GIÁO TRÌNH

MÔ HÌNH CLIENT/SERVER TRÊN SQL SERVER

MỤC LỤC

BÀI 1
TỔNG QUAN VỀ MÔ HÌNH CLIENT/SERVER

Mã bài : ITPRG3_17.1

Giới thiệu :

Nhìn chung mọi ứng dụng cơ sở dữ liệu đều bao gồm các phần:

- Thành phần xử lý ứng dụng (Application processing components)
- Thành phần phần mềm cơ sở dữ liệu (Database software componets)
- Bản thân cơ sở dữ liệu (The database itself)

Các mô hình về xử lý cơ sở dữ liệu khác nhau là bởi các trường hợp của 3 loại thành phần nói trên định vị ở đâu. Bài này xin giới thiệu các mô hình kiến trúc dựa trên cấu hình phân tán về truy nhập dữ liệu của hệ thống máy tính Client/Server.

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

- Mô tả chính xác các mô hình Client/server
- Xác định chính xác các nguyên tắc mô hình Client/server
- Xác định chính xác các đặc trưng của mô hình Client/server
- Mô tả được các tầng client và server của mô hình Client/server
- So sánh được sự khác nhau giữa mô hình Client/ server và các mô hình cơ sở dữ liệu khác.
- Tư vấn cho khách hàng về ý nghĩa của mô hình Client/Server và lợi ích khi sử dụng cơ sở dữ liệu theo mô hình này.

Nội dung chính:

1.1 Các kiến thức tổng quan về cơ sở dữ liệu.

1.2 Các giai đoạn phát triển của một hệ quản trị cơ sở dữ liệu.

1.3 Giới thiệu về mô hình Client server và các hệ quản trị cơ sở dữ liệu phục vụ cho mô hình Client/Server.

1.4 Các đặc trưng của mô hình Client/server

1.1 Các kiến thức tổng quan về cơ sở dữ liệu.

Ngôn ngữ CSDL được cài đặt khác nhau đối với các hệ quản trị CSDL khác nhau, tuy nhiên đều phải theo một chuẩn (Standard) nhất định. Bài học này sẽ cung cấp cho các học viên các kiến thức cơ bản về ngôn ngữ truy vấn có cấu trúc (Structured Query Language - SQL) CSDL, những cú pháp lệnh đã được chuẩn hóa trong hầu hết các hệ quản trị CSDL (DBMS).

1.2 Các giai đoạn phát triển của một hệ quản trị cơ sở dữ liệu.

Những năm 1975-1976, IBM lần đầu tiên đưa ra hệ quản trị CSDL kiểu quan hệ mang tên SYSTEM-R với ngôn ngữ giao tiếp CSDL là SEQUEL (Structured English QUEry Language), đó một ngôn ngữ con để thao tác với CSDL.

Năm 1976 ngôn ngữ SEQUEL được cải tiến thành SEQUEL2. Khoảng năm 1978-1979 SEQUEL2 được cải tiến và đổi tên thành Ngôn Ngữ Truy Vấn Có Cấu Trúc (Structured Query Language - SQL) và cuối năm 1979 hệ quản trị CSDL được cải tiến thành SYSTEM-R.

Năm 1986 Viện Tiêu Chuẩn Quốc Gia Mỹ (American National Standards Institute - ANSI) đã công nhận và chuẩn hóa ngôn ngữ SQL, và sau đó Tổ chức Tiêu chuẩn Thế giới (International Standards Organization - ISO) cũng đã công nhận ngôn ngữ này. Đó là chuẩn SQL-86.

Tới nay SQL đã qua 3 lần chuẩn hóa lại (1989, 1992, 1996) để mở rộng các phép toán và tăng cường khả năng bảo mật và tính toàn vẹn dữ liệu. Tài liệu này trình bày Ngôn ngữ truy vấn CSDL dựa trên chuẩn SQL-92 và có tham khảo với SQL, SQL*PLUS, PL/SQL của Oracle Server Release 7.3 (1996) và MicroSoft SQL Server 7.1 với các phạm trù nêu trên.

1.3 Giới thiệu về mô hình Client server và các hệ quản trị cơ sở dữ liệu phục vụ cho mô hình Client/Server.

Năm mô hình kiến trúc dựa trên cấu hình phân tán về truy nhập dữ liệu của hệ thống máy tính Client/Server.

- Mô hình cơ sở dữ liệu tập trung (Centralized database model)
- Mô hình cơ sở dữ liệu theo kiểu file - server (File - server database model)
- Mô hình xử lý từng phần cơ sở dữ liệu (Database extract processing model)
- Mô hình cơ sở dữ liệu Client/Server (Client/Server database model)
- Mô hình cơ sở dữ liệu phân tán (Distributed database model)

1.4 Các đặc trưng của mô hình Client/server

Mô hình Client/Server, mà cụ thể trong module này chúng ta sẽ tìm hiểu về mô hình của hệ quản trị cơ sở dữ liệu SQL. SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

1 • **Định nghĩa dữ liệu:** SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu cũng như mối quan hệ giữa các thành phần dữ liệu.

2 • **Truy xuất và thao tác dữ liệu:** Với SQL, người dùng có thể dễ dàng thực hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.

3 • **Điều khiển truy cập:** SQL có thể được sử dụng để cấp phát và kiểm soát các thao tác của người sử dụng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu

1 • **Đảm bảo toàn vẹn dữ liệu:** SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập nhật cũng như các lỗi của hệ thống.

Như vậy, có thể nói rằng SQL là một ngôn ngữ hoàn thiện được sử dụng trong các hệ thống cơ sở dữ liệu và là một thành phần không thể thiếu trong các hệ quản trị cơ sở dữ liệu. Mặc dù SQL không phải là một ngôn ngữ lập trình như C, C++, Java,... song các câu lệnh mà SQL cung cấp có thể được nhúng vào trong các ngôn ngữ lập trình nhằm xây dựng các ứng dụng tương tác với cơ sở dữ liệu.

Khác với các ngôn ngữ lập trình quen thuộc như C, C++, Java,... SQL là ngôn ngữ có tính khai báo. Với SQL, người dùng chỉ cần mô tả các yêu cầu cần phải thực hiện trên cơ

sở dữ liệu mà không cần phải chỉ ra cách thức thực hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ dễ tiếp cận và dễ sử dụng.

Bài tập:

Câu 1: Cho ví dụ về một số hệ quản trị cơ sở dữ liệu theo mô hình Client/Server ?

Câu 2: Hãy trình bày các đặt trưng của mô hình Client/Server?

BÀI 2
CẤU HÌNH CƠ SỞ DỮ LIỆU CLIENT/SERVER
Mã bài : ITPRG3_17.2

Giới thiệu :

Trong module này chúng ta sẽ tìm hiểu về cấu trúc, mô hình Client/Server để từ đó chúng ta có một cái nhìn xâu hơn về hệ thóng Client/Server.

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

- Mô tả được chính xác các tầng cấu trúc
- Diễn đạt được chính xác các mô hình cấu trúc dữ liệu của hệ thống Client/Server

Nội dung chính:

2.1 Tổng quan về cấu trúc Client/Server

2.2 Các tầng cấu trúc

2.3 Các mô hình dữ liệu của hệ thống Client/Server

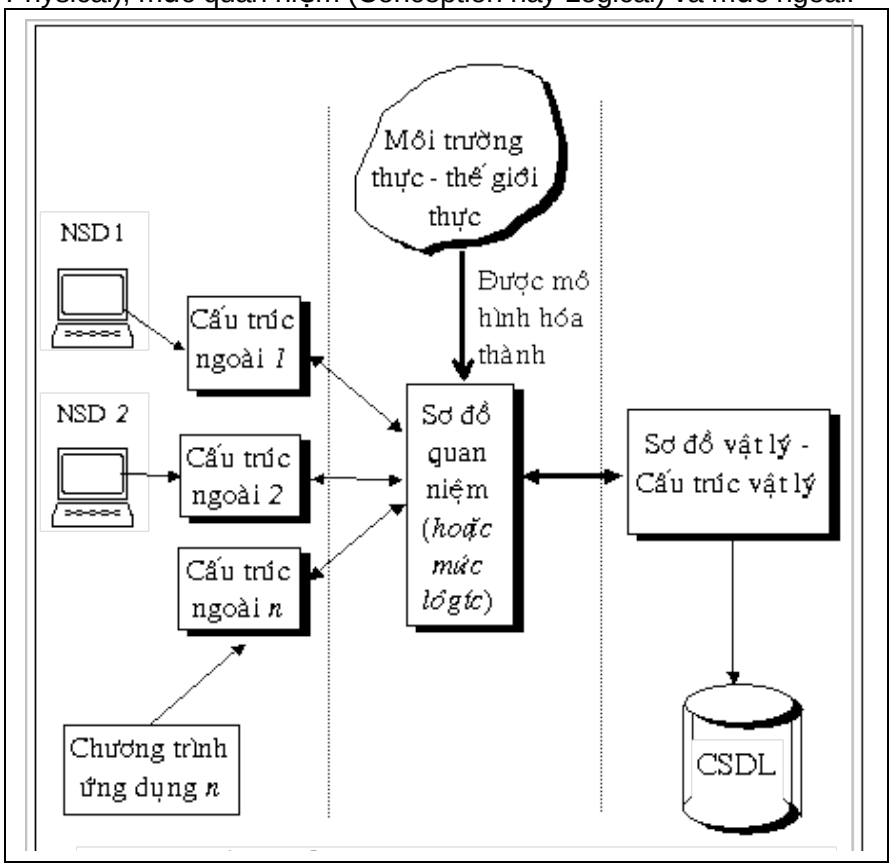
2.1 Tổng quan về cấu trúc Client/Server

Trong mô hình cơ sở dữ liệu Client/Server, cơ sở dữ liệu nằm trên một máy khác với các máy có thành phần xử lý ứng dụng. Nhưng phần mềm cơ sở dữ liệu được tách ra giữa hệ thống Client chạy các chương trình ứng dụng và hệ thống Server lưu trữ cơ sở dữ liệu.

Trong mô hình này, các thành phần xử lý ứng dụng trên hệ thống Client đưa ra yêu cầu cho phần mềm cơ sở dữ liệu trên máy client, phần mềm này sẽ kết nối với phần mềm cơ sở dữ liệu chạy trên Server. Phần mềm cơ sở dữ liệu trên Server sẽ truy nhập vào cơ sở dữ liệu và gửi trả kết quả cho máy Client.

2.2 Các tầng cấu trúc

Theo kiến trúc ANSI-PARC, một CSDL có 3 mức biểu diễn: Mức trong (còn gọi là mức vật lý - Physical), mức quan niệm (Conception hay Logical) và mức ngoài.



Kiến thức tổng quát (ANSI – PARC) của một cơ sở dữ liệu

2.2.1. Mức trong:

Đây là mức lưu trữ CSDL. Tại mức này, vấn đề cần giải quyết là, dữ liệu gì và được lưu trữ như thế nào? ở đâu (đĩa từ, băng từ, track, sector ... nào)? Cần các chỉ mục gì? Việc truy xuất là tuần tự (Sequential Access) hay ngẫu nhiên (Random Access) đối với từng loại dữ liệu.

Những người hiểu và làm việc với CSDL tại mức này là người quản trị CSDL (Administrator), những người sử dụng (NSD) chuyên môn.

2.2.2. Mức quan niệm:

Tại mức này sẽ giải quyết cho câu hỏi CSDL cần phải lưu giữ bao nhiêu loại dữ liệu? đó là những dữ liệu gì? Mối quan hệ giữa các loại dữ liệu này như thế nào?

Từ thế giới thực (Real Universe) các chuyên viên tin học qua quá trình khảo sát và phân tích, cùng với những người sẽ đảm nhận vai trò quản trị CSDL, sẽ xác định được những loại thông tin gì được cho là cần thiết phải đưa vào CSDL, đồng thời mô tả rõ mối liên hệ giữa các thông tin này. Có thể nói cách khác, CSDL mức quan niệm là một sự biểu diễn trừu tượng CSDL mức vật lý; hoặc ngược lại, CSDL vật lý là sự cài đặt cụ thể của CSDL mức quan niệm.

Từ môi trường thế giới thực, xuất phát từ nhu cầu quản lý, việc xác định các loại thông tin cần lưu trữ và các mối quan hệ giữa các thông tin đó như thế nào ... đó chính là công việc ở mức quan niệm.

2.2.3. Mức ngoài.

Đó là mức của người sử dụng và các chương trình ứng dụng. Làm việc tại mức này có các nhà chuyên môn, các kỹ sư tin học và những người sừ dụng không chuyên.

Mỗi người sử dụng hay mỗi chương trình ứng dụng có thể được "nhìn" (View) CSDL theo một góc độ khác nhau. Có thể "nhìn" thấy toàn bộ hay chỉ một phần hoặc chỉ là các thông tin tổng hợp từ CSDL hiện có. Người sử dụng hay chương trình ứng dụng có thể hoàn toàn không được biết về cấu trúc tổ chức lưu trữ thông tin trong CSDL, thậm chí ngay cả tên gọi của các loại dữ liệu hay tên gọi của các thuộc tính. Họ chỉ có thể làm việc trên một phần CSDL theo cách "nhìn" do người quản trị hay chương trình ứng dụng quy định, gọi là khung nhìn (View).

2.3 Các mô hình dữ liệu của hệ thống Client/Server

2.3.1 Mô hình cơ sở dữ liệu tập trung (Centralized database model)

Trong mô hình này, các thành phần xử lý ứng dụng, phần mềm cơ sở dữ liệu và bản thân cơ sở dữ liệu đều ở trên một bộ xử lý.

Ví dụ người dùng máy tính cá nhân có thể chạy các chương trình ứng dụng có sử dụng phần mềm cơ sở dữ liệu Oracle để truy nhập tới cơ sở dữ liệu nằm trên đĩa cứng

của máy tính cá nhân đó. Từ khi các thành phần ứng dụng, phần mềm cơ sở dữ liệu và bản thân cơ sở dữ liệu cùng nằm trên một máy tính thì ứng dụng đã thích hợp với mô hình tập trung.

Hầu hết công việc xử lý luồng thông tin chính được thực hiện bởi nhiều tổ chức mà vẫn phù hợp với mô hình tập trung. Ví dụ một bộ xử lý mainframe chạy phần mềm cơ sở dữ liệu IMS hoặc DB2 của IBM có thể cung cấp cho các trạm làm việc ở các vị trí phân tán sự truy nhập nhanh chóng tới cơ sở dữ liệu trung tâm. Tuy nhiên trong rất nhiều hệ thống như vậy, cả 3 thành phần của ứng dụng cơ sở dữ liệu đều thực hiện trên cùng một máy mainframe do vậy cấu hình này cũng thích hợp với mô hình tập trung.

2.3.2 Mô hình cơ sở dữ liệu theo kiểu file - server (File - server database model)

Trong mô hình cơ sở dữ liệu theo kiểu file - server các thành phần ứng dụng và phần mềm cơ sở dữ liệu ở trên một hệ thống máy tính và các file vật lý tạo nên cơ sở dữ liệu nằm trên hệ thống máy tính khác. Một cấu hình như vậy thường được dùng trong môi trường cục bộ, trong đó một hoặc nhiều hệ thống máy tính đóng vai trò của server, lưu trữ các file dữ liệu cho hệ thống máy tính khác thâm nhập tới. Trong môi trường file - server, phần mềm mạng được thi hành và làm cho các phần mềm ứng dụng cũng như phần mềm cơ sở dữ liệu chạy trên hệ thống của người dùng cuối coi các file hoặc cơ sở dữ liệu trên file server thực sự như là trên máy tính của người chính họ.

Mô hình file server rất giống với mô hình tập trung. Các file cơ sở dữ liệu nằm trên máy khác với các thành phần ứng dụng và phần mềm cơ sở dữ liệu; tuy nhiên các thành phần ứng dụng và phần mềm cơ sở dữ liệu có thể có cùng thiết kế để vận hành một môi trường tập trung. Thực chất phần mềm mạng đã làm cho phần mềm ứng dụng và phần mềm cơ sở dữ liệu tưởng rằng chúng đang truy nhập cơ sở dữ liệu trong môi trường cục bộ. Một môi trường như vậy có thể phức tạp hơn mô hình tập trung bởi vì phần mềm mạng có thể phải thực hiện cơ chế đồng thời cho phép nhiều người dùng cuối có thể truy nhập vào cùng cơ sở dữ liệu.

2.3.3 Mô hình xử lý từng phần cơ sở dữ liệu (Database extract processing model)

Một mô hình khác trong đó một cơ sở dữ liệu ở xa có thể được truy nhập bởi phần mềm cơ sở dữ liệu, được gọi là xử lý dữ liệu từng phần .

Với mô hình này, người sử dụng có thể tại một máy tính cá nhân kết nối với hệ thống máy tính ở xa nơi có dữ liệu mong muốn. Người sử dụng sau đó có thể tác động trực tiếp đến phần mềm chạy trên máy ở xa và tạo yêu cầu để lấy dữ liệu từ cơ sở dữ liệu đó.

Người sử dụng cũng có thể chuyển dữ liệu từ máy tính ở xa về chính máy tính của mình và vào đĩa cứng và có thể thực hiện việc sao chép bằng phần mềm cơ sở dữ liệu trên máy cá nhân.

Với cách tiếp cận này, người sử dụng phải biết chắc chắn là dữ liệu nằm ở đâu và làm như thế nào để truy nhập và lấy dữ liệu từ một máy tính ở xa. Phần mềm ứng dụng đi kèm cần phải có trên cả hai hệ thống máy tính để kiểm soát sự truy nhập dữ liệu và chuyển dữ liệu giữa hai hệ thống. Tuy nhiên, phần mềm cơ sở dữ liệu chạy trên hai máy không cần biết rằng việc xử lý cơ sở dữ liệu từ xa đang diễn ra vì người sử dụng tác động tới chúng một cách độc lập.

2.3.4 Mô hình cơ sở dữ liệu Client/Server (Client/Server database model)

Mới nhìn, mô hình cơ sở dữ liệu Client/Server có vẻ giống như mô hình file - server, tuy nhiên mô hình Client/Server có rất nhiều thuận lợi hơn mô hình file - server. Với mô hình file - server, thông tin gắn với sự truy nhập cơ sở dữ liệu vật lý phải chạy trên toàn mạng. Một giao tác yêu cầu nhiều sự truy nhập dữ liệu có thể gây ra tắc nghẽn lưu lượng truyền trên mạng.

Giả sử một người dùng cuối tạo ra một vấn tin để lấy dữ liệu tổng số, yêu cầu đòi hỏi lấy dữ liệu từ 1000 bản ghi, với cách tiếp cận file - server nội dung của tất cả 1000 bản ghi phải đưa lên mạng, vì phần mềm cơ sở dữ liệu chạy trên máy của người sử dụng phải truy nhập từng bản ghi để thoả mãn yêu cầu của người sử dụng. Với cách tiếp cận cơ sở dữ liệu Client/Server, chỉ có lời vấn tin khởi động ban đầu và kết quả cuối cùng cần đưa lên mạng, phần mềm cơ sở dữ liệu chạy trên máy lưu giữ cơ sở dữ liệu sẽ truy nhập các bản ghi cần thiết, xử lý chúng và gọi các thủ tục cần thiết để đưa ra kết quả cuối cùng.

Front-end software

Trong mô hình cơ sở dữ liệu Client/Server, thường nói đến các phần mềm front-end software và back-end software. Front-end software được chạy trên một máy tính cá nhân hoặc một workstation và đáp ứng các yêu cầu đơn lẻ riêng biệt, phần mềm này đóng vai trò của Client trong ứng dụng cơ sở dữ liệu Client/Server và thực hiện các chức năng hướng tới nhu cầu của người dùng cuối cùng, phần mềm Front-end software thường được chia thành các loại sau:

- End user database software: Phần mềm cơ sở dữ liệu này có thể được thực hiện bởi người sử dụng cuối trên chính hệ thống của họ để truy nhập các cơ sở dữ liệu cục bộ nhỏ cũng như kết nối với các cơ sở dữ liệu lớn hơn trên cơ sở dữ liệu Server.

- Simple query and reporting software: Phần mềm này được thiết kế để cung cấp các công cụ dễ dùng hơn trong việc lấy dữ liệu từ cơ sở dữ liệu và tạo các báo cáo đơn giản từ dữ liệu đã có.

- Data analysis software: Phần mềm này cung cấp các hàm về tìm kiếm, khôi phục, chúng có thể cung cấp các phân tích phức tạp cho người dùng.

- Application development tools: Các công cụ này cung cấp các khả năng về ngôn ngữ mà các nhân viên hệ thống thông tin chuyên nghiệp sử dụng để xây dựng các ứng dụng cơ sở dữ liệu của họ. Các công cụ ở đây bao gồm các công cụ về thông dịch, biên dịch đơn đến các công cụ CASE (Computer Aided Software Engineering), chúng tự động tất cả các bước trong quá trình phát triển ứng dụng và sinh ra chương trình cho các ứng dụng cơ sở dữ liệu.

- Database administration Tools: Các công cụ này cho phép người quản trị cơ sở dữ liệu sử dụng máy tính cá nhân hoặc trạm làm việc để thực hiện việc quản trị cơ sở dữ liệu như định nghĩa các cơ sở dữ liệu, thực hiện lưu trữ hay phục hồi.

Back-end software

Phần mềm này bao gồm phần mềm cơ sở dữ liệu Client/Server và phần mềm mạng chạy trên máy đóng vai trò là Server cơ sở dữ liệu.

2.3.5 Mô hình cơ sở dữ liệu phân tán (Distributed database model)

Cả hai mô hình File - Server và Client/Server đều giả định là dữ liệu nằm trên một bộ xử lý và chương trình ứng dụng truy nhập dữ liệu nằm trên một bộ xử lý khác, còn mô hình cơ sở dữ liệu phân tán lại giả định bản thân cơ sở dữ liệu có ở trên nhiều máy khác nhau.

Bài tập:

Câu 1: Hãy so sánh mô hình dữ liệu tập trung và mô hình dữ liệu phân tán ?

Câu 2: Cho ví dụ về mô hình dữ liệu tập trung và mô hình dữ liệu phân tán hiện nay mà bạn biết?

BÀI 3
HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER

Mã bài : ITPRG3-17.3

Giới thiệu :

SQL Server là một hệ thống quản lý cơ sở dữ liệu (Relational Database Management System (RDBMS)) sử dụng **Transact-SQL** để trao đổi dữ liệu giữa Client computer và SQL Server computer. Một RDBMS bao gồm databases, database engine và các ứng dụng dùng để quản lý dữ liệu và các bộ phận khác nhau trong RDBMS.

SQL Server được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến Tera-Byte và có thể phục vụ cùng lúc cho hàng ngàn user. SQL Server 2000 có thể kết hợp "ăn ý" với các server khác như Microsoft Internet Information Server (IIS), E-Commerce Server, Proxy Server....

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

- Trình bày được đặc điểm của hệ quản trị cơ sở dữ liệu SQL SERVER, lịch sử phát triển, các thành phần
- Cài đặt hệ quản trị CSDL Client/Server ở máy chủ và máy khách
- Thao tác chính xác các tiến trình càI đặt, gỡ bỏ bộ cài đặt
- Sử dụng thành thạo các điều khiển cơ bản trong cơ sở dữ liệu Client/Server.

Nội dung:

3.1 Giới thiệu hệ quản trị cơ sở dữ liệu SQL Server.

3.2 Cài đặt SQL Server

3.3 Các thành phần cua Sql Server

3.4 Các thao tác cơ bản trên môi trường SQL Server

3.1 Giới thiệu hệ quản trị cơ sở dữ liệu SQL Server.

Ngôn ngữ hỏi có cấu trúc (SQL) và các hệ quản trị cơ sở dữ liệu quan hệ là một trong những nền tảng kỹ thuật quan trọng trong công nghiệp máy tính. Cho đến nay, có thể nói rằng SQL đã được xem là ngôn ngữ chuẩn trong cơ sở dữ liệu. Các hệ quản trị

cơ sở dữ liệu quan hệ thương mại hiện có như Oracle, SQL Server, Informix, DB2,... đều chọn SQL làm ngôn ngữ cho sản phẩm của mình

Vậy thực sự SQL là gì? Tại sao nó lại quan trọng trong các hệ quản trị cơ sở dữ liệu? SQL có thể làm được những gì và như thế nào? Nó được sử dụng ra sao trong các hệ quản trị cơ sở dữ liệu quan hệ? Nội dung của chương này sẽ cung cấp cho chúng ta cái nhìn tổng quan về SQL và một số vấn đề liên quan.

SQL, viết tắt của Structured Query Language (ngôn ngữ hỏi có cấu trúc), là công cụ sử dụng để tổ chức, quản lý và truy xuất dữ liệu được lưu trữ trong các cơ sở dữ liệu. SQL là một hệ thống ngôn ngữ bao gồm tập các câu lệnh sử dụng để tương tác với cơ sở dữ liệu quan hệ.

Tên gọi ngôn ngữ hỏi có cấu trúc phần nào làm chúng ta liên tưởng đến một công cụ (ngôn ngữ) dùng để truy xuất dữ liệu trong các cơ sở dữ liệu. Thực sự mà nói, khả năng của SQL vượt xa so với một công cụ truy xuất dữ liệu, mặc dù đây là mục đích ban đầu khi SQL được xây dựng nên và truy xuất dữ liệu vẫn còn là một trong những chức năng quan trọng của nó. SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

1 • **Định nghĩa dữ liệu:** SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu cũng như mối quan hệ giữa các thành phần dữ liệu.

2 • **Truy xuất và thao tác dữ liệu:** Với SQL, người dùng có thể dễ dàng thực hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.

3 • **Điều khiển truy cập:** SQL có thể được sử dụng để cấp phát và kiểm soát các thao tác của người sử dụng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu

1 • **Đảm bảo toàn vẹn dữ liệu:** SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập nhật cũng như các lỗi của hệ thống.

Như vậy, có thể nói rằng SQL là một ngôn ngữ hoàn thiện được sử dụng trong các hệ thống cơ sở dữ liệu và là một thành phần không thể thiếu trong các hệ quản trị cơ sở dữ liệu. Mặc dù SQL không phải là một ngôn ngữ lập trình như C, C++, Java,... song các câu lệnh mà SQL cung cấp có thể được nhúng vào trong các ngôn ngữ lập trình nhằm xây dựng các ứng dụng tương tác với cơ sở dữ liệu.

Khác với các ngôn ngữ lập trình quen thuộc như C, C++, Java,... SQL là ngôn ngữ có tính khai báo. Với SQL, người dùng chỉ cần mô tả các yêu cầu cần phải thực hiện trên cơ sở dữ liệu mà không cần phải chỉ ra cách thức thực hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ dễ tiếp cận và dễ sử dụng.

3.2 Cài đặt SQL Server

Chúng ta cần có **Standard Edition** để có thể cài đặt SQL Server. Bạn có thể cài đặt **SQL Server 7.0** trên mọi hệ điều hành Win9x/Win2K và có thể cài đặt **SQL Server 2000** trên Windows Server hay Windows XP Professional, Windows 2000 Professional hay NT Workstation nhưng không thể cài đặt trên Win 98 family.

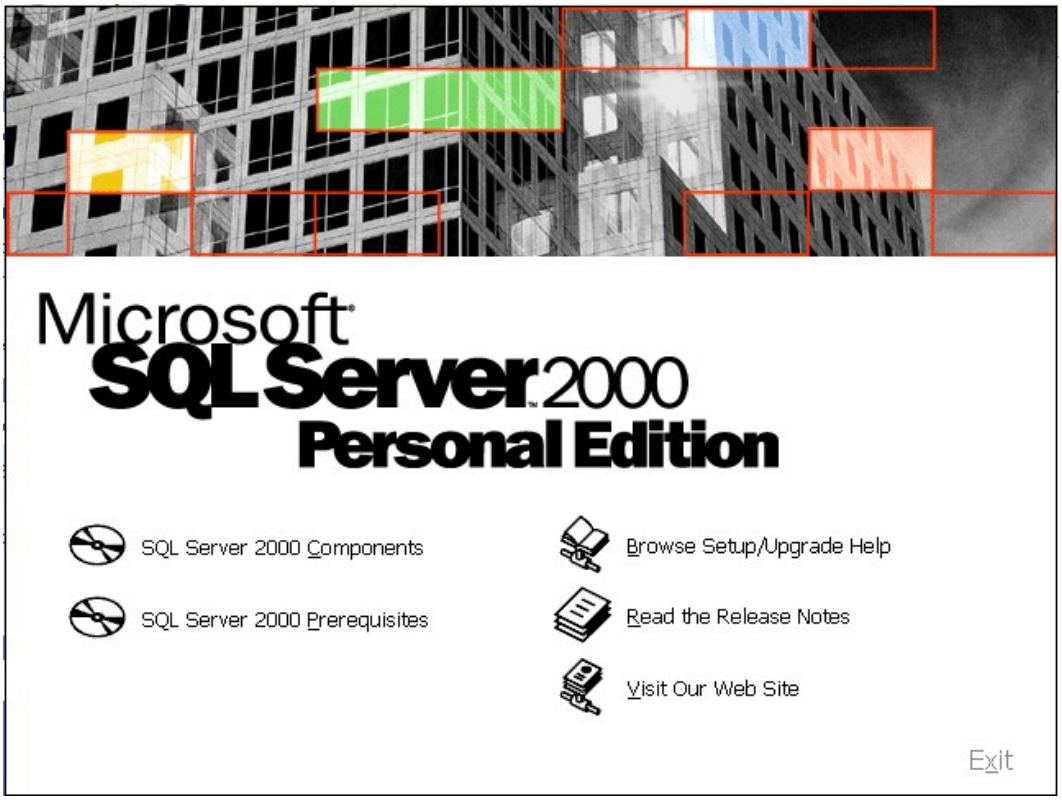
Sau khi cài đặt bạn sẽ thấy một biểu tượng nằm ở góc phải bên dưới màn hình, đây chính là Service Manager. Bạn có thể Start, Stop các SQL Server services dễ dàng bằng cách double-click vào biểu tượng này.

*** Kiến thức về các Version của SQL Server**

SQL Server của Microsoft được thị trường chấp nhận rộng rãi kể từ version 6.5. Sau đó Microsoft đã cải tiến và hầu như viết lại một engine mới cho SQL Server 7.0. Cho nên có thể nói từ version 6.5 lên version 7.0 là một bước nhảy vọt. Có một số đặc tính của SQL Server 7.0 không tương thích với version 6.5. Trong khi đó từ Version 7.0 lên version 8.0 (SQL Server 2000) thì những cải tiến chủ yếu là mở rộng các tính năng về web và làm cho SQL Server 2000 đáng tin cậy hơn.

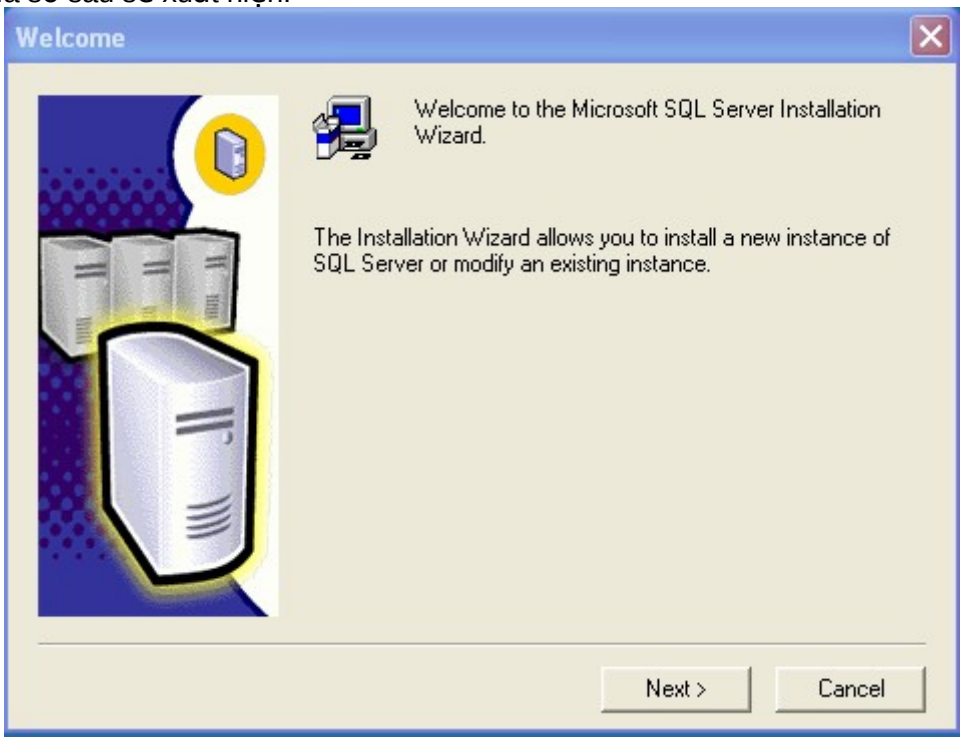
*** Các bước cài đặt SQL Server 2000**

- Đặt đĩa CD vào sẽ tự động xuất hện cửa sổ sau (nếu không thấy xuất hiện hãy chạy tập tin autorun.exe trong thư mục gốc của đĩa CD) như hình 1:



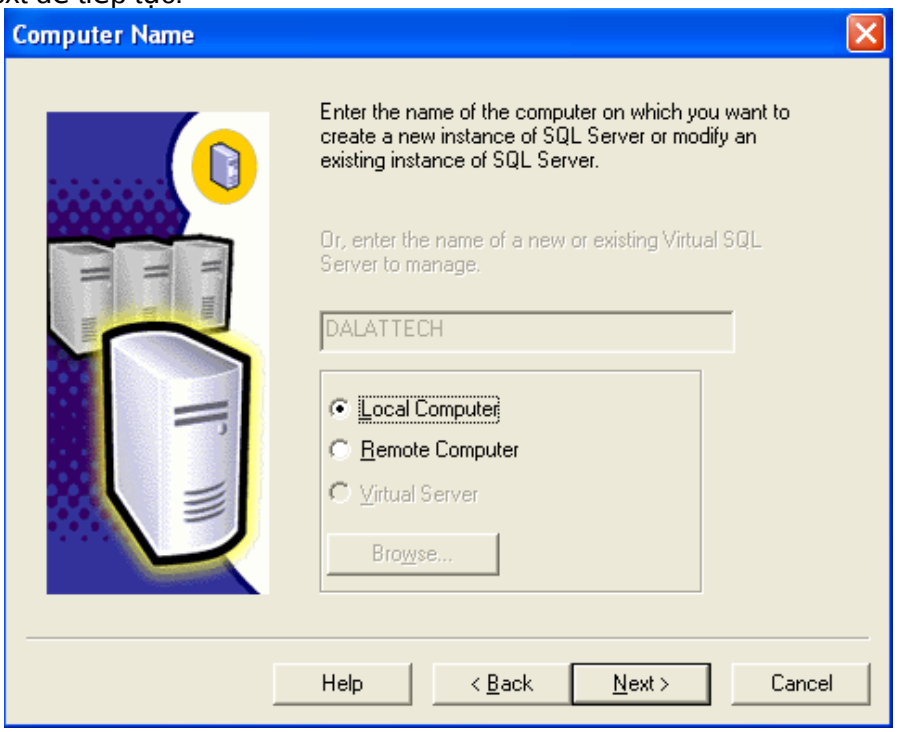
Hình 3.1 Sql server 2000

- Chọn SQL Server 2000 Components, và chọn Install Database Server ở bướckế tiếp, cửa sổ sau sẽ xuất hiện:



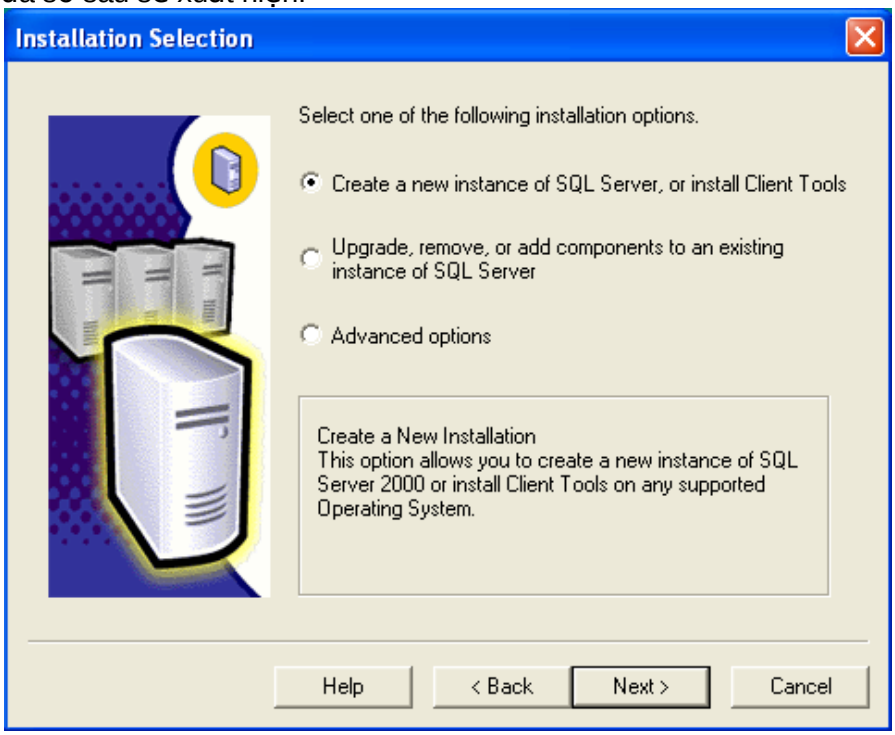
*Hình 3.2:*Install Database Server

- Nhấn Next để tiếp tục:



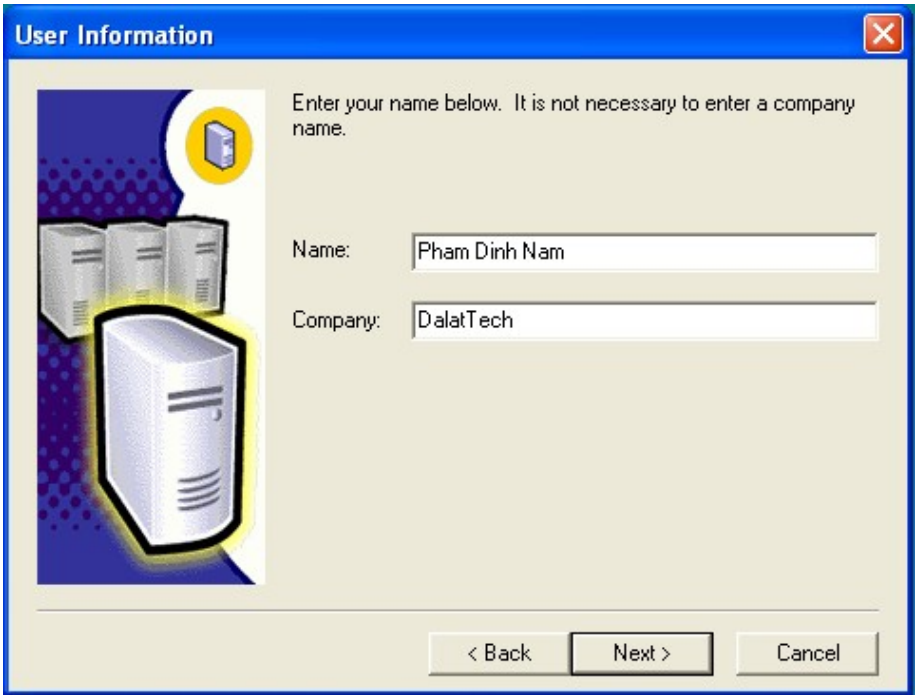
Hình 3.3: chọn Local Computer *nhấn Next để tiếp tục*

- Nếu cài trên máy cá nhân hãy để nguyên tùy chọn Local Computer và nhấn Next để tiếp tục, cửa sổ sau sẽ xuất hiện:



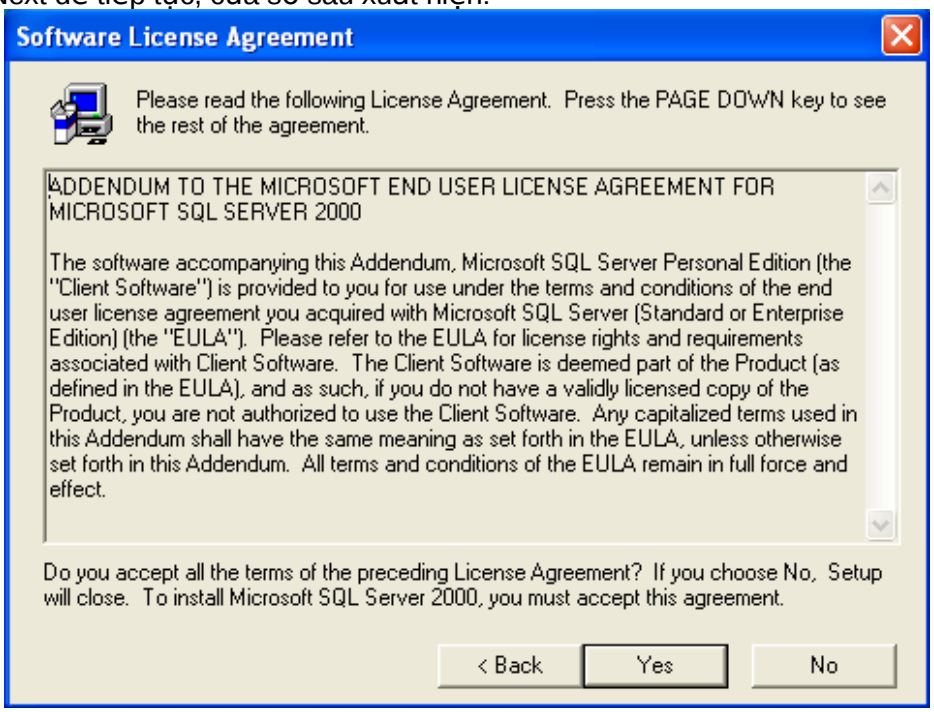
*Hình 3.4:*nhấn Next để tiếp tục

- Để nguyên tùy chọn Create a new instance of SQL Server,or install Client Tools và nhấn Next để tiếp tục rồi điền thông tin ở cửa sổ kế tiếp:



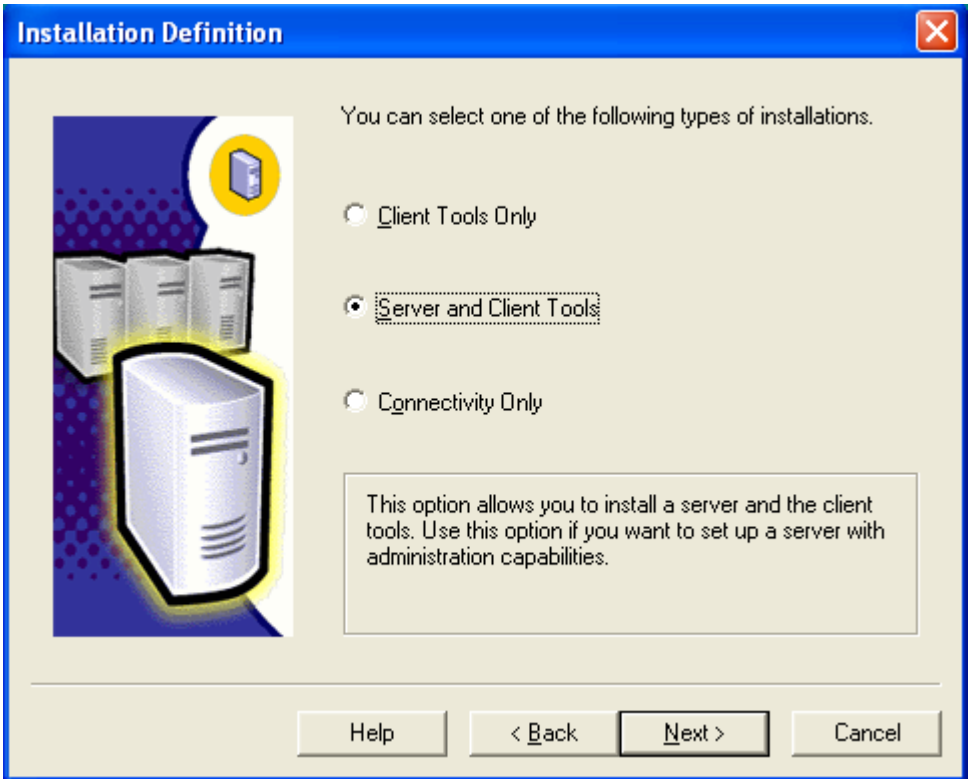
Hình 3.5 Nhấn Next để tiếp tục

- Nhấn Next để tiếp tục, cửa sổ sau xuất hiện:



*Hình 3.6:*Chọn Yes nếu đồng ý cài đặt

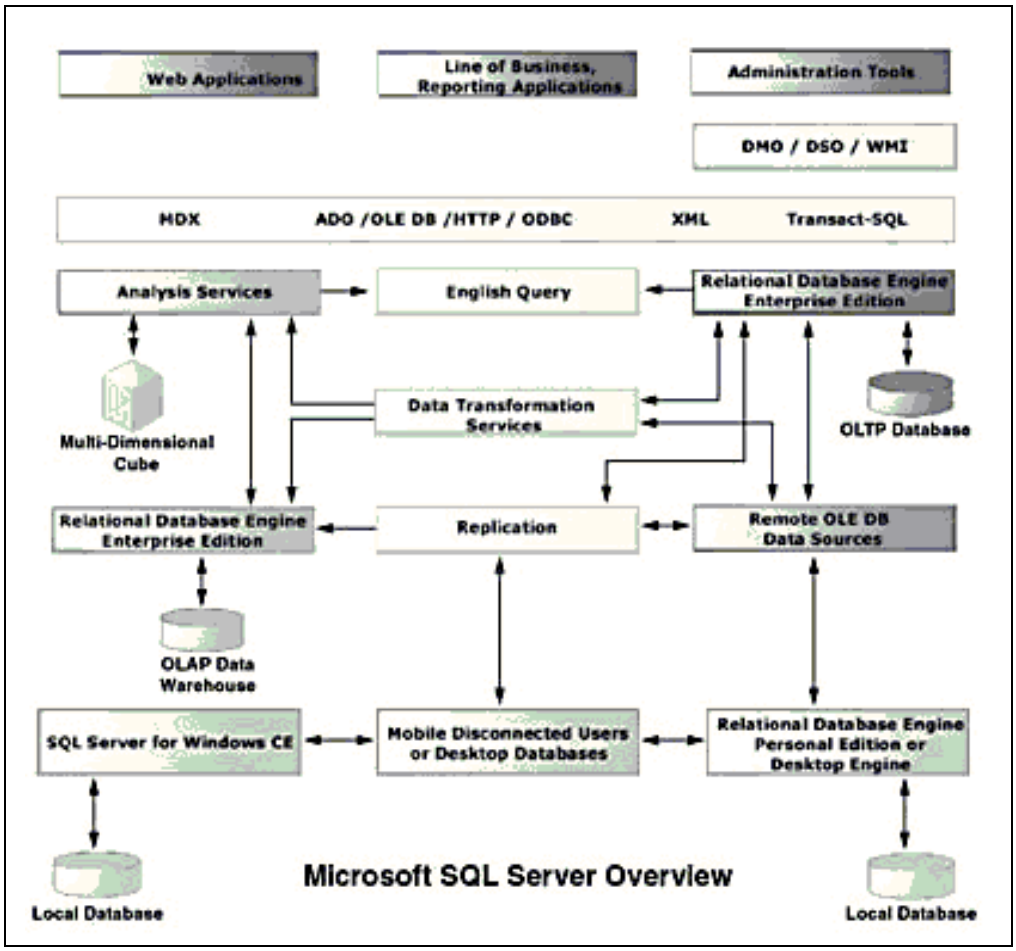
- Chọn Yes nếu đồng ý cài đặt hệ quản trị cơ sở dữ liệu SQL Server:



*Hình 3.7:*chọn Server and Client Tools để cài đặt

- Trên máy đơn, hệ thống vừa đóng vai trò là Server vừa là Client nên chúng ta nên để nguyên tùy chọn Server and Client Tools để cài đặt hết các công cụ cần thiết để quản trị. Ở các bước tiếp theo chúng ta nên chọn chế độ Local và sử dụng tài khoản đăng nhập là sa (password rỗng) để thuận tiện cho việc thực hành về sau.

3.3 Các thành phần cua Sql Server



3.4 *Hình 3.8* Các thành phần cua Sql Server

English Query - Một dịch vụ mà người Việt Nam chắc là ít muốn dùng.

Đây là một dịch vụ giúp cho việc query data bằng tiếng Anh "trơn" (plain English).

SQL Server Books Online - Cho dù bạn có đọc các sách khác nhau dạy về SQL server thì bạn cũng sẽ thấy books online này rất hữu dụng và không thể thiếu được.

SQL Server Tools - Đây là một bộ đồ nghề của người quản trị cơ sở dữ liệu (DBA)

Ở đây người viết chỉ kể ra một vài công cụ thông dụng mà thôi.

- Đầu tiên phải kể đến **Enterprise Manager**. Đây là một công cụ cho ta thấy toàn cảnh hệ thống cơ sở dữ liệu một cách rất trực quan. Nó rất hữu ích đặc biệt cho người mới học và không thông thạo lắm về SQL.
- Kế đến là **Query Analyzer**. Đối với một DBA giỏi thì hầu như chỉ cần công cụ này là có thể quản lý cả một hệ thống database mà không cần đến những thứ khác. Đây là một môi trường làm việc khá tốt vì ta có thể đánh bất kỳ câu lệnh SQL nào và chạy ngay lập tức đặc biệt là nó giúp cho ta debug mấy cái stored procedure dễ dàng.

3.5 Các thao tác cơ bản trên môi trường SQL Server

SQL chuẩn bao gồm khoảng 40 câu lệnh. Bảng sau đây liệt kê các lệnh SQL thường được sử dụng nhất trong số các câu lệnh của SQL Server:

Thao tác dữ liệu:

Câu lệnh	**Chức năng**
SELECT	Truy xuất dữ liệu
INSERT	Bổ sung dữ liệu
UPDATE	Cập nhật dữ liệu
DELETE	Xóa dữ liệu
TRUNCATE	Xóa toàn bộ dữ liệu trong bảng

Định nghĩa dữ liệu:

Câu lệnh	**Chức năng**
CREATE TABLE	Tạo bảng
DROP TABLE	Xóa bảng
ALTER TABLE	Sửa đổi bảng
CREATE VIEW	Tạo khung nhìn
DROP VIEW	Xóa khung nhìn
ALTER VIEW	Sửa đổi khung nhìn
CREATE INDEX	Tạo chỉ mục
DROP INDEX	Xóa chỉ mục
CREATE SCHEMA	Tạo lược đồ cơ sở dữ liệu
DROP SCHEMA	Xóa lược đồ cơ sở dữ liệu
CREATE PROCEDURE	Tạo thủ tục lưu trữ
DROP PROCEDURE	Xóa thủ tục lưu trữ
ALTER PROCEDURE	Sửa thủ tục lưu trữ
CREATE FUNCTION	Tạo hàm (do người sử dụng định nghĩa)
DROP FUNCTION	Xóa hàm
ALTER FUNCTION	Sửa đổi hàm
CREATE TRIGGER	Tạo Trigger
DROP TRIGGER	Xóa Trigger
ALTER TRIGGER	Sửa đổi Trigger

Điều khiển truy nhập:

Câu lệnh	**Chức năng**
GRANT	Cấp phát quyền cho người sử dụng
REVOKE	Thu hồi quyền đối với người sử dụng

Quản lý giao tác:

Câu lệnh	**Chức năng**
COMMIT	Uỷ thác (kết thúc thành công) giao tác
ROLLBACK	Quay lui giao tác
SAVE TRANSACTION	Đánh dấu một điểm trong giao tác

Lập trình:

Câu lệnh	**Chức năng**
DECLARE	Khai báo biến hoặc định nghĩa con trỏ
OPEN	Mở một con trỏ để truy xuất kết quả trừ
FETCH	Đọc một dòng trong kết quả của câu truy vấn (dùng con trỏ)
CLOSE	Đóng một con trỏ
EXECUTE	Thực thi một cấu lệnh SQL

BÀI TẬP THỰC HÀNH

1. Cài đăt SQL Server 2000 với công cụ quản trị cơ sở dữ liệu.

2. Sử dụng Books Online để tra cứu thông tin cú pháp của câu lệnh Create Table, Select.

BÀI 4
CÁC THAO TÁC TRÊN SQL SERVER

MÃ BÀI ITPRG3_17.4

Giới thiệu:

Trong bài này sẽ đề cập đến các thao tác trên SQL chủ yếu thông qua công cụ Enterprise Manager, không đề cập nhiều đến các lệnh của SQL Server. Trong bài sau sẽ tìm hiểu cụ thể các câu lệnh này.

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

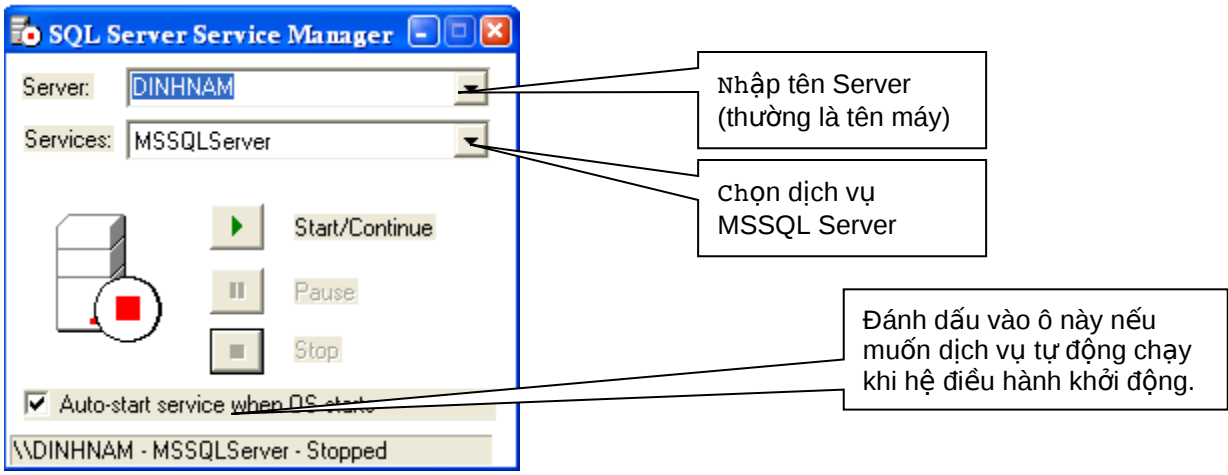
- Đăng nhập được vào SQL Server
- Phân biệt được các thành phần của SQL Server
- Khai báo đúng các kiểu dữ liệu và phạm vi ứng dụng
- Tạo được cơ sở dữ liệu trong SQL Server
- Tạo được bảng trong SQL Server
- Tạo được quan hệ trong SQL Server
- Nhập và trang trí được dữ liệu trong SQL Server

Nội dung:

4.1 Đăng nhập vào SQL Server

- Sau khi cài đặt thành công, việc đăng nhập sẽ được thực hiện qua các bước như sau:

1) Vào Start → programs → Microsoft SQL Server → Service Manager, cửa sổ sau sẽ xuất hiện:

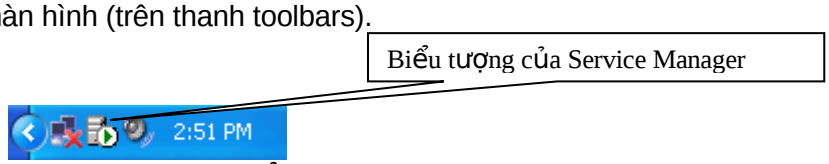


Hình 4.1 Cài đặt SQL Server

- Nhấn vào nút Start/Continue để bắt đầu chạy dịch vụ của SQL Server. Nếu thành công, cửa sổ sau sẽ xuất hiện:

Hình 4.2 :chạy dịch vụ của SQL Server

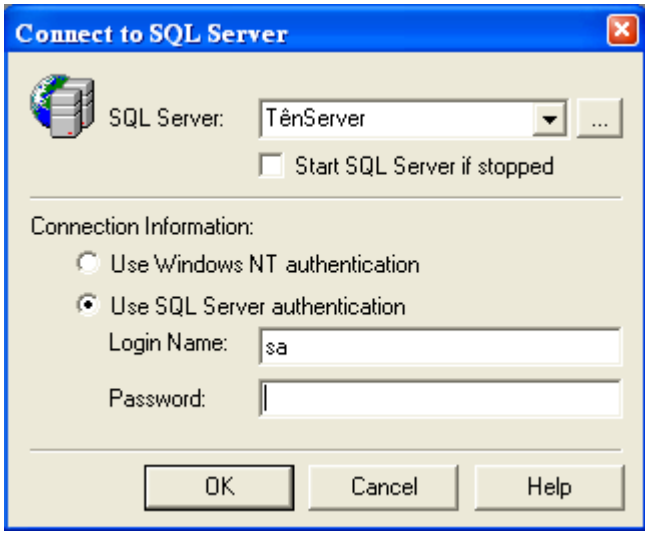
- Sau đó chúng ta có thể đóng cửa sổ này lại, một biểu tượng sẽ xuất hiện ở góc phải dưới của màn hình (trên thanh toolbars).



Hình 4.3 : biểu tượng trên thanh toolbars

2) Để đăng nhập vào hệ thống cơ sở dữ liệu của SQL Server, chúng ta thực hiện quy trình sau:

- Vào Start → Programs → Microsoft SQL Server → Query Analyzer, cửa sổ sau sẽ xuất hiện:



Hình 4.4:Query Analyzer

+ Nhập tên server vào hộp SQL Server, thường tên Server là tên máy chủ cài đặt SQL hoặc nếu muốn đăng nhập vào chính máy của người sử dụng thì có thể gõ vào ***localhost***.

+ Nhập tên đăng nhập vào hộp Login Name.

+ Nhập mật khẩu vào hộp Password.

→ Nhấn OK để truy nhập vào hệ thống cơ sở dữ liệu của SQL Server.

4.2 Các thành phần của SQL Server

SQL Server có nhiều dịch vụ chạy trên Windows NT/Windows 2000/XP/2003 hay các chương trình nền trên Windows 98. Những dịch vụ này là MSSQLServer, Search Service, MSDTC và SQLServerAgent. Ngoài ra OLAP là một dịch vụ khá hoàn hảo về kho dữ liệu.

Dịch vụ MSSQL Server là trình quản lý cơ sở dữ liệu cốt lõi và là dịch vụ duy nhất phải chạy. Bộ phối hợp giao tác phân phối (MSDTC) tương tác với các SQL Server khác và với MS Transaction Server (MTS) để phối hợp các giao tác với những trình ứng dụng chạy trên nhiều máy tính hay công nghệ. Dịch vụ tìm kiếm (Search service) điều khiển việc tìm kiếm text, cho phép bạn tìm kiếm qua dữ liệu text trong cơ sở dữ liệu của bạn; dịch vụ này không được cài đặt theo mặc định, do đó nó có thể không hiện diện trên server của bạn.

Dịch vụ SQLServerAgent là một bộ lập thời biểu tác vụ nhằm xử lý tất cả các tác vụ theo định kỳ, chẳng hạn như bảo trì và sao chép.

Những tiến trình này hoạt động cùng với nhau để quản lý cho phép bạn truy cập các cơ sở dữ liệu. một SQL Server có thể điều khiển nhiều cơ sở dữ liệu trên cùng một Server. Mỗi SQL Server có tối thiểu các cơ sở dữ liệu như model, msdb, master và tempdb. Chúng ta sẽ thường tìm thấy hai cơ sở dữ liệu mẫu là Northwind và pubs cũng như các cơ sỡ dữ liệu của công việc riêng của chúng ta.

4.3 Các kiểu dữ liệu trong SQL Server

Tên kiểu	**Mô tả**
CHAR (n)	Kiểu chuỗi với độ dài cố định
NCHAR (n)	Kiếu chuỗi với độ dài cố định hỗ trợ UNICODE
VARCHAR (n)	Kiểu chuỗi với độ dài chính xác
NVARCHAR (n)	Kiểu chuỗi với độ dài chính xác hỗ trợ UNICODE
INTEGER	Số nguyên có giá trị từ -2^{31} đến $2^{31} - 1$
INT	Như kiểu Integer
TINYTINT	Số nguyên có giá trị từ 0 đến 255.
SMALLINT	Số nguyên có giá trị từ -2^{15} đến $2^{15} – 1$
BIGINT	Số nguyên có giá trị từ -2^{63} đến $2^{63}-1$
NUMERIC (p,s)	Kiểu số với độ chính xác cố định.
DECIMAL (p,s)	Tương tự kiểu Numeric
FLOAT	Số thực có giá trị từ -1.79E+308 đến 1.79E+308
REAL	Số thực có giá trị từ -3.40E + 38 đến 3.40E + 38
MONEY	Kiểu tiền tệ
BIT	Kiểu bit (có giá trị 0 hoặc 1)
DATETIME	Kiểu ngày giờ (chính xác đến phần trăm của giây)
SMALLDATETIME	Kiểu ngày giờ (chính xác đến phút)
TIMESTAMP	
BINARY	Dữ liệu nhị phân với độ dài cố định (tối đa 8000 bytes)
VARBINARY	Dữ liệu nhị phân với độ dài chính xác (tối đa 8000 bytes)
IMAGE	Dữ liệu nhị phân với độ dài chính xác (tối đa 2,147,483,647 bytes)

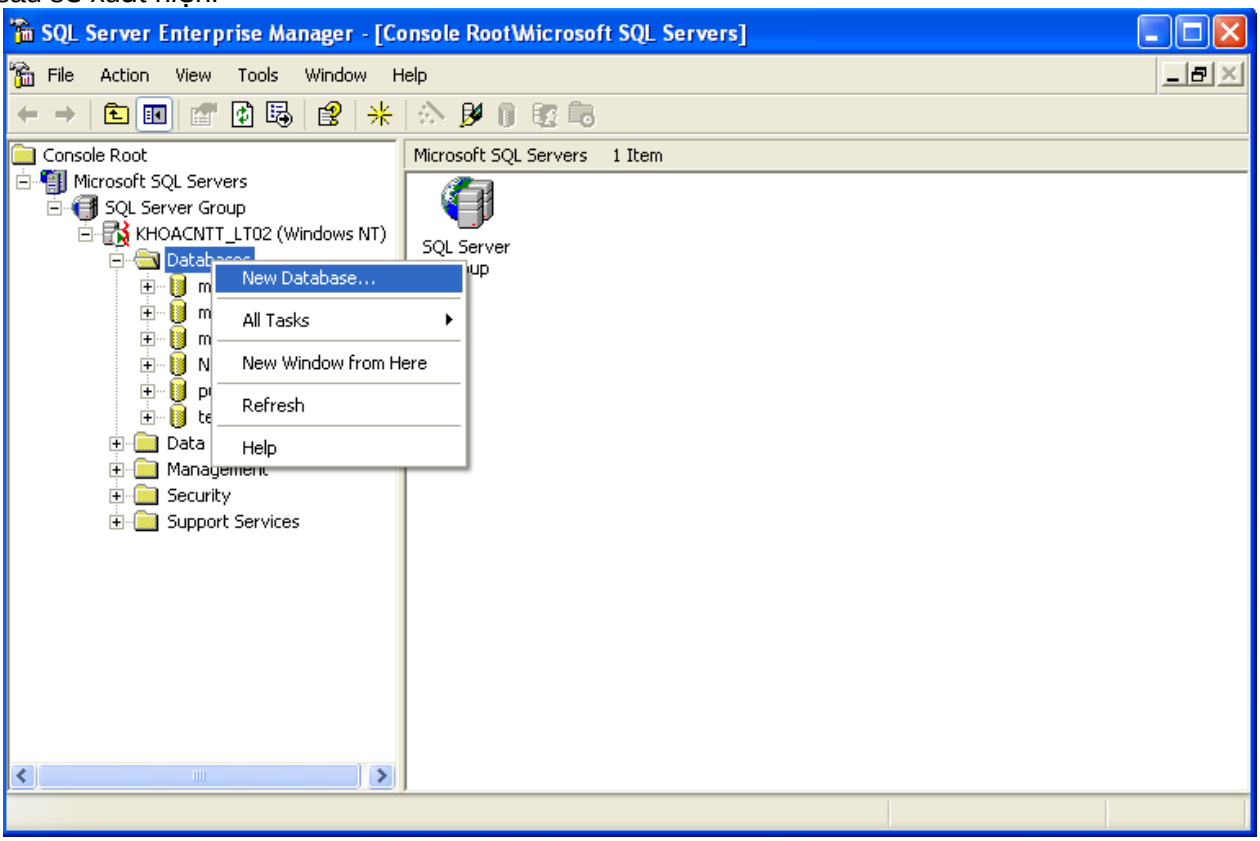
TEXT	Dữ liệu kiểu chuỗi với độ dài lớn (tối đa 2,147,483,647 ký tự)
NTEXT	Dữ liệu kiểu chuỗi với độ dài lớn và hỗ trợ UNICODE (tối đa 1,073,741,823 ký tự)

4.4 Tạo cơ sở dữ liệu trong SQL Server

Chúng ta có hai cách để tạo một cơ sở dữ liệu:

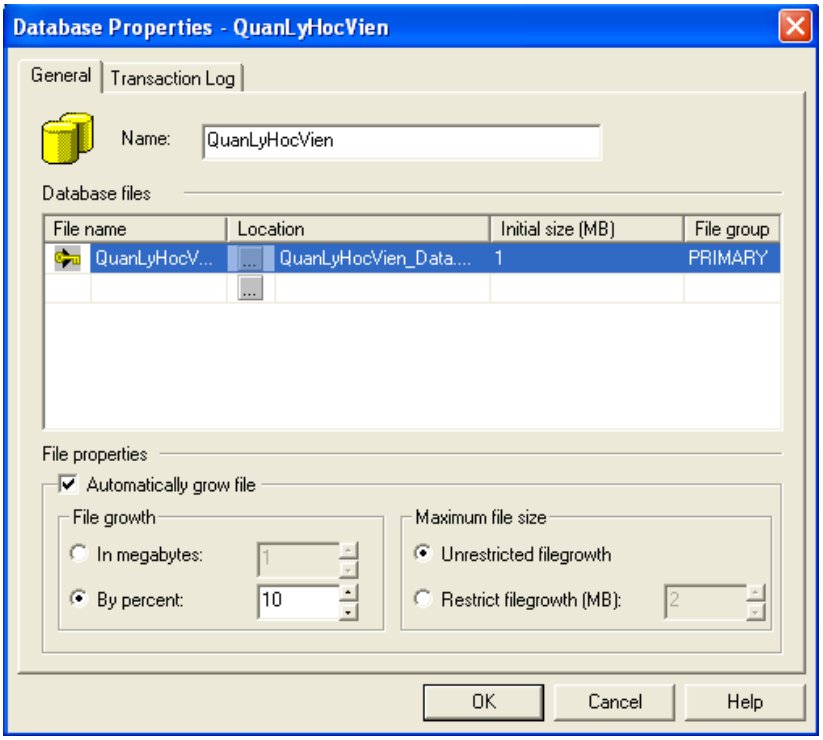
4.4.1 Tạo mới cơ sở dữ liệu từ công cụ Enterprise Manager:

Vào menu Start → Programs → Microsoft SQL Server → Enterprise Manager, cửa số sau sẽ xuất hiện:



Hình 4.5 :Enterprise Manager

Nhấn chuột phải vào mục Databases → New database…, một cửa sổ sẽ hiện ra yêu cầu chúng ta nhập tên cơ sở dữ liệu:



Hình 4.6 :nhập tên cơ sở dữ liệu

Nhập tên cơ sở dữ liệu vào hộp Name (ví dụ trên đã đặt tên cơ sở dữ liệu là QuanLyHocVien), chúng ta có thể thay đổi kích thước lưu trữ giới hạn cho cơ sở dữ liệu bằng cách chọn vào tùy chọn **In megabytes** và nhập dung lượng sẽ lưu trữ cơ sở dữ liệu.

4.4.2 Tạo mới cơ sở dữ liệu bằng lệnh SQL:

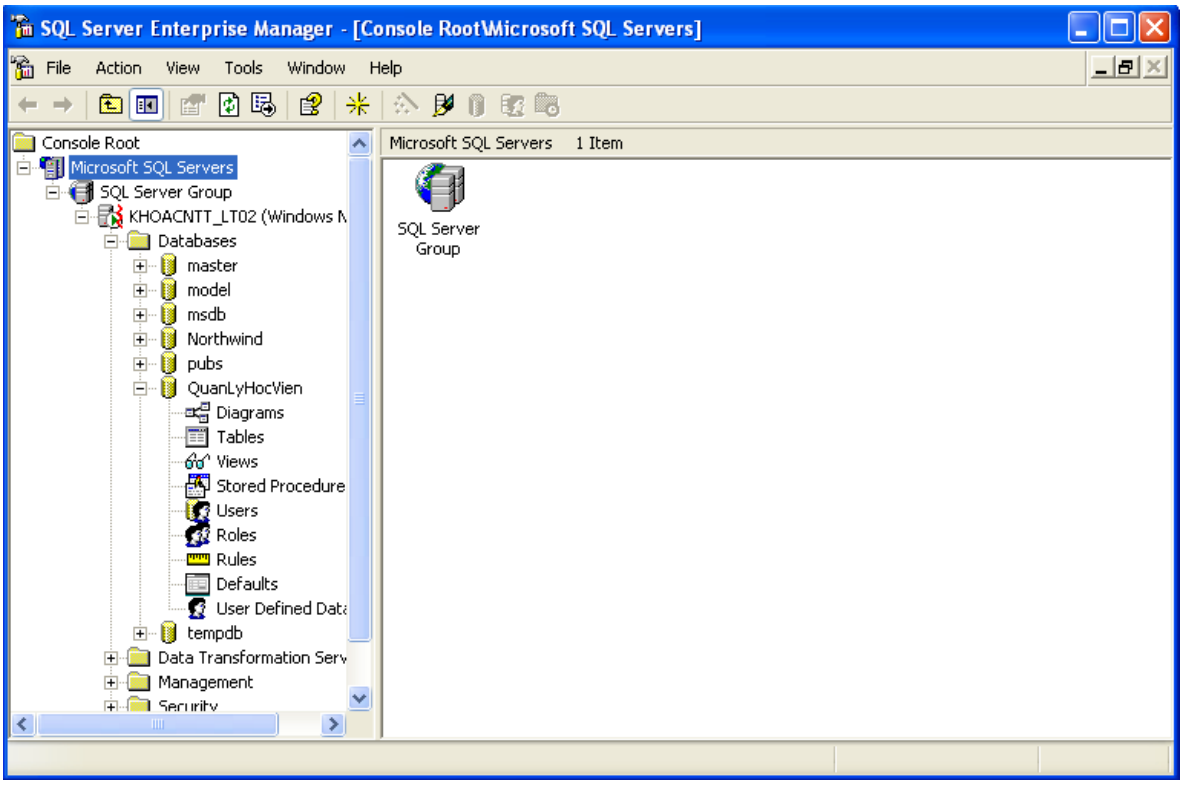
Đăng nhập với tài khoản sa, trong cửa sổ lệnh của công cụ Query Analyzer chúng ta gõ vào dòng lệnh Create Database với cú pháp như sau:

Create Database <tên cơ sở dữ liệu> [Các tham số nếu có]

Để tìm hiểu chi tiết về các tham số, chúng ta có thể tham khảo trong thành phần Books Online.

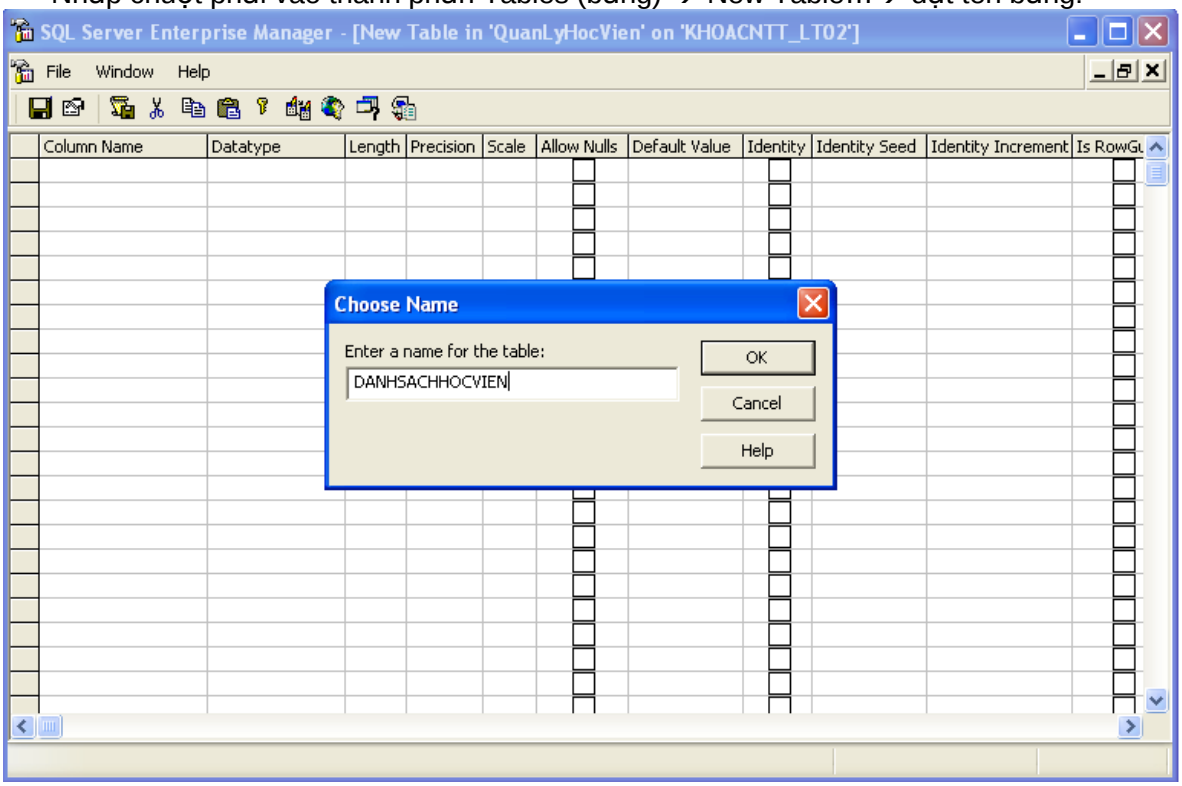
4.5 Tạo bảng trong SQL Server

Để tạo bảng cho môt cơ sở dữ liệu nào đó ta nhắp chuột vào dấu cộng (+) bên trái cơ sỡ dữ liệu tương ứng, chúng ta sẽ thấy một danh sách các thành phần của cơ sở dữ liệu sẽ mở ra:



Hình 4.7:thành phần của cơ sở dữ liệu

Nhấp chuột phải vào thành phần Tables (bảng) → New Table…→ đặt tên bảng:

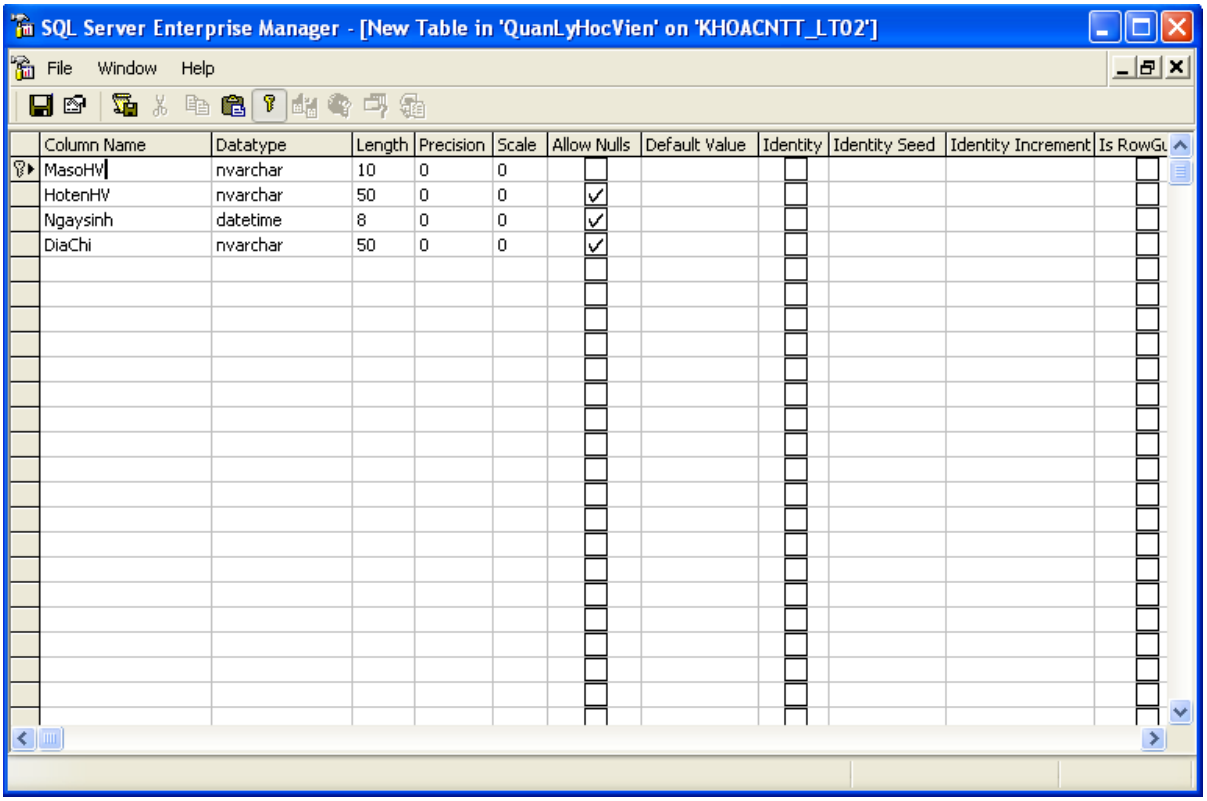


Hình 4.8:đặt tên bảng

Sau khi nhập tên bảng hãy nhấn OK để xác nhận, nếu muốn bỏ qua thao tác tạo bảng hãy nhấn nút Cancel.

Sau khi tạo bảng mới, chúng ta thiết kế bảng bằng cách nhập vào tên trường vào cột Column Name, chọn kiểu dữ liệu trong cột Datatype, đặt kích thước dữ liệu của trường trong cột Length, bỏ chọn Allow Nulls nếu muốn trường tương ứng không được để trống, chọn Identity nếu muốn tạo chỉ mục.

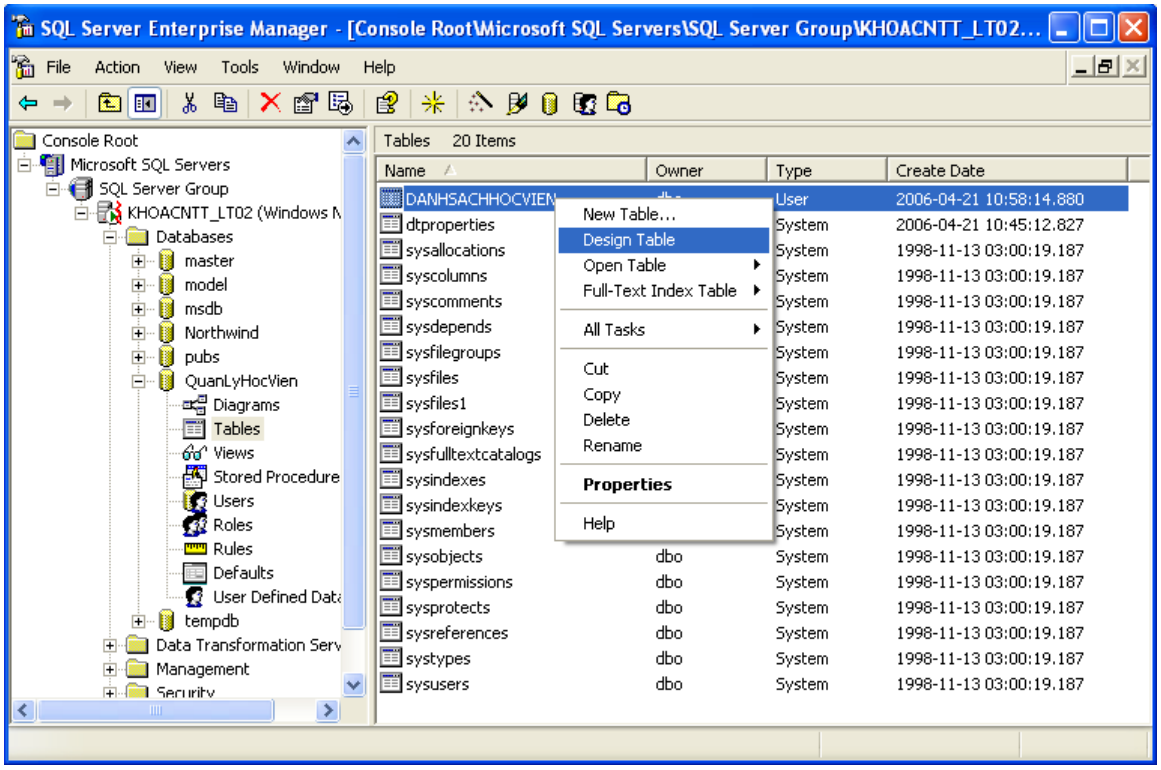
Muốn tạo khóa chính cho trường nào thì chọn dòng tương ứng sau đó nhắp vào biểu tượng chìa khóa trên thanh công cụ. Chúng ta có thể kết hợp với phím Shift và Ctrl nếu muốn chọn nhiều dòng.



Hình 4.9:tạo khoá chính

Sau khi thiết kế bảng xong, chúng ta nhắp chuột vào biểu tượng đĩa mềm trên thanh công cụ để lưu bảng và nhấn vào dấu (X) bên dưới của cửa sổ Enterprise Manager để đóng bảng lại.

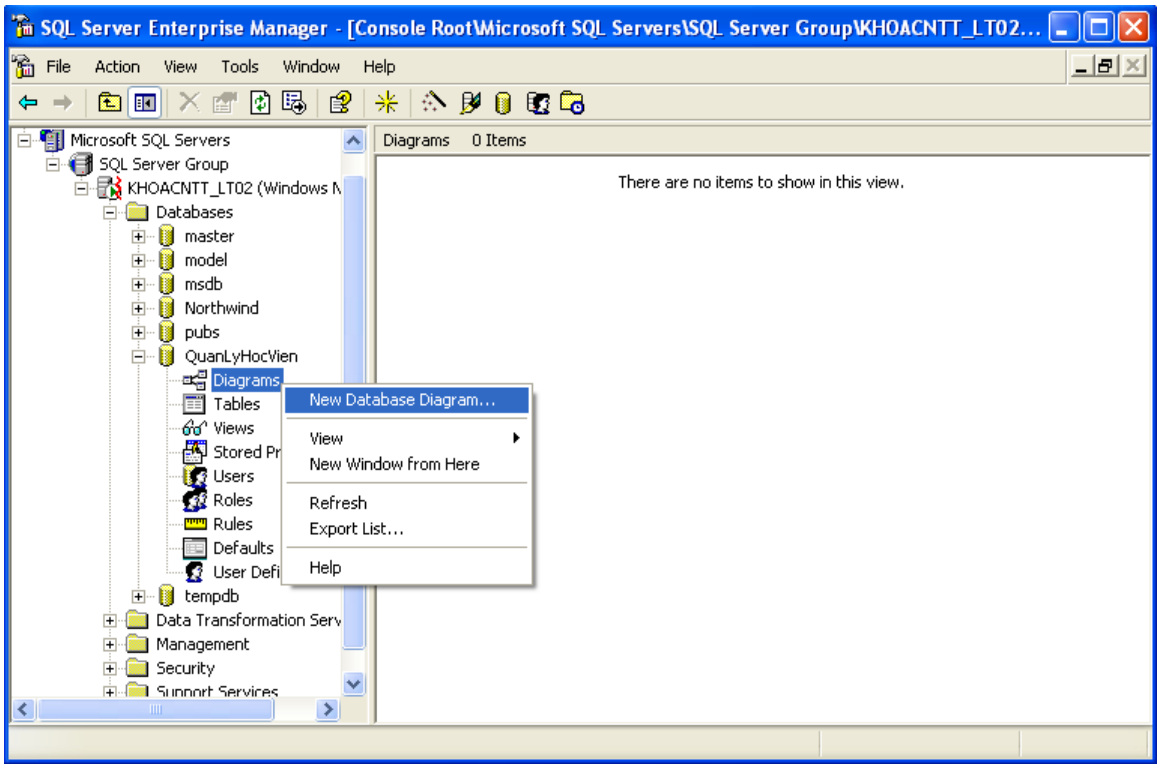
Để sửa đổi cấu trúc bảng đã tạo chúng ta nhắp chuột phải vào bảng tương ứng → chọn Design Table:



Hình 4.10 Design Table

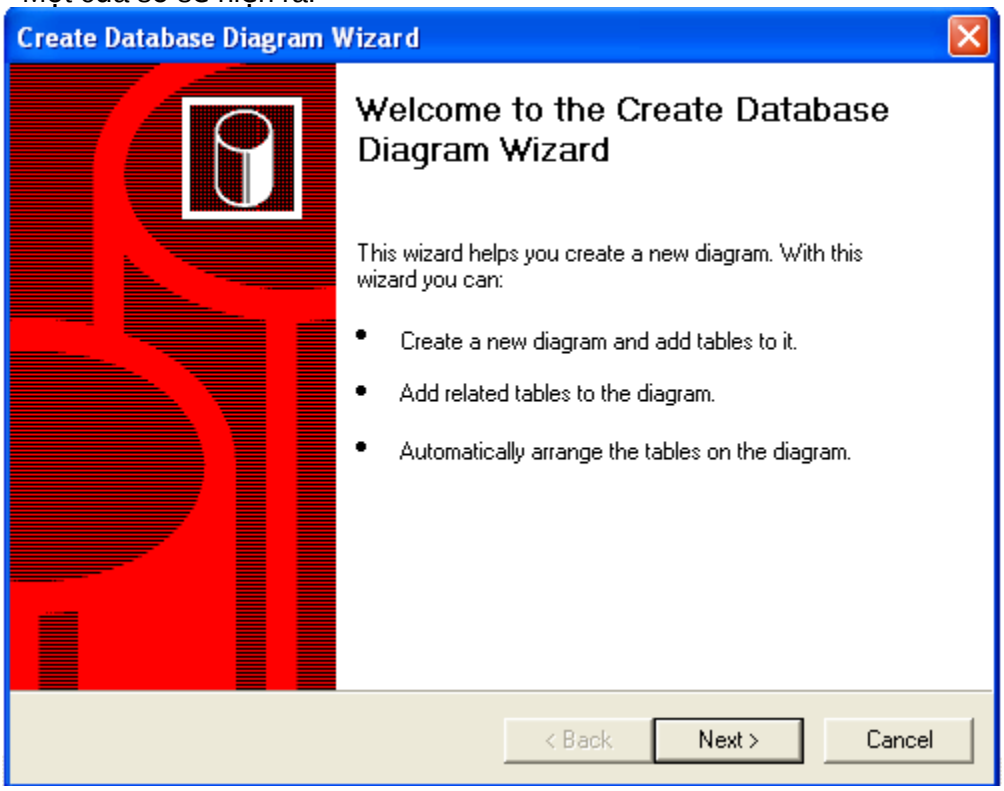
4.6 Tạo quan hệ trong SQL Server

Để tạo quan hệ cho một cơ sở dữ liệu trong SQL Server, chúng ta nhắp chuột phải vào thành phần Diagrams của cơ sở dữ liệu tương ứng → New Database Diagram…



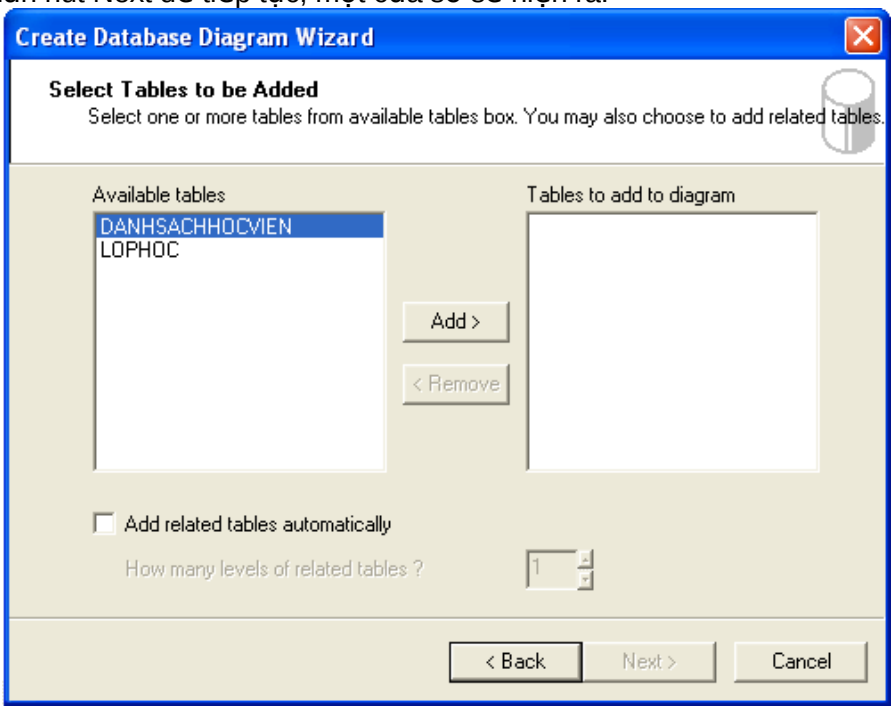
Hình 4.11:New Database Diagram…

Một cửa sổ sẽ hiện ra:



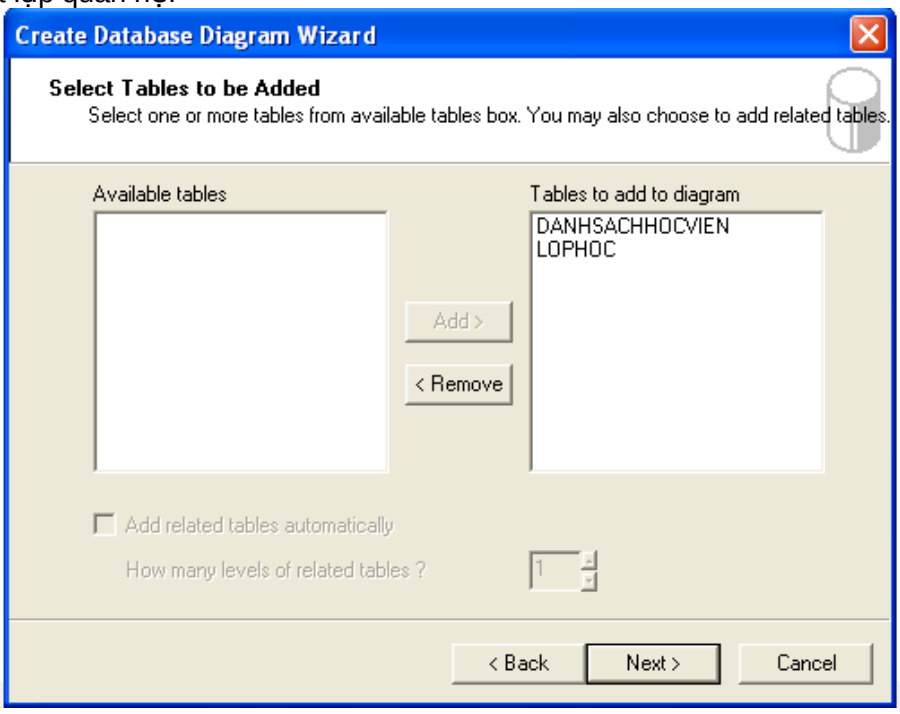
Hình 4.12 :Database Diagram

Nhấn nút Next để tiếp tục, một cửa sổ sẽ hiện ra:



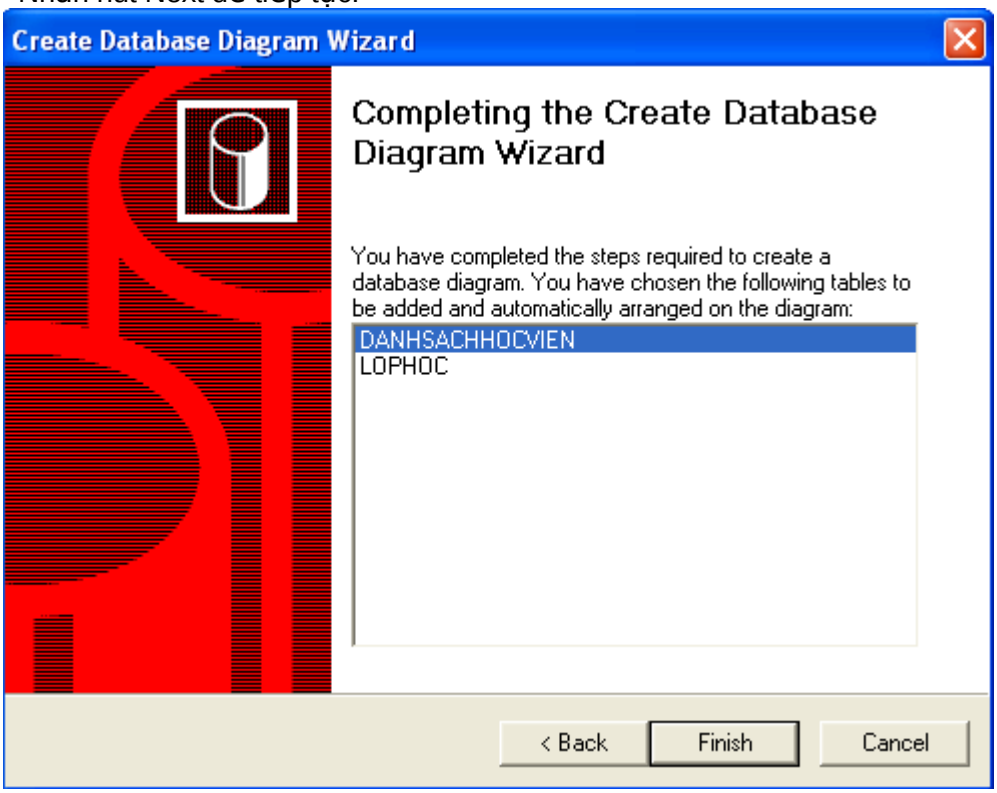
Hình 4.13:Select Table

Chọn bảng cần thiết lập quan hệ và nhấn nút Add để đưa các bảng này sang cửa sổ bên phải, chọn vào tùy chọn ***Add related tables automatically*** nếu muốn SQL tự động thiết lập quan hệ.



Hình 4.14:*Add related tables automatically*

Nhấn nút Next để tiếp tục:



Hình 4.15:completing the create Database diagram wizard

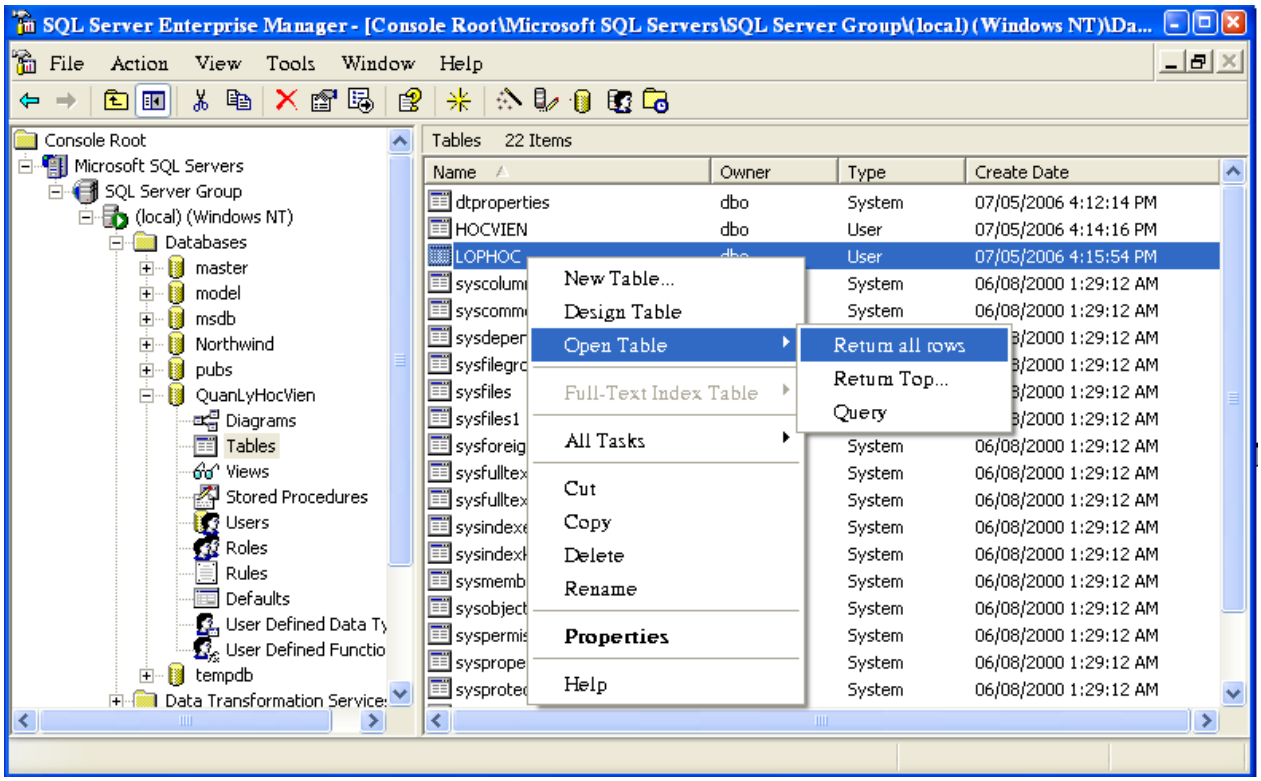
Nhấn nút Finish để hoàn tất.

4.7 Nhập dữ liệu trong SQL Server

Để nhập dữ liệu cho các bảng, chúng ta có thể sử dụng nhanh chóng công cụ Enterprise Manager thông qua các bước như sau:

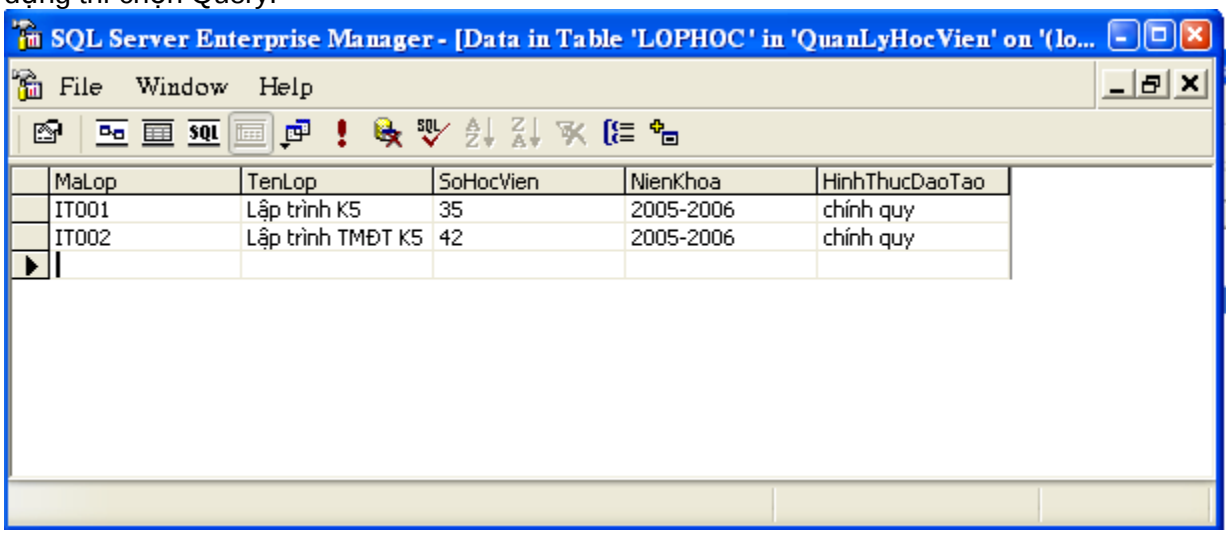
+ Trong cửa sổ Enterprise Manager, chúng ta chọn cơ sở dữ liệu → chọn Tables.

+ Nhấn chuột phải lên bảng cần nhập dữ liệu, chọn Open Table:



Hình 4.16:Nhập dữ liệu trong SQL Server

+ Nếu muốn nhập dữ liệu với hiện trạng xem tất cả các dòng dữ liệu thì chọn Return all rows (như hình minh họa), nếu muốn xem một số dòng đầu tiên thì chọn Reture Top..., hoặc muốn định nghĩa riêng một truy vấn trả về các dòng dữ liệu theo ý của người sử dụng thì chọn Query.



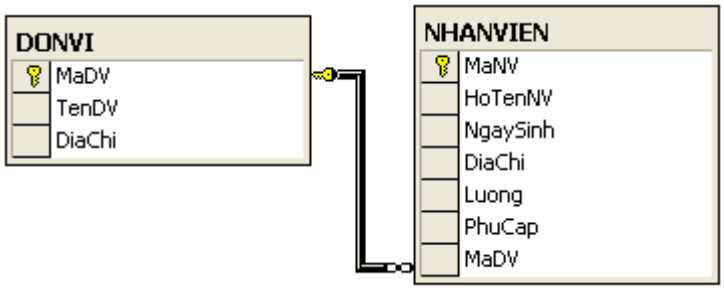
Hình 4.17:bảng dữ liệu

Trong cửa sổ hiện ra cho phép chúng ta nhập dữ liệu như cách thông thường chúng ta đã nhập dữ liệu với cơ sở dữ liệu Access.

BÀI TẬP THỰC HÀNH

1. Tạo một cơ sở dữ liệu tên QuanLyNhanVien.

2. Tạo và thiết lập quan hệ cho hai bảng như sơ đồ sau:



3. Nhập dữ liệu cho hai bảng, mỗi bảng ít nhất 10 bản ghi.

BÀI 5
THIẾT KẾ, BẢO TRÌ VÀ PHÁT TRIỂN MÔ HÌNH CLIENT/SERVER

MÃ BÀI: ITPRG3_17.5

Giới thiệu:

Đối với một doanh nghiệp thương mại điện tử, một cơ quan chính phủ, ... thì dữ liệu cực kỳ quan trọng và bảo mật. Vì vậy, đối với một chuyên gia quản lý cơ sở dữ liệu cần hết sức lưu ý về cách thức và giải pháp lưu trữ. Ngoài việc sử dụng công nghệ sao lưu tự động với nhiều bản, nhiều vị trí khác nhau, người quản trị còn phải thường xuyên theo dõi, điều hành để đảm bảo dữ liệu được an toàn tuyệt đối. Trong bài này chúng ta sẽ tìm ra những phương pháp để giải quyết vấn đề trên.

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

- Thiết kế được các cơ sở dữ liệu Client/Server với qui mô vừa và nhỏ
- Quản lý được các tập tin CSDL trên mô hình client/server
- Quản lý được các thực thể của CSDL và từ điển dữ liệu
- Xây dựng được một số ứng dụng vừa và nhỏ trên môi trường Client/Server

Nội dung:

5.1 Đọc hồ sơ thiết kế cơ sở dữ liệu

5.2 Thiết kế cơ sở dữ liệu hoàn hảo ứng dụng cơ sở dữ liệu

5.3 Bảo mật cơ sở dữ liệu

5.4 Chuyển đổi cơ sở dữ liệu từ các nguồn cơ sở dữ liệu

5.5 Sao lưu dự phòng cơ sở dữ lịêu

5.6 Bảo trì cơ sở dữ liệu

5.1 Đọc hồ sơ thiết kế cơ sở dữ liệu

Bước đầu tiên của việc thiết kế cơ sở dữ liệu là chúng ta phải biết cách đọc hộ sơ thiết kế cơ sở dữ liệu. Việc đọc được sơ đồ này đòi hỏi chúng ta phải nắm rõ các ký hiệu và quy định đã được trình bày trong môn học Cơ sở dữ liệu.

5.2 Thiết kế cơ sở dữ liệu hoàn hảo ứng dụng cơ sở dữ liệu

Khi thiết kế cơ sở dữ liệu chúng ta cần hết sức cẩn thận vì việc thiết kế sai, thiếu sót dẫn đến mất rất nhiều công sức và tiền bạc. Việc xây dựng một ứng dụng hay một Website có thành công hay không thì điều đầu tiên phải nói đến là cơ sở dữ liệu có tốt hay không, có an toàn và hiệu quả hay không. Do đó yêu cầu chúng ta phải quan tâm đúng mức để cơ sở dữ liệu được hoàn hảo trước khi chuyển sang bước lập trình.

5.3 Bảo mật cơ sở dữ liệu

5.3.1 Các khái niệm

Bảo mật là một trong những yếu tố đóng vai trò quan trọng đối với sự sống còn của cơ sở dữ liệu. Hầu hết các hệ quản trị cơ sở dữ liệu thương mại hiện nay đều cung cấp khả năng bảo mật cơ sở dữ liệu với những chức năng như:

1 • Cấp phát quyền truy cập cơ sở dữ liệu cho người dùng và các nhóm người dùng, phát hiện và ngăn chặn những thao tác trái phép của người sử dụng trên cơ sở dữ liệu.

2 • Cấp phát quyền sử dụng các câu lệnh, các đối tượng cơ sở dữ liệu đối với người dùng.

3 • Thu hồi (huỷ bỏ) quyền của người dùng.

Bảo mật dữ liệu trong SQL được thực hiện dựa trên ba khái niệm chính sau đây:

1 • **Người dùng cơ sở dữ liệu (Database user):** Là đối tượng sử dụng cơ sở dữ liệu, thực thi các thao tác trên cơ sở dữ liệu như tạo bảng, truy xuất dữ liệu,... Mỗi một người dùng trong cơ sở dữ liệu được xác định thông qua tên người dùng (User ID). Một tập nhiều người dùng có thể được tổ chức trong một nhóm và được gọi là nhóm người dùng (User Group). Chính sách bảo mật cơ sở dữ liệu có thể được áp dụng cho mỗi người dùng hoặc cho các nhóm người dùng.

2 • **Các đối tượng cơ sở dữ liệu (Database objects):** Tập hợp các đối tượng, các cấu trúc lưu trữ được sử dụng trong cơ sở dữ liệu như bảng, khung nhìn,

thủ tục, hàm được gọi là các đối tượng cơ sở dữ liệu. Đây là những đối tượng cần được bảo vệ trong chính sách bảo mật của cơ sở dữ liệu.

3 • **Đặc quyền (Privileges):** Là tập những thao tác được cấp phát cho người dùng trên các đối tượng cơ sở dữ liệu. Chẳng hạn một người dùng có thể truy xuất dữ liệu trên một bảng bằng câu lệnh SELECT nhưng có thể không thể thực hiện các câu lệnh INSERT, UPDATE hay DELETE trên bảng đó.

SQL cung cấp hai câu lệnh cho phép chúng ta thiết lập các chính sách bảo mật trong cơ sở dữ liệu:

1 • Lệnh GRANT: Sử dụng để cấp phát quyền cho người sử dụng trên các đối tượng cơ sở dữ liệu hoặc quyền sử dụng các câu lệnh SQL trong cơ sở dữ liệu.

2 • Lệnh REVOKE: Được sử dụng để thu hồi quyền đối với người sử dụng.

5.3.2 Cấp phát quyền

Câu lệnh GRANT được sử dụng để cấp phát quyền cho người dùng hay nhóm người dùng trên các đối tượng cơ sở dữ liệu. Câu lệnh này thường được sử dụng trong các trường hợp sau:

1 • Người sở hữu đối tượng cơ sở dữ liệu muốn cho phép người dùng khác quyền sử dụng những đối tượng mà anh ta đang sở hữu.

2 • Người sở hữu cơ sở dữ liệu cấp phát quyền thực thi các câu lệnh (như CREATE TABLE, CREATE VIEW,...) cho những người dùng khác.

5.3.2.1 Cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu

Chỉ có người sở hữu cơ sở dữ liệu hoặc người sở hữu đối tượng cơ sở dữ liệu mới có thể cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu. Câu lệnh GRANT trong trường hợp này có cú pháp như sau:

```
GRANT ALL [PRIVILEGES]| các_quyền_cấp_phát
[(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn
|ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]
|ON tên_thủ_tục
|ON tên_hàm
TO danh_sách_người_dùng | nhóm_người_dùng
[WITH GRANT OPTION ]
```

Trong đó:

ALL Cấp phát tất cả các quyền cho người dùng trên đối tượng cơ sở dữ

[PRIVILEGES] liệu được chỉ định. Các quyền có thể cấp phát cho người dùng bao gồm:

1 • Đối với bảng, khung nhìn, và hàm trả về dữ liệu kiểu bảng: SELECT, INSERT, DELETE, UPDATE và REFERENCES.

2 • Đối với cột trong bảng, khung nhìn: SELECT và UPDATE.

3 • Đối với thủ tục lưu trữ và hàm vô hướng:

1 EXECUTE.

Trong các quyền được đề cập đến ở trên, quyền REFERENCES được sử dụng nhằm cho phép tạo khóa ngoài tham chiếu đến bảng cấp phát.

các_quyền_cấp_phát	Danh sách các quyền cần cấp phát cho người dùng trên đối tượng cơ sở dữ liệu được chỉ định. Các quyền được phân cách nhau bởi dấu phẩy
tên_bảng | tên_khung_nhìn	Tên của bảng hoặc khung nhìn cần cấp phát quyền.
danh_sách_cột	Danh sách các cột của bảng hoặc khung nhìn cần cấp phát quyền.
tên_thủ_tục	Tên của thủ tục được cấp phát cho người dùng.
tên_hàm	Tên hàm (do người dùng định nghĩa) được cấp phát quyền.
danh_sách_người_dùng	Danh sách tên người dùng nhận quyền được cấp phát. Tên của các người dùng được phân cách nhau bởi dấu phẩy.
WITH GRANT OPTION	Cho phép người dùng chuyển tiếp quyền cho người dùng khác.

Các ví dụ dưới đây sẽ minh hoạ cho ta cách sử dụng câu lệnh GRANT để cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu.

Ví dụ 1: Cấp phát cho người dùng có tên thuchanh quyền thực thi các câu lệnh SELECT, INSERT và UPDATE trên bảng LOP

GRANT SELECT,INSERT,UPDATE

ON lop

TO thuchanh

Cho phép người dùng thuchanh quyền xem họ tên và ngày sinh của các sinh viên (cột HODEM,TEN và NGAYSINH của bảng SINHVIEN)

GRANT SELECT

(hodem,ten,ngaysinh) ON sinhvien

TO thuchanh

hoặc:

GRANT SELECT

ON sinhvien(hodem,ten,ngaysinh)

TO thuchanh

FROM sinhvien

Với quyền được cấp phát như trên, người dùng thuchanh có thể thực hiện câu lệnh sau trên bảng SINHVIEN

SELECT hoden,ten,ngaysinh

Nhưng câu lệnh dưới đây lại không thể thực hiện được

SELECT * FROM sinhvien

Trong trường hợp cần cấp phát tất cả các quyền có thể thực hiện được trên đối tượng cơ sở dữ liệu cho người dùng, thay vì liệt kê các câu lệnh, ta chỉ cần sử dụng từ khoá ALL PRIVILEGES (từ khóa PRIVILEGES có thể không cần chỉ định). Câu lệnh dưới đây cấp phát cho người dùng thuchanh các quyền SELECT, INSERT, UPDATE, DELETE VÀ REFERENCES trên bảng DIEMTHI

GRANT ALL

ON DIEMTHI

TO thuchanh

Khi ta cấp phát quyền nào đó cho một người dùng trên một đối tượng cơ sở dữ liệu, người dùng đó có thể thực thi câu lệnh được cho phép trên đối tượng đã cấp phát. Tuy nhiên, người dùng đó không có quyền cấp phát những quyền mà mình được phép cho những người sử dụng khác.Trong một số trường hợp, khi ta cấp phát quyền cho một người dùng nào đó, ta có thể cho phép người đó chuyển tiếp quyền cho người dùng khác bằng cách chỉ định tuỳ chọn WITH GRANT OPTION trong câu lệnh GRANT.

Ví dụ 2: Cho phép người dùng thuchanh quyền xem dữ liệu trên bảng SINHVIEN đồng thời có thể chuyển tiếp quyền này cho người dùng khác

GRANT SELECT

ON sinhvien

TO thuchanh

WITH GRANT OPTION

5.3.2.2 Cấp phát quyền thực thi các câu lệnh

Ngoài chức năng cấp phát quyền cho người sử dụng trên các đối tượng cơ sở dữ liệu, câu lệnh GRANT còn có thể sử dụng để cấp phát cho người sử dụng một số quyền trên hệ quản trị cơ sở dữ liệu hoặc cơ sở dữ liệu. Những quyền có thể cấp phát trong trường hợp này bao gồm:

1 • Tạo cơ sở dữ liệu: CREATE DATEBASE.

2 • Tạo bảng: CREATE RULE

3 • Tạo khung nhìn: CREATE VIEW

1 • Tạo thủ tục lưu trữ: CREATE PROCEDURE

2 • Tạo hàm: CREATE FUNCTION

3 • Sao lưu cơ sở dữ liệu: BACKUP DATABASE

Câu lệnh GRANT sử dụng trong trường hợp này có cú pháp như sau:

GRANT ALL | danh_sách_câu_lênh

TO danh_sách_người_dùng

Ví dụ 3: Để cấp phát quyền tạo bảng và khung nhìn cho người dùng có tên là thuchanh, ta sử dụng câu lệnh như sau:

GRANT CREATE TABLE,CREATE VIEW

TO thuchanh

Với câu lệnh GRANT, ta có thể cho phép người sử dụng tạo các đối tượng cơ sở dữ liệu trong cơ sở dữ liệu. Đối tượng cơ sở dữ liệu do người dùng nào tạo ra sẽ do người đó sở hữu và do đó người này có quyền cho người dùng khác sử dụng đối tượng và cũng có thể xóa bỏ (DROP) đối tượng do mình tạo ra.

Khác với trường hợp sử dụng câu lệnh GRANT để cấp phát quyền trên đối tượng cơ sở dữ liệu, câu lệnh GRANT trong trường hợp này không thể sử dụng tuỳ chọn WITH GRANT OPTION, tức là người dùng không thể chuyển tiếp được các quyền thực thi các câu lệnh đã được cấp phát.

5.3.3 Thu hồi quyền

Câu lệnh REVOKE được sử dụng để thu hồi quyền đã được cấp phát cho người dùng. Tương ứng với câu lệnh GRANT, câu lệnh REVOKE được sử dụng trong hai trường hợp:

1 • Thu hồi quyền đã cấp phát cho người dùng trên các đối tượng cơ sở dữ liệu.

2 • Thu hồi quyền thực thi các câu lệnh trên cơ sở dữ liệu đã cấp phát cho người dùng.

5.3.3.1 Thu hồi quyền trên đối tượng cơ sở dữ liệu:

Cú pháp câu lệnh REVOKE sử dụng để thu hồi quyền đã cấp phát trên đối tượng cơ sở dữ liệu có cú pháp như sau:

```
REVOKE [GRANT OPTION FOR]
ALL [PRIVILEGES]| các_quyền_cần_thu_hồi
[(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn
|ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]
|ON tên_thủ_tục
|ON tên_hàm
FROM danh_sách_người_dùng
[CASCADE]
```

Câu lệnh REVOKE có thể sử dụng để thu hồi một số quyền đã cấp phát cho người dùng hoặc là thu hồi tất cả các quyền (ALL PRIVILEGES).

Ví dụ 4: Thu hồi quyền thực thi lệnh INSERT trên bảng LOP đối với người dùng thuchanh.

```
REVOKE INSERT
ON lop
FROM thuchanh
```

Giả sử người dùng thuchanh đã được cấp phát quyền xem dữ liệu trên các cột HODEM, TEN và NGAYSINH của bảng SINHVIEN, câu lệnh dưới đây sẽ thu hồi quyền đã cấp phát trên cột NGAYSINH (chỉ cho phép xem dữ liệu trên cột HODEM và TEN)

```
REVOKE SELECT
ON sinhvien(ngaysinh)
FROM thuchanh
```

Khi ta sử dụng câu lệnh REVOKE để thu hồi quyền trên một đối tượng cơ sở dữ liêu từ một người dùng náo đó, chỉ những quyền mà ta đã cấp phát trước đó mới được thu hồi, những quyền mà người dùng này được cho phép bởi những người dùng khác vẫn còn có hiệu lực. Nói cách khác, nếu hai người dùng khác nhau cấp phát cùng các quyền trên cùng một đối tượng cơ sở dữ liệu cho một người dùng khác, sau đó người thu nhất thu hồi lại quyền đã cấp phát thì những quyền mà người dùng thứ hai cấp phát vẫn có hiệu lực.

Ví dụ 5: Giả sử trong cơ sở dữ liệu ta có 3 người dùng là A, B và C. A và B đều có quyền sử dụng và cấp phát quyền trên bảng R. A thực hiện lệnh sau để cấp phát quyền xem dữ liệu trên bảng R cho C:

GRANT SELECT

ON R TO C

và B cấp phát quyền xem và bổ sung dữ liệu trên bảng R cho C bằng câu lệnh:

GRANT SELECT, INSERT

ON R TO C

Như vậy, C có quyền xem và bổ sung dữ liệu trên bảng R. Bây giờ, nếu B thực hiện lệnh:

REVOKE SELECT, INSERT

ON R FROM C

Người dùng C sẽ không còn quyền bổ sung dữ liệu trên bảng R nhưng vẫn có thể xem được dữ liệu của bảng này (quyền này do A cấp cho C và vẫn còn hiệu lực).

Nếu ta đã cấp phát quyền cho người dùng nào đó bằng câu lệnh GRANT với tuỳ chọn WITH GRANT OPTION thì khi thu hồi quyền bằng câu lệnh REVOKE phải chỉ định tuỳ chọn CASCADE. Trong trường hợp này, các quyền được chuyển tiếp cho những người dùng khác cũng đồng thời được thu hồi.

Ví dụ 6: Ta cấp phát cho người dùng A trên bảng R với câu lệnh GRANT như sau:

GRANT SELECT

ON R TO A

WITH GRANT OPTION

sau đó người dùng A lại cấp phát cho người dùng B quyền xem dữ liệu trên R với câu lệnh:

GRANT SELECT

ON R TO B

Nếu muốn thu hồi quyền đã cấp phát cho người dùng A, ta sử dụng câu lệnh REVOKE như sau:

```
REVOKE SELECT
ON NHANVIEN
FROM A CASCADE
```

Câu lệnh trên sẽ đồng thời thu hồi quyền mà A đã cấp cho B và như vậy cả A và B đều không thể xem được dữ liệu trên bảng R.

Trong trường hợp cần thu hồi các quyền đã được chuyển tiếp và khả năng chuyển tiếp các quyền đối với những người đã được cấp phát quyền với tuỳ chọn WITH GRANT OPTION, trong câu lệnh REVOKE ta chỉ định mệnh đề GRANT OPTION FOR.

Ví dụ 7: Trong ví dụ trên, nếu ta thay câu lệnh:

```
REVOKE SELECT
ON NHANVIEN
FROM A CASCADE
```

bởi câu lệnh:

```
REVOKE GRANT OPTION FOR SELECT
ON NHANVIEN
FROM A CASCADE
```

Thì B sẽ không còn quyền xem dữ liệu trên bảng R đồng thời A không thể chuyển tiếp quyền mà ta đã cấp phát cho những người dùng khác (tuy nhiên A vẫn còn quyền xem dữ liệu trên bảng R).

5.3.3.2 Thu hồi quyền thực thi các câu lênh:

Việc thu hồi quyền thực thi các câu lệnh trên cơ sở dữ liệu (CREATE DATABASE, CREATE TABLE, CREATE VIEW,...) được thực hiện đơn giản với câu lệnh REVOKE có cú pháp:

```
REVOKE ALL | các_câu_lệnh_cần_thu_hồi
FROM danh_sách_người_dùng
```

Ví dụ 8: Để không cho phép người dùng thuchanh thực hiện lệnh CREATE TABLE trên cơ sở dữ liệu, ta sử dụng câu lệnh:

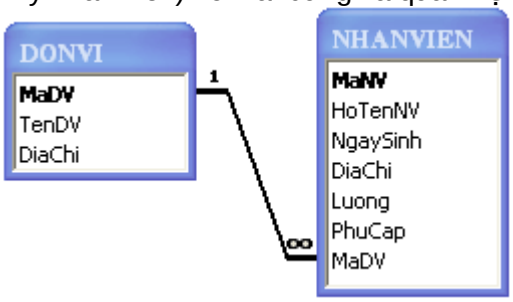
```
REVOKE CREATE TABLE
FROM thuchanh
```

5.4 Chuyển đổi cơ sở dữ liệu từ các nguồn cơ sở dữ liệu

Việc chuyển đổi cơ sở dữ liệu có thể thực hiện qua lại giữa các hệ quản trị cơ sở dữ liệu. Việc chuyển đổi chỉ thông qua một số bước đơn giản. Trong giới hạn của module này, chúng ta sẽ tìm hiểu về cách chuyển đổi qua lại giữa nguồn cơ sở dữ liệu SQL Server và cơ sở dữ liệu Access. Đối với các cơ sở dữ liệu khác, chúng ta cũng có cách làm tương tự.

5.4.1 Tạo cơ sở dữ liệu SQL Server từ nguồn cơ sở dữ liệu Access

Trước tiên, để thuận tiện cho việc thực hành chúng ta tạo một cơ sở dữ liệu Acces có tên QuanLyNhanVien (quản lý nhân viên) với hai bảng và quan hệ như sau:



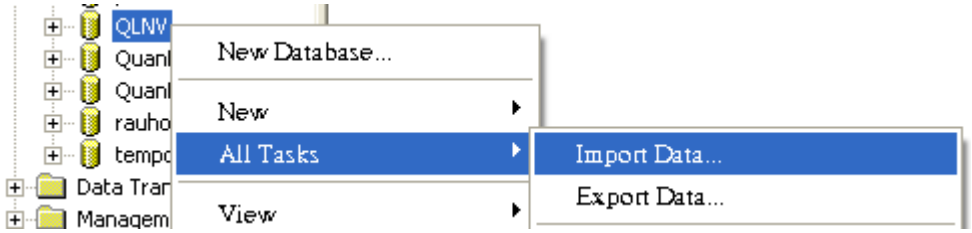
Hình 5.1: bảng quan hệ

Tiếp theo, chúng ta nhập một số bản ghi cho từng bảng và đóng cơ sở dữ liệu lại.

Các bước kết xuất dữ liệu từ Access:

1 - Tạo một cơ sở dữ liệu SQL mới , ở đây chúng ta đặt tên là **QLNV**.

2 - Click chuột phải lên cơ sở dữ liệu → All Tasks → Import Data…

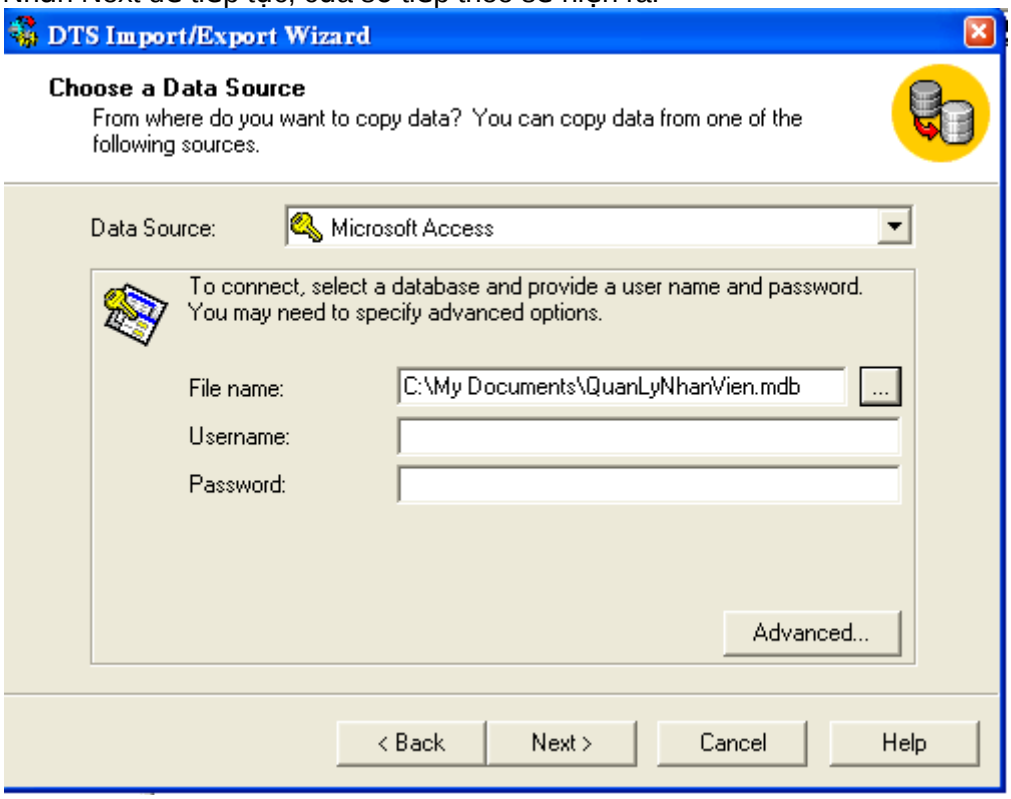


Hình 5.2:Import Data

một cửa sổ sẽ hiện ra như sau:

Hình 5.3:Data Transformation Services

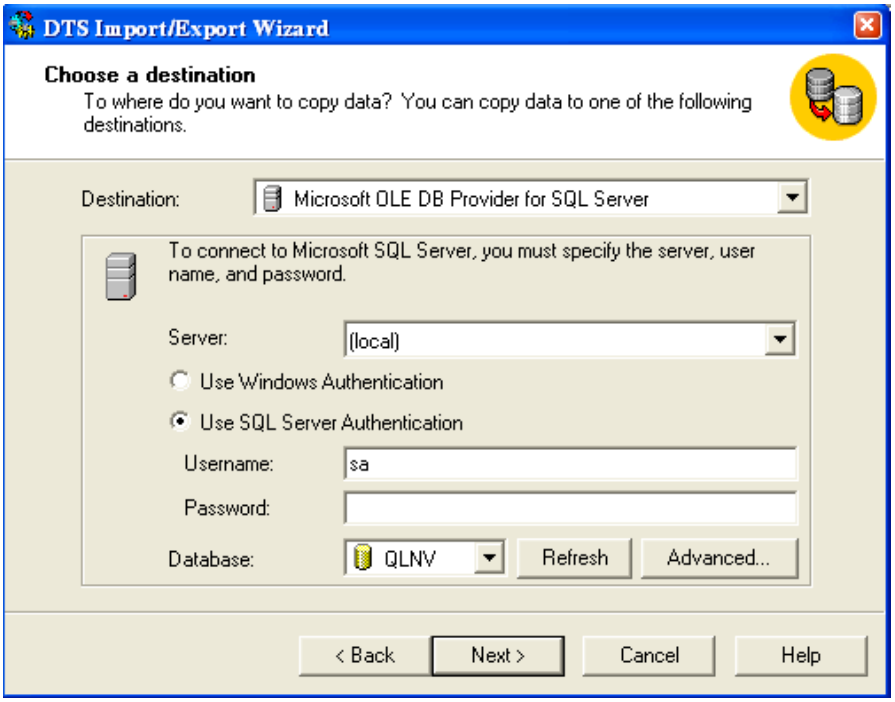
3 – Nhấn Next để tiếp tục, cửa sổ tiếp theo sẽ hiện ra:



Hình 5.4:Choose a data source

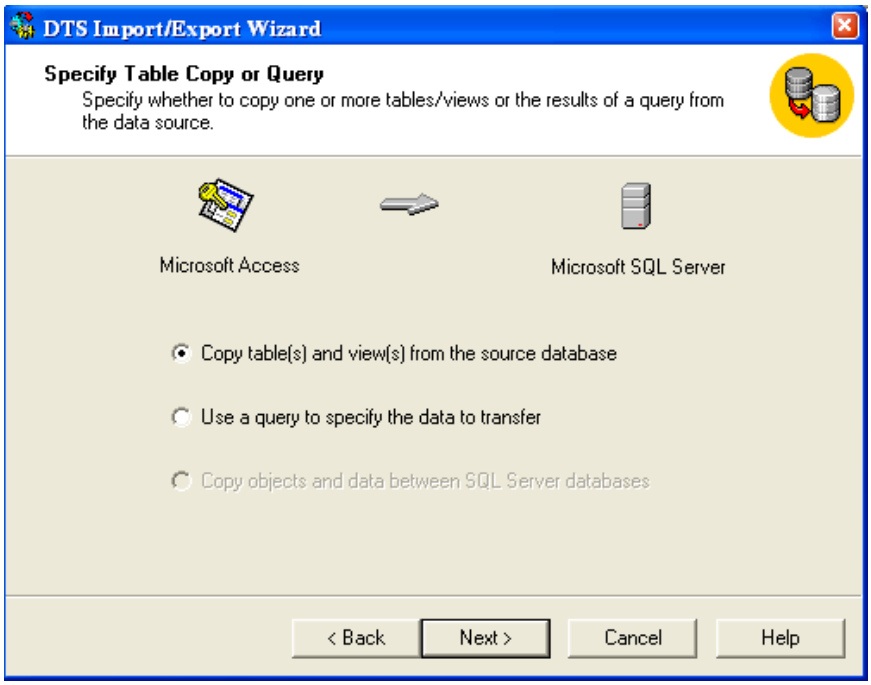
4 – Chọn nguồn cơ sở dữ liệu. Trong hộp Data Source chọn Microsoft Access, trong hộp File name chọn cơ sở dữ liệu Access đã được tạo trước (QuanLyNhanVien.mdb) như hình trên, nhấn Next để tiếp tục.

5 – Chọn cơ sở dữ liệu đích. Để mặc định như hình dưới đây, nếu bạn chọn kết nối với tài khoản của SQL Server thì đánh dấu vào ô **Use SQL Server Authentication**, trong hộp **Username** nhập tài khoản đăng nhập quản trị cơ sở dữ liệu như hình dưới đây:



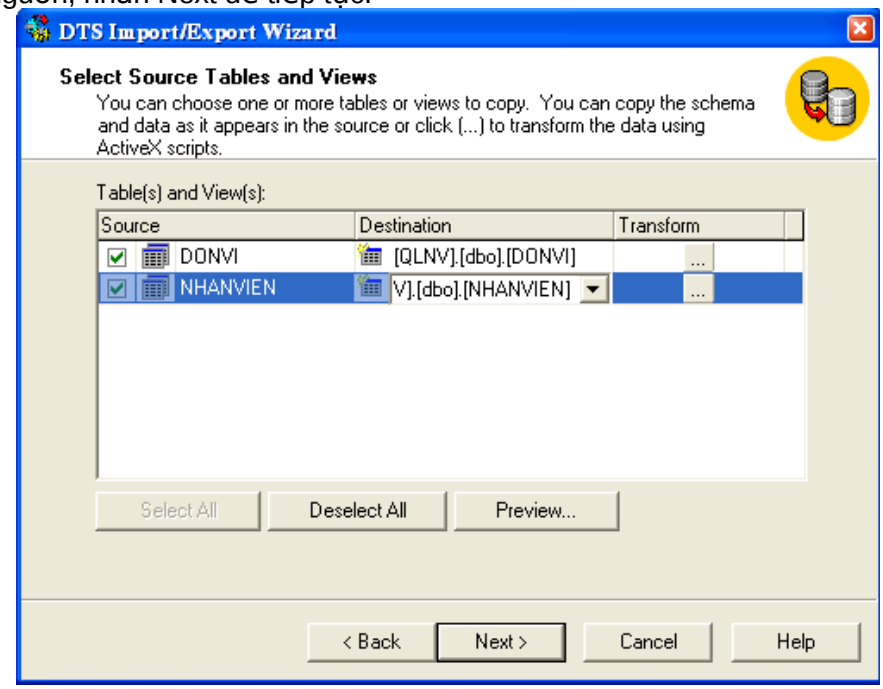
Hình 5.5:đăng nhập quản trị cơ sở dữ liệu

nhấn Next để tiếp tục, cửa sổ mới hiện ra:



Hình 5.6 :nhấn Next tiếp tục

6 – Để mặc định như hình trên nếu muốn sao chép toàn bộ bảng và truy vấn từ cơ sở dữ liệu nguồn, nhấn Next để tiếp tục.



Hình 5.7:truy vấn từ cơ sở dữ liệu nguồn

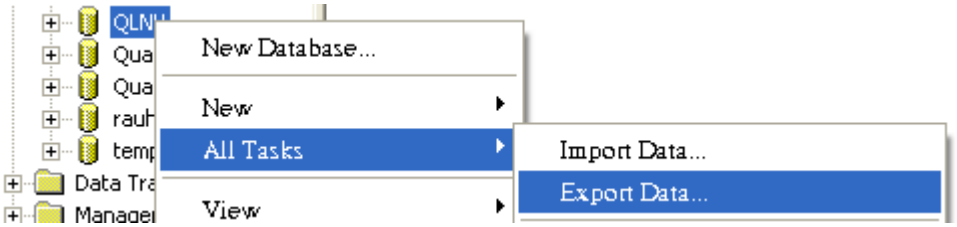
7 – Đánh dấu vào bảng và truy vấn cần sao chép, nhấn Next → Next để tiếp tục:

Hình 5.8:Next để tiếp tục

8. Nhấn **Finish**. Đợi trong vài giây, phụ thuộc vào dung lượng cơ sở dữ liệu nguồn mà thời gian này có thể lâu hơn, sau đó nhấn **Done** để hoàn tất công việc.

5.4.2 Xuất cơ sở dữ liệu SQL Server sang cơ sở dữ liệu Access

Nhấn chuột phải lên cơ sở dữ liệu SQL Server → chọn All Tasks → Export Data …



Hình 5.9:xuất cơ sở dữ liệu

Thực hiện lần lượt 8 bước tương tự như trên, chỉ thay đổi cơ sở dữ liệu nguồn và đích.

5.5 Sao lưu dự phòng cơ sở dữ liệu

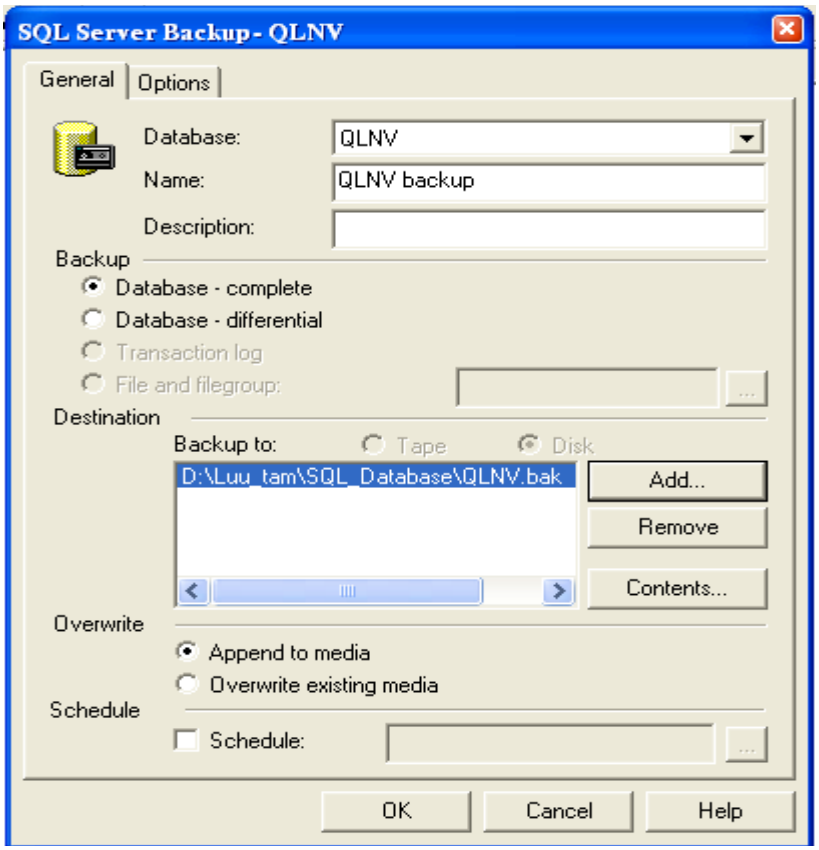
Trong phần này chúng ta sẽ bàn về cách sao lưu cơ sở dữ liệu (backup database).

Thuật Ngữ	Giải Thích
Backup	Quá trình copy toàn bộ hay một phần của database, transaction log, file hay file group hình thành một backup set. Backup set được chứa trên backup media (tape or disk) bằng cách sử dụng một backup device (tape drive name hay physical filename)
Backup Device	Một file vật lý (như C:\SQLBackups\Full.bak) hay tape drive cụ thể (như \\.\Tape0) dùng để record một backup vào một backup media.
Backup File	File chứa một backup set
Backup Media	Disk hay tape được sử dụng để chứa một backup set. Backup media có thể chứa nhiều backup sets (ví dụ như từ nhiều SQL Server 2000 backups và từ nhiều Windows 2000 backups).
Backup Set	Một bộ backup từ một lần backup đơn được chứa trên backup media.

Chúng ta có thể tạo một backup device cố định (permanent) hay tạo ra một backup file mới cho mỗi lần backup. Thông thường chúng ta sẽ tạo một backup device cố định để có thể dùng đi dùng lại đặc biệt cho việc tự động hóa công việc backup. Để tạo một backup device dùng Enterprise Manager bạn chọn **Management->Backup** rồi **Right-click->New Backup Device**. Ngoài ra bạn có thể dùng **sp_addumpdevice** system stored procedure như ví dụ sau:

```
USE Master
Go
Sp_addumpdevice 'disk' , 'FullBackupDevice' , 'E:\SQLBackups\Full.bak'
```

Để backup database bạn có thể dùng Backup Wizard hoặc click lên trên database muốn backup sau đó **Right-click->All Tasks->Backup Database...** sẽ hiện ra window như hình vẽ sau:



Hình 5.10:Backup Database

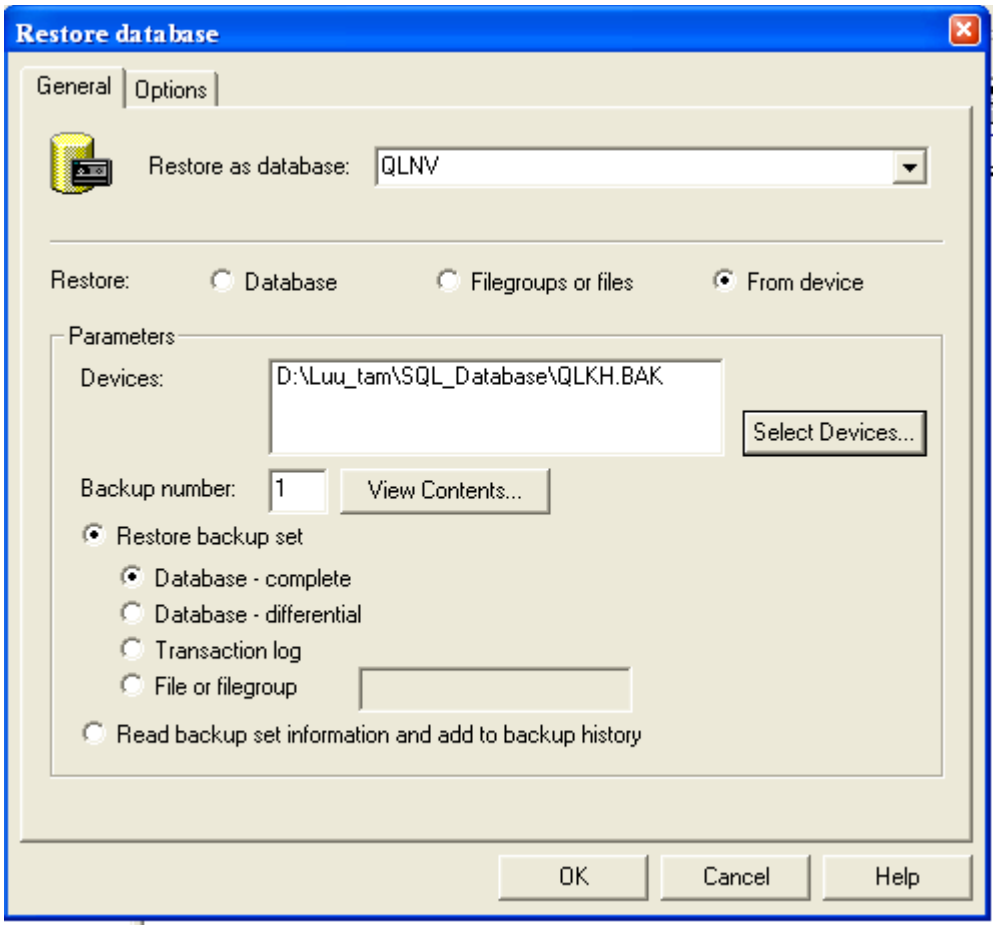
Sau đó dựa tùy theo yêu cầu của database mà chọn các option thích hợp. Ta có thể schedule cho SQL Server backup định kỳ.

5.6 Bảo trì cơ sở dữ liệu

Việc sao lưu dự phòng cơ sở dữ liệu nhằm phục hồi lại khi cần thiết, đó là một quá trình bảo trì cơ sở dữ liệu.

Trước khi phục hồi cơ sở dữ liệu (restore database) ta phải xác định được thứ tự file cần phục hồi. Các thông tin này được SQL Server chứa trong **msdb**. Việc phục hồi lại dữ liệu có thể thực hiện như sau:

Right-click->All Tasks->Restore database... sẽ thấy window như hình vẽ sau:



Hình 5.11:Restore database

Nếu chúng ta muốn phục hồi cơ sở dữ liệu từ một tập tin khác của SQL Server hay từ một server khác bạn có chọn tùy chọn **From device** và chọn backup device (file backup) tương ứng .

BÀI TẬP THỰC HÀNH

1. Tạo hai tài khoản đăng nhập là **nguoisudung1, nguoisudung2** và cho phép hai tài khoản này sử dụng cơ sở dữ liệu QuanLyNhanVien.

2. Cấp phát tất cả các quyền cho nguoisudung1, cấp quyền SELECT cho nguoisudung2 trên hai bảng của cơ sở dữ liệu QuanLyNhanVien.

3. Thu hồi quyền Delete đối với nguoisudung1 trên bảng DONVI.

4. Sao lưu cơ sở dữ liệu QuanLyNhanVien và lưu trên thư mục C:\BackUp\Data với tên **QLNV.bak**.

5. Xóa cơ sở dữ liệu QuanLyNhanVien khỏi hệ thống và phục hồi lại như cũ nhờ tập tin **QLNV.bak**.

BÀI 6
LẬP TRÌNH TRÊN SQL SERVER

MÃ BÀI: ITPRG3-17.6

Giới thiệu:

Hệ quản trị cơ sở dữ liệu SQL Server cung cấp cho ta một công cụ khá mạnh cho phép người quản trị có thể lập trình quản lý dữ liệu SQL nhờ ngôn ngữ SQL, người ta gọi là ngôn ngữ truy vấn có cấu trúc. Việc lập trình trên SQL Server cho phép chúng ta thực hiện các công việc có hiệu quả hơn, chẳng hạn các ràng buộc được chặc chẽ và toàn vẹn hơn, các thao tác được bảo mật hơn, …

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

- Sử dụng được các câu lệnh SQL trong Analysis
- Sử dụng được các tập tin batch
- Sử dụng được thủ tục lưu trữ trong lập trình ứng dụng
- Tạo được Trigger cho các bảng cơ sở dữ liệu

Nội dung:

6.1 Các câu lệnh SQL Server

6.1.1 Truy xuất dữ liệu với câu lệnh SELECT

Câu lệnh SELECT được sử dụng để truy xuất dữ liệu từ các dòng và các cột của một hay nhiều bảng, khung nhìn. Câu lệnh này có thể dùng để thực hiện phép chọn (tức là truy xuất một tập con các dòng trong một hay nhiều bảng), phép chiếu (tức là truy xuất một tập con các cột trong một hay nhiều bảng) và phép nối (tức là liên kết các dòng trong hai hay nhiều bảng để truy xuất dữ liệu). Ngoài ra, câu lệnh này còn cung cấp khả năng thực hiện các thao tác truy vấn và thống kê dữ liệu phức tạp khác.

Cú pháp chung của câu lệnh SELECT có dạng:

```
SELECT [ALL | DISTINCT][TOP n] danh_sách_chọn
[INTO tên_bảng_mới]
FROM danh_sách_bảng/khung_nhìn
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
[ORDER BY cột_sắp_xếp]
[COMPUTE danh_sách_hàm_gộp [BY danh_sách_cột]]
```

1) Mệnh đề FROM

Mệnh đề FROM trong câu lệnh SELECT được sử dung nhằm chỉ định các bảng và khung nhìn cần truy xuất dữ liệu. Sau FROM là danh sách tên của các bảng và khung nhìn tham gia vào truy vấn, tên của các bảng và khung nhìn được phân cách nhau bởi dấu phẩy.

Ví dụ 2.2: Câu lệnh dưới đây hiển thị danh sách các khoa trong trường

```
SELECT * FROM khoa
```

2) Danh sách chọn trong câu lệnh SELECT

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các trường, các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường, các biểu thức được chỉ định ngay sau từ khoá SELECT và phân cách nhau bởi dấu phẩy.

a. Chọn tất cả các cột trong bảng

Ví dụ: Câu lệnh

SELECT * FROM lop

b. Tên cột trong danh sách chọn

Ví dụ: Câu lệnh

SELECT malop,tenlop,namnhaphoc,khoa

FROM lop

Lưu ý: Nếu truy vấn được thực hiện trên nhiều bảng/khung nhìn và trong các bảng/khung nhìn có các trường trùng tên thì tên của những trường này nếu xuất hiện trong danh sách chọn phải được viết dưới dạng:

tên_bảng.tên_trường

Ví dụ:

SELECT malop, tenlop, lop.makhoa, tenkhoa

FROM lop, khoa

WHERE lop.malop = khoa.makhoa

c. Thay đổi tiêu đề các cột

Ta sử dụng cách viết:

tiêu_đề_cột = tên_trường

hoặc: tên_trường AS tiêu_đề_cột

hoặc: tên_trường tiêu_đề_cột

Ví dụ: Câu lệnh dưới đây:

SELECT 'Mã lớp'= malop,tenlop 'Tên lớp',khoa AS 'Khoá'

FROM lop

d. Sử dụng cấu trúc CASE trong danh sách chọn

Cấu trúc này có cú pháp như sau:

CASE biểu_thức

WHEN biểu_thức_kiểm_tra THEN kết_quả

[...]

[ELSE kết_quả_của_else]

END

hoặc:

CASE

WHEN điều_kiện THEN kết_quả

[...]

[ELSE kết_quả_của_else]

END

Ví dụ: Để hiển thị mã, họ tên và giới tính (nam hoặc nữ) của các sinh viên, ta sử dụng câu lệnh

```
SELECT masv,hodem,ten,
CASE gioitinh
WHEN 1 THEN 'Nam'
ELSE 'Nữ'
END AS gioitinh
FROM sinhvien
```

hoặc:

```
SELECT masv,hodem,ten,
CASE
WHEN gioitinh=1 THEN 'Nam'
ELSE 'Nữ'
END AS gioitinh
FROM sinhvien
```

Kết quả của hai câu lệnh trên đều như nhau.

e. Hằng và biểu thức trong danh sách chọn

Ví dụ: câu lệnh dưới đây cho biết tên và số tiết của các môn học

```
SELECT tenmonhoc,sodvht*15 AS sotiet
FROM monhoc
```

Nếu trong danh sách chọn có sự xuất hiện của giá trị hằng thì giá trị này sẽ xuất hiện trong một cột của kết quả truy vấn ở tất cả các dòng

Ví dụ: Câu lệnh

```
SELECT tenmonhoc,'Số tiết: ',sodvht*15 AS sotiet
FROM monhoc
```

f. Loại bỏ các dòng dữ liệu trùng nhau trong kết quả truy vấn

Ta chỉ định thêm từ khóa DISTINCT ngay sau từ khoá SELECT.

Ví dụ: Hai câu lệnh dưới đây

```
SELECT khoa FROM lop
```

và:

SELECT DISTINCT khoa FROM lop

g. Giới hạn số lượng dòng trong kết quả truy vấn

Ta chỉ định thêm mệnh đề TOP ngay trước danh sách chọn của câu lệnh SELECT.

Ví dụ: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của 5 sinh viên đầu tiên trong danh sách:

SELECT TOP 5 hodem,ten,ngaysinh

FROM sinhvien

Ngoài cách chỉ định cụ số lượng dòng cần hiển thị trong kết quả truy vấn, ta có thể chỉ định số lượng các dòng cần hiển thị theo tỷ lệ phần trăm bằng cách sử dụng thêm từ khoá PERCENT như ở ví dụ dưới đây.

Ví dụ: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của 10% số lượng sinh viên hiện có trong bảng SINHVIEN

SELECT TOP 10 PERCENT hodem,ten,ngaysinh

FROM sinhvien

3) Chỉ định điều kiện truy vấn dữ liệu

Mệnh đề WHERE trong câu lệnh SELECT được sử dụng nhằm xác định các điều kiện đối với việc truy xuất dữ liệu. Sau mệnh đề WHERE là một biểu thức logic và chỉ những dòng dữ liệu nào thoả mãn điều kiện được chỉ định mới được hiển thị trong kết quả truy vấn.

* Trong mệnh đề WHERE thường sử dụng:

+ Các toán tử kết hợp điều kiện (AND, OR)

+ Các toán tử so sánh

+ Kiểm tra giới hạn của dữ liệu (BETWEEN/ NOT BETWEEN)o Danh sách

+ Kiểm tra khuôn dạng dữ liệu.

+ Các giá trị NULL

a. Các toán tử so sánh

Toán tử	ý nghĩa
=	Bằng
>	Lớn hơn

<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

b. Kiểm tra giới hạn của dữ liệu

Để kiểm tra xem giá trị dữ liệu nằm trong (ngoài) một khoảng nào đó, ta sử dụng toán tử BETWEEN (NOT BETWEEN) như sau:

Cách sử dụng	**Ý nghĩa**
giá_trị BETWEEN a AND b	a ≤ giá_trị ≤ b
giá_trị NOT BETWEEN a AND b	(giá_trị < a) AND (giá_trị>b)

Ví dụ: Câu lệnh dưới đây cho biết họ tên và tuổi của các sinh viên có tên là Bình và có tuổi nằm trong khoảng từ 20 đến 22

```
SELECT hodem,ten,year(getdate())-year(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten='Bình' AND
YEAR(GETDATE())-YEAR(ngaysinh) BETWEEN 20 AND 22
```

c. Danh sách (IN và NOT IN)

Ví dụ: Để biết danh sách các môn học có số đơn vị học trình là 2, 4 hoặc 5, thay vì sử dụng câu lệnh

```
SELECT * FROM monhoc
WHERE sodvht=2 OR sodvht=4 OR sodvht=5
```

ta có thể sử dụng câu lệnh

```
SELECT * FROM monhoc
WHERE sodvht IN (2,4,5)
```

d. Toán tử LIKE và các ký tự đại diện

Từ khoá LIKE (NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm kiếm. Chúng thường được kết hợp với các ký tự đại diện sau đây:

% :Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự

_ : Ký tự đơn bất kỳ

[] Ký tự đơn bất kỳ trong giới hạn được chỉ định (ví dụ [a-f]) hay một tập (ví dụ [abcdef])

[^] Ký tự đơn bất kỳ không nằm trong giới hạn được chỉ định (ví dụ [^a-f] hay một tập (ví dụ [^abcdef]).

Ví dụ: Câu lệnh dưới đây

```
SELECT hodem,ten FROM sinhvien
WHERE hodem LIKE 'Lê%'
```

Câu lệnh:

```
SELECT hodem,ten FROM sinhvien
WHERE hodem LIKE 'Lê%' AND ten LIKE '[AB]%'
```

e. Giá trị NULL: là một giá trị đặc biệt trong SQL Server dùng để chỉ một giá trị dữ liệu không chứa gì cả.

4) Tạo mới bảng dữ liệu từ kết quả của câu lệnh SELECT

Câu lệnh SELECT ... INTO có tác dụng tạo một bảng mới có cấu trúc và dữ liệu được xác định từ kết quả của truy vấn. Bảng mới được tạo ra sẽ có số cột bằng số cột được chỉ định trong danh sách chọn và số dòng sẽ là số dòng kết quả của truy vấn

Ví dụ:

```
SELECT hodem,ten,YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
INTO tuoisv
FROM sinhvien
```

Lưu ý: Nếu trong danh sách chọn có các biểu thức thì những biểu thức này phải được đặt tiêu đề.

5) Sắp xếp kết quả truy vấn

Câu lệnh ở ví dụ trên có thể được viết lại như sau:

```
SELECT hodem,ten,gioitinh,
YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
```

FROM sinhvien
WHERE ten='Bình'
ORDER BY 3, 4

6) Phép hợp

Cú pháp như sau:

Câu_lệnh_1
UNION [ALL] Câu_lệnh_2
[UNION [ALL] Câu_lệnh_3]
...
[UNION [ALL] Câu_lệnh_n]
[ORDER BY cột_sắp_xếp]
[COMPUTE danh_sách_hàm_gộp [BY danh_sách_cột]]

Trong đó

Câu_lệnh_1 có dạng

SELECT danh_sách_cột
[INTO tên_bảng_mới]
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]

và Câu_lệnh_i (i = 2,..,n) có dạng

SELECT danh_sách_cột
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]

7) Phép nối

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép nối. Một câu lệnh nối kết hợp các dòng dữ liệu trong các bảng khác nhau lại theo một hoặc nhiều điều kiện nào đó và hiển thị chúng trong kết quả truy vấn.

Xét hai bảng sau đây:

Bảng KHOA

Bảng LOP

o Chọn ra dòng trong bảng KHOA có tên khoa là Khoa Công nghệ Thông tin,

từ đó xác định được mã khoa (MAKHOA) là DHT02.

o Tìm kiếm trong bảng LOP những dòng có giá trị trường MAKHOA là DHT02 (tức là bằng MAKHOA tương ứng trong bảng KHOA) và đưa những dòng này vào kết quả truy vấn

Như vậy, để thực hiện được yêu cầu truy vấn dữ liệu trên, ta phải thực hiện phép nối giữa hai bảng KHOA và LOP với điều kiện nối là MAKHOA của KHOA bằng với MAKHOA của LOP.

a) Sử dụng phép nối

Ví dụ: Câu lệnh dưới đây hiển thị danh sách các sinh viên với các thông tin: mã sinh viên, họ và tên, mã lớp, tên lớp và tên khoa

```
SELECT masv,hodem,ten,sinhvien. malop, tenlop, tenkhoa
FROM sinhvien,lop,khoa
WHERE sinhvien.malop = lop.malop AND
lop.makhoa=khoa.makhoa
```

b) Các loại phép nối

Phép nối bằng và phép nối tự nhiên

Ví dụ: Câu lệnh dưới đây thực hiện phép nối bằng giữa hai bảng LOP và KHOA

```
SELECT *
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

Ví dụ: Để thực hiện phép nối tự nhiên, câu lệnh trong ví dụ trên được viết lại như sau

```
SELECT malop,tenlop,khoa,hedaotao,namnhaphoc,
siso,lop.makhoa,tenkhoa,dienthoai
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

hoặc viết dưới dạng ngắn gọn hơn:

```
SELECT lop.*,tenkhoa,dienthoai
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

8) Thống kê dữ liệu với GROUP BY

Ngoài khả năng thực hiện các yêu cầu truy vấn dữ liệu thông thường (chiếu, chọn, nối,…) như đã đề cập như ở các phần trước, câu lệnh SELECT còn cho phép thực hiện các thao tác truy vấn và tính toán thống kê trên dữ liệu như: cho biết tổng số tiết dạy của mỗi giáo viên, điểm trung bình các môn học của mỗi sinh viên,…

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu, và trên mỗi nhóm dữ liệu thực hiện tính toán các giá trị thống kê như tính tổng, tính giá trị trung bình,...

Các hàm gộp được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như là các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được phép xuất hiện trong mệnh đề WHERE

SQL cung cấp các hàm gộp dưới đây:

Hàm gộp Chức năng

SUM([ALL | DISTINCT] biểu_thức) Tính tổng các giá trị.

AVG([ALL | DISTINCT] biểu_thức) Tính trung bình của các giá trị

COUNT([ALL | DISTINCT] biểu_thức) Đếm số các giá trị trong biểu thức.

COUNT(*) Đếm số các dòng được chọn.

MAX(biểu_thức) Tính giá trị lớn nhất

MIN(biểu_thức) Tính giá trị nhỏ nhất

Trong đó:

o Hàm SUM và AVG chỉ làm việc với các biểu thức số.

o Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.

o Hàm COUNT(*) không bỏ qua các giá trị NULL.

Ví dụ: Để thống kê trung bình điểm lần 1 của tất cả các môn học, ta sử dụng câu lệnh như sau:

```
SELECT AVG(diemlan1)
FROM diemthi
```

còn câu lệnh dưới đây cho biết tuổi lớn nhất, tuổi nhỏ nhất và độ tuổi trung bình của tất

cả các sinh viên sinh tại Dalat:

```
SELECT MAX(YEAR(GETDATE())-YEAR(ngaysinh)),
```

MIN(YEAR(GETDATE())-YEAR(ngaysinh)),

AVG(YEAR(GETDATE())-YEAR(ngaysinh))

FROM sinhvien

WHERE noisinh=' Dalat'

Thống kê dữ liệu trên các nhóm

Trong trường hợp cần thực hiện tính toán các giá trị thống kê trên các nhóm dữ liệu, ta sử dụng mệnh đề GROUP BY để phân hoạch dữ liệu vào trong các nhóm. Các hàm gộp được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo các nhóm dữ liệu.

Ví dụ: Câu lệnh dưới đây cho biết sĩ số (số lượng sinh viên) của mỗi lớp

SELECT lop.malop,tenlop,COUNT(masv) AS siso

FROM lop,sinhvien

WHERE lop.malop=sinhvien.malop

GROUP BY lop.malop,tenlop

còn câu lệnh:

SELECT sinhvien.masv,hodem,ten,

sum(diemlan1*sodvht)/sum(sodvht)

FROM sinhvien,diemthi,monhoc

WHERE sinhvien.masv=diemthi.masv AND

diemthi.mamonhoc=monhoc.mamonhoc

GROUP BY sinhvien.masv,hodem,ten

cho biết trung bình điểm thi lần 1 các môn học của các sinh viên

Lưu ý: Trong trường hợp danh sách chọn của câu lệnh SELECT có cả các hàm gộp và

những biểu thức không phải là hàm gộp thì những biểu thức này phải có mặt đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Ví dụ: Dưới đây là một câu lệnh sai

SELECT lop.malop,tenlop,COUNT(masv)

FROM lop,sinhvien

WHERE lop.malop=sinhvien.malop

GROUP BY lop.malop

do thiếu trường TENLOP sau mệnh đề GROUP BY.

Chỉ định điều kiện đối với hàm gộp

Mệnh đề HAVING được sử dụng nhằm chỉ định điều kiện đối với các giá trị thống kê được sản sinh từ các hàm gộp tương tự như cách thức mệnh đề WHERE thiết lập các điều kiện cho câu lệnh SELECT.

6.1.2 Bổ sung, cập nhật và xoá dữ liệu

Các câu lệnh thao tác dữ liệu trong SQL không những chỉ sử dụng để truy vấn dữ liệu mà còn để thay đổi và cập nhật dữ liệu trong cơ sở dữ liệu. So với câu lệnh SELECT, việc sử dụng các câu lệnh để bổ sung, cập nhật hay xoá dữ liệu đơn giản hơn nhiều. Trong phần còn lại của chương này sẽ đề cập đến 3 câu lệnh:

o Lệnh **INSERT**

o Lệnh **UPDATE**

o Lệnh **DELETE**

1) Bổ sung dữ liệu

Dữ liệu trong các bảng được thể hiện dưới dạng các dòng (bản ghi). Để bổ sung thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT. Hầu hết các hệ quản trị CSDL dựa trên SQL cung cấp các cách dưới đây để thực hiện thao tác bổ sung dữ liệu cho bảng:

o + Bổ sung từng dòng dữ liệu với mỗi câu lệnh INSERT. Đây là các sử dụng thường gặp nhất trong giao tác SQL.

o + Bổ sung nhiều dòng dữ liệu bằng cách truy xuất dữ liệu từ các bảng dữ liệu khác.

+ Bổ sung từng dòng dữ liệu với lệnh INSERT

Để bổ sung một dòng dữ liệu mới vào bảng, ta sử dụng câu lệnh INSERT với cú pháp như sau:

INSERT INTO tên_bảng[(danh_sách_cột)]

VALUES(danh_sách_trị)

Trong câu lệnh INSERT, danh sách cột ngay sau tên bảng không cần thiết phải chỉ định nếu giá trị các trường của bản ghi mới được chỉ định đầy đủ trong danh sách trị.

Trong trường hợp này, thứ tự các giá trị trong danh sách trị phải bằng với số lượng các trường của bảng cần bổ sung dữ liệu cũng như phải tuân theo đúng thứ tự của các trường như khi bảng được định nghĩa.

Ví dụ: Câu lệnh dưới đây bổ sung thêm một dòng dữ liệu vào bảng KHOA

INSERT INTO khoa

VALUES('DHT10','Khoa Luật','054821135')

Trong trường hợp chỉ nhập giá trị cho một số cột trong bảng, ta phải chỉ định danh sách các cột cần nhập dữ liệu ngay sau tên bảng. Khi đó, các cột không được nhập dữ liệu sẽ nhận giá trị mặc định (nếu có) hoặc nhận giá trị NULL (nếu cột phép chấp nhận giá trị NULL). Nếu một cột không có giá trị mặc định và không chấp nhận giá trị NULL mà không được nhập dữ liệu, câu lệnh sẽ bị lỗi.

Ví dụ: Câu lệnh dưới đây bổ sung một bản ghi mới cho bảng SINHVIEN

INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)

VALUES('0241020008','Nguyễn Công','Chính',1,'C24102')

câu lệnh trên còn có thể được viết như sau:

INSERT INTO sinhvien

VALUES('0241020008','Nguyễn Công','Chính',

NULL,1,NULL,'C24102')

Bổ sung nhiều dòng dữ liệu từ bảng khác

Một cách sử dụng khác của câu lệnh INSERT được sử dụng để bổ sung nhiều dòng dữ liệu vào một bảng, các dòng dữ liệu này được lấy từ một bảng khác thông qua câu lệnh SELECT. Ở cách này, các giá trị dữ liệu được bổ sung vào bảng không được chỉ định tường minh mà thay vào đó là một câu lệnh SELECT truy vấn dữ liệu từ bảng khác.

Cú pháp câu lệnh INSERT có dạng như sau:

INSERT INTO tên_bảng[(danh_sách_cột)] câu_lệnh_SELECT

Ví dụ: Giả sử ta có bảng LUUSINHVIEN bao gồm các trường HODEM, TEN, NGAYSINH. Câu lệnh dưới đây bổ sung vào bảng LUUSINHVIEN các dòng dữ liệu có được từ câu truy vấn SELECT:

INSERT INTO luusinhvien

SELECT hodem,ten,ngaysinh

FROM sinhvien

WHERE noisinh like '%Huế%'

Khi bổ sung dữ liệu theo cách này cần lưu ý một số điểm sau:

o Kết quả của câu lệnh SELECT phải có số cột bằng với số cột được chỉ định trong bảng đích và phải tương thích về kiểu dữ liệu.

2) Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng.

Câu lệnh này có cú pháp như sau:

UPDATE tên_bảng

SET tên_cột = biểu_thức

[, ..., tên_cột_k = biểu_thức_k]

[FROM danh_sách_bảng]

[WHERE điều_kiện]

Sau UPDATE là tên của bảng cần cập nhật dữ liệu. Một câu lệnh UPDATE có thể cập nhật dữ liệu cho nhiều cột bằng cách chỉ định các danh sách tên cột và biểu thức tương ứng sau từ khoá SET. Mệnh đề WHERE trong câu lệnh UPDATE thường được sử dụng để chỉ định các dòng dữ liệu chịu tác động của câu lệnh (nếu không chỉ định, phạm vi tác động của câu lệnh được hiểu là toàn bộ các dòng trong bảng)

Ví dụ: Câu lệnh dưới đây cập nhật lại số đơn vị học trình của các môn học có số đơn vị học trình nhỏ hơn 2

UPDATE monhoc

SET sodvht = 3

WHERE sodvht = 2

Sử dụng cấu trúc CASE trong câu lệnh UPDATE

Cấu trúc CASE có thể được sử dụng trong biểu thức khi cần phải đưa ra các quyết định khác nhau về giá trị của biểu thức

Ví dụ: Giả sử ta có bảng NHATKYPHONG sau đây

Sau khi thực hiện câu lệnh:

UPDATE NHATKYPHONG

SET tienphong=songay*CASE WHEN loaiphong='A' THEN 100

WHEN loaiphong='B' THEN 70

ELSE 50

END

Điều kiện cập nhật dữ liệu liên quan đến nhiều bảng

Mệnh đề FROM trong câu lệnh UPDATE được sử dụng khi cần chỉ định các điều kiện liên quan đến các bảng khác với bảng cần cập nhật dữ liệu. Trong trường hợp này, trong mệnh đề WHERE thường có điều kiện nối giữa các bảng.

Ví dụ: Giả sử ta có hai bảng MATHANG và NHATKYBANHANG như sau:

Câu lệnh dưới đây sẽ cập nhật giá trị trường THANHTIEN của bảng NHATKYBANHANG theo công thức THANHTIEN = SOLUONG × GIA:

UPDATE nhatkybanhang

SET thanhtien = soluong*gia

FROM mathang

WHERE nhatkybanhang.mahang = mathang.mahang

Câu lệnh UPDATE với truy vấn con

Tương tự như trong câu lệnh SELECT, truy vấn con có thể được sử dụng trong mệnh đề WHERE của câu lệnh UPDATE nhằm chỉ định điều kiện đối với các dòng dữ liệu cần cập nhật dữ liệu.

Ví dụ: Câu lệnh ở trên có thể được viết như sau:

UPDATE nhatkybanhang

SET thanhtien = soluong*gia

FROM mathang

WHERE mathang.mahang =(SELECT mathang.mahang

FROM mathang

WHERE mathang.mahang=nhatkybanhang.mahang)

3) Xoá dữ liệu

Để xoá dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE. Cú pháp của câu lệnh này như sau:

DELETE FROM tên_bảng

[FROM danh_sách_bảng]

[WHERE điều_kiện]

Trong câu lệnh này, tên của bảng cần xoá dữ liệu được chỉ định sau DELETE FROM.

Mệnh đề WHERE trong câu lệnh được sử dụng để chỉ định điều kiện đối với các dòng dữ liệu cần xoá. Nếu câu lệnh DELETE không có mệnh đề WHERE thì toàn bộ các dòng dữ liệu trong bảng đều bị xoá.

Ví dụ: Câu lệnh dưới đây xoá khỏi bảng SINHVIEN những sinh viên sinh tại Huế

DELETE FROM sinhvien

WHERE noisinh LIKE '%Huế%'

Xoá dữ liệu khi điều kiện liên quan đến nhiều bảng

Nếu điều kiện trong câu lệnh DELETE liên quan đến các bảng không phải là bảng cần xóa dữ liệu, ta phải sử dụng thêm mệnh đề FROM và sau đó là danh sách tên các bảng đó. Trong trường hợp này, trong mệnh đề WHERE ta chỉ định thêm điều kiện nối giữa các bảng

Ví dụ: Câu lệnh dưới đây xoá ra khỏi bảng SINHVIEN những sinh viên lớp Tin K3

DELETE FROM sinhvien

FROM lop

WHERE lop.malop=sinhvien.malop AND tenlop='Tin K3'

Thay vì sử dụng câu lệnh DELETE, ta có thể sử dụng câu lệnh TRUNCATE có cú pháp như sau:

TRUNCATE TABLE tên_bảng

Ví dụ: Câu lệnh sau xoá toàn bộ dữ liệu trong bảng diemthi:

DELETE FROM diemthi

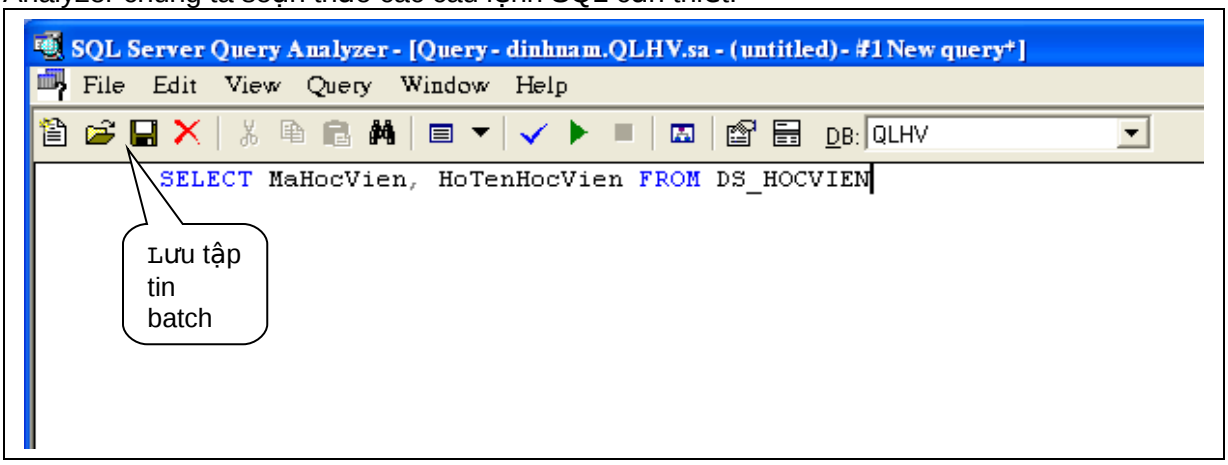
có tác dụng tương tự với câu lệnh

TRUNCATE TABLE diemthi

6.2 Tập tin batch

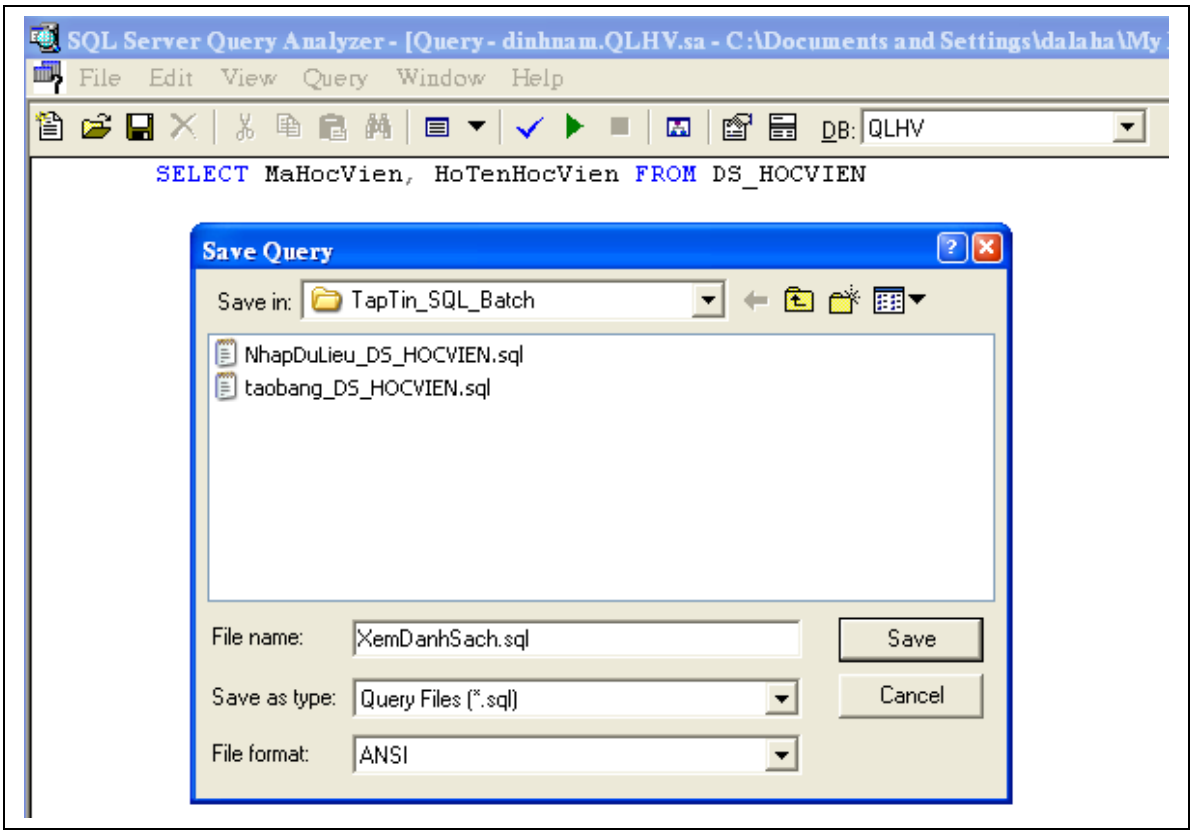
- Tập tin batch là tập tin văn bản chứa các câu lệnh SQL được lập trình sẵn, giúp việc cài đặt và sao chép các ứng dụng sử dụng cơ sở dữ liệu SQL mang lại hiệu quả nhanh chóng và dễ dàng.

+ Để tạo ra một tập tin batch trong môi trường SQL Server, trong cửa sổ Query Analyzer chúng ta soạn thảo các câu lệnh SQL cần thiết:



Hình 6.1 : Tập tin batch

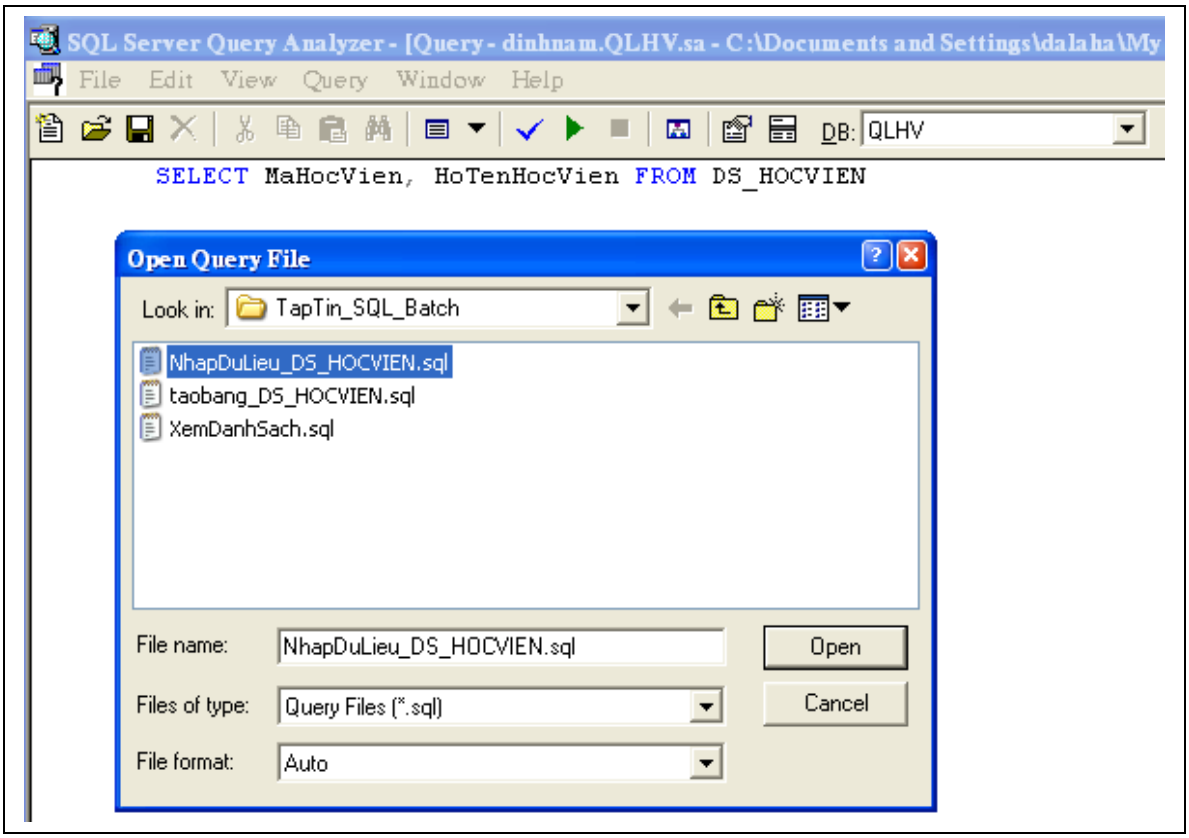
sau đó nhấn vào nút Save (hình đĩa mềm) trên thanh công cụ hoặc vào thực đơn File → Save, một cửa sổ sẽ xuất hiện:



Hình 6.2 : Lưu query

Mặc định, tập tin batch sẽ tự động lưu với phần mở rộng .sql, người sử dụng chỉ cần nhập tên tập tin, sau đó nhấn vào nút Save để lưu tập tin lên thư mục hoặc ổ đĩa.

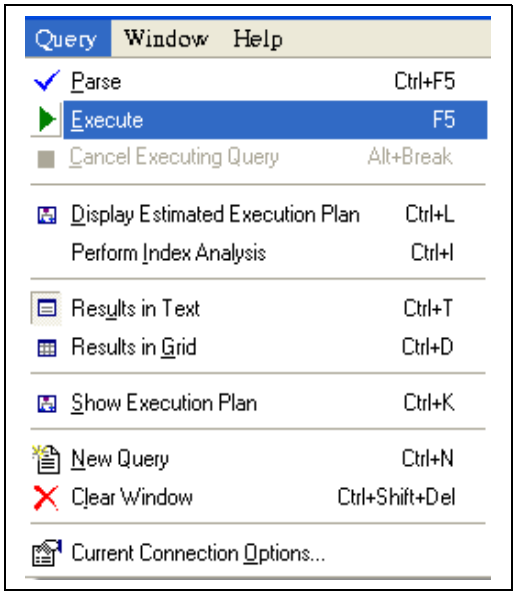
+ Để mở một tập tin batch, chúng ta nhấn vào nút Open trên thanh công cụ hoặc vào thực đơn File→ Open, một cửa sổ sẽ xuất hiện:



Hình 6.3: Mở tập query

Chọn tập tin batch cần mở, sau đó nhấn nút Open.

+ Để thực hiện tập tin batch, chúng ta có thể nhấn phím F5 hoặc vào thực đơn Query → Execute:



Hình 6.4 :Execute

6.3 Stored Procedure (thủ tục lưu trữ)

6.3.1 Các khái niệm

Như đã đề cập ở các chương trước, SQL được thiết kế và cài đặt như là một ngôn ngữ để thực hiện các thao tác trên cơ sở dữ liệu như tạo lập các cấu trúc trong cơ sở dữ liệu, bổ sung, cập nhật, xoá và truy vấn dữ liệu trong cơ sở dữ liệu. Các câu lệnh SQL được người sử dụng viết và yêu cầu hệ quản trị cơ sở dữ liệu thực hiện theo chế độ tương tác.

Các câu lệnh SQL có thể được nhúng vào trong các ngôn ngữ lập trình, thông qua đó chuỗi các thao tác trên cơ sở dữ liệu được xác định và thực thi nhờ vào các câu lệnh, các cấu trúc điều khiển của bản thân ngôn ngữ lập trình được sử dụng.

Với thủ tục lưu trữ, một phần nào đó khả năng của ngôn ngữ lập trình được đưa vào trong ngôn ngữ SQL. Một thủ tục là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm với những khả năng sau:

1 • Các cấu trúc điều khiển (IF, WHILE, FOR) có thể được sử dụng trong thủ tục.

2 • Bên trong thủ tục lưu trữ có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất được từ cơ sở dữ liệu.

3 • Một tập các câu lệnh SQL được kết hợp lại với nhau thành một khối lệnh bên trong một thủ tục. Một thủ tục có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình). Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

Sử dụng các thủ tục lưu trữ trong cơ sở dữ liệu sẽ giúp tăng hiệu năng của cơ sở dữ liệu, mang lại các lợi ích sau:

1 • Đơn giản hoá các thao tác trên cơ sở dữ liệu nhờ vào khả năng module hoá các thao tác này.

1 • Thủ tục lưu trữ được phân tích, tối ưu khi tạo ra nên việc thực thi chúng nhanh hơn nhiều so với việc phải thực hiện một tập rời rạc các câu lệnh SQL tương đương theo cách thông thường.

2 • Thủ tục lưu trữ cho phép chúng ta thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL. Điều này sẽ làm giảm thiểu sự lưu thông trên mạng.

3 • Thay vì cấp phát quyền trực tiếp cho người sử dụng trên các câu lệnh SQL và trên các đối tượng cơ sở dữ liệu, ta có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, nhờ đó tăng khả năng bảo mật đối với hệ thống.

6.3.2 Tạo thủ tục lưu trữ

Thủ tục lưu trữ được tạo bởi câu lệnh CREATE PROCEDURE với cú pháp như sau:

CREATE PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]
[WITH RECOMPILE|ENCRYPTION|RECOMPILE**,**ENCRYPTION]
AS
Các_câu_lệnh_của_thủ_tục

Trong đó:

tên_thủ_tục	Tên của thủ tục cần tạo. Tên phải tuân theo qui tắc định danh và không được vượt quá 128 ký tự.
danh_sách_tham_số	Các tham số của thủ tục được khai báo ngay sau tên thủ tục và nếu thủ tục có nhiều tham số thì các khai báo phân cách nhau bởi dấu phẩy. Khai báo của mỗi một tham số tối thiểu phải bao gồm hai phần: 1 • tên tham số được bắt đầu bởi dấu @. 2 • kiểu dữ liệu của tham số **Ví dụ:** @mamonhoc nvarchar(10)
RECOMPILE	Thông thường, thủ tục sẽ được phân tích, tối ưu và dịch sẵn ở lần gọi đầu tiên. Nếu tuỳ chọn WITH RECOMPILE được chỉ định, thủ tục sẽ được dịch lại mỗi khi được gọi.

ENCRYPTION	Thủ tục sẽ được mã hoá nếu tuỳ chọn WITH ENCRYPTION được chỉ định. Nếu thủ tục đã được mã hoá, ta không thể xem được nội dung của thủ tục.
Các_câu_lệnh_của_thủ_tục	Tập hợp các câu lệnh sử dụng trong nội dung thủ tục. Các câu lệnh này có thể đặt trong cặp từ khoá BEGIN...END hoặc có thể không.

Ví dụ 1: Giả sử ta cần thực hiện một chuỗi các thao tác như sau trên cơ sở dữ liệu

1. 1. Bổ sung thêm môn học cơ sở dữ liệu có mã TI-005 và số đơn vị học trình là 5 vào bảng MONHOC
2. 2. Lên danh sách nhập điểm thi môn cơ sở dữ liệu cho các sinh viên học lớp có mã C24102 (tức là bổ sung thêm vào bảng DIEMTHI các bản ghi với cột MAMONHOC nhận giá trị TI-005, cột MASV nhận giá trị lần lượt là mã các sinh viên học lớp có mã C24105 và các cột điểm là NULL).

Nếu thực hiện yêu cầu trên thông qua các câu lệnh SQL như thông thường, ta phải thực thi hai câu lệnh như sau:

```
INSERT INTO MONHOC
VALUES('TI-005','Cơ sở dữ liệu',5)
INSERT INTO DIEMTHI(MAMONHOC,MASV)
SELECT 'TI-005',MASV
FROM SINHVIEN
WHERE MALOP='C24102'
```

Thay vì phải sử dụng hai câu lệnh như trên, ta có thể định nghĩa môt thủ tục lưu trữ với các tham số vào là @mamonhoc, @tenmonhoc, @sodvht và @malop như sau:

```
CREATE PROC sp_LenDanhSachDiem(
@mamonhoc NVARCHAR(10),
@tenmonhoc NVARCHAR(50),
@sodvht SMALLINT,
@malop NVARCHAR(10))
AS
BEGIN
INSERT INTO monhoc
```

```
VALUES(@mamonhoc,@tenmonhoc,@sodvht)
INSERT INTO diemthi(mamonhoc,masv)

SELECT @mamonhoc,masv
FROM sinhvien
WHERE malop=@malop
END
```

Khi thủ tục trên đã được tạo ra, ta có thể thực hiện được hai yêu cầu đặt ra ở trên một cách đơn giản thông qua lời gọi thủ tục:

```
sp_LenDanhSachDiem 'TI-005','Cơ sở dữ liệu',5,'C24102'
```

6.3.3 Lời gọi thủ tục lưu trữ

Như đã thấy ở ví dụ ở trên, khi một thủ tục lưu trữ đã được tạo ra, ta có thể yêu cầu hệ quản trị cơ sở dữ liệu thực thi thủ tục bằng lời gọi thủ tục có dạng:

tên_thủ_tục [danh_sách_các_đối_số]

Số lượng các đối số cũng như thứ tự của chúng phải phù hợp với số lượng và thứ tự của các tham số khi định nghĩa thủ tục.

Trong trường hợp lời gọi thủ tục được thực hiện bên trong một thủ tục khác, bên trong một trigger hay kết hợp với các câu lệnh SQL khác, ta sử dụng cú pháp như sau:

EXECUTE tên_thủ_tục [danh_sách_các_đối_số]

Thứ tự của các đối số được truyền cho thủ tục có thể không cần phải tuân theo thứ tự của các tham số như khi định nghĩa thủ tục nếu tất cả các đối số được viết dưới dạng:

@tên_tham_số = giá_trị

Ví dụ 2: Lời gọi thủ tục ở ví dụ trên có thể viết như sau:

```
sp_LenDanhSachDiem @malop='C24102',
@tenmonhoc='Cơ sở dữ liệu',
@mamonhoc='TI-005',
@sodvht=5
```

6.3.4 Sử dụng biến trong thủ tục

Ngoài những tham số được truyền cho thủ tục, bên trong thủ tục còn có thể sử dụng các biến nhằm lưu giữ các giá trị tính toán được hoặc truy xuất được từ cơ sở dữ liệu. Các biến trong thủ tục được khai báo bằng từ khoá DECLARE theo cú pháp như sau:

DECLARE @tên_biến kiểu_dữ_liệu

Tên biến phải bắt đầu bởi ký tự @ và tuân theo qui tắc về định danh. Ví dụ dưới đây minh hoạ việc sử dụng biến trong thủ tục

Ví dụ 3: Trong định nghĩa của thủ tục dưới đây sử dung các biến chứa các giá trị truy xuất được từ cơ sở dữ liệu.

```
CREATE PROCEDURE sp_Vidu(
@malop1 NVARCHAR(10),
@malop2 NVARCHAR(10))
AS
DECLARE @tenlop1 NVARCHAR(30)
DECLARE @namnhaphoc1 INT
DECLARE @tenlop2 NVARCHAR(30)
DECLARE @namnhaphoc2 INT
SELECT @tenlop1=tenlop,
@namnhaphoc1=namnhaphoc
FROM lop WHERE malop=@malop1
SELECT @tenlop2=tenlop,
@namnhaphoc2=namnhaphoc
FROM lop WHERE malop=@malop2
PRINT @tenlop1+' nhap hoc nam '+str(@namnhaphoc1)
print @tenlop2+' nhap hoc nam '+str(@namnhaphoc2)
IF @namnhaphoc1=@namnhaphoc2
PRINT 'Hai lớp nhập học cùng năm'
ELSE
PRINT 'Hai lớp nhập học khác năm'
```

6.3.5 Giá trị trả về của tham số trong thủ tục lưu trữ

Trong các ví dụ trước, nếu đối số truyền cho thủ tục khi có lời gọi đến thủ tục là biến, những thay đổi giá trị của biền trong thủ tục sẽ không được giữ lại khi kết thúc quá trình thực hiện thủ tục.

Ví dụ 4: Xét câu lệnh sau đây

```
CREATE PROCEDURE sp_Conghaiso(@a INT,@b INT, @c INT)
AS
SELECT @c=@a+@b
```

Nếu sau khi đã tạo thủ tục với câu lệnh trên, ta thực thi một tập các câu lệnh như sau:

```
DECLARE @tong INT
SELECT @tong=0
EXECUTE sp_Conghaiso 100,200,@tong
SELECT @tong
```

Câu lệnh "SELECT @tong" cuối cùng trong loạt các câu lệnh trên sẽ cho kết quả là: 0

Trong trường hợp cần phải giữ lại giá trị của đối số sau khi kết thúc thủ tục, ta phải khai báo tham số của thủ tục theo cú pháp như sau:

@tên_tham_số kiểu_dữ_liệu OUTPUT

hoặc:

@tên_tham_số kiểu_dữ_liệu OUT

và trong lời gọi thủ tục, sau đối số được truyền cho thủ tục, ta cũng phải chỉ định thêm từ khoá OUTPUT (hoặc OUT)

Ví dụ 5: Ta định nghĩa lại thủ tục ở ví dụ 6.4 như sau:

```
CREATE PROCEDURE sp_Conghaiso(
@a INT,
@b INT,
@c INT OUTPUT)
AS
SELECT @c=@a+@b
```

và thực hiện lời gọi thủ tục trong một tập các câu lệnh như sau:

```
DECLARE @tong INT
SELECT @tong=0
EXECUTE sp_Conghaiso 100,200,@tong OUTPUT
SELECT @tong
```

thì câu lệnh "SELECT @tong" sẽ cho kết quả là: 300

6.3.6 Tham số với giá trị mặc định

Các tham số được khai báo trong thủ tục có thể nhận các giá trị mặc định. Giá trị mặc định sẽ được gán cho tham số trong trường hợp không truyền đối số cho tham số khi có lời gọi đến thủ tục.

Tham số với giá trị mặc định được khai báo theo cú pháp như sau:

@tên_tham_số kiểu_dữ_liệu = giá_trị_mặc_định

Ví dụ 6: Trong câu lệnh dưới đây:

```
CREATE PROC sp_TestDefault(
@tenlop NVARCHAR(30)=NULL,
@noisinh NVARCHAR(100)='Huế')
AS
BEGIN
IF @tenlop IS NULL
SELECT hodem,ten
FROM sinhvien INNER JOIN lop
ON sinhvien.malop=lop.malop
WHERE noisinh=@noisinh
ELSE
SELECT hodem,ten
FROM sinhvien INNER JOIN lop
ON sinhvien.malop=lop.malop
WHERE noisinh=@noisinh AND
tenlop=@tenlop
END
```

thủ tục sp_TestDefault được định nghĩa với tham số @tenlop có giá trị mặc định là NULL và tham số @noisinh có giá trị mặc định là Huế. Với thủ tục được định nghĩa như trên, ta có thể thực hiện các lời gọi với các mục đích khác nhau như sau:

1 • Cho biết họ tên của các sinh viên sinh tại Huế:

```
sp_testdefault
```

1 • Cho biết họ tên của các sinh viên lớp Tin K24 sinh tại Huế:

```
sp_testdefault @tenlop='Tin K24'
```

1 • Cho biết họ tên của các sinh viên sinh tại Nghệ An:

```
sp_testDefault @noisinh=N'Nghệ An'
```

1 • Cho biết họ tên của các sinh viên lớp Tin K26 sinh tại Đà Nẵng:

```
sp_testdefault @tenlop='Tin K26',@noisinh='Đà Nẵng'
```

6.3.7 Sửa đổi thủ tục

Khi một thủ tục đã được tạo ra, ta có thể tiến hành định nghĩa lại thủ tục đó bằng câu lệnh ALTER PROCEDURE có cú pháp như sau:

```
ALTER PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]
[WITH RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION]
AS
Các_câu_lệnh_Của_thủ_tục
```

Câu lệnh này sử dụng tương tự như câu lệnh CREATE PROCEDURE. Việc sửa đổi lại một thủ tục đã có không làm thay đổi đến các quyền đã cấp phát trên thủ tục cũng như không tác động đến các thủ tục khác hay trigger phụ thuộc vào thủ tục này.

6.3.8 Xoá thủ tục

Để xoá một thủ tục đã có, ta sử dụng câu lệnh DROP PROCEDURE với cú pháp như sau:

```
DROP PROCEDURE tên_thủ_tục
```

Khi xoá một thủ tục, tất cả các quyền đã cấp cho người sử dụng trên thủ tục đó cũng đồng thời bị xoá bỏ. Do đó, nếu tạo lại thủ tục, ta phải tiến hành cấp phát lại các quyền trên thủ tục đó.

6.4 Trigger

Ta đã biết các ràng buộc được sử dụng để đảm bảo tính toàn vẹn dữ liệu trong cơ sở dữ liệu. Một đối tượng khác cũng thường được sử dụng trong các cơ sở dữ liệu cũng với mục đích này là các trigger. Cũng tương tự như thủ tục lưu trữ, một trigger là một đối tượng chứa một tập các câu lệnh SQL và tập các câu lệnh này sẽ được thực thi khi trigger được gọi. Điểm khác biệt giữa thủ tục lưu trữ và trigger là: các thủ tục lưu trữ được thực thi khi người sử dụng có lời gọi đến chúng còn các trigger lại được “gọi” tự động khi xảy ra những giao tác làm thay đổi dữ liệu trong các bảng.

Mỗi một trigger được tạo ra và gắn liền với một bảng nào đó trong cơ sở dữ liệu. Khi dữ liệu trong bảng bị thay đổi (tức là khi bảng chịu tác động của các câu lệnh INSERT, UPDATE hay DELETE) thì trigger sẽ được tự đông kích hoạt.

Sử dụng trigger một cách hợp lý trong cơ sở dữ liệu sẽ có tác động rất lớn trong việc tăng hiệu năng của cơ sở dữ liệu. Các trigger thực sự hữu dụng với những khả năng sau:

1 • Một trigger có thể nhận biết, ngăn chặn và huỷ bỏ được những thao tác làm thay đổi trái phép dữ liệu trong cơ sở dữ liệu.

2 • Các thao tác trên dữ liệu (xoá, cập nhật và bổ sung) có thể được trigger phát hiện ra và tự động thực hiện một loạt các thao tác khác trên cơ sở dữ liệu nhằm đảm bảo tính hợp lệ của dữ liệu.

1 • Thông qua trigger, ta có thể tạo và kiểm tra được những mối quan hệ phức tạp hơn giữa các bảng trong cơ sở dữ liệu mà bản thân các ràng buộc không thể thực hiện được.

6.4.1 Định nghĩa trigger

Một trigger là một đối tượng gắn liền với một bảng và được tự động kích hoạt khi xảy ra những giao tác làm thay đổi dữ liệu trong bảng. Định nghĩa một trigger bao gồm các yếu tố sau:

1 • Trigger sẽ được áp dụng đối với bảng nào?

2 • Trigger được kích hoạt khi câu lệnh nào được thực thi trên bảng: INSERT, UPDATE, DELETE?

3 • Trigger sẽ làm gì khi được kích hoạt?

Câu lệnh CREATE TRIGGER được sử dụng để đinh nghĩa trigger và có cú pháp như sau:

```
CREATE TRIGGER tên_trigger
ON tên_bảng
FOR {[INSERT][,][UPDATE][,][DELETE]}
AS
[IF UPDATE(tên_cột)
[AND UPDATE(tên_cột)|OR UPDATE(tên_cột)]
...]
các_câu_lệnh_của_trigger
```

Ví dụ 1: Ta định nghĩa các bảng như sau:

Bảng MATHANG lưu trữ dữ liệu về các mặt hàng:

```
CREATE TABLE mathang
(
mahang NVARCHAR(5) PRIMARY KEY, /*mã hàng*/
tenhang NVARCHAR(50) NOT NULL, /*tên hàng*/
soluong INT, /*số lượng hàng hiện có*/
)
```

Bảng NHATKYBANHANG lưu trữ thông tin về các lần bán hàng

```
CREATE TABLE nhatkybanhang
(
stt INT IDENTITY PRIMARY KEY,
ngay DATETIME, /*ngày bán hàng*/
nguoimua NVARCHAR(30), /*tên người mua hàng*/
mahang NVARCHAR(5) /*mã mặt hàng được bán*/
FOREIGN KEY REFERENCES mathang(mahang),
soluong INT, /*số lượng hàng bán được*/
giaban MONEY /*giá bán*/
)
```

Câu lệnh dưới đây định nghĩa trigger trg_nhatkybanhang_insert. Trigger này có chức năng tự động giảm số lượng hàng hiện có khi một mặt hàng nào đó được bán (tức là khi câu lệnh INSERT được thực thi trên bảng NHATKYBANHANG).

```
CREATE TRIGGER trg_nhatkybanhang_insert
ON nhatkybanhang
FOR INSERT
AS
UPDATE mathang
SET mathang.soluong=mathang.soluong-inserted.soluong
FROM mathang INNER JOIN inserted
ON mathang.mahang=inserted.mahang
```

Với trigger vừa tạo ở trên, nếu dữ liệu trong bảng MATHANG là:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	30
H2	Kem đánh răng	45

thì sau khi ta thực hiện câu lênh:

```
INSERT INTO nhatkybanhang
(ngay,nguoimua,mahang,soluong,giaban)
VALUES('5/5/2004','Tran Ngoc Thanh','H1',10,5200)
```

dữ liệu trong bảng MATHANG sẽ như sau:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	20
H2	Kem đánh răng	45

Trong câu lệnh CREATE TRIGGER ở ví dụ trên, sau mệnh đề ON là tên của bảng mà trigger cần tạo sẽ tác động đến. Mệnh đề tiếp theo chỉ định câu lệnh sẽ kích hoạt trigger (FOR INSERT). Ngoài INSERT, ta còn có thể chỉ định UPDATE hoặc DELETE cho mệnh đề này, hoặc có thể kết hợp chúng lại với nhau. Phần thân của trigger nằm sau từ khoá AS bao gồm các câu lệnh mà trigger sẽ thực thi khi được kích hoạt.

Chuẩn SQL định nghĩa hai bảng logic INSERTED và DELETED để sử dụng trong các trigger. Cấu trúc của hai bảng này tương tự như cấu trúc của bảng mà trigger tác động. Dữ liệu trong hai bảng này tuỳ thuộc vào câu lệnh tác động lên bảng làm kích hoạt trigger; cụ thể trong các trường hợp sau:

1 • Khi câu lệnh DELETE được thực thi trên bảng, các dòng dữ liệu bị xoá sẽ được sao chép vào trong bảng DELETED. Bảng INSERTED trong trường hợp này không có dữ liệu.

2 • Dữ liệu trong bảng INSERTED sẽ là dòng dữ liệu được bổ sung vào bảng gây nên sự kích hoạt đối với trigger bằng câu lệnh INSERT. Bảng DELETED trong trường hợp này không có dữ liệu.

3 • Khi câu lệnh UPDATE được thực thi trên bảng, các dòng dữ liệu cũ chịu sự tác động của câu lệnh sẽ được sao chép vào bảng DELETED, còn trong bảng INSERTED sẽ là các dòng sau khi đã được cập nhật.

6.4.2 Sử dụng mệnh đề IF UPDATE trong trigger

Thay vì chỉ định một trigger được kích hoạt trên một bảng, ta có thể chỉ định trigger được kích hoạt và thực hiện những thao tác cụ thể khi việc thay đổi dữ liệu chỉ liên quan đến một số cột nhất định nào đó của cột. Trong trường hợp này, ta sử dụng mệnh đề IF UPDATE trong trigger. IF UPDATE không sử dụng được đối với câu lệnh DELETE.

Ví dụ 2: Xét lại ví dụ với hai bảng MATHANG và NHATKYBANHANG, trigger dưới đây được kích hoạt khi ta tiến hành cập nhật cột SOLUONG cho một bản ghi của bảng NHATKYBANHANG (lưu ý là chỉ cập nhật đúng một bản ghi)

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
```

```
FOR UPDATE
AS
IF UPDATE(soluong)
UPDATE mathang
SET mathang.soluong = mathang.soluong –
(inserted.soluong-deleted.soluong)
FROM (deleted INNER JOIN inserted ON
    deleted.stt = inserted.stt) INNER JOIN mathang
ON mathang.mahang = deleted.mahang
```

Với trigger ở ví dụ trên, câu lệnh:

```
UPDATE nhatkybanhang
SET soluong=soluong+20
WHERE stt=1
```

sẽ kích hoạt trigger ứng với mệnh đề IF UPDATE (soluong) và câu lệnh UPDATE trong trigger sẽ được thực thi. Tuy nhiên câu lệnh:

```
UPDATE nhatkybanhang
SET nguoimua='Mai Hữu Toàn'
WHERE stt=3
```

lại không kích hoạt trigger này.

Mệnh đề IF UPDATE có thể xuất hiện nhiều lần trong phần thân của trigger. Khi đó, mệnh đề IF UPDATE nào đúng thì phần câu lệnh của mệnh đề đó sẽ được thực thi khi trigger được kích hoạt.

Ví dụ 3: Giả sử ta định nghĩa bảng R như sau:

```
CREATE TABLE R
(
A INT,
B INT,
C INT
)
```

và trigger trg_R_update cho bảng R:

```
CREATE TRIGGER trg_R_test
ON R
FOR UPDATE
AS
```

```
IF UPDATE(A)
Print 'A updated'
IF UPDATE(C)
Print 'C updated'
```

Câu lệnh:

UPDATE R SET A=100 WHERE A=1

sẽ kích hoạt trigger và cho kết quả là:

A updated

và câu lệnh:

```
UPDATE R SET C=100 WHERE C=2
```

cũng kích hoạt trigger và cho kết quả là:

C updated

còn câu lệnh:

```
UPDATE R SET B=100 WHERE B=3
```

hiển nhiên sẽ không kích hoạt trigger

6.4.3 ROLLBACK TRANSACTION và trigger

Một trigger có khả năng nhận biết được sự thay đổi về mặt dữ liệu trên bảng dữ liệu, từ đó có thể phát hiện và huỷ bỏ những thao tác không đảm bảo tính toàn vẹn dữ liệu. Trong một trigger, để huỷ bỏ tác dụng của câu lệnh làm kích hoạt trigger, ta sử dụng câu lệnh(1):

ROLLBACK TRANSACTION

Ví dụ 4: Nếu trên bảng MATHANG, ta tạo một trigger như sau:

```
CREATE TRIGGER trg_mathang_delete
ON mathang
FOR DELETE
AS
ROLLBACK TRANSACTION
```

Thì câu lệnh DELETE sẽ không thể có tác dụng đối với bảng MATHANG. Hay nói cách khác, ta không thể xoá được dữ liệu trong bảng.

Ví dụ 5: Trigger dưới đây được kích hoạt khi câu lệnh INSERT được sử dụng để bổ sung một bản ghi mới cho bảng NHATKYBANHANG. Trong trigger này kiểm tra điều kiện hợp lệ của dữ liệu là số lượng hàng bán ra phải nhỏ hơn hoặc bằng số lượng hàng hiện có. Nếu điều kiện này không thoả mãn thì huỷ bỏ thao tác bổ sung dữ liệu.

```
CREATE TRIGGER trg_nhatkybanhang_insert
ON NHATKYBANHANG
FOR INSERT
AS
DECLARE @sl_co int /* Số lượng hàng hiện có */
DECLARE @sl_ban int /* Số lượng hàng được bán */
DECLARE @mahang nvarchar(5) /* Mã hàng được bán */
SELECT @mahang=mahang,@sl_ban=soluong
FROM inserted
SELECT @sl_co = soluong
FROM mathang where mahang=@mahang
/*Nếu số lượng hàng hiện có nhỏ hơn số lượng bán
thì huỷ bỏ thao tác bổ sung dữ liệu */
```

(1) Cách sử dụng và ý nghĩa của câu lệnh ROLLBACK TRANSACTION để bỏ qua thao tác SQL vừa thực thi.

```
IF @sl_co<@sl_ban
ROLLBACK TRANSACTION
/* Nếu dữ liệu hợp lệ
thì giảm số lượng hàng hiện có */
ELSE
UPDATE mathang
SET soluong=soluong-@sl_ban
WHERE mahang=@mahang
```

6.4.4 Sử dụng trigger trong trường hợp câu lệnh INSERT, UPDATE và DELETE có tác động đến nhiều dòng dữ liệu

Trong các ví dụ trước, các trigger chỉ thực sự hoạt động đúng mục đích khi các câu lệnh kích hoạt trigger chỉ có tác dụng đối với đúng một dòng dữ liêu. Ta có thể nhận thấy là câu lệnh UPDATE và DELETE thường có tác dụng trên nhiều dòng, câu lệnh INSERT mặc dù ít rơi vào trường hợp này nhưng không phải là không gặp; đó là khi ta sử dụng câu lệnh có dạng INSERT INTO ... SELECT ... Vậy làm thế nào để trigger hoạt động đúng trong trường hợp những câu lệnh có tác động lên nhiều dòng dữ liệu?

Có hai giải pháp có thể sử dụng đối với vấn đề này:

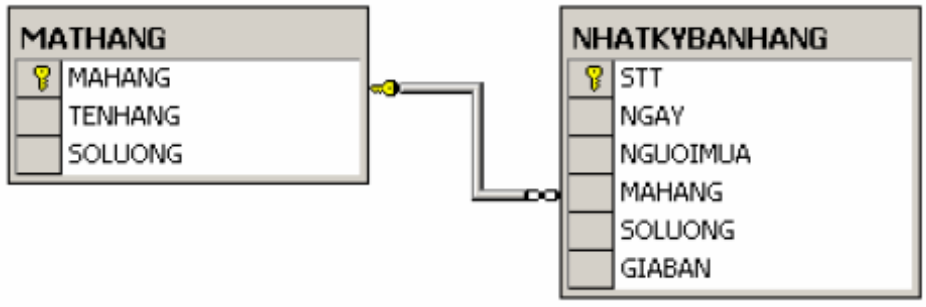
1 • Sử dụng truy vấn con.

2 • Sử dụng biến con trỏ.

6.4.4.1 Sử dụng truy vấn con

Ta hình dung vấn đề này và cách khắc phục qua ví dụ dưới đây:

Ví dụ 6: Ta xét lại trường hợp của hai bảng MATHANG và NHATKYBANHANG:



MAHANG	TENHANG	SOLUONG
H1	Xà phòng	30
H2	Kem đánh răng	45

STT	NGAY	NGUOIMUA	MAHANG	SOLUONG	GIABAN
1	1-1-2004	Ha	H1	10	10000.0000
2	2-2-2004	Phong	H2	20	5000.0000
3	3-3-2004	Thuy	H2	30	6000.0000

Hình 6.5:Sử dụng truy vấn con

Trigger dưới đây cập nhật lại số lượng hàng của bảng MATHANG khi câu lệnh UPDATE được sử dụng để cập nhật cột SOLUONG của bảng NHATKYBANHANG.

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
UPDATE mathang
SET mathang.soluong = mathang.soluong –
(inserted.soluong-deleted.soluong)
FROM (deleted INNER JOIN inserted ON
      deleted.stt = inserted.stt) INNER JOIN mathang
ON mathang.mahang = deleted.mahang
```

Với trigger được định nghĩa như trên, nếu thực hiện câu lệnh:

```
UPDATE nhatkybanhang
SET soluong = soluong + 10
WHERE stt = 1
```

thì dữ liệu trong hai bảng MATHANG và NHATKYBANHANG sẽ là:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	20
H2	Kem đánh răng	45

STT	NGAY	NGUOIMUA	MAHANG	SOLUONG	GIABAN
1	1-1-2004	Ha	H1	20	10000.0000
2	2-2-2004	Phong	H2	20	5000.0000
3	3-3-2004	Thuy	H2	30	6000.0000
4	4-4-2004	Dung	H1	40	9000.0000

Bảng MATHANG Bảng NHATKYBANHANG

Hình 6.6:xuất dữ liệu

tức là số lượng của mặt hàng có mã H1 đã được giảm đi 10. Nhưng nếu thực hiện tiếp câu lệnh:

```
UPDATE nhatkybanhang
SET soluong=soluong + 5
WHERE mahang='H2'
```

dữ liệu trong hai bảng sau khi câu lệnh thực hiện xong sẽ như sau:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	20
H2	Kem đánh răng	40

STT	NGAY	NGUOIMUA	MAHANG	SOLUONG	GIABAN
1	1-1-2004	Ha	H1	20	10000.0000
2	2-2-2004	Phong	H2	25	5000.0000
3	3-3-2004	Thuy	H2	35	6000.0000

Bảng MATHANG Bảng NHATKYBANHANG

Hình 6.7 :minh hoạ dữ liệu

tức là số lượng của mặt hàng có mã H2 còn lại 40 (giảm đi 5) trong khi đúng ra phải là 35 (tức là phải giảm 10). Như vậy, trigger ở trên không hoạt động đúng trong trường hợp này.

Để khắc phục lỗi gặp phải như trên, ta định nghĩa lại trigger như sau:

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
UPDATE mathang
SET mathang.soluong = mathang.soluong -
(SELECT SUM(inserted.soluong-deleted.soluong)
FROM inserted INNER JOIN deleted
        ON inserted.stt=deleted.stt
```

```
        WHERE inserted.mahang = mathang.mahang)
        WHERE mathang.mahang IN (SELECT mahang
        FROM inserted)
```

hoặc:

```
        CREATE TRIGGER trg_nhatkybanhang_update_soluong
        ON nhatkybanhang
        FOR UPDATE
        AS
        IF UPDATE(soluong)
        /* Nếu số lượng dòng được cập nhật bằng 1 */
        IF @@ROWCOUNT = 1
        BEGIN
        UPDATE mathang
        SET mathang.soluong = mathang.soluong –
        (inserted.soluong-deleted.soluong)
        FROM (deleted INNER JOIN inserted ON
                deleted.stt = inserted.stt) INNER JOIN mathang
        ON mathang.mahang = deleted.mahang
        END
        ELSE
        BEGIN
        UPDATE mathang
        SET mathang.soluong = mathang.soluong -
        (SELECT SUM(inserted.soluong-deleted.soluong)
        FROM inserted INNER JOIN deleted
                ON inserted.stt=deleted.stt
        WHERE inserted.mahang = mathang.mahang)
        WHERE mathang.mahang IN (SELECT mahang
        FROM inserted)
END
```

6.4.4.2 Sử dụng biến con trỏ

Một cách khác để khắc phục lỗi xảy ra như trong ví dụ 5.17 là sử dụng con trỏ để duyệt qua các dòng dữ liệu và kiểm tra trên từng dòng. Tuy nhiên, sử dụng biến con trỏ trong trigger là giải pháp nên chọn trong trường hợp thực sự cần thiết.

Một biến con trỏ được sử dụng để duyệt qua các dòng dữ liệu trong kết quả của một truy vấn và được khai báo theo cú pháp như sau:

DECLARE tên_con_trỏ CURSOR

FOR câu_lệnh_SELECT

Trong đó câu lệnh SELECT phải có kết quả dưới dạng bảng. Tức là trong câu lệnh không sử dụng mệnh đề COMPUTE và INTO.

Để mở một biến con trỏ ta sử dụng câu lệnh:

OPEN tên_con_trỏ

Để sử dụng biến con trỏ duyệt qua các dòng dữ liệu của truy vấn, ta sử dụng câu lệnh FETCH. Giá trị của biến trạng thái @@FETCH_STATUS bằng không nếu chưa duyệt hết các dòng trong kết quả truy vấn.

Câu lệnh FETCH có cú pháp như sau:

FETCH [[NEXT|PRIOR|FIST|LAST] FROM] tên_con_trỏ

[INTO danh_sách_biến]

Trong đó các biến trong danh sách biến được sử dụng để chứa các giá trị của các trường ứng với dòng dữ liệu mà con trỏ trỏ đến. Số lượng các biến phải bằng với số lượng các cột của kết quả truy vấn trong câu lệnh DECLARE CURSOR.

Ví dụ 7: Tập các câu lệnh trong ví dụ dưới đây minh hoạ cách sử dụng biến con trỏ để duyệt qua các dòng trong kết quả của câu lệnh SELECT

```
DECLARE contro CURSOR
FOR SELECT mahang,tenhang,soluong FROM mathang
OPEN contro
DECLARE @mahang NVARCHAR(10)
DECLARE @tenhang NVARCHAR(10)
DECLARE @soluong INT
/*Bắt đầu duyệt qua các dòng trong kết quả truy vấn*/
FETCH NEXT FROM contro
INTO @mahang,@tenhang,@soluong
WHILE @@FETCH_STATUS=0
BEGIN
PRINT 'Ma hang:'+@mahang
PRINT 'Ten hang:'+@tenhang
PRINT 'So luong:'+STR(@soluong)
FETCH NEXT FROM contro
```

```
INTO @mahang,@tenhang,@soluong
END
/*Đóng con trỏ và giải phóng vùng nhớ*/
CLOSE contro
DEALLOCATE contro
```

Ví dụ 8: Trigger dưới đây là một cách giải quyết khác của trường hợp được đề cập ở ví dụ 6

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
BEGIN
        DECLARE @mahang NVARCHAR(10)
DECLARE @soluong INT
DECLARE contro CURSOR FOR
SELECT inserted.mahang,
inserted.soluong-deleted.soluong AS soluong
FROM inserted INNER JOIN deleted
ON inserted.stt=deleted.stt
OPEN contro
FETCH NEXT FROM contro INTO @mahang,@soluong
WHILE @@FETCH_STATUS=0
BEGIN
UPDATE mathang SET soluong=soluong-@soluong
WHERE mahang=@mahang
FETCH NEXT FROM contro INTO @mahang,@soluong
END
CLOSE contro
DEALLOCATE contro
END
END
```

BÀI TẬP THỰC HÀNH

1 6.1 Tạo thủ tục lưu trữ để thông qua thủ tục này có thể bổ sung thêm một bản ghi mới cho bảng MATHANG (thủ tục phải thực hiện kiểm tra tính hợp lệ của dữ liệu cần bổ sung: không trùng khoá chính và đảm bảo toàn vẹn tham chiếu)

2 6.2 Tạo thủ tục lưu trữ có chức năng thống kê tổng số lượng hàng bán được của một mặt hàng có mã bất kỳ (mã mặt hàng cần thống kê là tham số của thủ tục).

3 6.3 Viết trigger cho bảng CHITIETDATHANG theo yêu cầu sau:

0 • Khi một bản ghi mới được bổ sung vào bảng này thì giảm số lượng hàng hiện có nếu số lượng hàng hiện có lớn hơn hoặc bằng số lượng hàng được bán ra. Ngược lại thì huỷ bỏ thao tác bổ sung.

1 • Khi cập nhật lại số lượng hàng được bán, kiểm tra số lượng hàng được cập nhật lại có phù hợp hay không (số lượng hàng bán ra không được vượt quá số lượng hàng hiện có và không được nhỏ hơn 1). Nếu dữ liệu hợp lệ thì giảm (hoặc tăng) số lượng hàng hiện có trong công ty, ngược lại thì huỷ bỏ thao tác cập nhật.

4 6.4 Viết trigger cho bảng CHITIETDATHANG để sao cho chỉ chấp nhận giá hàng bán ra phải nhỏ hơn hoặc bằng giá gốc (giá của mặt hàng trong bảng MATHANG)

Lời giải:

6.1 CREATE PROCEDURE sp_insert_mathang(

@mahang NVARCHAR(10),
@tenhang NVARCHAR(50),
@macongty NVARCHAR(10) = NULL,
@maloaihang INT = NULL,
@soluong INT = 0,
@donvitinh NVARCHAR(20) = NULL,
@giahang money = 0)
AS
IF NOT EXISTS(SELECT mahang FROM mathang
WHERE mahang=@mahang)
IF (@macongty IS NULL OR EXISTS(SELECT macongty
FROM nhacungcap
WHERE macongty=@macongty))
AND
(@maloaihang IS NULL OR
EXISTS(SELECT maloaihang FROM loaihang
WHERE maloaihang=@maloaihang))
INSERT INTO mathang
VALUES(@mahang,@tenhang,
@macongty,@maloaihang,
@soluong,@donvitinh,@giahang)

6.2 CREATE PROCEDURE sp_thongkebanhang(@mahang NVARCHAR(10))

AS
SELECT mathang.mahang,tenhang,
SUM(chitietdathang.soluong) AS tongsoluong
FROM mathang LEFT OUTER JOIN chitietdathang
ON mathang.mahang=chitietdathang.mahang
WHERE mathang.mahang=@mahang
GROUP BY mathang.mahang,tenhang

6.3 CREATE TRIGGER trg_chitietdathang_insert

```
ON chitietdathang
FOR INSERT
AS
BEGIN
DECLARE @mahang NVARCHAR(100)
DECLARE @soluongban INT
DECLARE @soluongcon INT
SELECT @mahang=mahang,@soluongban=soluong
FROM inserted
SELECT @soluongcon=soluong FROM mathang
WHERE mahang=@mahang
IF @soluongcon>=@soluongban
UPDATE mathang SET soluong=soluong-@soluongban
WHERE mahang=@mahang
ELSE
ROLLBACK TRANSACTION
END
CREATE TRIGGER trg_chitietdathang_update_soluong
ON chitietdathang
FOR UPDATE
AS
IF UPDATE(soluong)
BEGIN
IF EXISTS(SELECT sohoadon FROM inserted WHERE soluong<0)
ROLLBACK TRANSACTION
ELSE
BEGIN
UPDATE mathang
SET soluong=soluong-
(SELECT SUM(inserted.soluong-deleted.soluong)
FROM inserted INNER JOIN deleted
```

```
ON inserted.sohoadon=deleted.sohoadon AND

inserted.mahang=deleted.mahang
WHERE inserted.mahang=mathang.mahang
GROUP BY inserted.mahang)
WHERE mahang IN (SELECT DISTINCT mahang
FROM inserted)
IF EXISTS(SELECT mahang FROM mathang
WHERE soluong<0)
ROLLBACK TRANSACTION
END
END
```

6.4 CREATE TRIGGER trg_chitietdathang_giaban

```
ON chitietdathang
FOR INSERT,UPDATE
AS
IF UPDATE(giaban)
IF EXISTS(SELECT inserted.mahang
FROM mathang INNER JOIN inserted
ON mathang.mahang=inserted.mahang
WHERE mathang.giahang>inserted.giaban)
ROLLBACK TRANSACTION
```

BÀI 7
KẾT NỐI ÚNG DỤNG VỚI CƠ SỞ DỮ LIỆU

MÃ BÀI: ITPRG3-17.7

Giới thiệu:

Việc xây dựng một ứng dụng có thành công hay không phụ thuộc vào cơ sở dữ liệu của ứng dụng, còn việc sử dụng một ngôn ngữ lập trình nào đó là sở thích của bạn và yêu cầu của đối tác, của hệ thống,… Trong bài này chúng ta sẽ tìm hiểu cách thức kết nối đế cơ sở dữ liệu Access thông qua ngôn ngữ Visual Basic và .NET.

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

- Tạo và sử dụng được ODBC
- Sử dụng được ADO
- Sử dụng được Data Environment
- Sử dụng được OLE_DB
- Lập trình được trên các đối tượng RecordSet

Nội dung:

7.1 ODBC, JDBC

7.2 ADO

7.3 Data Environment

7.4 OLE_DB

7.5 Lập trình trên các đối tượng Record Set

7.1 ODBC, JDBC

7.1.1 Giới thiệu chung.

Cùng với sự phát triển của CNTT, nhu cầu xây dựng, lưu trữ các CSDL lớn và nhu cầu về chia sẻ dữ liệu ngày càng gia tăng. Điều đó dẫn đến sự ra đời của các Hệ quản trị cơ sở dữ liệu khác nhau (DBMS - Database Management System: Là phần mềm thực thi các lệnh để truy xuất dữ liệu trong Data Storage. Thường thì một DBMS bao gồm một SQL parser - module phân tích cú pháp các lệnh SQL, một Optimizer - module đánh giá, tối ưu các câu lệnh, một module thực thi, và một vài thành phần khác mà nó cung cấp các phục vụ quản lý dữ liệu như security, transactions, recovery. Đôi khi DBMS được nhắc tới như một server, khi nói đến Microsoft SQL

Server, Oracle, hoặc DB2 chúng ta thường nghĩ ngay tới DBMS. Trong nội dung trình bày ở đây chúng ta sẽ xem các DBMS bao gồm cả các Desktop Database Product như Foxpro, Access và Paradox). Mỗi DBMS khác nhau được triển khai tuỳ thuộc vào các nhu cầu riêng của từng bộ phận.

Trước những nhu cầu ngày càng cao của người sử dụng, họ cần có những ứng dụng mà nó có thể được phát triển và sửa đổi nhanh chóng và các ứng dụng đó phải phải khai thác được các khả năng đặc biệt của mỗi DBMS và cuối cùng là nó phải đơn giản dễ sử dụng. Và như vậy các nhà phát triển ứng dụng đã gặp phải nhiều khó khăn vì mỗi DBMS sử dụng một version SQL của riêng họ và do đó nó có một API riêng rất phức tạp. Điều này đã tạo ra một nhu cầu cần thiết phải có một middleware layer đảm nhận chức năng của một translator, nó sẽ chuyển đổi các lệnh SQL chuẩn thành các câu lệnh SQL sử dụng bởi DBMSs đặc biệt. Microsoft's ODBC và Sun Microsystem's JDBC chính là các translator như vậy.

7.1.2 ODBC - open database connectivity

a. ODBC là gì ?

ODBC (Open Database Connectivity) là một standard Database API. Ở ĐÂY CẦN PHÂN BIỆT 2 KHÁI NIỆM "standard API" và "native API": Standard API là một API chuẩn, điều đó có nghĩa là nó được sự chấp thuận và hỗ trợ từ các nhà cung cấp DBMSs và các nhà phát triển ứng dụng. Còn các native API là các API do từng nhà cung cấp DBMS đưa ra để truy xuất và khai thác hiệu quả các khả năng đặc trưng của DBMS do họ cung cấp (ví dụ Oracle OCI, Sybase DB-Library là các native API).

Các bạn đã làm quen với Windows API khi lập trình trong môi trường Windows. Trong Visual Basic có thể bạn sẽ ít dùng trực tiếp các hàm, thủ tục do Windows API cung cấp nhưng trong Visual C++ thì gần như liên tục sử dụng các hàm, thủ tục này để ứng dụng của bạn giao tiếp được với hệ điều hành Windows và để tạo ra các giao diện của người sử dụng. Cho ví dụ, trong vấn đề hiển thị video để đưa ra một cửa sổ trên màn hình thì bạn không cần phải biết đến loại màn hình nào đang được sử dụng, Windows API cung cấp cho bạn khả năng hiển thị video độc lập với các thiết bị. Cũng tương tự như vậy, ODBC là một Database API chuẩn, nó cung cấp cho các bạn khả năng truy nhập đến các CSDL một cách độc lập với các DBMS, qua ODBC các bạn có thể truy xuất được tới các CSDL trên các DBMS khác nhau.

Các đặc điểm của ODBC:

+ ODBC là một giao diện lập trình sử dụng SQL: ODBC sẽ sử dụng các lệnh SQL để truy xuất các CSDL.

+ ODBC tách các nhà phát triển ứng dụng khỏi sự phức tạp của việc kết nối tới một nguồn dữ liệu: Mục tiêu chính được đề cho ODBC là nó phải dễ dàng cho người lập trình ứng dụng có thể tạo ra các kết nối của người sử dụng cuối tới nguồn dữ liệu thích hợp mà không phải trở thành một chuyên gia về mạng.

+ Kiến trúc của ODBC cho phép nhiều ứng dụng truy xuất nhiều nguồn dữ liệu.

+ ODBC cung cấp một mô hình lập trình "thích ứng" (adaptive): ODBC cung cấp các chức năng mà nó có thể được sử dụng với tất cả các DBMS trong khi vẫn cho phép một ứng dụng khai thác các khả năng riêng của mỗi DBMS. Nó cung cấp các interrogation function mà một ứng dụng có thể chủ động sử dụng để xác định các khả năng của một DBMS. Các interrogation function cho phép một ứng dụng hỏi một driver về một vài chức năng đặc biệt có được cung cấp trong một DBMS nào đó hay không.

b. Kiến trúc của ODBC.

*** ODBC được xây dựng trên mô hình kiến trúc Client/Server.**

Trong kiến trúc Client/Server bao gồm một một client, một server, và một data protocol mà nó cho phép client và server giao tiếp với nhau. Mô hình này rất lý tưởng cho một Traditional Relational DBMS, trong đó một mạng vật lý kết nối client PC tới DBMS ở trên một máy khác. ODBC được thiết kế để sử dụng với các hệ thống nằm trong mô hình kiến trúc client/server, đáp ứng được các yêu cầu cần thiết cho các Traditional Relational DBMS như:

+ Cung cấp một standard API.

+ Khai thác tất cả các chức năng của bất cứ một DBMS nào.

+ Cung cấp một sự thực thi tương đương với native API của bất cứ một DBMS nào.

Kiến trúc của ODBC đặt trên nền tảng mô hình kiến trúc client/server và sự đảm nhận cho bất cứ một giao diện lập trình nào cũng có thể phát và thu trên giao thức truyền dữ liệu của bất cứ một SQL DBMS, sẽ hoạt động và thực thi như native API cho DBMS đó.

ODBC không chỉ giới hạn với các client/server DBMS, nó cũng làm việc với các desktop database và các file-oriented store như bảng tính và text.

*** Các thành phần cơ bản trong kiến trúc của ODBC.**

File - Oriented Data Store Client/Server DBMS

Applications: Các ứng dụng đảm nhận việc tương tác với người sử dụng qua user interface và gọi các ODBC function để đưa ra các câu lệnh SQL và nhận các kết quả trả về.

Driver Manager: Như tên gọi của nó, nhiệm vụ của nó là quản lý sự tương tác giữa các chương trình ứng dụng và các driver, nhiều ứng dụng và nhiều driver có thể được quản lý cùng

một lúc. Driver Manager cung cấp sự liên kết giữa các ứng dụng và các driver, cho phép nhiều ứng dụng truy xuất dữ liệu qua nhiều driver. Driver Manager load hay unload một hoặc nhiều driver cho một hoặc nhiều ứng dụng. Khi một ứng dụng cần truy xuất một nguồn dữ liệu, Driver Manager sẽ load đúng driver cần thiết. Driver Manager xác định các ODBC function được cung cấp bởi driver đó và ghi các địa chỉ trong bộ nhớ của chúng vào một bảng. Khi một ứng dụng gọi một function trong một driver, Driver Manager sẽ xác định vào gọi function đó. Bằng cách này, nhiều driver có thể được quản lý đồng thời và người lập trình ứng dụng không phải lo lắng đến việc quản lý chi tiết các từng driver. Một ứng dụng có thể sử dụng ODBC tại cùng một thời điểm với một ứng dụng khác mà không cần phải biết đến ứng dụng này.

Drivers: Các driver xử lý các ODBC function được gọi, đưa ra các yêu cầu SQL để chỉ định các nguồn dữ liệu, và trả về kết quả cho các ứng dụng. Các driver cũng đảm nhận việc tương tác với bất cứ các lớp phần mềm nào cần thiết để truy xuất nguồn dữ liệu.

Data sources: Bao gồm các tập hợp dữ liệu và các môi trường tương ứng của chúng, bao gồm các hệ điều hành, các DBMS, và các phần mềm mạng.

c. Các mô hình hoạt động của ODBC.

- Mô hình One-Tier

Mô hình này được dùng để truy xuất các Desktop Database/ISAM (Indexed Sequential Access Method) file (các file dữ liệu Foxpro, Access, Paradox, dBase), hoặc các flat file (các file text hoặc spreadsheet).

Trong mô hình này One-Tier driver sẽ đảm nhiệm vai trò của một SQL Database Engine, thực hiện xử lý tất cả các câu lệnh SQL (parse, optimize, execute).

- Mô hình Two-Tier

Đây là một mô hình kinh điển trong kiến trúc Client/Server. Các Two-Tier driver trực tiếp gửi và nhận thông tin trên giao thức truyền dữ liệu của một DBMS hoặc ánh xạ tới các native Database API, không trực tiếp truy xuất dữ liệu. DBMS Server nhận các yêu cầu SQL từ Client, thực hiện chúng và gửi kết quả trở lại Client.

Cho ví dụ, Two-Tier driver truy xuất CSDL trên Microsoft SQL Server sẽ trực tiếp truyền và nhận thông tin trên giao thức truyền dữ liệu, Two-Tier driver truy xuất CSDL trên Oracle sẽ ánh xạ tới Oracle's Native API đó là OCI (Oracle Call Interface).

Một sự biến đổi khác của mô hình này:

- Mô hình Three-Tier

Client trong mô hình Three-Tier thay vì kết nối trực tiếp tới DBMS, nó được kết nối qua một Gateway Server.

Trong thực tế, Gateway Server sẽ kết nối tới nhiều DBMS. Trong hệ thống triển khai các ứng dụng xây dựng trên ODBC để truy xuất nhiều nguồn dữ liệu, mô hình Three-Tier đã đưa hầu hết những sự phức tạp trên Client lên Server. Nó trợ giúp rất nhiều trong việc đơn giản hoá sự cài đặt, quản lý các driver trên Client.

7.1.3 JDBC - java database connectivity

- JDBC là gì?

Tương tự như ODBC thì JDBC cũng là một Database API chuẩn. JDBC API định nghĩa các lớp Java để đưa ra các kết nối CSDL, các câu lệnh SQL, các tập hợp kết quả, các siêu dữ liệu,... Nó cho phép một người lập trình Java đưa ra các câu lệnh SQL và xử lý các kết quả được trả về. JDBC là primary API cho việc truy xuất dữ liệu trong Java.

JDBC API được thực hiện qua một Driver Manager mà nó thể cung cấp nhiều driver kết nối tới các kiểu CSDL khác nhau. Các JDBC driver hoặc có thể được viết hoàn toàn bằng Java (pure java) để cho chúng có thể được download như một phần của applet, hoặc chúng có thể được thực thi sử dụng các native method để nối với các thư viện truy xuất CSDL đã có (Database Access Libraries).

- Kiến trúc của jdbc

Cũng như ODBC thì JDBC được thiết kế cho mô hình kiến trúc Client/Server với các ứng dụng Java truy xuất CSDL. Các thành phần và chức năng chúng trong kiến trúc của JDBC cũng tương tự như trong ODBC.

+ **Java Applications**: Các ứng dụng truy nhập CSDL viết bằng Java (Java applet/Java stand-alone application).
+ **JDBC Driver Manager**.
+ **JDBC Drivers**.
+ **Data Source**.

- Các mô hình hoạt động của JDBC

1. Mô hình Two-Tier:

Trong mô hình Two-Tier, một Java applet/application qua các driver trực tiếp gọi tới CSDL. Mô hình yêu cầu một JDBC driver có thể giao tiếp với một DBMS đặc biệt được truy xuất. Qua đó các câu lệnh SQL của người sử dụng được chuyển tới DBMS, và kết quả của được gửi trở lại cho người sử dụng. DBMS được đặt trên một Database Server.

- **Native-API party-Java driver**: Kiểu driver này chuyển các yêu cầu JDBC thành các yêu cầu tương ứng trên các native API cho DBMS tương ứng. Nó là một bridge driver nên yêu cầu phải có một vài mã được nạp trên Client.

- **JDBC-ODBC bridge driver**: Kiểu driver này truy xuất DBMS qua các ODBC driver. Yêu cầu mã ODBC phải được nạp trên Client. Kiểu driver hầu như chỉ dùng trên các mạng tổ hợp - nơi mà việc cài đặt client không phải là vấn đề chuyên môn hoặc cho application server viết bằng Java trong kiến trúc three-tier.

- **Native-protocol pure Java driver**: Kiểu driver này chuyển các yêu cầu JDBC trên giao thức được sử dụng trực tiếp bởi DBMS. Nó cho phép một yêu cầu trực tiếp từ Client tới DBMS Server, đây là giải pháp thích hợp cho truy xuất Intranet.

2. Mô hình Three-Tier:

Trong mô hình Three-Tier, các yêu cầu được JDBC-Net driver gửi tới một middle-tier qua một giao thức độc lập với DBMS, sau đó qua một giao thức đặc biệt middle-tier gửi các câu lệnh SQL tới DBMS. DBMS xử lý các câu lệnh này rồi gửi kết quả trở lại cho middle-tier, middle-tier gửi kết quả này tới ứng dụng.

Pure Java driver như JDBC-Net pure Java driver và Native-protocol pure Java driver sẽ thường được sử dụng để truy xuất các CSDL, nó khai thác được lợi thế của Java và tốc độ truy xuất sẽ nhanh hơn các bridge driver. Còn các kiểu bridge driver như JDBC-ODBC bridge driver và Native-API partly Java driver chỉ là các giải pháp tạm thời được sử dụng trong những trường hợp mà ở đó không thể sử dụng các pure Java driver.

7.2 ADO – ActiveX Data Object

ADO là công nghệ truy cập cơ sở dữ liệu hướng đối tượng, được xem là kỹ thuật để truy cập cơ sở dữ liệu từ Web Server của Microsoft. ADO được cung cấp dưới dạng thư viện ActiveX

Server, chúng ta có thể thỏa mái dùng ADO trong ứng dụng Visual Basic, trong thực tế sử dụng ADO để làm việc với cơ sở dữ liệu Client/Server thì dễ dàng hơn các kỹ thuật khác.

Tuy nhiên, với xu hướng phát triển của công nghệ Internet thì việc kết nối dự liệu từ xa là một nhu cầu rất thực tế và thông dụng. Do đó, chúng ta sẽ tìm hiểu về đối tượng ADO trong môi trường Web, mà cụ thể là môi trường của ngôn ngữ kịch bản ASP.

Để tạo một đối tượng Recordset ASP, chúng ta dùng phương thức CreateObject với cú pháp như sau:

Dim rs 'Khai báo biến recordset

Set rs = Server.CreateObject("ADODB.Recordset")

Bởi vì đối tượng Server là mặc định của ASP nên ta không cần phải tham chiếu đến nó trực tiếp, có nghĩa là chúng ta có thể dùng ***CreateObject()*** thay vì ***Server.CreateObject()***.

Để tham chiếu đến một trường của đối tượng Recordset, chúng ta sử dụng một trong hai cú pháp sau:

rs.Fields("Tên trường")

Hoặc:

rs!<Tên trường>

Để đóng đối tượng Recordset lại, chúng ta dùng cú pháp:

rs.Close

set rs = nothing 'hủy đối tượng Recordset

Ví dụ sau sẽ mô tả tổng quát một cách sử dụng đối tượng ADO để hiển thị dang sách nhân viên trong cơ sở dữ liệu QuanLyNhanVien (quản lý nhân viên) đã được tạo bởi ODBC với tên DSN là QLNV:

```
<html>
<head>
<title> Hien thi dang sach nhan vien</title>
</head>
<body>
<%
set rs = Server.CreateObject("ADODB.Recordset")
rs.ActiveConnection = "DSN = QLNV"
rs.Open "SELECT * FROM NHANVIEN WHERE Luong>=1000000"
Do Until rs.EOF
    Response.write("rs.Fields("HotenNV") & " – ")
```

```
    Response.write("rs.Fields("NgaySinh") & "– ")
    Response.write("rs.Fields("Luong") & "<br>")
    rs.MoveNext
Loop
rs.Close
set rs = nothing
%>
</body>
</html>
```

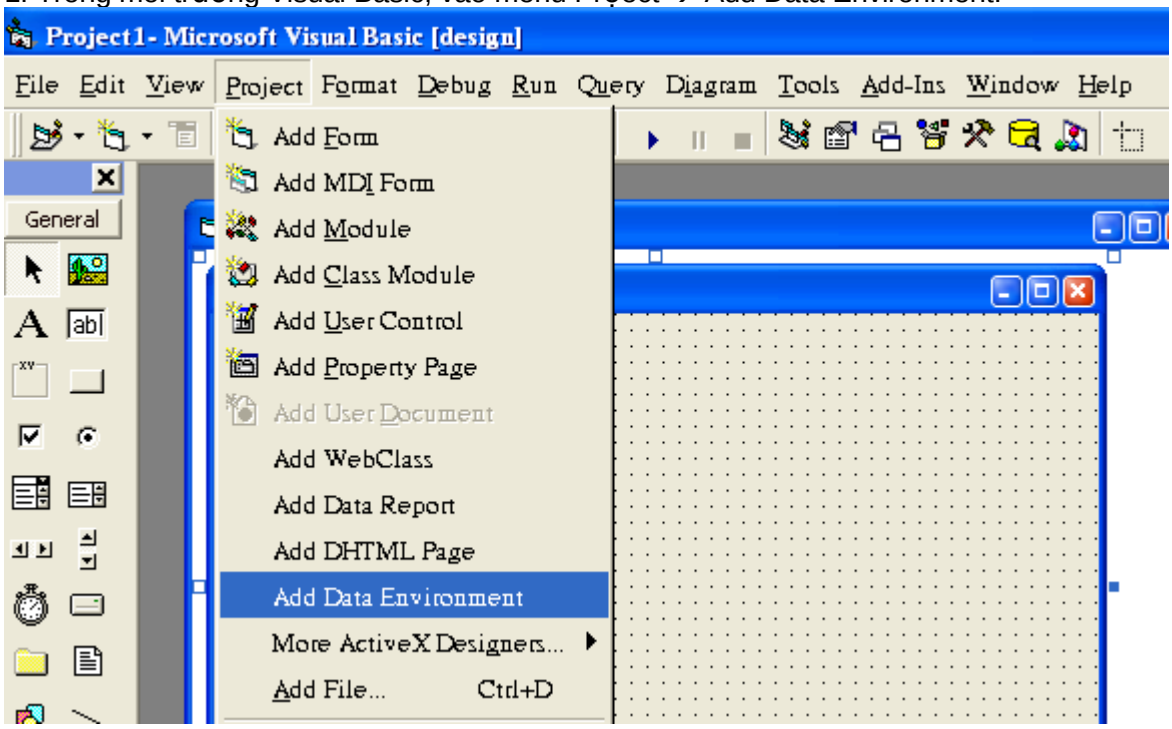
Chú ý: Ở ví dụ trên, đoạn lệnh VB Script được đặt trong cặp dấu <% và %>.

7.3 Data Environment

Trong phần này chúng ta sẽ tìm hiểu về cách sử dụng môi trường dữ liệu (Data Environment) để xây dựng một ứng dụng Visual Basic 6.0 kết nối đến cơ sở dữ liệu SQL, đối với các hệ quản trị cơ sở dữ liệu khác chúng ta cũng có cách làm tương tự.

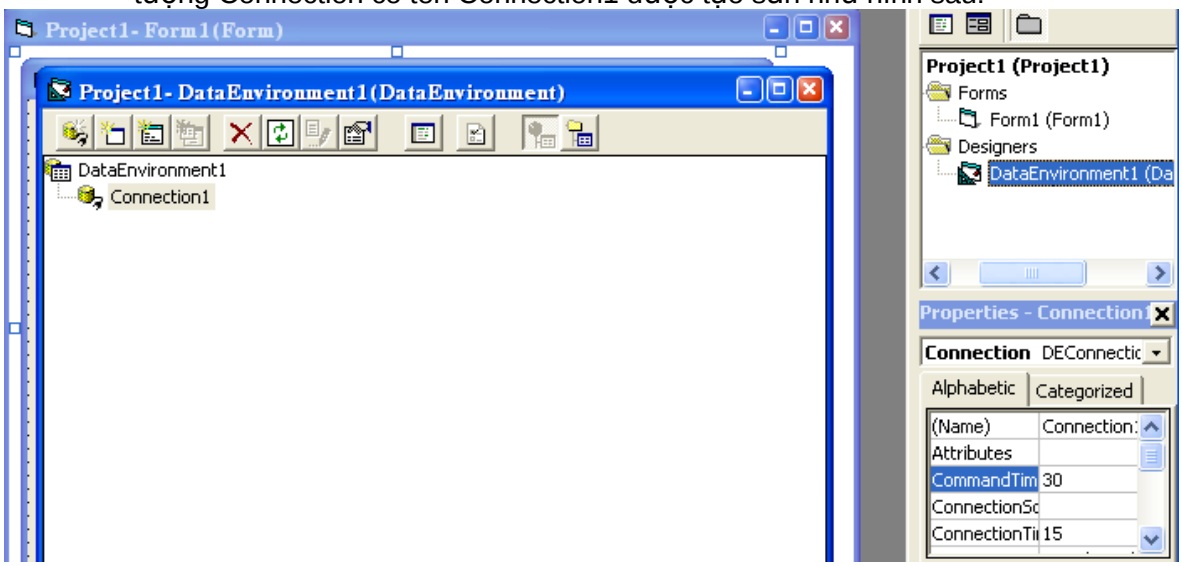
Trước tiên, chúng ta chắc chắn rằng đã thiết kế cơ sở dữ liệu QuanLyNhanVien như trên. Các bước thực hiện sau sẽ cho phép sử dụng đối tượng Textbox để điều khiển các bản ghi trong bảng NHANVIEN của cơ sở dữ liệu:

1. Trong môi trường Visual Basic, vào menu Project → Add Data Environment:



Hình 7.1:Add Data Environment

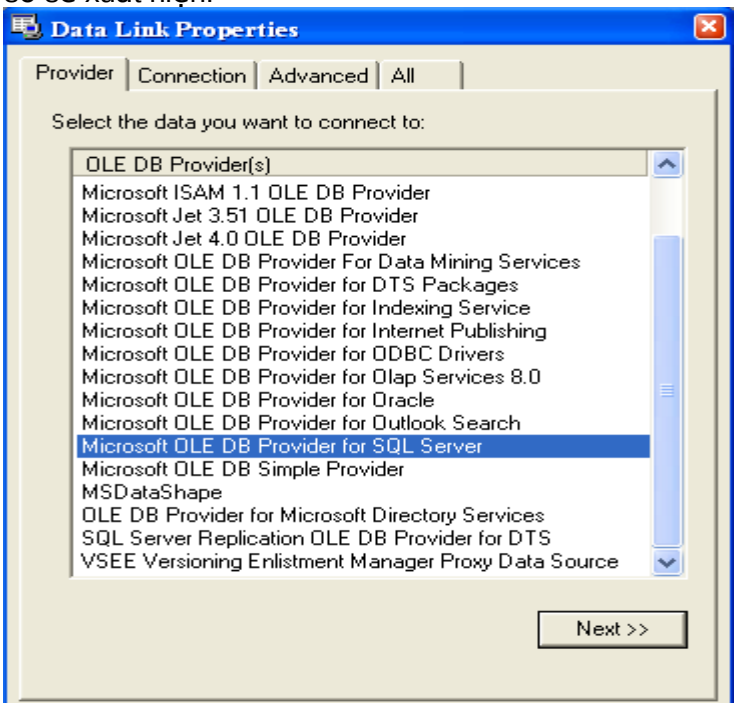
một đối tượng Data Environment sẽ được bổ sung vào có tên DataEnvironment1 và một dối tượng Connection có tên Connection1 được tạo sẵn như hình sau:



Hình 7.2: Data Environment 1

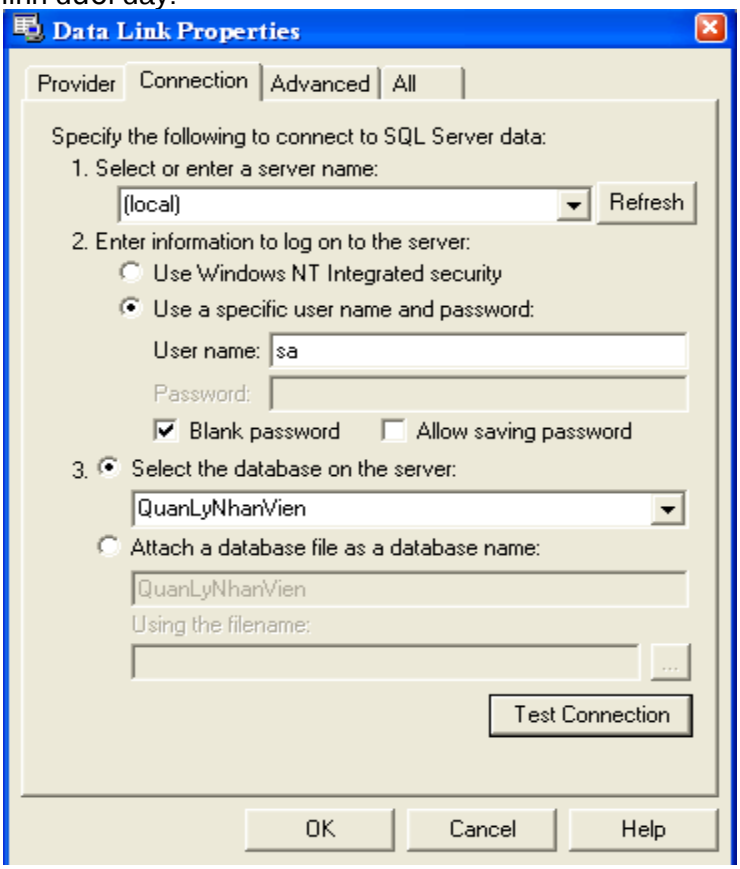
2. Để với tên mặc định hoặc có thể thay đổi bằng cách nhắp chuột phải lên đối tượng → Rename để đổi tên lại nếu bạn muốn.

3. Tạo kết nối cho đối tượng Connection1 bằng cách nhắp chuột phải lện tên Connection1 → Properties, một cửa sổ sẽ xuất hiện:



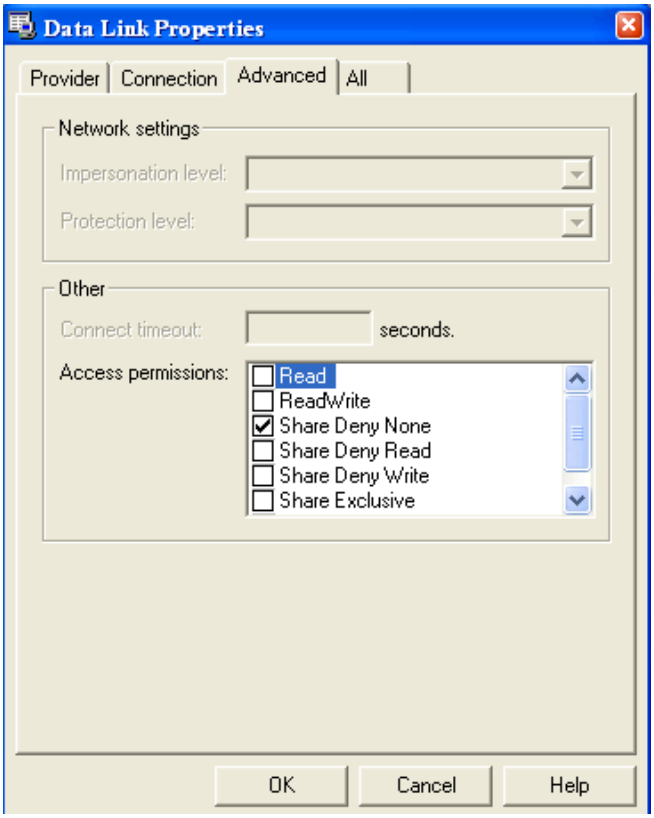
Hình 7.3:Properties

Ở đây chung ta muốn kết nối tới cơ sở dữ liệu SQL Server nên sẽ chọn chuỗi kết nối ***Microsoft OLE DB Provider for SQL Server*** như hình trên, lưu ý nếu sử dụng cơ sở dữ liệu khác thì sẽ dùng chuỗi kết nối khác trong danh sách tương ứng. Nhấn **Next** để tiếp tục và chọn tên cơ sở dữ liệu như hình dưới đây:



Hình 7.4:Connection and select database

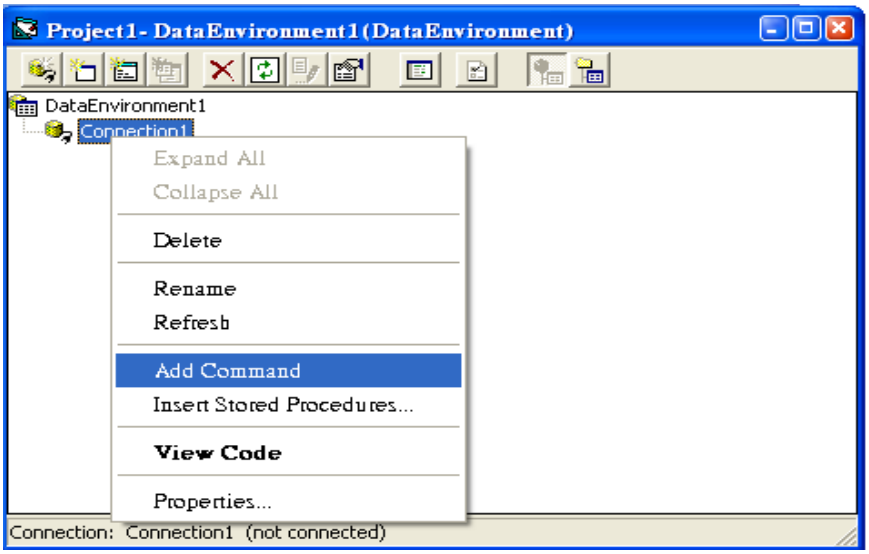
- Nếu muốn hạn chế quyền truy cập thì chọn vào tab Advanced và thiết lập như hình dưới đây:



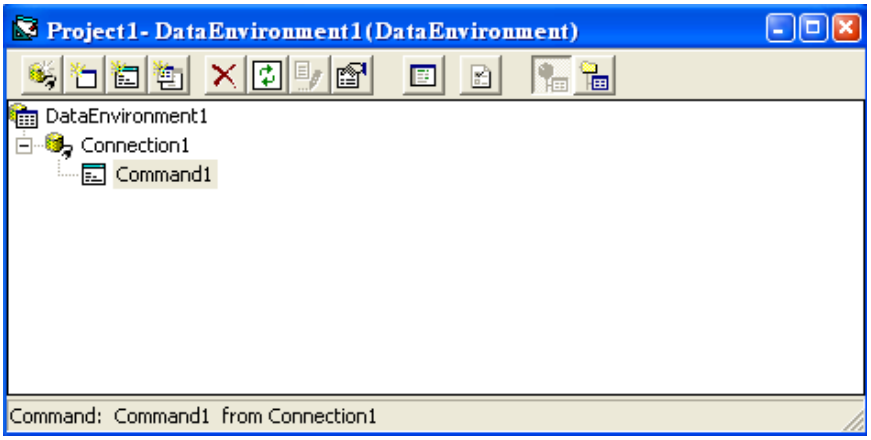
Hình 7.5: Advanced access permissions

Nhấn OK để hoàn tất

4. Nhắp chuột phải lên tên đối tượng Connection1 → Add Command để bổ sung vào một đối tượng Command1:

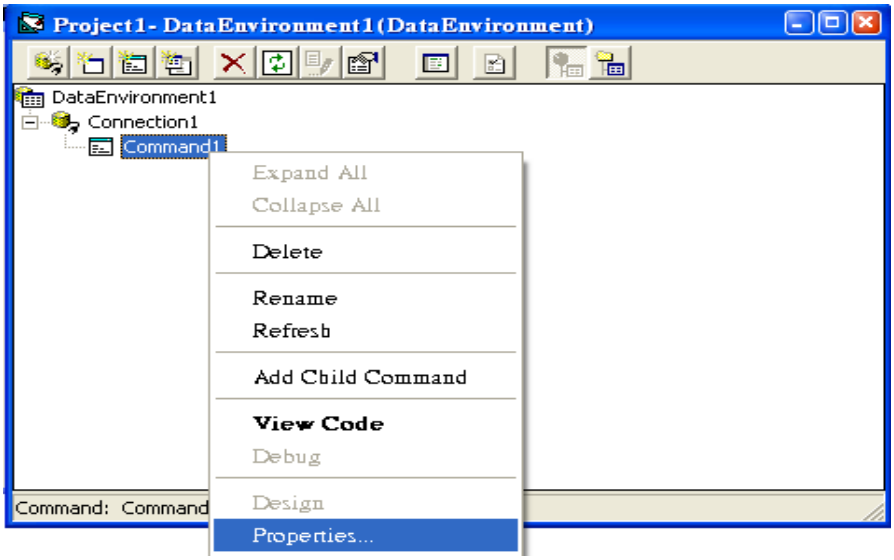


Hình 7.6:Add Command



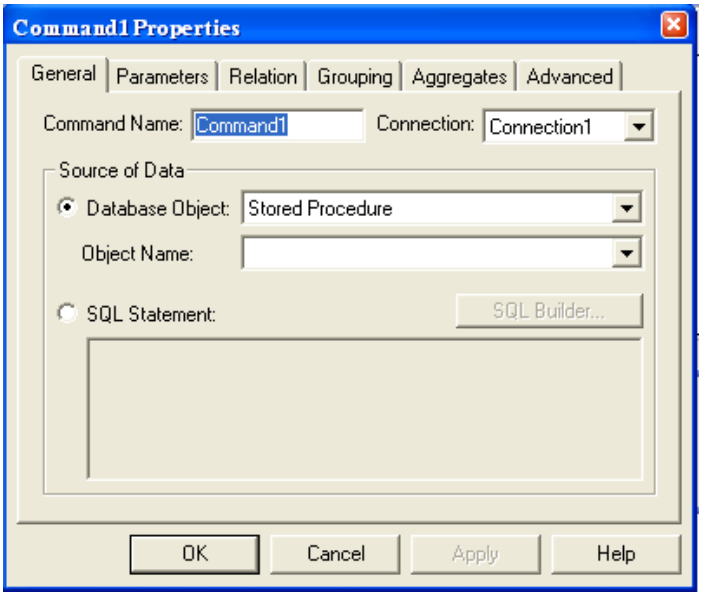
Hình 7.7:Command

5. Tiếp theo, nhắp chuột phải lên tên đối tượng Command1 → Properties:



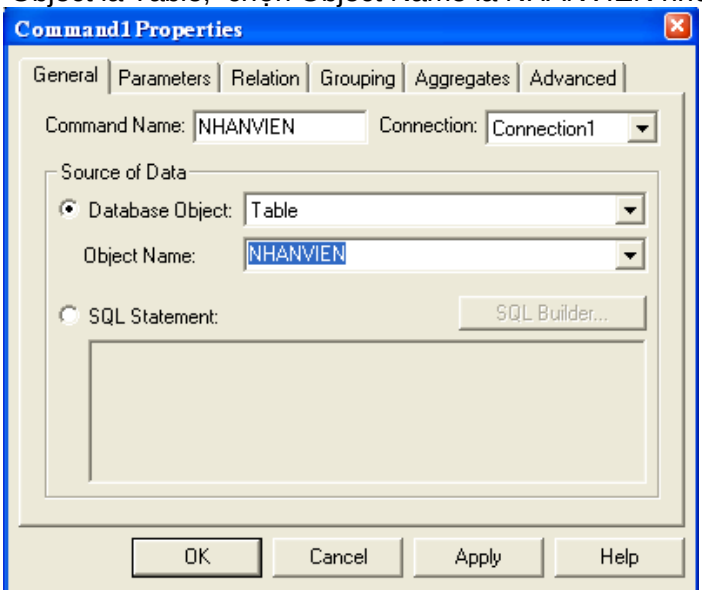
Hình 7.8:Properties

Một cửa sổ sau sẽ hiện ra:



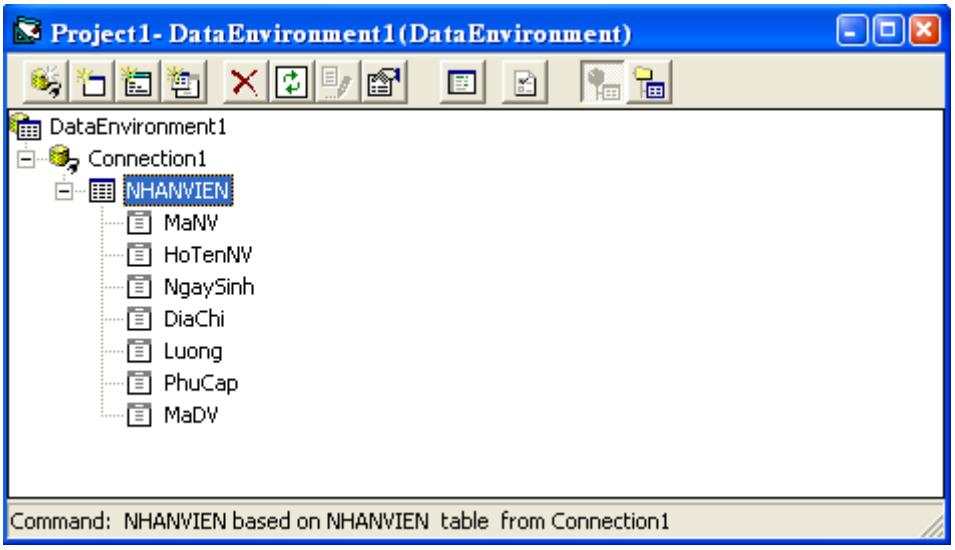
Hình 7.9:General command properties

6. Đặt tên cho lệnh, giả sử ở đây chúng ta đặt tên là NHANVIEN, và trong **Source of Data**, chúng ta chọn Database Object là Table, chọn Object Name là NHANVIEN như hình sau:



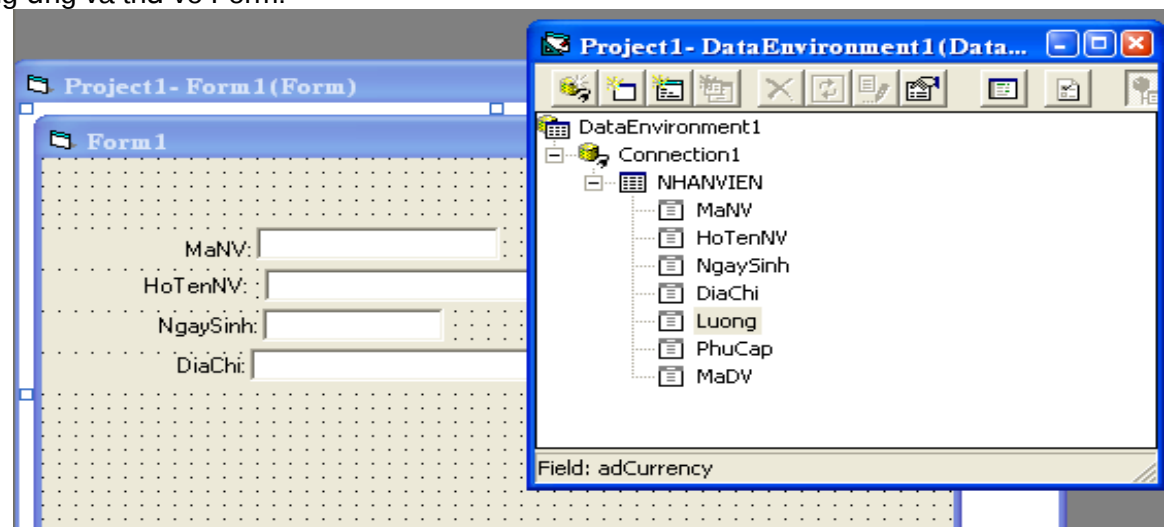
Hình 7.10:Object Name

Nhấn OK để hoàn tất, chúng ta sẽ thấy xuất hiện một bảng NHANVIEN trong môi trường dữ liệu như sau:



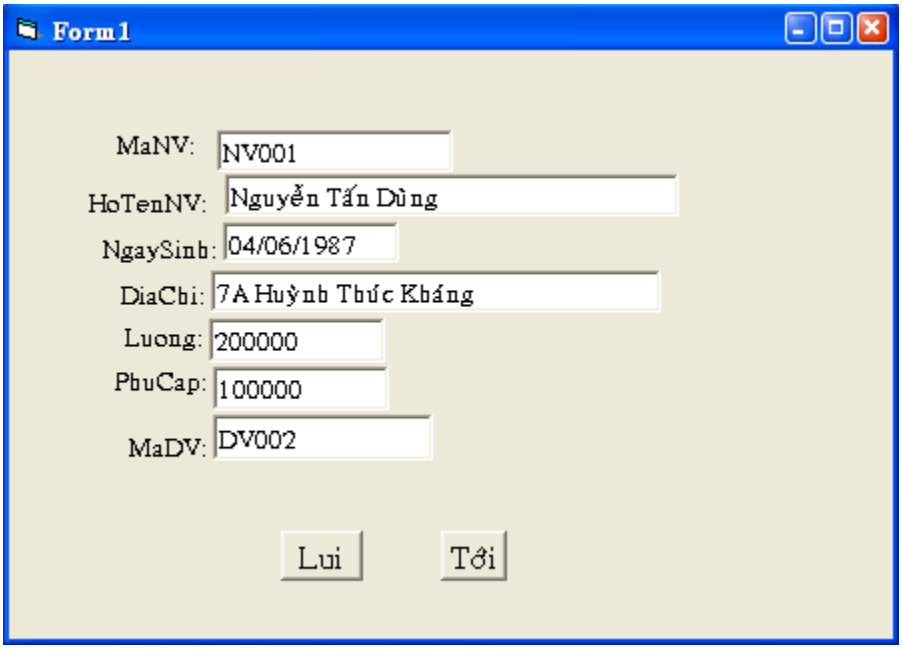
Hình 7.11:bảng trong môi trường dữ liệu

7. Kết gán các trường cho các trường dữ liệu vào Form bằng cách nhấp chuột vào trường tương ứng và thả vò Form:



Hình 7.12:Tạo Form

8. Sau khi hoàn tất, nhấn F5 để chạy thử chương trình, chúng ta sẽ thấy chương trình được hiển thị như sau:



Hình 7.13: chương trình được hiển thi

Quả là rất đơn giản, chúng ta đã có thể hiển thị được dữ liệu của bảng nhân viên, nếu chúng ta thay đổi thông tin này thì dữ liệu cũng sẽ được cập nhật đối với cơ sở dữ liệu. Và dĩ nhiên sẽ chỉ thay đổi đối với bản ghi hiện hành, đối với ví dụ trên là bản ghi đầu tiên, nếu muốn di chuyển đến bản ghi khác chúng ta có thể dùng các phương thức MoveNext, MovePrevious,… của đối tượng Recordset trong DataEnvironment1:

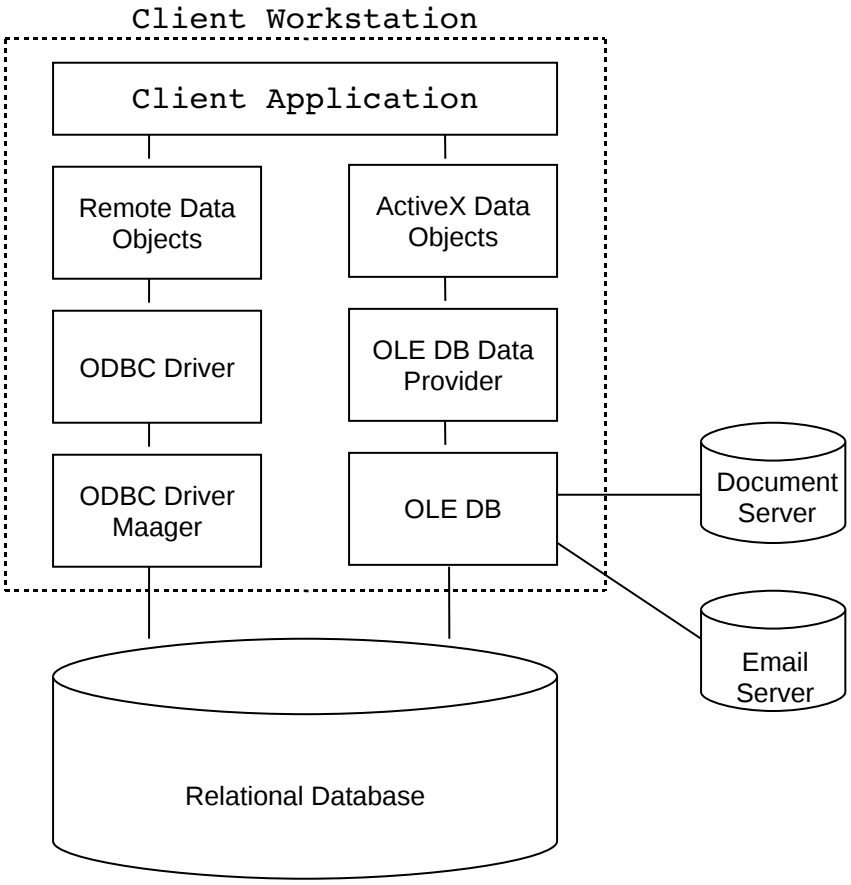
```
Private Sub cmdLui_Click()
   DataEnvironment1.rsNHANVIEN.MovePrevious
End Sub

Private Sub cmdToi_Click()
   DataEnvironment1.rsNHANVIEN.MoveNext
End Sub
```

Lưu ý, ở đoạn lệnh trên chúng ta đã sử dụng một đối tượng Recordset mặc định tên là rsNHANVIEN để trỏ đến bảng NHANVIEN do môi trường tự động sinh ra. Đoạn lệnh trên là thủ tục sự kiện của hai nút lệnh Lui và Tới.

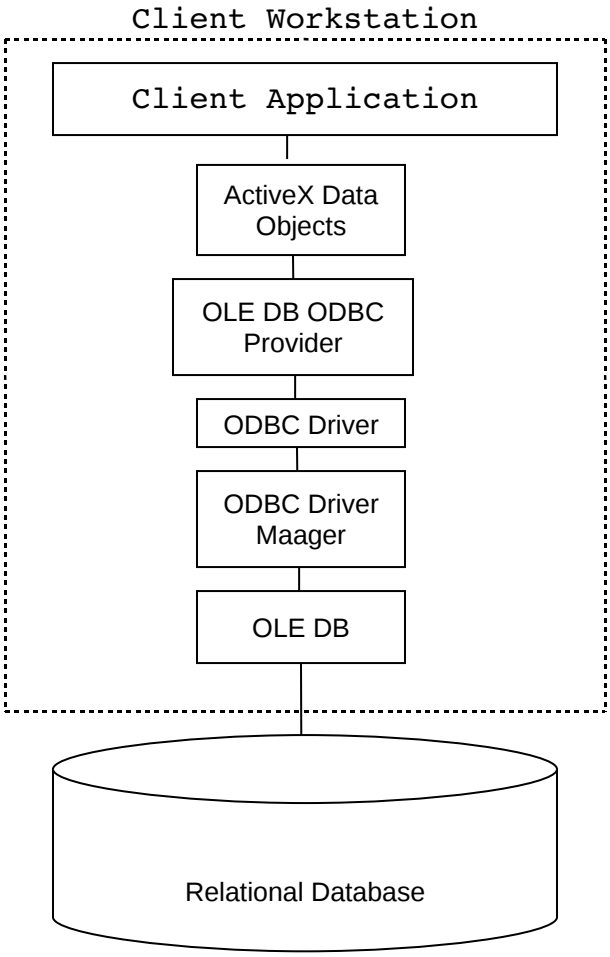
7.4 OLE_DB

Phần lớn các nhà lập trình không tương tác trực tiếp với OLE DB. Thay vào đó họ lập trình với đối tượng ADO, tuy nhiên chúng ta cũng cần phải biết mô hình đối tượng cung cấp giao diện với OLE DB.

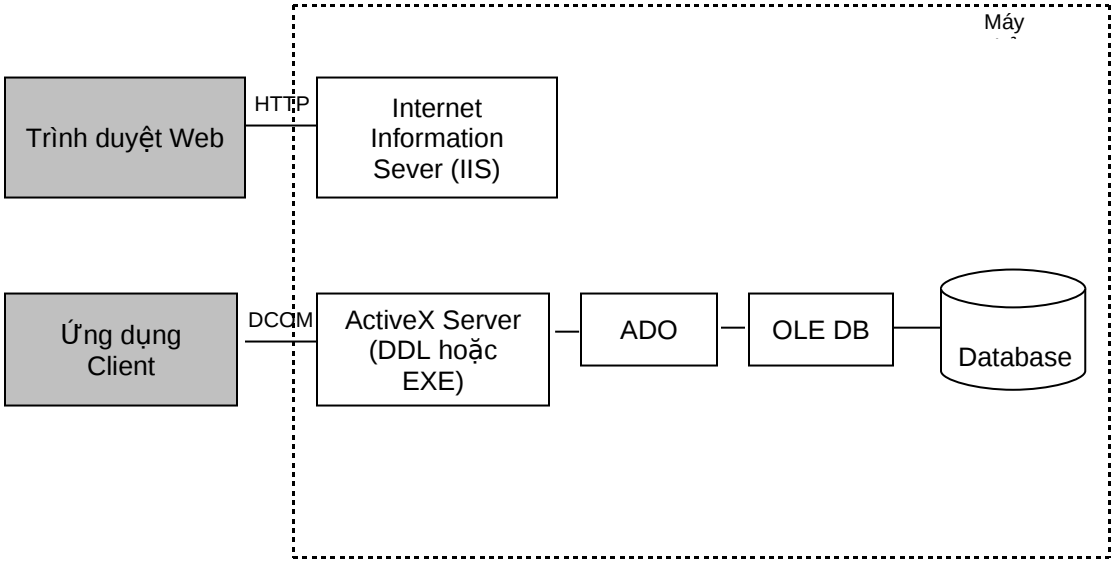
Chúng ta có thể tham khảo ba mô hình sau để có cái nhìn sâu hơn về OLE DB:



Mô hình 1: Sử dụng ADO và OLE DB truy cập thông tin trong một cơ sở dữ liệu



Mô hình 2: Cấu trúc truy cập cơ sở dữ liệu ODBC dùng trình cung cấp ODBC OLE DB.



Mô hình 3: Cấu trúc sử dụng một thành phần chương trình ActiveX chung với cả trình duyệt Web và các ứng dụng Client.

7.5 Lập trình trên các đối tượng RecordSet

Như một số ví dụ trên, chúng ta đã làm quen với đối tượng Recordset. Bây giờ chúng ta sẽ tìm hiểu rõ hơn về các phương thức, thuộc tính của đối tượng Recordset.

*** Các phương thức của đối tượng Recordset:**

PHƯƠNG THỨC	DIỄN GIẢI
AddNew	Tạo một bản ghi mới
Cancel	Hủy bỏ thao tác đang thực thi
CancelBatch	Hủy bỏ các cập nhật bị treo
CancelUpdate	Hủy bỏ các thay đổi với bản ghi hiện hành
Clone	Tạo một bản sao của đối tượng Recordset
Close	Đóng đối tượng Recordset và các đối tượng liên quan
CompareBookmarks	So sánh 2 chổ đánh dấu
Delete	Xóa bản ghi hay một tập bản ghi hiện hành
Find	Tìm một bản ghi thỏa điều kiện
GetRows	Lấy nhiều bản ghi đưa vào một mảng
GetString	Trả recordset về dưới dạng một chuổi
Move	Di chuyển vị trí của bản ghi hiện hành
MoveFirst	Đưa vị trí của bản ghi hiện hành đến bản ghi đầu tiên trong Recordset
MoveLast	Đưa vị trí của bản ghi hiện hành đến bản ghi cuối cùng trong Recordset
MoveNext	Đưa vị trí của bản ghi hiện hành đến bản ghi tiếp theo trong Recordset
MovePrevious	Đưa vị trí của bản ghi hiện hành đến bản ghi trước đó trong Recordset
NextRecordset	Xóa đối tượng recordset hiện hành và trả về đối tượng recordset kế tiếp
Open	Mở một Recordset
Requery	Cập nhật lại dữ liệu bằng cách thực thi lại câu truy vấn ban đầu
Resync	Refresh lại dữ liệu trong đối tượng Recordset hiện hành
Save	Lưu recordset xuống file
Seek	Tìm chỉ mục của Recordset
Supports	Xác định xem đối tượng recordset có hổ trợ chức năng gì đặc biệt
Update	Lưu các thay đổi
UpdateBatch	Lưu các khối thay đổi xuống đĩa

*** Các thuộc tính của đối tượng Recordset:**

THUỘC TÍNH	DIỄN GIẢI
BOF	Trả về giá trị là TRUE nếu vị trí bản ghi hiện thời nằm phía trước bản ghi đầu tiên, ngược lại là FALSE
EOF	Trả về giá trị là TRUE nếu vị trí bản ghi hiện thời nằm phía sau bản ghi cuối cùng, ngược lại là FALSE
RecordCount	Trả về số bản ghi trong Recordset
Sort	Sắp xếp

Để thấy rõ hơn lợi ích và tính khả thi của đối tượng Recordset trong các thao tác dữ liệu, chúng ta hãy xem xét cách thức kết nối và thao tác trên cơ sở dữ liệu SQL Server.

Trong thực tế, người ta ít khi thực hiện việc kết nối trực tiếp qua đối tượng Recordset mà thường thông qua một đối tượng kết nối cơ sở dữ liệu gọi là đối tượng Connection nhằm tăng tính linh động và hiệu quả cho ứng dụng cũng như Website. Chuỗi kết nối OLE DB của hệ quản trị cơ sở dữ liệu SQL Server được cung cấp như sau:

“Data Source = tên_server; Initial Catalog = tên_cơ_sở_dữ_liệu; User ID = tên_sử_dụng; Password = mật_khẩu”

Ví dụ sau sẽ dùng đối tượng Connection kết nối dữ liệu, dùng đối tượng Recordset hiển thị thông tin nhân viên:

```
<%
Dim Conn
Dim rs
Set Conn = server.CreateObject(“ADODB.Connection”)
Conn.Open “DSN = QLNV”
Dim sqlText
sqlText = “SELECT * FROM NHANVIEN”
Set rs = Conn.Execute(sqlText)
While Not rs.EOF
    Response.Write(rs(“HotenNV”) & “ – “)
    Response.Write(rs(“NgaySinh”) & “ – “)
    Response.Write(rs(“Luong”) & “<br> “)
    rs.MoveNext
WEnd
Rs.Close
```

```
Set rs = nothing
Conn.Close
%>
```

BÀI TẬP THỰC HÀNH

1. Xây dựng một ứng dụng WEB bằng ngôn ngữ VB Script kết nối đến cơ sở dữ liệu SQL Server QuanLyNhanVien và thực hiện các công việc sau:

- Hiển thị danh sách nhân viên và đơn vị.
- Bổ sung các nhân viên và đơn vị.
- Sửa đổi thông tin một nhân viên và đơn vị dựa vào mã nhân viên, mã đơn vị.
- Xóa các nhân viên và đơn vị dựa vào mã nhân viên, mã đơn vị.

CÁC THUẬT NGỮ CHUYÊN MÔN

1. Database: Cơ sở dữ liệu
2. Server: Máy chủ
3. Client: Máy khách

TÀI LIỆU THAM KHẢO

1. Giáo trình SQL Server – Trần Nguyên Phong – Trường Đại học Khoa học Huế
2. Website: http://vovisoft.com
3. ASP.NET- Kỹ thuật và ứng dụng – Nhà xuất bản thống kê 2002.