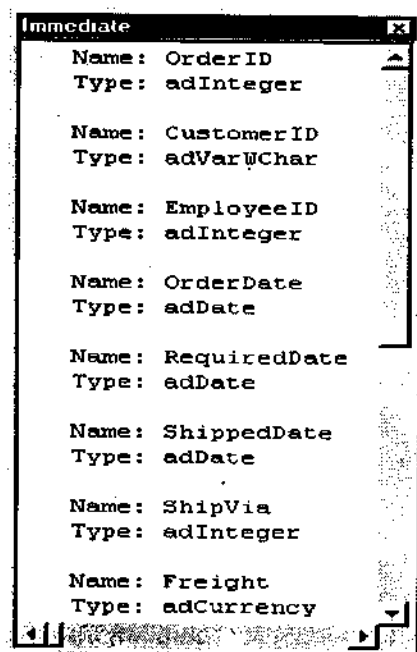
```
Case adDate
```

```
FieldType = "adDate"
```

```
End Select
```

```
End Function
```

Hình 5.9 biểu diễn một trích dẫn từ đầu ra của *FieldNameType*. Trích dẫn này bao gồm ít nhất một trường với kiểu dữ liệu tương ứng được giải mã trong hàm *FieldType*. Có thể dễ dàng chạy *FieldNameType* và *FieldType* trên các recordset của các nguồn dữ liệu khác thay vì chỉ trong bảng *Orders*. Ngoài bốn kiểu trong danh sách, ta cũng có thể làm với các kiểu khác. Trong trường hợp này, trường *Type* trong báo cáo sẽ bị để trống. Khi đó, có thể sửa chữa bằng cách quyết định giá trị của các trường. Để làm được điều này cần đặt một điểm dừng trên câu lệnh *Debug.Print* bên trong vòng lặp *Do* của thủ tục *FieldNameType*. Kiểm tra giá trị của *fldloop.Type* cho mỗi trường có kiểu không được hiển thị và sau đó gán với các hằng số liên kết với tên hằng bên trong cửa sổ *Object Browser* cho khóa *DataTypeEnum* (xem hình 5.8). Cuối cùng, hoàn thiện câu lệnh *Select Case* trong thủ tục *FieldType* để giải mã các hằng còn lại.



Immediate	
Name:	Order ID
Type:	adInteger
Name:	Customer ID
Type:	adVarChar
Name:	EmployeeID
Type:	adInteger
Name:	OrderDate
Type:	adDate
Name:	RequiredDate
Type:	adDate
Name:	ShippedDate
Type:	adDate
Name:	ShipVia
Type:	adInteger
Name:	Freight
Type:	adCurrency

Hình 5.9. Trích từ kết quả của thủ tục *FieldNameType*.

4. Tìm thành phần dài nhất của trường

Thủ tục *FieldSizes* dưới đây áp dụng thuộc tính *ActualSize* để tìm ra thành phần dài nhất của trường *CompanyName* trong bảng *Shippers* của cơ sở dữ liệu *Northwind*. Thủ tục bắt đầu bằng việc tạo kết nối tới cơ sở dữ liệu *Northwind* và sau đó mở một recordset trên bảng *Shippers*. Phần tiếp theo của thủ tục là tìm ra tên một công ty dài nhất và hiển thị hộp thông báo số ký tự và tên của công ty đó.

```

Sub FieldSizes()
Dim cnn1 As ADODB.Connection
Dim rsShippers As ADODB.Recordset
Dim fldLoop As ADODB.Field
Dim intMaxChars As Integer, strMsg As String
Dim strName As String
'Open connection and recordset.
strCnn = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Program Files\Microsoft Office" & _
    "\Office\Samples\Northwind.mdb;"
Set cnn1 = New ADODB.Connection
cnn1.Open strCnn
Set rsShippers = New ADODB.Recordset
rsShippers.ActiveConnection = cnn1
rsShippers.Open "SELECT * FROM Shippers" _
    , , , adCmdText

'Find longest shipper's name.
intMaxChars = 0
Do Until rsShippers.EOF
    If rsShippers!CompanyName.ActualSize / 2 _
        > intMaxChars Then
        intMaxChars _
            = rsShippers!CompanyName.ActualSize / 2
        strName = rsShippers.Fields("CompanyName")
    End If
    rsShippers.MoveNext
Loop
strMsg = "The longest shipper's name is " & _
    strName & " (" & intMaxChars & " characters)."

```

MsgBox strMsg, vbInformation, "Programming Microsoft Access 2000"

rsShippers.Close

End Sub

Biến *intMaxChars* kiểm soát độ dài của trường dài nhất. Phần thứ hai của *FieldSizes* sẽ khởi tạo biến bằng 0 trước khi bắt đầu vòng lặp duyệt qua tất cả các bản ghi của recordset. Điều này thực sự cần thiết vì VBA không tự động khởi tạo nó bằng không. Tuy nhiên, điều đó có thể tự chú thích cho thủ tục. Bất kỳ một công ty nào có tên dài hơn *intMaxChars* sẽ cập nhật *intMaxChars* và lưu trữ lại tên đó. Cần chú ý rằng thủ tục sử dụng các quy tắc cú pháp khác nhau để tham chiếu vào cùng một trường. Một số câu lệnh sử dụng dấu chấm than (!) theo kí hiệu của Access và kí hiệu này sẽ phân tách các tên của recordset và tên các trường. Phương thức phổ biến hơn để tham chiếu vào các trường là sử dụng tập hợp *Fields* và cũng có thể sử dụng các chỉ số nếu đã biết rõ nó hoặc sử dụng tên trường trong dấu nháy kép.

5.4.4. Các đối tượng Command và Parameter

Trong thư viện ADODB, đối tượng Command có 3 ưu điểm chính :

- Có thể thực thi một câu lựa chọn để trả về một tập các hàng từ dữ liệu nguồn.
- Xử lý tham số của câu hỏi đáp vì thế ta có thể nhập các điều kiện tìm kiếm thời gian chạy.
- Hỗ trợ dữ liệu nguồn nhằm thực thi các hoạt động như cập nhật, xoá và thêm các bản ghi.

Đối tượng Command đóng vai trò quan trọng hơn so với các thư viện khác, điều này sẽ được thảo luận ở các phần sau.

Điều kiện bắt buộc là phải tạo ra một đối tượng Connection để chạy một lệnh ở trên đó. Một đối tượng Connection được tạo ra khi chỉ định một câu lệnh hoặc khởi gán một đối tượng Connection đã tồn tại cho một câu lệnh. Có những sự lựa chọn tương tự cho tập bản ghi.

1. Thuộc tính *CommandTimeout*

Thuộc tính *CommandTimeout* quyết định thời gian ADO sẽ chờ việc thực hiện câu lệnh để kết thúc. Thuộc tính này nhận giá trị kiểu Long, thể hiện thời gian chờ đợi tối đa tính theo giây và giá trị mặc định của nó là 30. Nếu khoảng thời gian này kết thúc trước khi đối tượng *Command* hoàn thành công việc, ADO sẽ hủy bỏ lệnh và đưa ra thông báo lỗi. Đối tượng *Connection* cũng hỗ trợ thuộc tính *CommandTimeout*. Nó có cùng tên nhưng lại độc lập với thuộc tính *CommandTimeout* của đối tượng *Command*. Thuộc tính *CommandTimeout* của đối tượng *Command* không được thừa hưởng các sắp đặt của thuộc tính *CommandTimeout* của đối tượng *Connection*.

2. Thuộc tính *CommandType*

Thuộc tính *CommandType* xác định loại của đối tượng *Command*. Câu lệnh của người sử dụng có thể dựa trên một mệnh đề SQL, một bảng hoặc một thủ tục và được thể hiện ở bảng 5.1. Lý do chính sắp xếp lại thuộc tính *CommandType* là để cho phép tạo ra một đối tượng *Command* dựa trên một câu lệnh SQL. Thay đổi hằng *CommandType* khác với giá trị mặc định có thể làm tăng tốc độ xử lý của câu lệnh. Vì vậy, nếu đã có mã nguồn trong tay thì nên thiết lập giá trị cho hằng này.

Bảng 5.1. CÁC HẰNG COMMANDTYPE

HẰNG	GIÁ TRỊ	HOẠT ĐỘNG
<i>adCmdText</i>	1	Cho phép chạy một lệnh dựa trên một mệnh đề SQL, một thủ tục có sẵn hay thậm chí một bảng. Ta thường giữ thiết lập này cho mệnh đề SQL.
<i>adCmTable</i>	2	Tập trả về dựa trên một bảng đã được xây dựng trước đó. Trả về tất cả các cột từ bảng dựa trên các nội dung mà mệnh đề SQL tạo ra.
<i>adCmdStoredProc</i>	4	Chạy một lệnh dựa trên một đoạn văn bản cho một thủ tục đã có sẵn.

<i>adCmdUnknown</i>	8	Không có sự chỉ định đặc biệt nào cho loại câu lệnh. Đây là giá trị mặc định.
<i>adCmdFile</i>	256	Đánh giá một câu lệnh dựa theo tên file nhằm đảm bảo sự toàn vẹn của một recordset.
<i>adCmdTableDirect</i>	512	Đánh giá một câu lệnh giống như tên một bảng. Trả về mọi cột trong bảng mà không có bất kỳ lệnh SQL xen giữa.

3. Thuộc tính *CommandText*

Thuộc tính *CommandText* của đối tượng *Command* dùng để viết một mệnh đề SQL cho command thực hiện, cũng có thể đặt thuộc tính này thành tên của một thủ tục đã được lưu trữ. Khi chạy một câu lệnh SQL, ta có thể sử dụng thuộc tính *Prepared* để thông báo rằng mệnh đề này đã được biên dịch và được lưu trữ trên cơ sở dữ liệu... Điều này sẽ làm chậm xử lý đầu tiên của lệnh nhưng lại tăng tốc độ của chuỗi xử lý sau đó. Khởi gán giá trị *True* cho thuộc tính *Prepared* để gọi trình biên dịch cho câu lệnh SQL.

4. Phương thức *Execute*

Phương thức *Execute* dành cho đối tượng *Command* gọi đến đoạn mã sau đối tượng *Command* (một câu hỏi, một mệnh đề SQL hay một thủ tục đã có). Có thể chỉ định 3 đối số cho phương thức *Execute*. Đối số đầu tiên cho phép đối tượng *Command* thông báo với thủ tục gọi đến số lượng bản ghi mà nó tác động lên. Đối số thứ hai có thể là một mảng các giá trị *Variant* với các tham số để điều khiển câu lệnh. Đối số thứ ba cho ADO biết cách đánh giá mã nguồn. Nó có thể là bất kỳ hằng nào được liệt kê trong bảng 5.1.

5. Phương thức *CreateParameter*

Phương thức *CreateParameter* của đối tượng *Command* tạo ra tham số mới cho câu lệnh. Sau khi tham số được tạo, ta có thể sử dụng phương thức *Append* để thêm tham số vào tập tham số cho câu lệnh. Trước khi chạy một câu truy vấn có tham số, phải khởi gán giá trị cho tham số.

Tạo một tập bản ghi với câu truy vấn lựa chọn : một trong những nhiệm vụ hàng đầu phải thực hiện với đối tượng *Command* là tạo ra một tập bản

ghi dựa trên câu truy vấn lựa chọn. Đối tượng *Command* chạy câu truy vấn và hiển thị tập hợp mà nó trả về. Chương trình sau đó có thể mở đối tượng *Recordset* dựa trên tập trả về từ đối tượng *Command*. Thủ tục *SelectCommand* sẽ hoàn thành công việc đó. Nó bao gồm hai phần : phần một tạo ra đối tượng *Command* và tạo kết nối cho đối tượng đến cơ sở dữ liệu, phần hai xử lý tập bản ghi dựa trên tập trả về từ đối tượng *Command*.

```
Sub SelectCommand()
```

```
Dim cmd1 As Command
```

```
Dim rs1 As Recordset, str1 As String
```

```
Dim fldLoop As ADODB.Field
```

'Define and execute command.

```
Set cmd1 = New ADODB.Command
```

```
With cmd1
```

```
    .ActiveConnection = CurrentProject.Connection
```

```
    .CommandText = "SELECT MyTable.* FROM MyTable"
```

```
    .CommandType = adCmdText
```

```
    .Execute
```

```
End With
```

'Open and print recordset.

```
Set rs1 = New ADODB.Recordset
```

```
rs1.Open cmd1
```

```
Do Until rs1.EOF
```

```
    str1 = ""
```

```
    For Each fldLoop In rs1.Fields
```

```
        str1 = str1 & fldLoop.Value & Chr(9)
```

```
    Next fldLoop
```

```
Debug.Print str1
```

```
rs1.MoveNext
```

```
Loop
```

```
End Sub
```

Phần đầu khai báo *cmd1* như một đối tượng *Command* và sau đó thiết lập ba thuộc tính điều kiện của đối tượng. Mỗi một câu lệnh cần phải có một thuộc tính *ActiveConnection* để chạy trên cơ sở dữ liệu. Đối tượng *Command* dựa trên một câu lệnh SQL để biểu diễn câu truy vấn lựa chọn của nó (cũng có thể thay bằng một câu truy vấn đã được lưu trữ trước). Một câu lệnh *Execute* sẽ chạy câu truy vấn lựa chọn. Sau khi phương thức *Execute* thực hiện, *cmd1* chứa một tham chiếu đến một recordset.

Phần thứ hai của thủ tục thực hiện mở một đối tượng *Recordset* dựa trên *cmd1* và in ra kết quả với dấu phân tách *tab* trong cửa sổ *Immediate*. Thủ tục này có thể xử lý số lượng hàng và cột bất kỳ.

6. Tạo một recordset với câu truy vấn có tham số

Đoạn mã sau là một ví dụ của câu truy vấn có tham số. Đoạn mã này được thiết kế hai phần riêng rẽ. Tham số trong đoạn thứ nhất có một số dòng mã của ADO và một ngữ pháp của câu lệnh SQL khác hẳn so với câu truy vấn chọn trước đó. Đoạn thứ hai in ra kết quả trả về giống như câu truy vấn chọn lựa trước đó.

```
Sub ParameterQCommand()
```

```
Dim cmd1 As Command
```

```
Dim rs1 As Recordset, str1 As String
```

```
Dim fldLoop As ADODB.Field
```

```
Dim prm1 As ADODB.Parameter, int1 As Integer
```

```
'Create and define command.
```

```
Set cmd1 = New ADODB.Command
```

With cmd1

.ActiveConnection = CurrentProject.Connection

.CommandText = "Parameters [Lowest] Long;" & _

"SELECT Column1, Column2, Column3 " & _

"FROM MyTable " & _

"WHERE Column1>=[Lowest]"

.CommandType = adCmdText

End With

'Create and define parameter.

Set prm1 = cmd1.CreateParameter("[Lowest]", _
adInteger, adParamInput)

cmd1.Parameters.Append prm1

int1 = Trim(InputBox("Lowest value?", _
"Programming Microsoft Access 2000"))

prm1.Value = int1

'Run parameter query.

cmd1.Execute

'Open recordset on cmd1 and print it out.

Set rs1 = New ADODB.Recordset

rs1.Open cmd1

Do Until rs1.EOF

str1 = ""

For Each fldLoop In rs1.Fields

str1 = str1 & fldLoop.Value & Chr(9)

Next fldLoop

Debug.Print str1

rs1.MoveNext

Loop

End Sub

Cú pháp của câu lệnh SQL sử dụng một khai báo mới là *Parameters* xác định tên và kiểu của tham số. Mệnh đề *Where* chiếu đến một hoặc nhiều các tham số tác động lên kết quả trả về. Những thay đổi về cú pháp câu lệnh SQL không tự tạo được hiệu quả cho tham số thực sự làm việc, ta phải thêm vào các tham số và ghép nó vào các câu lệnh sử dụng mã lệnh của ADO. Sử dụng phương thức *CreateParameter* để thêm các tham số. Đoạn mã trên sử dụng ba tham trị với phương thức *CreateParameter*. Tham trị thứ nhất đặt tên cho tham số, tham trị thứ hai chỉ ra loại dữ liệu và thứ ba khai báo hướng cho tham số. Hằng *adParamInput* là giá trị mặc định chỉ ra những tham số là đầu vào cho các câu truy vấn. Các hằng khác chỉ đầu ra, vào/ra và trả về giá trị cho tham số. Khi đã tạo được các tham số, cần ghép nó vào tập hợp các *Parameter* của câu lệnh.

Sau khi viết đoạn mã trên để tạo tham số, phải gán những giá trị cho tham số để các câu lệnh truy vấn có tham số hoạt động chính xác. Đoạn mã trên sử dụng một hàm *InputBox* để thu thập thông tin từ người sử dụng. Thủ tục đó tiếp tục gọi phương thức *Execute* của đối tượng *Command* để tạo ra kết quả.

7. Xóa bản ghi

Có thể sử dụng đối tượng *Command* để xóa, cập nhật hoặc thêm các bản ghi vào nguồn dữ liệu. Đối tượng *Command* cung cấp một cách duy trì dữ liệu nguồn bằng cách lập trình. Hai thủ tục *DeleteARecord* và *DeleteAllRecords* dưới đây sẽ xóa các bản ghi với dữ liệu nguồn. Ta có thể chỉ ra dữ liệu nguồn và điều kiện chọn các bản ghi sử dụng câu lệnh *Delete* của ngôn ngữ SQL. Cửa sổ khung nhìn trong câu truy vấn của Access cho phép thiết kế câu truy vấn một cách hình học và sau đó copy đoạn mã để đưa vào thuộc tính *CommandText* của một câu lệnh. Cụ thể hơn, nếu muốn thay đổi câu lệnh SQL từ màn hình thiết kế câu truy vấn của Access để loại bỏ các chỗ trống thừa thì có thể xóa các đoạn tiền tố chỉ tên bảng trước mỗi trường nếu câu truy vấn chỉ hoạt động trên một bảng đơn lẻ. Như vậy, sự khác nhau giữa hai truy vấn để xóa chỉ là cú pháp của câu lệnh SQL.

```
Sub DeleteARecord()
```

```
Dim cmd1 As ADODB.Command
```

```
Set cmd1 = New ADODB.Command
```

```
With cmd1
```

```
    .ActiveConnection = CurrentProject.Connection
```

```
    .CommandText = "DELETE MyTable.Column1 FROM " & _  
        "MyTable WHERE (((MyTable.Column1)=13));"
```

```
    .CommandType = adCmdText
```

```
    .Execute
```

```
End With
```

```
End Sub
```

```
Sub DeleteAllRecords()
```

```
Dim cmd1 As ADODB.Command
```

```
Set cmd1 = New ADODB.Command
```

```
With cmd1
```

```
    .ActiveConnection = CurrentProject.Connection
```

```
    .CommandText = "DELETE MyTable.* FROM MyTable"
```

```
    .CommandType = adCmdText
```

```
    .Execute
```

```
End With
```

```
End Sub
```

8. Chèn bản ghi

Thủ tục *InsertRecords* dưới đây sử dụng đối tượng *Command* để lưu một bảng với các giá trị của nó. Ta có thể sử dụng thủ tục này để kết hợp với thủ tục *DeleteAllRecords* để làm tươi lại một bảng với tập các bản ghi nhỏ gọn.

```
Sub InsertRecords()
```

```
Dim cmd1 As ADODB.Command
```

```
Set cmd1 = New Command
```

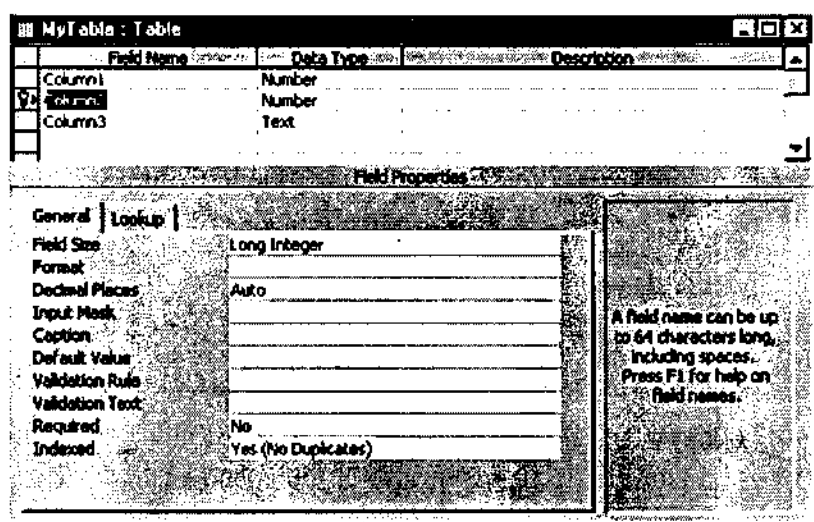
With cmd1

```
.ActiveConnection = CurrentProject.Connection  
.CommandText = "INSERT INTO MyTable(Column1, " & _  
    "Column2, Column3) VALUES (1,2,'3')"  
.CommandType = adCmdText  
.Execute  
.CommandText = "INSERT INTO MyTable(Column1, " & _  
    "Column2, Column3) VALUES (4,5,'6')"  
.CommandType = adCmdText  
.Execute  
.CommandText = "INSERT INTO MyTable(Column1, " & _  
    "Column2, Column3) VALUES (7,8,'9')"  
.CommandType = adCmdText  
.Execute  
.CommandText = "INSERT INTO MyTable(Column1, " & _  
    "Column2, Column3) VALUES (10,11,'12')"  
.CommandType = adCmdText  
.Execute  
.CommandText = "INSERT INTO MyTable(Column1, " & _  
    "Column2, Column3) VALUES (13,14,'15')"  
.CommandType = adCmdText  
.Execute  
.CommandText = "INSERT INTO MyTable(Column1, " & _  
    "Column2, Column3) VALUES (16,17,'18')"  
.CommandType = adCmdText  
.Execute
```

End With

End Sub

Thủ tục *InsertRecords* có các phần tử cụ thể, phần tử chung. Các phần tử chung này không phụ thuộc vào thiết kế của một bảng cụ thể. Trong đoạn mã trên, các phần tử này biến đổi các phần tử chung của bảng *MyTable*. Hình 5.10 hiển thị khung nhìn thiết kế của bảng *MyTable* bao gồm ba cột, được đặt tên là *Column1*, *Column2* và *Column3*. Hai cột đầu tiên có kiểu dữ liệu là số nguyên dài (Long Integer) và cột thứ ba có kiểu dữ liệu dạng văn bản (khi thêm các bản ghi vào một bảng cần phải xem xét các kiểu dữ liệu của các trường).



Hình 5.10. Khung nhìn thiết kế bảng tiến hành thủ tục *InsertRecord* để thêm các bản ghi.

Các đối tượng chung của thủ tục *InsertRecords* được chia sẻ với các ứng dụng khác của đối tượng *Command* vì vậy cần tạo một tham chiếu tới đối tượng *Command* và thiết lập các thuộc tính kết nối của nó. Đối với mỗi hàng cần thêm một bản ghi, có ba dòng buộc phải có : thiết lập thuộc tính *CommandText* – chỉ ra câu lệnh đó sẽ làm gì ; thiết lập thuộc tính *CommandType* – chỉ ra định dạng của chỉ lệnh và phương thức *Execute* – kích hoạt việc thêm vào một bản ghi. Ta có thể lặp lại ba dòng trên cho mỗi hàng được thêm vào dữ liệu nguồn. Nếu ta chỉ ra một dynaset cập nhật như là mục tiêu, các bước trên có thể thêm các bản ghi một cách đồng thời vào hai hai nhiều hơn các bảng cùng một thời điểm.

Cú pháp cho đối tượng *CommandText* của câu lệnh SQL không hiện hữu trong khung nhìn thiết kế câu truy vấn của Access. Chúng có ba đặc điểm sau:

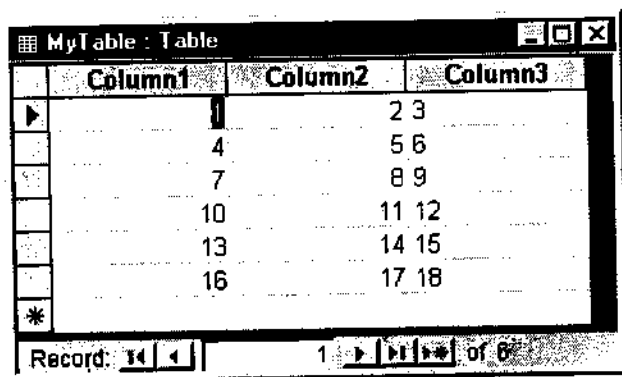
– Từ khoá **INSERT INTO** đi kèm với tên của dữ liệu nguồn nơi cần thêm vào các bản ghi.

– Có một bước không bắt buộc là liệt kê các tên trường mà nó gửi dữ liệu tới. Nếu không tiến hành bước này, các giá trị trong bước thứ ba sẽ phải tuân theo một thứ tự tuần tự, điều đó có thể là một vấn đề nếu thiết kế dữ liệu nguồn bị thay đổi theo thời gian.

– Từ khoá **VALUES** xuất hiện trước các giá trị của các trường cho bản ghi mới.

9. Cập nhật các giá trị cho bản ghi

Thủ tục *OddToEven* và *EvenToOdd* dưới đây cập nhật giá trị trong cột *Column1* của dữ liệu nguồn sử dụng đối tượng *Command*. Hình 5.11 hiển thị một khung nhìn của bảng sau khi hai thủ tục *DeleteAllRecords* và *InsertRecords* được chạy. Trên hình 5.11 giá trị của cột *Column1* được thay đổi xen kẽ giữa chẵn và lẻ. Nếu giá trị *Column1* là lẻ, giá trị trong *Column2* sẽ là chẵn. Thủ tục này sử dụng các thông tin đó để quản lý nội dung của bảng.



	Column1	Column2	Column3
▶	1	2	3
	4	5	6
	7	8	9
	10	11	12
	13	14	15
	16	17	18
*			

Record: 1 of 6

Hình 5.11. Khung nhìn Datasheet của bảng sau khi *OddToEven* và *EvenToOdd* thực hiện.

Sub *OddToEven*()

Dim cmdO2E As ADODB.Command

Dim intRowsChanged As Integer

Set cmdO2E = New ADODB.Command

With cmdO2E

```

.ActiveConnection = CurrentProject.Connection
.CommandText = "UPDATE MyTable SET Column1 = " & _
    "Column1+1 WHERE ((1*(Column1 Mod 2))=True)"
.CommandType = adCmdText
.Execute intRowsChanged
Debug.Print intRowsChanged & " rows were affected."
End With

```

End Sub

Sub EvenToOdd()

Dim cmdE2O As ADODB.Command

Set cmdE2O = New ADODB.Command

With cmdE2O

.ActiveConnection = CurrentProject.Connection

.CommandText = "UPDATE MyTable SET Column1 = " & _
 "Column1-1 WHERE ((-1*(Column2 Mod 2))=False)"

.CommandType = adCmdText

.Execute

End With

End Sub

Thiết kế tổng quan của các thủ tục trên cũng tương tự như trong các thủ tục trước đó. Sự khác biệt đáng kể nhất giữa chúng là ở cú pháp của câu lệnh SQL cho thuộc tính *CommandText*. Trong trường hợp này, ta có thể suy ra một cách dễ dàng từ cú pháp tổng quát của bộ thiết kế câu truy vấn trong Access. Mệnh đề Where trong thủ tục *OddToEven* chọn các bản ghi có giá trị của cột *Column1* là lẻ. Phần UPDATE của đoạn cú pháp cộng 1 vào giá trị để chuyển giá trị đó từ một số lẻ sang một số chẵn. Phương thức *Execute* sử dụng một trong các tham trị sẵn có của nó để trả về một số hàng mà câu lệnh làm thay đổi. Ví dụ đơn giản : phương thức *Print* gửi các giá trị này tới cửa sổ Immediate để quan sát.

Thủ tục *EvenToOdd* kiểm tra điểm vào trong *Column2* để quyết định nó nên trừ 1 từ giá trị nào trong *Column1*. Khi điểm vào trong *Column2* là chẵn, câu lệnh SQL thực hiện thao tác trên giá trị của *Column1*. Điều này khôi phục lại giá trị ban đầu của *Column1* nếu *EvenToOdd* được thực hiện ngay sau khi *OddToEven* được chạy.

5.4.5. Tập hợp lỗi (Errors)

Tập các lỗi có vai trò kiểm soát lỗi xuất hiện trong một ứng dụng ADO (tuy nhiên không phải toàn bộ các lỗi) và trả về các lỗi từ nguồn cung cấp OLEDB. Một lỗi điều kiện có thể trả về rất nhiều lỗi, một trong số đó là nguyên nhân mà một đối tượng Error mới được thêm vào trong tập các lỗi. Một số lỗi sẽ làm treo chương trình, một số khác thì không. Một số lỗi trong ADO thuộc về đối tượng Error (*Err*) hơn là trong tập các lỗi. Tập các lỗi được sử dụng dành riêng cho việc kiểm soát các lỗi trả về từ cơ sở dữ liệu thông qua nhà cung cấp OLEDB.

Đối tượng *Err* trong tập các lỗi có 5 thuộc tính do đó đưa ra nhiều thông tin hơn và vì thế có thể phản hồi lại chúng với một chương trình đúng đắn. Thuộc tính *Number* và *Description* luôn đi cùng đối tượng *Err*. Các đối tượng đó bổ sung cho nhau. Thuộc tính *Number* trả về mã lỗi và thuộc tính *Description* trả về một chuỗi mô tả lỗi. Thuộc tính *NativeError* đưa ra mã lỗi của một nhà cung cấp xác định. Nếu người sử dụng thường xuyên làm việc với một nhà cung cấp cụ thể, thuộc tính này sẽ cung cấp nhiều thông tin bổ ích về việc xử lý lỗi. Thuộc tính *Source* cho biết tên của đối tượng hay chương trình gây ra lỗi. Thuộc tính *SQLState* có thể chứa thông báo lỗi cú pháp trong câu lệnh SQL từ cơ sở dữ liệu mà người sử dụng đưa ra yêu cầu.

Thủ tục *OpenLookOnlyError* dưới đây mô phỏng cho một thủ tục trước đó đã bộc lộ các thuộc tính *Mode* của đối tượng *Connection*. Một thiết lập chỉ đọc cho thuộc tính này sẽ tạo ra một lỗi khi ta cố gắng cập nhật cơ sở dữ liệu. Điều đặc biệt ở chỗ lỗi này không phải là một phần trong tập hợp *Errors*, ta có thể bắt chúng và đáp ứng bằng cách sử dụng đối tượng *Err*. Thành viên cuối cùng của tập hợp *Errors* cũng sẽ xuất hiện trong đối tượng *Err*. Bẫy lỗi logic ở đoạn cuối của thủ tục này sẽ in ra hai dòng chứa mã lỗi và mô tả lỗi.

```

Sub OpenLookOnlyErrors()
Dim cnn1 As New Connection
Dim rsCustomers As Recordset
Dim errLoop As Error, intInErrors As Integer
On Error GoTo LookOnlyTrap
    cnn1.Mode = adModeRead
    cnn1.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=C:\Program Files\Microsoft Office\Office\" & _
        "Samples\Northwind.mdb;"
'Spell Northwind incorrectly to generate trappable error.
'    cnn1.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
'        "Data Source=C:\Program Files\Microsoft Office\Office\" & _
'        "Samples\Northwinds.mdb;"
'No Errors element for faulty provider spelling
'    cnn1.Open "Provider=Microsoft.Jets.OLEDB.4.0;" & _
'        "Data Source=C:\Program Files\Microsoft Office\Office\" & _
'        "Samples\Northwind.mdb;"

Set rsCustomers = New ADODB.Recordset
rsCustomers.ActiveConnection = cnn1
'Spell rsCustomers incorrectly to make a 424 non-trappable error.
'Spell cnn1 as cnn to make 3001 non-trappable error.
'    rsCustomer.ActiveConnection = cnn1
'Spell table name as "Customer" to make -2147217900 trappable error.
    rsCustomers.Open "Customers"
'adModeRead setting for cnn1.Mode causes an error (3251) here.
'Comment out cnn1.Mode line to enable updates.
    rsCustomers.Fields("CustomerID") = "xxxxx"

```

```
rsCustomers.Update
```

```
Debug.Print rsCustomers.Fields("CustomerID")
```

```
rsCustomers.Close
```

```
LookOnlyTrap :
```

```
intInErrors = 0
```

```
For Each errLoop In cnn1.Errors
```

```
    Debug.Print errLoop.Number, errLoop.Description
```

```
    intInErrors = intInErrors + 1
```

```
Next errLoop
```

```
If intInErrors = 0 Then
```

```
    Debug.Print Err.Number, Err.Description
```

```
End If
```

```
End Sub
```

Thủ tục *OpenLookOnly Errors* tạo ra một số dạng lỗi khác nhau và cố gắng ghi vào kết nối chỉ đọc. Hình 5.12 chỉ ra đoạn mã VBE và cửa sổ hiện hành cho những lỗi đó, thông báo với những mã lỗi từ tập các lỗi. Các lỗi còn lại là các lỗi ADO được thông báo qua đối tượng Err. Chỉ có hai trong số các lỗi có số âm lớn trong tập *Errors*. Một lỗi trong xâu kết nối (lỗi số 3706) không thông báo qua tập *Errors*.

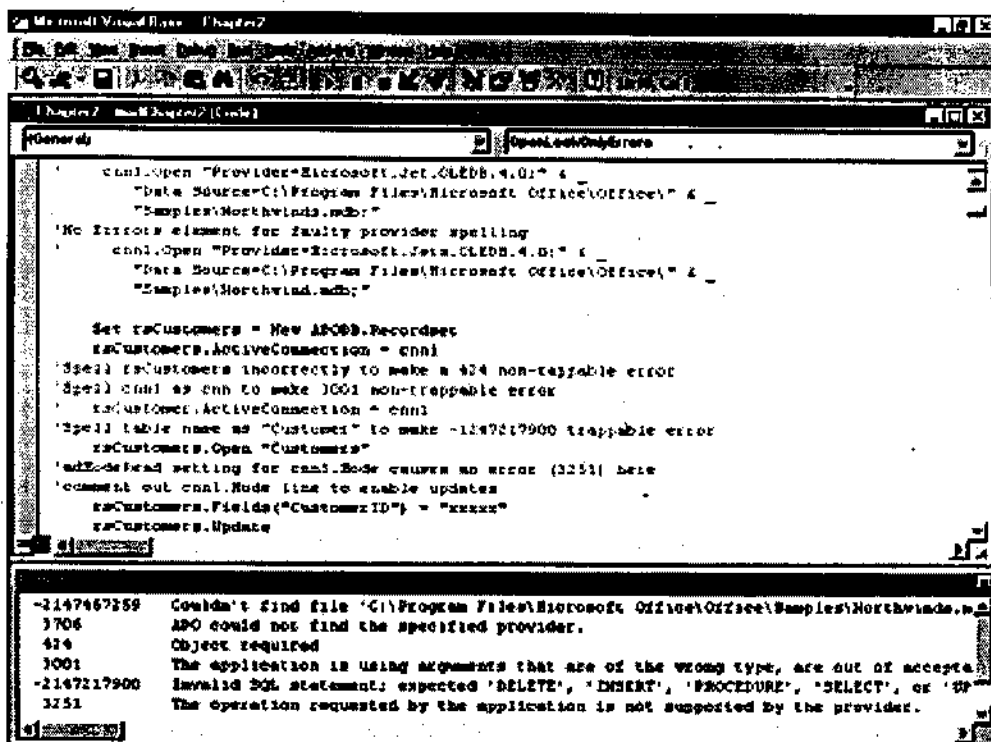
Chú ý : Ta có thể chèn một khai báo *Option Explicit* trong vùng khai báo chung của một module để loại đi khả năng gây ra lỗi như việc tham chiếu đến một đối tượng không tồn tại.

Thủ tục *LoopToUsingErrors* sau đây sẽ đưa ra một số cách tiếp cận tới xử lý lỗi với đối tượng Err. Thủ tục sinh ra một lỗi và bao gồm một chú thích miêu tả sự thay đổi tới mã cần thiết tạo ra một lỗi khác.

– Trong trường hợp lỗi 3251, thủ tục thay đổi kiểu khoá do đó tập bản ghi có thể cập nhật được. Lỗi này xảy ra vì kiểu khoá bị lỗi. Để sửa vấn đề này, mã xử lý lỗi đóng tập bản ghi cũ, khởi tạo lại thuộc tính LockType và mở lại đối tượng tập bản ghi.

– Với lỗi 424, thủ tục không cố sửa lỗi nhưng thông báo cho người sử dụng khả năng gây ra lỗi. Lỗi này xảy ra khi một phương thức được gọi hay một thuộc tính đặt cho một giá trị không được khai báo như một đối tượng. Ví dụ, một lỗi typographical có thể gây ra lỗi này.

– Nếu lỗi không phải là 3251 hay 424, thủ tục in ra số và thuộc tính miêu tả cho đối tượng *Err*.



Hình 5.12. Mã VBE và cửa sổ thực thi chỉ ra mã lỗi từ một kiểu lỗi điển hình.

```
Sub LoopToUsingErrors()
```

```
On Error GoTo DErrorsTrap
```

```
Dim cnn1 As Connection
```

```
Dim rsMyTable As Recordset
```

```
Set cnn1 = New ADOConnection
```

```
cnn1.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
"Data Source=C:\Program Files\Microsoft Office\Office\" & _  
"Samples\Northwind.mdb;"
```

```
Set rsMyTable = New ADODB.Recordset
```

```
'Open recordset with defaults.
```

```
OpenRSMYTable:
```

```
rsMyTable.Open "MyTable", cnn1
```

```
'Loop through recordset.
```

```
Do Until rsMyTable.EOF
```

```
'Make 424 error by using next instead of preceding line.
```

```
' Do Until rsMyTable.EOF
```

```
If rsMyTable.Fields(0) = 4 Then
```

```
'This line makes 3251 error because recordset is read-only.
```

```
rsMyTable.Fields(0) = 88
```

```
Else
```

```
Debug.Print rsMyTable.Fields(0), rsMyTable.Fields(1)
```

```
End If
```

```
rsMyTable.MoveNext
```

```
Loop
```

```
rsMyTable.Close
```

```
ErrorsExit:
```

```
Exit Sub
```

```
DErrorsTrap:
```

```
If Err.Number = 3251 Then
```

```
MsgBox "OLEDB Provider does not support operation. " & _
```

```
"Find another way to get the job done or get a new " & _
```

```

    "OLEDB Provider. Error happened in LoopToDeleteErrors."
    Debug.Print rsMyTable.LockType
    rsMyTable.Close
    rsMyTable.LockType = adLockOptimistic
    Resume OpenRSMMyTable
ElseIf Err.Number = 424 Then
    MsgBox "The code tried to do something requiring an " & _
        "object, such as set a property or invoke a method, " & _
        "but the code did not have an object. Check spelling."
Else
    MsgBox "Check Immediate window for error # and desc."
    Debug.Print Err.Number, Err.Description
End If
Resume ErrorsExit

End Sub

```

5.5. CÂU HỎI ÔN TẬP – BÀI TẬP

1. Xem kỹ ví dụ 5.11. Hãy soạn đoạn chương trình tương đương để mở tệp cơ sở dữ liệu Quanlysinhvien.mdb.
2. Sử dụng đối tượng DAO để truy nhập tệp CSDL quanlysinhvien.mdb và hiển thị các bản ghi của một bảng nào đó trong tệp này.
3. Sử dụng đối tượng ADO để truy nhập tệp CSDL quanlysinhvien.mdb và hiển thị các bản ghi của một bảng nào đó trong tệp này.

Chương VI

SỬ DỤNG CƠ SỞ DỮ LIỆU ACCESS TRONG VISUAL BASIC

6.1. GIỚI THIỆU NGÔN NGỮ VISUAL BASIC

BASIC là viết tắt của Beginner's All – purpose Symbolic Instruction Code do John Kemeny và Thomas Kurtz thuộc trường Dartmouth viết ra năm 1964 với mục đích phục vụ cho những người mới học tiếp cận một cách dễ dàng ngôn ngữ lập trình. Basic đã trải qua nhiều thế hệ : BASICA, GW – BASIC, QBASIC, QuickBASIC, đặc biệt Basic đã được hãng Microsoft của Bill Gate hỗ trợ và phát triển thành Visual Basic (VB).

Phần Visual đề cập đến các phương pháp thiết lập giao diện đồ hoạ cho người sử dụng (GUI). Với những bộ phận hình ảnh trực quan có sẵn (gọi là các Controls), ta có thể bố trí sắp đặt vị trí và quyết định các đặc tính của chúng trên một khung màn hình (gọi là Form). Người viết phần Visual cho VB là Alan Cooper của hãng Microsoft, ông đã làm cho môi trường hoạt động của Basic một mặt trở nên dễ hiểu, dễ sử dụng, không rắc rối như Windows mặt khác vẫn sử dụng được các chức năng của Windows một cách hiệu quả. Đến nay VB đã trải qua các thế hệ 3.0, 4.0, 5.0, 6.0 và gần đây là VB.NET. Các ấn phẩm của VB gồm 3 loại :

1. Learning Edition, dùng cho học tập.
2. Professional Edition, dùng cho các nhà lập trình chuyên nghiệp.
3. Enterprise Edition, thương phẩm cho các công ty, xí nghiệp.

Ngoài VB còn có 2 dạng khác :

– **VBA** (Visual Basic for Application) : là ngôn ngữ nằm sau các chương trình Office như WORD, EXCEL, MS PROJECT, ACCESS ..., còn gọi là các Macros dùng để tăng chức năng của các ứng dụng trên bằng cách tự động hoá một số tiến trình.

– **VBScript** : Dùng để xây dựng các ứng dụng web (ASP) cũng như cho chính hệ điều hành.

Tuy đơn giản, dễ học, dễ dùng và cho phép xây dựng ứng dụng nhanh chóng, tiện lợi nhưng VB cũng có thể được dùng để phát triển những hệ thống mạnh, đồ sộ. Ví dụ : nhiều trò chơi được viết bằng VB, giới doanh nghiệp dùng VB để quản trị CSDL, các nhà quản trị mạng dùng VB để xây dựng các trang WEB và chúng ta có thể dùng VB để xây dựng các loại hình ứng dụng windows cần thiết.

6.1.1. Các đặc điểm chính của VB

- Môi trường lập trình trực quan cho phép xây dựng ứng dụng windows nhanh chóng, dễ dàng.
- Lập trình viên thiết kế giao diện đồ hoạ bằng cách lấy các biểu tượng thành phần (control) từ hộp đồ nghề (toolbox), rồi viết mã lệnh cho mỗi thành phần.

Các thủ tục được gọi tự động khi người sử dụng tạo ra các sự kiện tương ứng.

VB là ngôn ngữ lập trình trực quan bậc cao, hướng sự kiện, có một phần của lập trình hướng sự kiện, đa năng vừa là biên dịch vừa là thông dịch. Giới hạn của VB là chỉ làm việc trong môi trường Windows.

6.1.2. Các thành phần cơ bản của VB

- *Control tools* – Các công cụ điều khiển để tạo giao diện.
- *Editor* – Bộ soạn thảo để thảo chương.
- *Debugger* – Bộ gỡ rối, để kiểm tra, tìm và sửa lỗi cho ứng dụng.
- *Compiler* – Biên dịch để tạo chương trình chạy được hoặc các ứng dụng độc lập.

6.1.3. Kiến trúc của VB (hình 6.1)

Khi đang thiết kế, xây dựng, ứng dụng trong VB gọi là dự án (Project), sau khi hoàn thiện và bàn giao sử dụng, gọi là ứng dụng hoàn chỉnh. Trong môi trường tích hợp phát triển ứng dụng VB, một dự án có ba trạng thái : Thiết kế (Design mode), thực hiện (Run mode) và tạm dừng để gỡ rối (Break mode). Từ dự án cho đến ứng dụng hoàn chỉnh là quá trình lặp đi lặp lại của 3 trạng thái trên.

1. Về mặt hình thức, một dự án VB có cấu trúc phân cấp như sau :

- Project : dự án.
- Form(s) : biểu mẫu.
- Code module(s) : các mô đun chứa các hàm, mã.
- Class module(s) : các mô đun chứa các lớp .
- Resource file(s) : các tệp tài nguyên.
- Control(s) : điều khiển.

- Properties : thuộc tính.
- Event procedures : hàm.
- Methods : phương thức.

+ Controls (*Điều khiển*) : là các phần tử giao diện người sử dụng Windows (button, text box, label, ...)

+ Form (*Mẫu biểu*) : khung chứa các controls ; window hoặc dialog box

+ Objects (*Đối tượng*) : các controls, forms ... là các đối tượng với các thuộc tính, phương thức khác nhau, đồng thời mỗi đối tượng lại có các bộ sự kiện khác nhau.

+ Property (*Thuộc tính*) : xác định đặc tính của đối tượng (color, size, name, caption ...)

+ Event (*Sự kiện*) : là hành vi nhận biết bởi đối tượng (mouse click, load, system error,...)

Ví dụ : Command1_Click

+ Event procedures (*Thủ tục sự kiện*) : là các câu lệnh được thực hiện khi sự kiện tương ứng xảy ra với đối tượng.

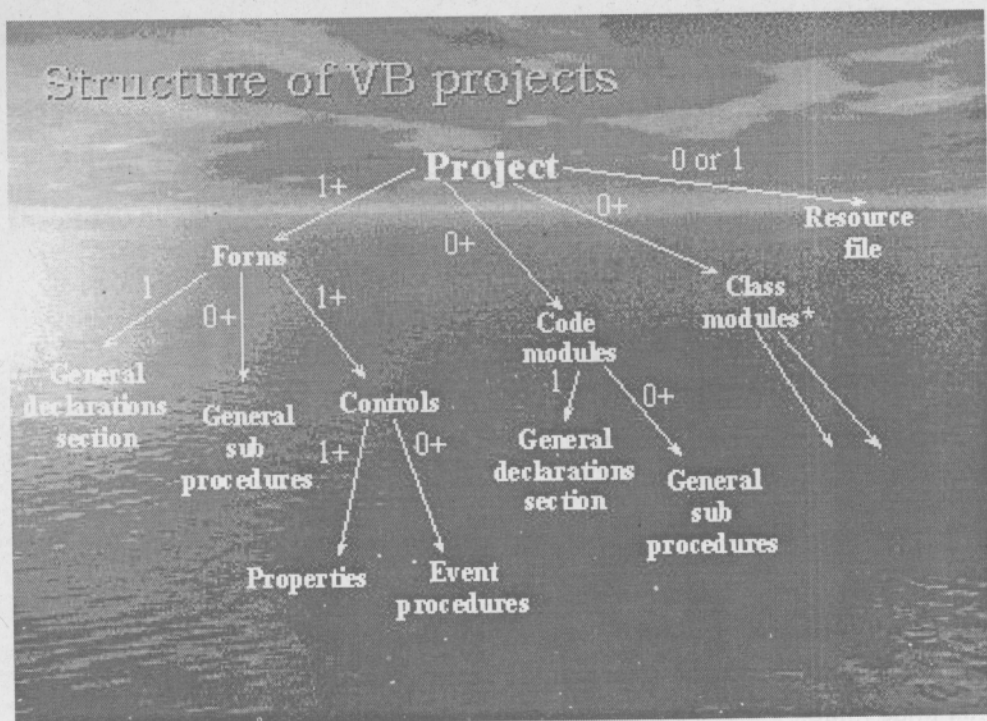
Ví dụ : End

+ Instruction (*Câu lệnh*) : Trong VB cho phép trên một dòng viết 1 câu lệnh. Nếu muốn viết các câu lệnh trên một dòng, các lệnh cần phân cách bởi dấu :

+ Statement (*Lệnh*) : từ khoá để yêu cầu thực hiện 1 việc gì đó (end, print).

+ Method (*Phương thức*) : là đoạn mã riêng biệt gắn với một hành động trên đối tượng tương ứng.

+ Syntax : object.method



Hình 6.1. Cấu trúc của các dự án VB.

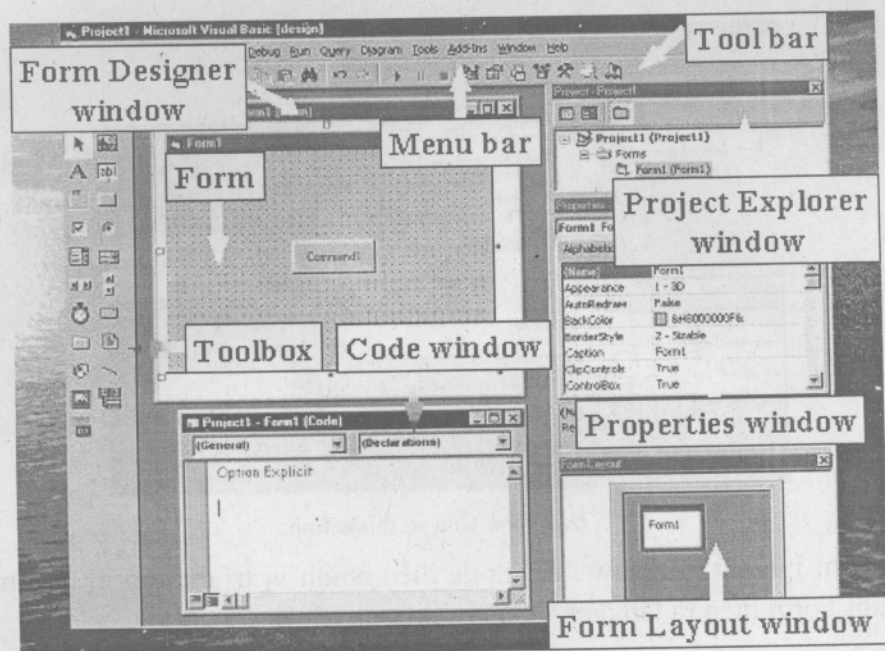
2. Theo chức năng các Controls chia làm 7 loại :

- Trigger (command button, menu) : các đối tượng sẵn sàng kích hoạt.
- Input (text box, option button) : các đối tượng phục vụ nhập dữ liệu.
- Output (label, image) : các đối tượng phục vụ hiển thị dữ liệu.
- Organize (form, frame) : các đối tượng chứa các đối tượng khác.
- Beautify (line, shape) : các đối tượng trang trí.
- Data access (data, datagrid) : các đối tượng sử dụng để truy nhập CSDL
- Integrate (OLE) : các đối tượng kết hợp.

Môi trường phát triển ứng dụng tích hợp IDE (Integrated Development Environment)

IDE của VB 6.0 được mô tả trên hình 6.2, gồm các thành phần sau :

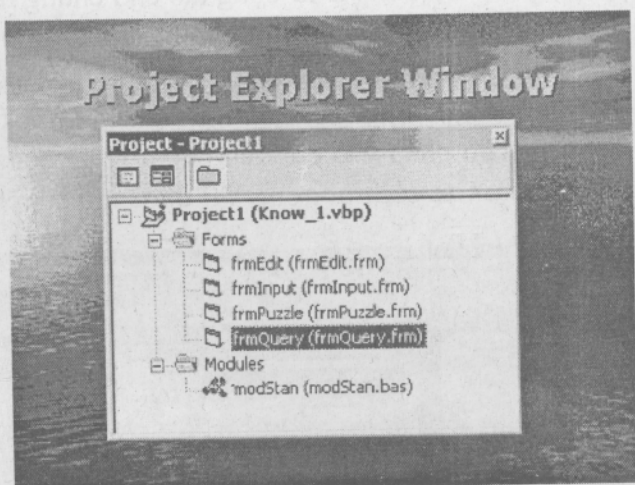
- Menu Bar : Chứa đầy đủ các chức năng để làm việc với VB 6.0, cho phép tác động, quản lý trực tiếp trên toàn bộ dự án.



Hình 6.2. Màn hình VB.

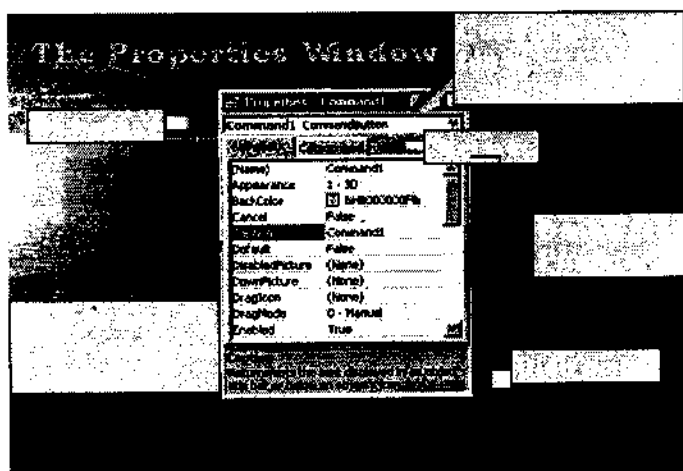
– Tool Bar : Là tập hợp các nút bấm mang biểu tượng (Icons) đảm nhiệm một số chức năng thông dụng nhưng nhanh và tiện lợi hơn menu.

– Project Explorer Window : Để quản trị các thành phần của dự án. Chi tiết của sổ này được mô tả trên hình 6.3 :



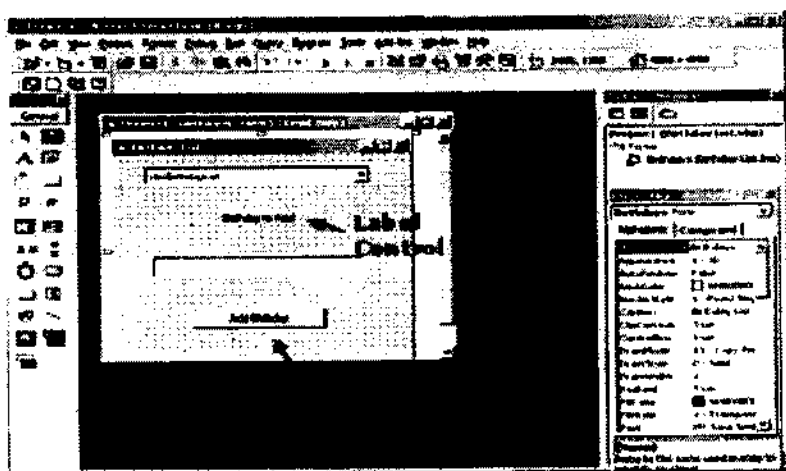
Hình 6.3. Màn hình Project Explorer Window.

– Properties Window (hình 6.4) : Cửa sổ thuộc tính, liệt kê các thuộc tính của đối tượng đang thiết kế. Khi thay đổi 1 thuộc tính ta sẽ thấy ngay kết quả trên đối tượng đang chọn.



Hình 6.4. Cửa sổ thuộc tính.

- Form Layout Window : Dùng để điều chỉnh vị trí của Form trên màn hình khi Form hiện ra lần đầu lúc chạy chương trình
- Form Designer Window : Cửa sổ thiết kế form, dùng để thiết kế giao diện.
- Form : Là thành phần gần như không thể thiếu của ứng dụng VB, khi bắt đầu một dự án, VB tự động tạo cho chúng ta một form có tên là Form1. Khi chương trình chạy, form chính sẽ hiện ra để đợi một sự kiện do người sử dụng tạo ra. Sẽ không thao tác được trên một form trống do vậy ta cần đặt thêm vào form các Controls như Textbox, Command button ... Các Controls cho phép vào dữ kiện để xử lý và cũng chính các controls sẽ hiện thị kết quả. Các control trong VB được mô tả trên hình 6.5.



Hình 6.5. Các control trong VB.

6.2. TẠO GIAO DIỆN CHO CHƯƠNG TRÌNH VB

Khi đã tạo được các form trong Access thành thạo, người sử dụng cũng rất dễ dàng tạo giao diện cho chương trình VB. Chỉ khác ở chỗ trong Access để mở thuộc tính của một đối tượng phải nhấn chuột phải và chọn Properties, còn trong VB để chọn đối tượng và thay đổi các thuộc tính cần nhìn vào cửa sổ properties.

6.3. ĐỐI TƯỢNG VÀ CÁCH BIỂU DIỄN ĐỐI TƯỢNG TRONG VB

VB 6.0 là ngôn ngữ lập trình hướng đối tượng mạnh. Các loại đối tượng của VB rất đa dạng, bao gồm nhiều loại :

– *Đối tượng chuẩn* : là các đối tượng có sẵn của VB, chúng gồm các kiểu sau :

CheckBox	ComboBox	CommandButton
Data	DirListBox	DriveListBox
FileListBox	Form	Frame
Grid	HScrollBar	Image
Label	Line	ListBox
Menu	MDIForm	OptionButton
OLE	PictureBox	Shape
TextBox	Timer	VScrollBar

– *Đối tượng từ các thư viện tham chiếu* rất đa dạng và ngày càng mở rộng.

Các đối tượng do người sử dụng tạo ra thông qua cơ chế Class Module. Trong lớp, ta có thể định nghĩa các thuộc tính (thành viên dữ liệu) và các phương thức (hàm thành viên) của một đối tượng.

Những đặc tính chung của đối tượng

– Từng đối tượng có các chức năng tổng quát được định nghĩa đầy đủ nhưng để hiểu để có thể sử dụng được và đồng thời cho phép phát triển khi cần thiết.

– Đối tượng giao tiếp với bên ngoài thông qua các thuộc tính, phương thức và sự kiện được định nghĩa trước cho nó. Tổ hợp của 3 khái niệm này gọi là giao diện (interface). Đó là những yếu tố cần biết về một đối tượng để sử dụng chúng.

– Có thể dùng nhiều đối tượng trong một dự án, đồng thời cũng có nhiều thể hiện khác nhau của một kiểu đối tượng.

– Người sử dụng không quan tâm đến cách lập trình bên trong đối tượng mà chỉ thấy đối tượng điều khiển và thay đổi hoạt động bên trong của chúng sao cho những thay đổi đó không ảnh hưởng đến ứng dụng đang dùng đối tượng, tức là không thay đổi Interface.

Khác với VBA, VB chỉ dùng toán tử chấm (.) để biểu diễn đối tượng, với cú pháp dạng : TênĐốiTượng.TênThuộcTính

hoặc : TênĐốiTượng.TênPhươngThức

Ví dụ 6.1 :

Form1.Caption

TxtName.SetFocus

Form1.Font.Size

MyRS.Fields("Hoten").Value hay MyRS.Fields(0).value

Khi sử dụng các tập hợp mặc định của đối tượng ta sử dụng dấu chấm than (!). Cách này, cùng với việc dùng các thuộc tính mặc định sẽ làm cho chương trình được đơn giản hoá rất nhiều.

Ví dụ 6.2 : Lấy về một giá trị từ thuộc tính Hoten của một RecordSet

HT = MyRS.Fields("Hoten").Value

Ta có thể viết :

HT = MyRS!Hoten.value

Tập hợp Fields là một tập hợp mặc định đối với đối tượng RecordSet nên

HT = MyRS!Hoten (thuộc tính Value là thuộc tính mặc định).

VB cho phép tạo các bản sao (Instance) của một Form. Từng bản sao có các điều khiển và menu như nhau nhưng có những dữ liệu khác nhau. Mặc dù chương trình cũng như tên biến và tên điều khiển như nhau, nhưng dữ liệu được chứa ở những nơi khác nhau trong bộ nhớ. Chính vì vậy việc xác định cửa sổ làm việc là rất cần thiết. Ở những trường hợp như vậy, thay vì sử dụng tên form, ta có thể dùng từ khoá ActiveForm hoặc từ khoá Me, VD :

ActiveForm.TxtHoten = "Hoàng Vũ "

Hoặc thông dụng hơn : Me.TxtHoten = "Hoàng Vũ ".

Một điểm cần lưu ý khi sử dụng biến đối tượng trong VB là phải khai báo tường minh (mặc dù không có Option Explicit). Sở dĩ như vậy là do các đối tượng thường có cấu trúc phức tạp, đòi hỏi nhiều vùng bộ nhớ hơn các kiểu biến khác. Để khai báo biến đối tượng ta sử dụng từ khoá DIM.

Ví dụ 6.3 :

Dim DB As Database

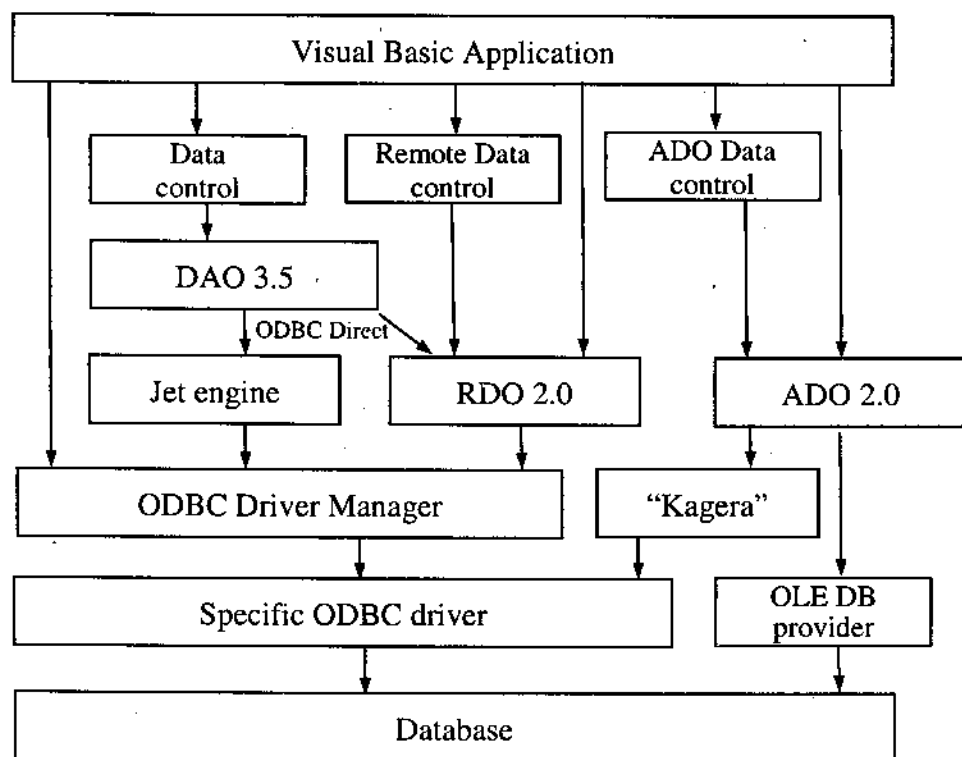
Khi gán một giá trị cho biến đối tượng ta phải sử dụng từ khoá Set

Set DB = OpenDatabase ("C:\QLVT\QLVT.MDB")

Sau khi đã gán, ta có thể lập trình trên biến đối tượng DB, thi hành các phương thức và đọc hoặc đổi giá trị các thuộc tính của nó. Khi không tiếp tục làm việc với CSDL QLVT.MDB, dùng lệnh : Set DB = Nothing để giải phóng biến.

6.4. LẬP TRÌNH CƠ SỞ DỮ LIỆU SỬ DỤNG ĐỐI TƯỢNG DỮ LIỆU DAO

VB có các mô hình truy cập dữ liệu như sơ đồ hình 6.8 :



Hình 6.8. Mô hình truy cập dữ liệu của VB.

Như đã trình bày ở trên, mô hình DAO khá phức tạp nhưng ta vẫn có thể sử dụng DAO khi đã hiểu được một số thuộc tính quan trọng của những đối tượng cơ bản nhất của DAO là : Database, Recordsets và Fields và các tập hợp của chúng.

Để dùng DAO, trước hết phải tham chiếu đến thư viện Microsoft DAO trong menu Project/Referenses của VB. Ta thực hiện theo các bước sau :

1. Từ menu Project, chọn References. Hộp thoại References xuất hiện.
2. Từ danh sách các thư viện, chọn hộp kiểm "Microsoft DAO 3.51 Object Library" (nếu sử dụng CSDL ACCESS 2000 thì chọn DAO 3.6).
3. Nhấn nút OK. Như vậy, ta có thể dùng đối tượng được cung cấp bởi thư viện đối tượng DAO.

6.4.1. Kết nối với một cơ sở dữ liệu

- Để kết nối với một CSDL dùng DAO cần phải khai báo một đối tượng kiểu Database và dùng phương thức OpenDatabase để tạo kết nối.

Dim db As Database

Set db = OpenDatabase("c:\QLVT\QLVT.MDB")

MsgBox " CSDL " & db.name & " Đã được mở. "

- Phương thức OpenDatabase có một tham số bắt buộc – tên CSDL muốn mở và một số tham số tùy chọn. Cú pháp đầy đủ của OpenDatabase là :

OpenDatabase (dbname, [Option], [Readonly], [Connect])

với :

+ Option (*True*) : Độc quyền mở dữ liệu, ngăn không cho người khác mở CSDL. Nếu là *False* : Mở dùng chung

+ Read only (*True*) thì chỉ mở để đọc, không sửa đổi

+ Connect : là một chuỗi chỉ ra cách thức kết nối với CSDL, thường chỉ dùng với những nguồn dữ liệu Client/Server và ODBC.

Chú ý : Opendatabase là một phương thức của đối tượng Workspace của DAO. Để viết một cách đầy đủ ta phải dùng :

DAO.Workspace(0).OpenDatabase, tuy vậy do Workspace(0) là ngầm định, nên để đơn giản, ta bỏ qua.

6.4.2. Dùng phương thức Execute để thi hành truy vấn hành động

Phương thức Execute của đối tượng Database dùng để thi hành một câu lệnh SQL trên CSDL, chủ yếu là để :

- Cập nhật, xoá hay sao chép mẫu tin (Truy vấn hành động)
- Sửa cấu trúc CSDL (Ngôn ngữ định nghĩa dữ liệu DDL)

Đối với các câu truy vấn SELECT sẽ được thi hành qua phương thức OpenRecordset của đối tượng Database (nội dung cụ thể được trình bày trong phần 6.4.3).

Ví dụ 6.4 :

Tăng tiền thưởng của những người có số ngày công đi làm lớn hơn một giá trị vào từ Textbox NCongThuong của Form FrmThuong lên gấp đôi trong một CSDL về tiền lương.

Dim db As Database

Private Sub Form_Load()

set db = OpenDatabase(app.Path & "QLLuong.Mdb")

End Sub

Private Sub CmdOK_click()

db.Execute " Update Soluong Set Thuong = thuong*2 " & _ "where
NCong >= " & Me.NCongThuong

End Sub

Dùng Execute để thi hành lệnh DDL cũng tương tự :

Private Sub CmdCreat_click()

db.Execute " CREAT TABLE tblthem ([Hoten] TEXT (50), " & _ "
[Diachi] TEXT (50))"

end Sub

6.4.3. Phương thức OpenRecordset

Trong DAO các đối tượng Database, Connection, QueryDef, TableDef và Recordset đều có phương thức OpenRecordset để truy cập dữ liệu chứa trong CSDL

Phương thức OpenRecordset luôn trả về một đối tượng Recordset, vì vậy cần khai báo đối tượng Recordset trước khi gọi phương thức này.

Dim db As Database

Dim rs As Recordset

set db = OpenDatabase (app.Path & "QLVT.MDB")

set rs = db.OpenRecordset ("Phieunhap")

Tham số bắt buộc duy nhất của phương thức này là nguồn dữ liệu. Nguồn dữ liệu có thể là tên một bảng hoặc một định nghĩa truy vấn chứa sẵn hoặc cũng có thể là một câu lệnh SELECT SQL.

Ví dụ 6.5 :

...

```
set rs = db.OpenRecordset( " SELECT * FROM dmkh " & _ " ORDE  
BY Tenkh"
```

Bên cạnh tham số bắt buộc là nguồn dữ liệu, có thể chỉ ra tham số tùy chọn Option để xác định kiểu Recordset tức là cách thức thao tác với mẫu tin. Tham số Option có các giá trị được ghi trong bảng 6.1.

Bảng 6.1

HÀNG SỐ	Ý NGHĨA
dbOpenTable	Tạo đối tượng Recordset kiểu bảng : Cập nhật được và nhanh vì có chỉ mục, nhưng không thể trình bày kết quả của một truy vấn nhiều bảng.
dbOpenDynaset	Mở đối tượng kiểu dynaset : Cập nhật được và rất hiệu quả vì nó là tham chiếu đến dữ liệu trong truy vấn, có thể lấy về các mẫu tin từ một hay nhiều bảng, thậm chí từ nhiều CSDL khác nhau. Tuy nhiên cập nhật chậm hơn kiểu bảng vì không dùng chỉ mục.
dbOpenDynamic	Kiểu Dynamic : Cập nhật được, có thể trả về các mẫu tin từ một hay nhiều bảng. Đặc biệt thích hợp với CSDL dùng chung vì có thể tự cập nhật khi những người sử dụng khác sửa đổi mẫu tin chứa trong đó. Tuy nhiên không hiệu quả bằng Dynaset.

dbOpenSnapshot	Kiểu SnapShot : Nhanh hơn bảng và Dynaset, có thể lấy các mẫu tin từ nhiều hơn một bảng và có thể cập nhật được, trừ Microsoft Jet. Khác với Dynaset là tham chiếu, Snapshot trả về một bản sao dữ liệu nên chậm hơn.
dbOpenForwardOnly	Tương tự Snapshot, nhưng nhanh hơn. Chỉ có thể dịch chuyển tiến.

6.4.4. Tham chiếu đến một trường (field)

Đối tượng Field thể hiện một trường trong một cấu trúc dữ liệu. Các đối tượng TableDef, Recordset, QueryDef, Relation và Index đều chứa các tập hợp trường. Ta có thể lấy về giá trị một trường bằng cách kiểm tra giá trị của thuộc tính Value của một đối tượng Field. Mặt khác thuộc tính Value là mặc định của đối tượng Field, nên chỉ cần tham chiếu đến đối tượng đó mà không nhất thiết phải chỉ ra một cách tường minh thuộc tính Value. Ví dụ :

rs!Hovaten rs! [Ho va ten]

rs("Hovaten") rs.fields(1)

6.4.5. Các phương thức định vị (duyệt) bản ghi

Các phương thức duyệt của một đối tượng Recordset gồm có :

MoveFirst : Di chuyển đến bản ghi đầu tiên trong Recordset

MoveNext : Di chuyển đến bản ghi tiếp theo

MovePrevious : Di chuyển về bản ghi trước đó

MoveLast : Di chuyển đến bản ghi cuối cùng

Move : Di chuyển đến một bản ghi được chỉ định trước

Cần chú ý có những Recordset với phương pháp duyệt dữ liệu chỉ cho phép di chuyển tới, khi đó phương thức MoveFirst và MovePrevious sẽ tạo lỗi. Ngoài ra nếu đang ở bản ghi đầu tiên, phương thức MovePrevious sẽ gây lỗi, tương tự đang ở bản ghi cuối cùng, phương thức MoveNext cũng gây lỗi. Để tránh xảy ra lỗi khi dịch chuyển các bản ghi, ta dùng các thuộc tính BOF (Begin Of File – Đầu Tập) và EOF (End Of File – Cuối tập) cũng như RecordCount để đếm số bản ghi.

Ví dụ 6.6 :

Dim db As Database

Dim rs As Recordset

set db = OpenDatabase (app.Path & "QLVT.MDB")

set rs = db.OpenRecordset ("DMKH")

Do Until rs.EOF

Debug.Print rs!Makh; rs!tenKh;rs!Diachi

rs.MoveNext

Loop

rs.MoveFirst

MsgBox "Có tất cả " & rs.RecordCount & " Khách hàng "

Để kiểm tra xem recordset có rỗng hay không ta cần kiểm tra cả hai thuộc tính EOF và BOF. Đồng thời để xác định chính xác số bản ghi trong Recordset, phải dịch chuyển tới bản ghi cuối cùng (nếu không kết quả sẽ không chính xác vì có thể ban đầu JET chỉ trả về một phần các bản ghi đủ để làm việc).

If rs.EOF and rs.BOF Then

MsgBox " Không có bản ghi nào "

Else

Rs.MoveLast

MsgBox "Có tất cả " & rs.RecordCount & " Khách hàng "

End If

6.4.6. Phương thức Find

Để tìm kiếm một bản ghi trong recordset ta sử dụng một trong 4 phương thức tìm kiếm của đối tượng Recordset. Cần có điều kiện tìm kiếm (search criteria) với các phương thức này :

FindFirst (criteria)

FindNext

FindLast

FindPrevious

Trong đó Search criteria là một biểu thức quan hệ (1 string). Ví dụ :
soluong > 25)

Phải dùng chính xác các tên trường của bảng dữ liệu trong biểu thức.

Các strings dùng trong search criteria phải được đặt trong dấu nháy đơn ' '.

Dữ liệu kiểu date dùng trong criteria phải được đặt trong # ..#

Ví dụ 6.7 :

"EmpID < 50"

"EmpID < " & txtID.Text

"EmpName = 'Brian Kane' "

"EmpName Like '*ith' "

EmpID, .. Là tên của các trường trong 1 table.

Ví dụ 6.8 :

Dim sCriteria As String

sCriteria = "EmpName Like '*mas' "

rs.FindFirst(sCriteria)

Ví dụ 6.9 :

Dim Salary As Currency

Salary = Ccur(txtSalary.Text)

sCriteria = "EmpSalary >=" & CStr(Salary)

rs.FindFirst(sCriteria)

Ví dụ 6.10 :

Dim SearchCriteria As String

sCriteria = "HiredDate > #01/01/99# "

rs.FindFirst(sCriteria)

Câu hỏi: Làm thế nào để kiểm tra kết quả tìm kiếm?

Trả lời: Dùng thuộc tính NoMatch của recordset.

Ví dụ 6.11 :

Hiện tên của những người có tienluong >= 150000.

sCriteria = "Tienluong >=150000"

Dim Continue As Boolean

Continue = True

Do While Continue = true

 rs.FindFirst(sCriteria)

If NoMatch Then

 Continue = False

Else

 Call lstEmp.AddItem(datx.Recordset!EmpName)

End If

Loop

Ngoài ra ta có thể dùng phương thức Seek để tìm kiếm theo chỉ mục (chỉ dùng với Recordset kiểu bảng và bảng phải có chỉ mục).

Ví dụ 6.12 :

VD sau đây cho phép tìm khách hàng có tên chứa trong TextBox TxtName, bảng khách hàng có chỉ mục trên trường Hoten và tên tệp chỉ mục là HotenIndex.

Dim db As Database

Dim rs As Recordset

Private Sub Form_load()

 set db = OpenDatabase (app.Path & "QLVT.MDB")

 set rs = db.OpenRecordset ("DMKH", dbOpenTable)

end sub

Private sub CmdSeek_click()

 rs.Index = "HotenIndex"

 rs.Seek "=", txtHoten.Text

 If rs.NoMatch Then

 MsgBox " Không có khách hàng tên là " & txtHoten.Text

 Else

 MsgBox rs!Hoten & ", " & rs!Diachi

 End If

End Sub

Chú ý trong đoạn mã lệnh trên ta dùng tham số dbOpenTable để mở Recordset kiểu bảng. Phương thức Seek chỉ cho phép tìm kiếm trên một trường duy nhất và khi dùng Seek, ta chỉ được dùng các toán tử so sánh :

- = Bằng
- < Nhỏ hơn
- <= Nhỏ hơn hoặc bằng
- > Lớn hơn
- >= Lớn hơn hoặc bằng

6.4.7. Cập nhật Recordset

Đối tượng RecordSet có các phương thức để cập nhật : Sửa, thêm hay xóa một bản ghi. Tất cả các phương thức này đòi hỏi 2 bước :

1. Thực hiện việc thay đổi
2. Lưu sự thay đổi vào CSDL

* *Sửa đổi nội dung bản ghi*, ta theo các bước sau :

1. Dùng các phương thức duyệt của Recordset để di chuyển đến bản ghi cần sửa đổi

2. Gọi phương thức Edit của Recordset : rs.Edit

3. Gán giá trị mới cho các trường của Recordset

rs!hoten = "Trần Văn An"

rs!Diachi = " 1. Đại Cồ Việt – Hà nội"

.....

4. Lưu bản ghi đã sửa vào CSDL bằng cách dùng phương thức Update : rs.Update

* *Để thêm bản ghi mới vào Recordset*, ta theo các bước sau :

1. Gọi phương thức AddNew của Recordset : rs.AddNew

2. Gán giá trị cho bản ghi mới bằng cách dùng câu lệnh gán các trường tương ứng của Recordset

3. Gọi phương thức Update để lưu bản ghi vào CSDL

* *Để xóa một bản ghi của Recordset*, ta theo các bước sau :

1. Tìm, định vị tới bản ghi cần xóa.

2. Gọi phương thức Delete, xoá bản ghi hiện tại.

3. Bản ghi bị xoá vẫn còn trong CSDL, nhưng không truy cập được. Để loại bỏ hẳn ta cần dịch chuyển sang một bản ghi hợp lệ khác. Kiểm tra nếu EOF và BOF đều là true, khi đó recordset là rỗng, không có bản ghi nào.

```
Private Sub CmdDel_Click()  
    on error goto DelErr  
    With rs  
        .Delete  
        .MoveNext  
        If . EOF then .MoveLast  
    Exit Sub  
End With  
DelErr :  
    MsgBox Err.Description  
    Exit Sub  
End Sub
```

6.4.8. Giải phóng đối tượng

Sau khi dùng xong đối tượng cần đóng và giải phóng đối tượng để tránh các xung đột có thể xảy ra và giải phóng bộ nhớ.

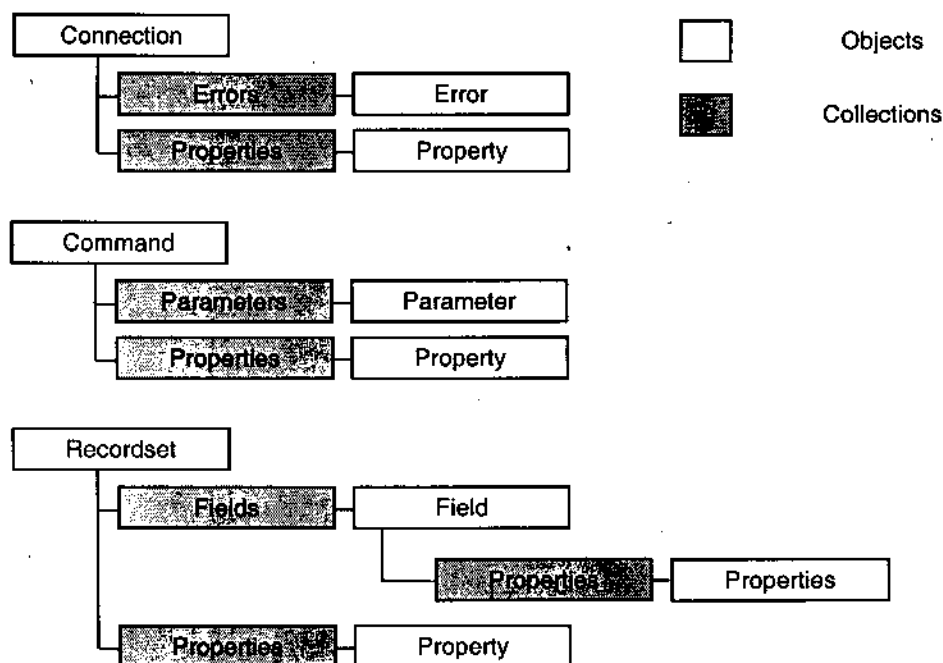
```
rs.Close    ' Đóng Recordset  
set rs = Nothing ' Giải phóng bộ nhớ  
db.close  
set db = Nothing
```

6.5. LẬP TRÌNH CƠ SỞ DỮ LIỆU SỬ DỤNG ĐỐI TƯỢNG DỮ LIỆU ADO

6.5.1. Mô hình ADO

Visual Basic 6 cho ta sự lựa chọn về kỹ thuật khi lập trình với database, hoặc là sử dụng DAO như trong phần trước, hoặc là dùng ADO (ActiveX Data Objects). Sự khác biệt cơ bản giữa ADO và DAO là ADO cho phép ta

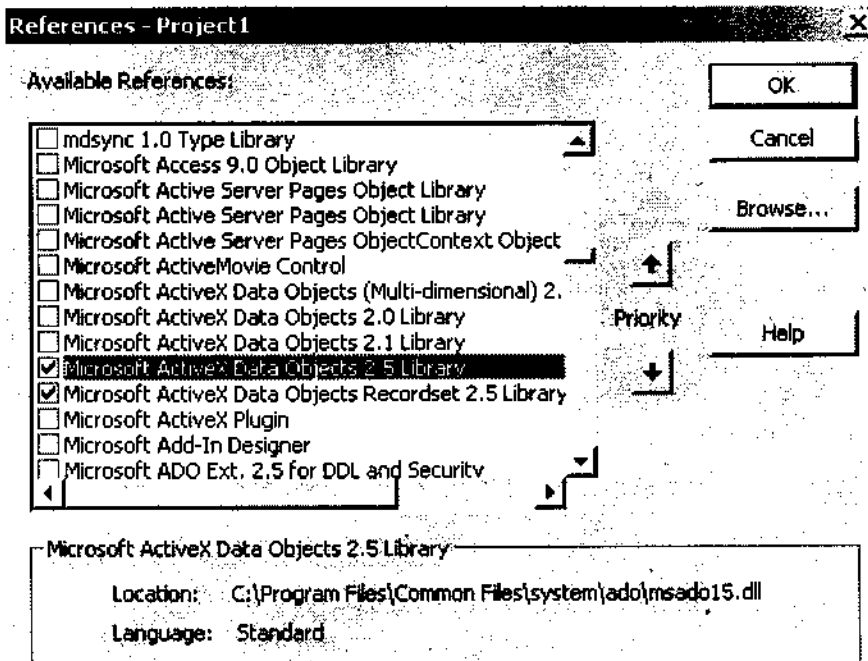
làm việc với mọi loại nguồn dữ kiện (data sources), không nhất thiết phải là Access database hay ODBC. Nguồn dữ kiện có thể là danh sách các địa chỉ Email, hay một file text string, trong đó mỗi hàng là một record gồm những fields ngăn cách bởi các dấu phẩy (comma separated values). ADO đặc biệt phù hợp với mô hình dữ liệu phân tán.



Hình 6.9. Mô hình truy cập dữ liệu ADO.

Nếu trong DAO sử dụng trực tiếp của MSAccess Database thì trong ADO cho phép *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 logo một database... Sau đó ta có thể *lấy về (extract)* những recordsets và cập nhật hóa các records bằng cách sử dụng những *lệnh SQL* trên các tables hay những *stored procedures* bên trong database.

Khi mới khởi động một dự án VB6 mới, Control Data ADO thường không có sẵn trong IDE. Để control data ADO xuất hiện, sử dụng Menu *Command Project | Components...*, rồi chọn *Microsoft ADO Data Control 6.0 (OLEDB)* từ giao diện Components như hình 6.10 :



Hình 6.10. Giao diện Components.

6.5.2. Điều khiển ADODC

ADODC là yếu tố cơ bản trong việc tạo ra các ứng dụng truy cập CSDL với khối lượng mã nguồn tối thiểu.

ADODC cung cấp :

- Một liên kết giữa ứng dụng và dữ liệu ta cần truy cập.
- Một giao diện trực quan để thiết lập các thuộc tính truy cập dữ liệu.

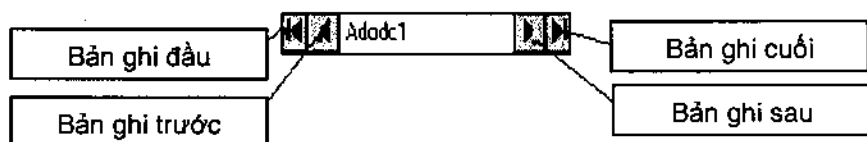
Bằng cách bind (gắn) các điều khiển với ADODC, dữ liệu có thể hiển thị mà không cần viết mã.

Các điều khiển có thể làm việc với ADODC :

Checkbox, Textbox, CheckBox, ComboBox, Image, Label, ListBox, PictureBox và TextBox controls.

DataCombo, DataGrid, Chart và DataList.

Để thêm ADODC vào Toolbox ta chọn “Microsoft ADO Data Control 6.0” trong mục Components của menu Project, khi đó trên Toolbox sẽ xuất hiện thêm điều khiển ADODC (hình 6.11).

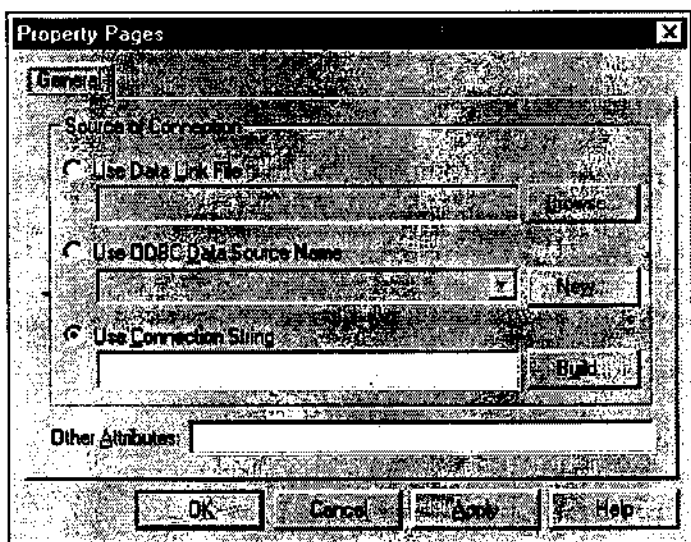


Hình 6.11. Điều khiển ADODC.

Để kết nối thanh ADODC với nguồn dữ liệu (datasource) có nhiều cách như sau :

– Thuộc tính connectionString của ADODC

- + Tạo ra kết nối đến data source
- + Chỉ định kiểu và vị trí của CSDL cần truy nhập.
- + Mở cửa sổ thuộc tính trong cửa sổ properties (hình 6.12).

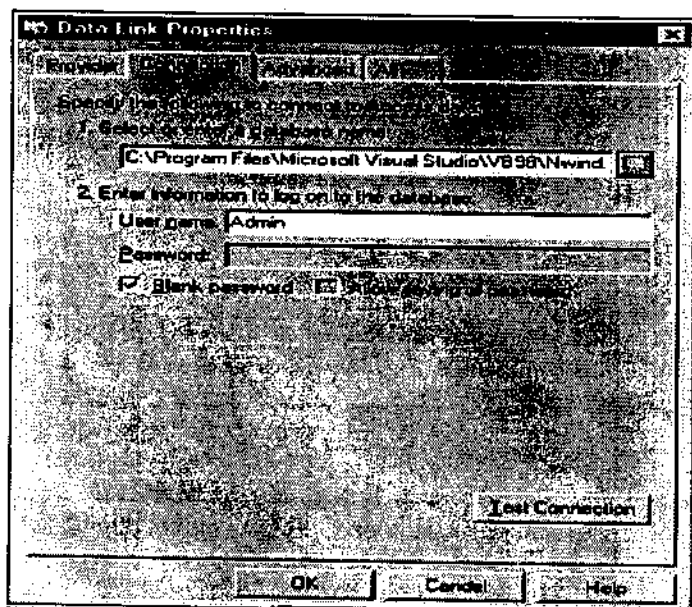
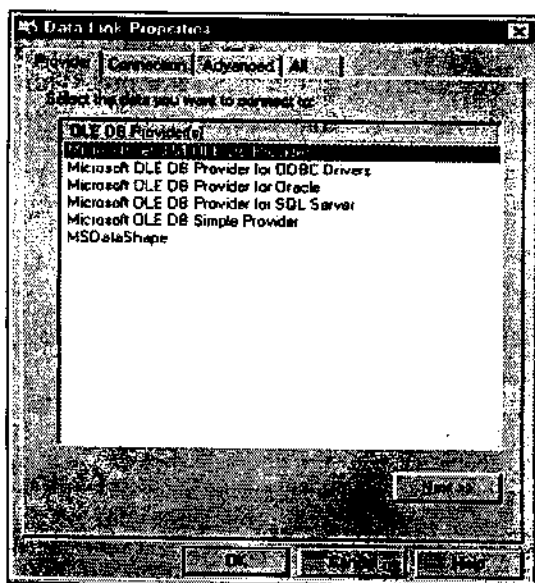


Hình 6.12. Cửa sổ thuộc tính.

– Sử dụng File Data Link

- + Chỉ định một kết nối tùy biến đến một data source
- + Data Link file được tạo ra bằng Data Environment. Browse
- + Sử dụng ODBC Data Source Name

- + Cho phép sử dụng DSN để tạo connection string.
 - + Hộp chọn hiện ra các DSN của hệ thống để người dùng chọn.
- *Sử dụng Connection String*
- + Hiện thị hộp thoại Datalink (hình 6.13) thông qua nút lệnh Build
 - + Chỉ định tên của Provider, Connection và các thông tin khác.



Hình 6.13. Data Link Properties

6.5.3. Lập trình với ADO

1. Tạo kết nối (Connection)

ADO đặc biệt linh hoạt trong việc kết nối với CSDL, có nhiều cách khác nhau, ví dụ, có thể tạo một đối tượng Recordset độc lập và gán một chuỗi kết nối (connection string) vào phương thức *Open* của đối tượng. Hoặc có thể tạo một đối tượng Command độc lập và gán chuỗi kết nối cho thuộc tính *ActiveConnection* của nó. Bất kể đối tượng dùng để kết nối với một nguồn dữ liệu ADO là gì đều cần xây dựng chuỗi kết nối. Chuỗi đó có thể gán cho thuộc tính *ConnectionString* của đối tượng Connection hoặc thuộc tính *ActiveConnection* của đối tượng Recordset hay Command, hoặc để truyền cho phương thức *Execute* của đối tượng Connection hay phương thức *Open* của đối tượng Recordset.

Chuỗi connection có thể chứa một hay nhiều phần tử dưới dạng *argument = value*. Danh sách các tham số phụ thuộc nhà cung cấp CSDL mà ta kết nối đến, nhưng ít nhất phải có 2 tham số là : *Provider* - Nhà cung cấp và *File Name* - Tên file. Dưới đây là danh sách các tham số cơ bản dùng cho chuỗi kết nối :

ARGUMENT	DIỄN GIẢI
<i>Data Source</i>	Tên của nguồn CSDL cần kết nối.
<i>DSN</i>	Tên nguồn ODBC đã được đăng ký trên máy ; tham số này thay cho tên nguồn CSDL <i>Data Source</i> .
<i>Filename</i>	Tên tệp chứa thông tin về kết nối ; có thể là tệp ODBC DSN hoặc tệp Microsoft Data Link (UDL).
<i>Initial Catalog</i>	Tên của CSDL ngầm định. Khi kết nối với nguồn ODBC, có thể dùng tham số <i>Database</i> .
<i>Password</i>	Mật khẩu. Với nguồn ODBC, có thể dùng tham số <i>PWD</i> . (Cần phải truyền user ID và password khi kết nối với SQL Server).
<i>Persist Security Info</i>	Là True nếu ADO lưu user ID và password trong chuỗi kết nối.
<i>Provider</i>	Tên của nhà cung cấp OLE DB ; Ngầm định là MSDASQL, nhà cung cấp nguồn ODBC.
<i>User ID</i>	Tên người sử dụng.

Xác định những tham số nào phù hợp với nhà cung cấp CSDL là một việc không đơn giản đối với người lập trình. Cách tốt nhất là tạo một điều khiển ADODC trên một Form, thiết lập kết nối trên trang thuộc tính của nó rồi lấy kết quả cuối cùng trong thuộc tính *ConnectionString* đưa vào chương trình.

Nếu sử dụng nguồn ODBC, ta chỉ cần chỉ ra tên của CSDL với tham số *Data Source* hoặc *DSN*.

Ví dụ 6.13 :

Mở tệp Authors của CSDL Pubs trên Microsoft SQL Server mà không tạo đối tượng Connection :

```
Dim rs As New ADODB.Recordset
```

```
rs.Open "Authors", "Provider=MSDASQL.1;User ID=sa;Data Source=Pubs"
```

Hoặc có thể viết gọn hơn (vì ngầm định)

```
rs.Open "Authors", "DSN=Pubs"
```

Khi kết nối với Microsoft Jet database chỉ cần chỉ ra tên của nhà cung cấp và tên CSDL với đầy đủ đường dẫn :

```
cn.Open "Provider=Microsoft.Jet.OLEDB.3.51;" _
```

```
& "Data Source=E:\Microsoft Visual Studio\VB98\Biblio.mdb"
```

Với CSDL SQL Server thì công việc phức tạp hơn ; sau đây là kết nối đến CSDL Pubs của SQL Server nằm trên máy trạm MyServer dùng cơ chế bảo mật Windows NT :

```
cn.Open "Provider=SQLOLEDB.1;Integrated Security=SSPI;" _
```

```
& "Data Source=MyServer;Initial Catalog=Pubs;"
```

Trong trường hợp này, *Data Source* là tên của server và xác định cơ chế bảo mật bằng cách gán giá trị SSPI cho tham số *Integrated Security*. Lệnh sau mở một kết nối đến CSDL SQL Server trên nhưng dùng tường minh user ID và password:

```
cn.Open "Provider=SQLOLEDB.1;Data Source=MyServer;User ID=sa;" _
```

```
& "Password=mypwd;Initial Catalog=Pubs"
```

Connection Timeout cũng là một thuộc tính hữu dụng của chuỗi kết nối, thường dùng với phương thức *Open* của *Recordset* hay *Execute* của *Command* :

```
rs.Open "Authors", "Provider=SQLOLEDB.1;Data Source=MyServer;User ID=sa;" _  
& "Connection Timeout=10;Initial Catalog=Pubs"
```

Nếu kết nối thông qua nhà cung cấp ngầm định *MSDASQL* provider thì cần xác định thêm hàng loạt tham số phụ khác, trong đó quan trọng nhất là *ODBC driver* :

```
cn.ConnectionString = "Provider=MSDASQL.1;User ID=sa;" _  
& "ExtendedProperties=""DRIVER= SQL Server;SERVER=ServerNT;" _  
& "MODE=Read;WSID=P2;DATABASE=pubs"""
```

2. Mở Connection

Sau khi đã tạo chuỗi kết nối, cần tạo lệnh thực sự mở kết nối tùy theo đối tượng sử dụng.

Các đối tượng kết nối tường minh (explicit Connection objects)

Với đối tượng *Connection* độc lập ta phải chuẩn bị trước cho việc kết nối, ví dụ :

' Chuẩn bị cho một kết nối chỉ đọc read – only connection.

```
Dim cn As New ADODB.Connection
```

```
cn.ConnectionTimeout = 30 ' Ngầm định = 15 seconds.
```

```
cn.Mode = adModeRead ' Ngầm định = adModeUnknown.
```

Sau đó ta có hàng loạt cách để mở kết nối. Có thể gán chuỗi kết nối cho thuộc tính *ConnectionString* rồi gọi bằng phương thức *Open* :

```
cn.ConnectionString="Provider=SQLOLEDB.1;Data Source=MyServer;" _  
& "Initial Catalog=Pubs"
```

' The second and third arguments are the user name and the password.

```
cn.Open , "sa", "mypwd"
```

Hoặc có thể truyền chuỗi kết nối như tham số thứ nhất của phương thức *Open* :

```
cn.Open "Provider=SQLOLEDB.1;Data Source=MyServer;User ID=sa;" _  
    & "Password=mypwd;Initial Catalog=Pubs"
```

Không sử dụng tường minh đối tượng kết nối (Implicit Connection objects)

Ta có thể bỏ qua đối tượng Connection, mở kết nối trực tiếp trong các đối tượng khác, ví dụ : dùng phương thức *Open* của Recordset

```
Dim rs As New ADODB.Recordset  
rs.Open "Authors", "Provider=SQLOLEDB.1;Data Source=MyServer;User  
ID=sa;" _  
    & "Password=mypwd;Initial Catalog=Pubs"
```

Tương tự với đối tượng Command, nhưng cần gán trước vào các thuộc tính *ActiveConnection* và *CommandText* rồi dùng phương thức *Execute* :

```
Dim cmd As New ADODB.Command  
cmd.ActiveConnection = "Provider=SQLOLEDB.1;Data Source=  
    MyServer;" _& "userID=sa;Password=mypwd;Initial Catalog=Pubs"  
cmd.CommandText = "DELETE FROM Authors WHERE State =  
    'WA'"  
cmd.Execute
```

Với cách trên, bạn vẫn có thể truy cập đến đối tượng *active connection* qua thuộc tính *ActiveConnection* của *Recordset* hay *Command* :

' Display the errors in the connection created by the previous example.

```
Dim er As ADODB.Error
```

```
For Each er In cmd.ActiveConnection.Errors
```

```
    Debug.Print er.Number, er.Description
```

```
Next
```

3. Mở một đối tượng Recordset

Trước khi mở một Recordset, cần xác định những bản ghi nào cần lấy, kiểu bản ghi là gì, vị trí ...

Thuộc tính *source* (nguồn dữ liệu) : Là thuộc tính quan trọng nhất của đối tượng Recordset, nó chỉ ra những bản ghi nào cần lấy, có thể là tên bảng hay query, tên của thủ tục chứa sẵn hay câu lệnh SQL SELECT.

Ví dụ 6.14 :

' Chọn các nguồn khác nhau dựa trên mảng các nút lựa chọn option buttons.

```
Dim rs As New ADODB.Recordset
```

```
If optSource(0).Value Then      ' Database table
```

```
    rs.Source = "Authors"
```

```
ElseIf optSource(1).Value Then  ' Stored procedure
```

```
    rs.Source = "reptq1"
```

```
ElseIf optSource(2) Then        ' SQL query
```

```
    rs.Source = "SELECT * FROM Authors" WHERE au_lname LIKE 'A*'"
```

```
End If
```

Khi mở một Recordset, phải xác định các đặc điểm của kết nối bằng các cách sau :

- Tạo đối tượng Connection độc lập với các thuộc tính cần thiết (connection timeout, user name and password...). Mở đối tượng đó và gán cho thuộc tính *ActiveConnection* của recordset trước khi mở Recordset.

- Tạo đối tượng Connection độc lập và truyền nó như tham số thứ hai trong phương thức *Open* của Recordset.

- Truyền chuỗi connection như tham số thứ hai trong phương thức *Open* của Recordset (ADO sẽ tự tạo đối tượng Connection ẩn – không tường minh).

Tạo đối tượng Connection độc lập và truyền vào tham số thứ nhất của phương thức *Execute* sau đó gán kết quả cho biến Recordset.

Ví dụ 6.15 :

Sau đây là một số cách mở bảng Authors của CSDL Pubs từ SQL Server có tên P2:

' Cách 1: Tạo đối tượng Connection độc lập, gán cho thuộc tính Active Connection

```
Dim cn As New ADODB.Connection, rs As New ADODB.Recordset
```

```
cn.ConnectionTimeout = 5
```

```
cn.Open "Provider=sqloledb;Data Source=P2;Initial Catalog=pubs;", "sa"
```

```
Set rs.ActiveConnection = cn
```

```
rs.Open "Authors"
```

' Cách 2 : Tạo đối tượng Connection độc lập, truyền cho phương thức Open.

```
Dim cn As New ADODB.Connection, rs As New ADODB.Recordset
```

```
cn.ConnectionTimeout = 5
```

```
cn.Open "Provider=sqloledb;Data Source=P2;Initial Catalog=pubs;", "sa"
```

```
rs.Open "Authors", cn
```

' Cách 3: Dùng phương thức Open của Recordset, tạo Connection ẩn. (chú ý cần thêm các thuộc tính của connection như connection timeout và user ID trong connection string).

```
Dim rs As New ADODB.Recordset
```

```
rs.Open "Authors", "Provider=sqloledb;Data Source=P2;" _
```

```
& "Initial Catalog=pubs;User ID=sa;Connection Timeout=10"
```

' Cách 4: Dùng phương thức Execute của đối tượng Connection, theo ngầm định nó sẽ tạo một Recordset ở máy khách kiểu forward – only, read – only với CacheSize=1.

```
Dim cn As New ADODB.Connection, rs As New ADODB.Recordset
```

```
cn.Open "Provider=sqloledb;Data Source=P2;Initial Catalog=pubs;", "sa"
```

```
Set rs = cn.Execute("Authors")
```

' Mở 1 Recordset mới với cùng connection "rs".

Dim rs2 As New ADODB.Recordset

rs2.Open "Publishers", rs.ActiveConnection

' Chọn từ các nguồn khác nhau dựa trên mảng các nút lựa chọn.

If optSource(0).Value Then ' Database table

rs.Open "Publishers", , , , adCmdTable

Else optSource(1).Value Then ' Stored procedure

rs.Open "reptq1", , , , adCmdStoredProc

ElseIf optSource(2) Then ' SQL query

rs.Open "SELECT * FROM Authors", , , , adCmdText

End If

' Mở tập các bản ghi ở máy chủ (server – side) theo kiểu dynamic.

' (Giả sử thuộc tính ActiveConnection đã được xác lập.)

rs.CursorType = adOpenDynamic

rs.Open "SELECT * FROM Authors", , , , adCmdText

' Mở ở máy chủ kiểu keyset bằng một lệnh đơn.

rs.Open "SELECT * FROM Authors", , , adOpenKeyset,
adLockOptimistic, adCmdText

Cũng có thể tạo tập bản ghi tĩnh (static Cursors) ở máy khách (client – side) bằng cách đặt thuộc tính *CursorLocation* = adUseClient trước khi mở Recordset.

' Mở 1 client – side static cursor.

rs.CursorLocation = adUseClient

rs.CursorType = adOpenStatic ' Có thể thay đổi.

rs.Open "SELECT * FROM Authors", , , , adCmdText

4. Đối tượng Recordset độc lập (hình 6.14)

Đối tượng Recordset của ADO linh hoạt hơn nhiều so với của DAO và RDO, thậm chí không cần mở kết nối. Thực ra, ADO hỗ trợ 2 kiểu Recordsets : stand – alone Recordsets và file – based Recordsets.

Recordsets tương đối đơn giản : tạo một đối tượng Recordset mới, thêm một hay nhiều trường vào tập hợp Fields của đối tượng sau đó mở nó và kết quả là một client – side Recordset kiểu tĩnh (static cursor) và optimistic batch locking :

' Tạo 1 Recordset độc lập với 3 fields

Dim rs As New ADODB.Recordset

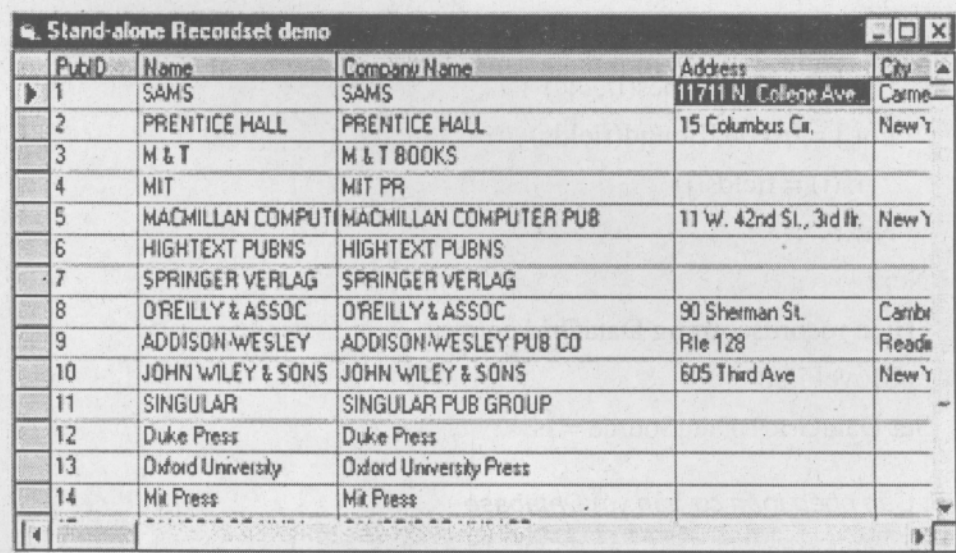
rs.Fields.Append "FirstName", adChar, 40, adFldIsNullable

rs.Fields.Append "LastName", adChar, 40, adFldIsNullable

rs.Fields.Append "BirthDate", adDate

rs.Open

Sau khi đã mở Recordset, ta có thể thêm các bản ghi vào đó, thậm chí gán nó cho một điều khiển ADO Data hoặc vào thuộc tính *DataSource* của một bound control. Điều này cho phép kết nối với một điều khiển data thuộc mọi kiểu, kể cả khi nó không được lưu trong CSDL. Ví dụ, để hiện một file TEXT, ta dùng dấu chấm phẩy làm phân cách trong một điều khiển DataGrid.



PubID	Name	Company Name	Address	City
1	SAMS	SAMS	11711 N. College Ave	Carme
2	PRENTICE HALL	PRENTICE HALL	15 Columbus Cir.	New Y
3	M & T	M & T BOOKS		
4	MIT	MIT PR		
5	MACMILLAN COMPUT	MACMILLAN COMPUTER PUB	11 W. 42nd St., 3rd flr.	New Y
6	HIGHTEXT PUBNS	HIGHTEXT PUBNS		
7	SPRINGER VERLAG	SPRINGER VERLAG		
8	O'REILLY & ASSOC	O'REILLY & ASSOC	90 Sherman St.	Cambr
9	ADDISON-WESLEY	ADDISON-WESLEY PUB CO	Rte 128	Readi
10	JOHN WILEY & SONS	JOHN WILEY & SONS	605 Third Ave	New Y
11	SINGULAR	SINGULAR PUB GROUP		
12	Duke Press	Duke Press		
13	Oxford University	Oxford University Press		
14	Mit Press	Mit Press		

Hình 6.14. Một đối tượng cơ sở dữ liệu.

Ví dụ 6.16 :

```
Dim rs As New ADODB.Recordset
Dim lines() As String, fields() As String
Dim i As Long, j As Long

' Mở file text Publishers.dat .
Open "Publishers.dat" For Input As #1
' Đọc nội dung và cắt từng dòng.
lines() = Split(Input(LOF(1), 1), vbCrLf)
Close #1
' Xử lý dòng đầu tiên : chứa danh sách các trường.
fields() = Split(lines(0), ";")
For j = 0 To UBound(fields)
    rs.Fields.Append fields(j), adChar, 200
Next
rs.Open

' Xử lý các dòng còn lại, đưa vào Recordset.
For i = 1 To UBound(lines)
    rs.AddNew
    fields() = Split(lines(i), ";")
    For j = 0 To UBound(fields)
        rs(j) = fields(j)
    Next
Next
' Hiện recordset trong DataGrid control.
rs.MoveFirst
Set DataGrid1.DataSource = rs
```

5. Các phép toán cơ bản với Database

Mục đích của việc kết nối CSDL là để đọc nội dung của chúng và cập nhật chúng. ADO cung cấp nhiều cách để thực hiện việc đó.

- *Đọc bản ghi (Read records)* : Sau khi tạo Recordset, ta đọc nội dung bằng cấu trúc *Do...Loop* như sau :

' Đưa toàn bộ tên tác giả vào 1 list box.

Dim rs As New ADODB.Recordset

rs.Open "Authors", "Provider=sqloledb;Data Source=P2;" _

& "Initial Catalog=pubs;User ID=sa;Connection Timeout=10"

Do Until rs.EOF

lstAuthors.AddItem rs("au_fname") & " " & rs("au_lname")

rs.MoveNext

Loop

rs.Close

- *Chèn, xóa, cập nhật bản ghi (Insert, delete, update records)* : Nếu Recordset thuộc loại cập nhật được, ta có thể thêm records mới bằng phương thức *AddNew*. Dùng phương thức *Supports* để xác định có thêm records mới được hay không :

If rs.Supports(adAddNew) Then. . .

rs.AddNew

rs.Fields("Name") = "MSPress"

rs.Fields("City") = "Seattle"

rs.Fields("State") = "WA"

If MsgBox("Bạn có muốn thêm hay không?", vbYesNo) = vbYes Then

rs.Update

Else

rs.CancelUpdate

End If

'Một cách khác so với DAO là ta có thể dùng tập hợp Fields để gán trước giá trị, ví dụ :

Dim fieldNames(0 To fieldMax) As Variant

```
For j = 0 To fieldMax
```

```
    fieldNames(j) = fields(j)
```

```
Next
```

```
' Xử lý file text, nhưng dùng mảng giá trị trong AddNew.
```

```
For i = 1 To UBound(lines)
```

```
    fields() = Split(lines(i), ";")
```

```
    ReDim fieldValues(0 To fieldMax) As Variant
```

```
    For j = 0 To UBound(fields)
```

```
        fieldValues(j) = fields(j) ' Move values into the Variant arrays.
```

```
    Next
```

```
    rs.AddNew fieldNames(), fieldValues()
```

```
Next
```

ADO cho phép thay đổi giá trị các trường mà không cần dùng chế độ EDIT :

```
' Tăng đơn giá 10%.
```

```
Do Until rs.EOF
```

```
    rs("Dongia") = rs("Dongia") * 1.1
```

```
    rs.MoveNext
```

```
Loop
```

+ Cú pháp của phương thức *Delete* rất đơn giản : Tùy theo tham số truyền cho phương thức, ta có thể xóa bản ghi hiện tại (ngầm định) hoặc tất cả các bản ghi chỉ ra bởi thuộc tính lọc *Filter*. Tuy nhiên cần chú ý sau khi gọi phương thức này, các bản ghi vẫn còn nhưng không truy cập được, phải dịch chuyển con trỏ bản ghi sang một vị trí hợp lệ khác.

```
rs.Delete
```

```
rs.MoveNext
```

```
If rs.EOF Then rs.MoveLast
```

+ Cập nhật thông qua các câu lệnh SQL

Nếu đang làm việc với các bản ghi chỉ đọc (read – only Recordset), ta có thể cập nhật một dòng nào đó bằng lệnh UPDATE :

' Hỏi xem có muốn sửa giá các mặt hàng có đơn giá > \$40.

Dim rs As New ADODB.Recordset, cn As New ADODB.Connection

Dim newValue As String

cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _

& "Data Source=C:\Program Files\Microsoft Visual
Studio\VB98\NWind.mdb"

rs.Open "Products", cn

Do Until rs.EOF

If rs("UnitPrice") > 40 Then

newValue = InputBox("Insert a new price for product " & _
rs("ProductName"), , rs("UnitPrice"))

If Len(newValue) Then

cn.Execute "UPDATE Products SET UnitPrice=" & newValue & _
" WHERE ProductID =" & rs("ProductID")

End If

End If

rs.MoveNext

Loop

+ Xoá một record dùng lệnh SQL cũng tương tự, nhưng sử dụng lệnh DELETE :

' Xoá các nhà cung cấp từ Italy, theo quyết định của người sử dụng :

Dim rs As New ADODB.Recordset, cn As New ADODB.Connection

cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _

& "Data Source=E:\Microsoft Visual Studio\VB98\NWind.mdb"

rs.Open "Suppliers", cn

Do Until rs.EOF

If rs("Country") = "Italy" Then

```

If MsgBox("Do you want to delete supplier " & rs("Company
Name")) _
    & "?", vbYesNo) = vbYes Then
    cn.Execute "DELETE FROM Suppliers WHERE SupplierID =" _
        & rs("SupplierID")
End If
End If
rs.MoveNext
Loop

```

+ Để thêm các bản ghi mới ta dùng lệnh INSERT INTO :

```

cn.Execute "INSERT INTO Employees (LastName, FirstName,
BirthDate) " _
    & "VALUES ('Smith', 'Robert', '2/12/1953')".

```

Cần chú ý khi nhận các giá trị từ các điều khiển , ví dụ :

```

cn.Execute "INSERT INTO Employees (LastName, FirstName,
BirthDate) " _
    & "VALUES ('" & txtLastName & "', '" & txtFirstName _
    & "', '" & txtBirthDate & "')"

```

Đoạn mã trên có thể viết cách khác đơn giản hơn nếu dùng thủ tục *ReplaceParams* :

```

sql = "INSERT INTO Employees (LastName, FirstName, BirthDate) " _
    & "VALUES ('@1', '@2', '@3')
cn.Execute ReplaceParams(sql, txtLastName, txtFirstName,
txtBirthDate)

```

6. Sử dụng đối tượng Command

Đối tượng Command là lựa chọn thích hợp nhất khi thực hiện các câu truy vấn có tham số đồng thời là cách duy nhất để gọi các thủ tục chứa sẵn có tham số và trả về các giá trị.

– *Action commands* : Một đối tượng Command đại diện cho một lệnh thực hiện trên nguồn số liệu. Trước khi thực hiện một lệnh, cần phải kết nối với CSDL tuy nhiên cũng tương tự Recordset, có thể khai báo ngầm định đối tượng Connection bằng cách thiết lập giá trị cho thuộc tính *ActiveConnection* của đối tượng Command độc lập. Sau đây là ví dụ về một lệnh đơn để thực hiện lệnh UPDATE SQL trên bảng Publishers của CSDL Pubs trong SQL Server :

' Xác lập các giá trị cho thuộc tính của đối tượng Command.

```
Dim cmd As New ADODB.Command
```

```
cmd.CommandText = "UPDATE Publishers SET city = 'London' " _  
& "WHERE Pub_ID = '9999'"
```

```
cmd.CommandTimeout = 10
```

```
cmd.CommandType = adCmdText
```

' Lưu lệnh để sử dụng sau này khi cần sẽ mở kết nối và gọi cho lệnh thực hiện.

```
Dim cn As New ADODB.Connection
```

```
Dim recs As Long
```

```
cn.Open "Provider=sqloledb;Data source=p2;user id=sa;initial  
catalog=pubs"
```

```
Set cmd.ActiveConnection = cn
```

```
cmd.Execute recs
```

```
Print "RecordsAffected = " & recs
```

Ngoài ra cũng có thể gán chuỗi kết nối cho thuộc tính *ActiveConnection* của đối tượng Command, khi đó ADO sẽ tạo kết nối không tường minh :

```
cmd.ActiveConnection = "Provider=sqloledb;Data Source=p2;User  
Id=sa;" _  
& "Initial Catalog=pubs"
```

```
cmd.Execute recs
```

– *Thực hiện các truy vấn tuyến chọn* : Có thể dùng đối tượng Command để chạy một truy vấn tuyến chọn theo ba cách khác nhau. Cả ba cách đều

cho kết quả tương tự, tuy nhiên tùy từng nhiệm vụ hay phong cách lập trình mà lựa chọn.

Cách 1 : *gán một đối tượng Recordset cho giá trị trả về từ phương thức Execute của đối tượng Command.*

' Giả sử các thuộc tính của Command's đã được xác lập chính xác.

Dim rs As ADODB.Recordset.

cmd.CommandText = "SELECT * FROM Publishers WHERE country = 'USA'"

Set rs = cmd.Execute

' Từ đây, Recordset được mở.

Cách 2 : *gán đối tượng Command cho thuộc tính Source của recordset*

Set rs.Source = cmd

rs.Open

Cách 3 : *dùng phương thức Open của Recordset*

rs.Open cmd

- *Các truy vấn và lệnh có tham số* : Qua các ví dụ trên, đối tượng Command chưa thể hiện được ưu thế so với các câu lệnh SQL, chỉ với các query có tham số thì thế mạnh thực sự của đối tượng Command mới được thể hiện. Ví dụ, để chọn các nhà xuất bản theo tên nước, ta làm như sau :

Dim cmd As New ADODB.Command, rs As ADODB.Recordset

cmd.ActiveConnection = "Provider=sqloledb;Data source=p2;user id=sa;" _
& "initial catalog=pubs"

' Dùng dấu ? làm chỗ chứa cho tham số.

cmd.CommandText = "SELECT * FROM Publishers WHERE country = ?"

' Có thể truyền tham số cụ thể khi thực hiện lệnh.

Set rs = cmd.Execute("USA", adCmdText)

+ Khi dùng nhiều tham số, phải truyền các giá trị của chúng vào mảng kiểu Variants, khi đó sử dụng hàm Array :

```
cmd.CommandText = "SELECT * FROM Publishers WHERE country = ? "_  
& " AND Pub_Name LIKE ?"
```

' Chú ý rằng toán tử LIKE theo cú pháp của SQL Server .

```
Set rs = cmd.Execute(, Array("USA", "N%"), adCmdText)
```

+ Chương trình sẽ mạch lạc hơn khi sử dụng tập hợp tham số :
Parameters Collection:

```
cmd.Parameters.Refresh ' Tạo tập hợp (optional).
```

```
cmd.Parameters(0) = "USA"
```

```
cmd.Parameters(1) = "N%"
```

```
Set rs = cmd.Execute()
```

+ Cũng có thể dùng phương thức CreateParameter của đối tượng Command để tạo các tham số :

' Tạo tập hợp tham số (Chỉ làm 1 lần)

With cmd.Parameters

```
.Append cmd.CreateParameter("Country", adChar, adParamInput, 20)
```

```
.Append cmd.CreateParameter("Name", adChar, adParamInput, 20)
```

End With

' Gán giá trị cho tham số.

```
cmd.Parameters("Country") = "USA"
```

```
cmd.Parameters("Name") = "N%"
```

```
Set rs = cmd.Execute()
```

6.6. CÂU HỎI ÔN TẬP – BÀI TẬP

1. Tạo một form. Lập các thao tác để kết nối với bảng trong tệp cơ sở dữ liệu Quanlysinhvien.mdb
2. Xây dựng form màn hình tương tự như bài tập 1 chương 2 để lấy các thông tin về học sinh, lớp

MỤC LỤC

	<i>Trang</i>
<i>Lời giới thiệu</i>	3
<i>Mở đầu</i>	4
Chương I. GIỚI THIỆU HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ACCESS	5
1.1. Khái niệm và các tính năng của hệ quản trị cơ sở dữ liệu quan hệ	5
1.2. Giới thiệu hệ quản trị cơ sở dữ liệu ACCESS	5
1.3. Hệ thống menu chính của ACCESS	8
1.4. Cách tổ chức dữ liệu trong ACCESS	9
1.5. Công cụ Wizard và Builder	10
1.6. Các ví dụ	11
1.7. Câu hỏi ôn tập – bài tập	13
Chương II. TẠO LẬP CƠ SỞ DỮ LIỆU	14
2.1. Tạo bảng	14
2.2. Thiết kế các kiểm soát dữ liệu trên bảng	17
2.3. Quan hệ giữa các bảng	22
2.4. Tạo truy vấn bằng ngôn ngữ QBE	26
2.5. Tạo truy vấn bằng ngôn ngữ SQL	39
2.6. Câu hỏi ôn tập – bài tập	40

Chương III. THIẾT KẾ BIỂU MẪU VÀ BÁO CÁO	44
3.1. Giới thiệu biểu mẫu	44
3.2. Thiết kế các điều khiển trên form	47
3.3. Biểu mẫu liên quan đến nhiều bảng – biểu mẫu chính phụ	57
3.4. Giới thiệu Report Wizard	59
3.5. Thiết kế các thành phần của báo cáo	61
3.6. Báo cáo chính – phụ	64
3.7. Sắp xếp, phân nhóm trên báo cáo	65
3.8. Câu hỏi ôn tập – bài tập	66
 Chương IV. LẬP TRÌNH TRÊN ACCESS	 69
4.1. Lập trình đơn giản bằng Macro – Ứng dụng của Macro	69
4.2. Giới thiệu ngôn ngữ VBA và phương pháp lập trình hướng sự kiện	74
4.3. Các yếu tố cơ bản trong chương trình VBA	80
4.4. Các cấu trúc lập trình	87
4.5. Các thao tác trên tệp	94
4.6. Câu hỏi ôn tập – bài tập	94
 Chương V. SỬ DỤNG ĐỐI TƯỢNG TRONG VBA	 96
5.1. Khái niệm đối tượng – Tập hợp đối tượng	96
5.2. Các đối tượng của ACCESS	99
5.3. Đối tượng dữ liệu DAO	103
5.4. Đối tượng dữ liệu ADO	107
5.5. Câu hỏi ôn tập – bài tập	152

Chương VI. SỬ DỤNG CƠ SỞ DỮ LIỆU ACCESS	
TRONG VISUAL BASIC	153
6.1. Giới thiệu ngôn ngữ Visual Basic	153
6.2. Tạo giao diện cho chương trình VB	160
6.3. Đối tượng và cách biểu diễn đối tượng trong VB	160
6.4. Lập trình cơ sở dữ liệu sử dụng đối tượng dữ liệu DAO	162
6.5. Lập trình cơ sở dữ liệu sử dụng đối tượng dữ liệu ADO	171
6.6. Câu hỏi ôn tập – bài tập	191
<i>Mục lục</i>	192

Chịu trách nhiệm xuất bản :

Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI
Phó Tổng Giám đốc kiêm Tổng biên tập VŨ DƯƠNG THỤY

Biên tập nội dung :

BÙI MINH HIỂN

Trình bày bìa :

TÀO THANH HUYỀN

Sửa bản in :

PHƯƠNG NGỌC MINH

Chế bản :

CHU ĐAN NGỌC

GIÁO TRÌNH ACCESS VÀ ỨNG DỤNG

Mã số : 6H163 M4

In 3.000 bản, khổ 16 x 24 cm. Tại Công ty in Thái Nguyên. Số in: 733. Số xuất bản: 65/119-04 CXB. In xong và nộp lưu chiểu tháng 8 năm 2004.



NGÔI SAO BẠCH KIM
CHẤT LƯỢNG
QUỐC TẾ

**TÌM ĐỌC GIÁO TRÌNH DÙNG CHO CÁC TRƯỜNG
ĐÀO TẠO HỆ TRUNG HỌC CHUYÊN NGHIỆP - DẠY NGHỀ
CỦA NHÀ XUẤT BẢN GIÁO DỤC
(NGÀNH ĐIỆN TỬ - TIN HỌC)**

- | | |
|---|---------------------------------------|
| 1. Lĩnh kiện điện tử và ứng dụng | TS. Nguyễn Viết Nguyên |
| 2. Điện tử dân dụng | ThS. Nguyễn Thanh Trà |
| 3. Điện tử công suất | Trần Trọng Minh |
| 4. Mạch điện tử | TS. Đặng Văn Chuyết |
| 5. Kỹ thuật số | TS. Nguyễn Viết Nguyên |
| 7. Kỹ thuật điều khiển | Vũ Quang Hồi |
| 8. Kỹ thuật xung - số | TS. Lương Ngọc Hải |
| 9. Điện tử công nghiệp | Vũ Quang Hồi |
| 10. Toán ứng dụng trong tin học | PGS. TS. Bùi Minh Trí |
| 11. Nhập môn tin học | Tô Văn Nam |
| 12. Cấu trúc máy vi tính và vi xử lý | Lê Hải Sâm - Phạm Thanh Liêm |
| 13. Hệ các chương trình ứng dụng
(Window, Word, Excel) | GVC. Trần Viết Thường - Tô Văn Nam |
| 14. Cơ sở dữ liệu | Tô Văn Nam |
| 15. Lập trình C | GVC. Tiêu Kim Cương |
| 16. Cấu trúc dữ liệu và giải thuật | PGS. TS. Đỗ Xuân Lôi |
| 17. Cài đặt và điều hành mạng | TS. Nguyễn Vũ Sơn |
| 18. Phân tích thiết kế hệ thống | GVC. Tô Văn Nam |
| 19. ACCESS và ứng dụng | TS. Huỳnh Quyết Thắng |
| 20. Sử dụng Corel Draw | Nguyễn Phú Quảng |
| 21. Bảo trì và quản lý phòng máy tính | Phạm Thanh Liêm |
| 22. Kinh tế và quản trị doanh nghiệp
(kinh tế và TCQLSX) | TS. Ngô Xuân Bình - TS. Hoàng Văn Hải |

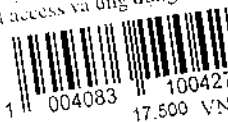
Bạn đọc có thể tìm mua tại các Công ti Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng sách của Nhà xuất bản Giáo dục:

Tại Hà Nội : 25 Hàn Thuyên, 81 Trần Hưng Đạo, 187 Giảng Võ.
23 Tràng Tiền.

Tại Đà Nẵng : 15 Nguyễn Chí Thanh.

Tại Thành phố Hồ Chí Minh : 104 Mai Thị Lựu, Quận

gi access và ứng dụng



Giá: 17.500 đ

