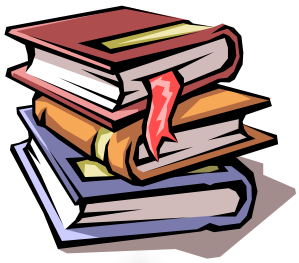
BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ...

KHOA ...



Bài giảng
Hệ quản trị cơ sở dữ liệu

GV: Chu Thị Hường

MỤC LỤC

Chương 1: TỔNG QUAN VỀ HỆ QUẢN TRỊ CSDL

1.1. Định nghĩa:

- Hệ quản trị cơ sở dữ liệu (Database Management System - DBMS): Là một hệ thống phần mềm cho phép tạo lập cơ sở dữ liệu và điều khiển mọi truy nhập đối với cơ sở dữ liệu đó.

Trên thị trường phần mềm hiện nay ở Việt Nam đã xuất hiện khá nhiều phần mềm hệ quản trị cơ sở dữ liệu như: Microsoft Access, Foxpro, DB2, SQL Server, Oracle,.v.v…

- Hệ quản trị cơ sở dữ liệu quan hệ (Relation Database Management System - RDBMS) là một hệ quản trị cơ sở dữ liệu theo mô hình quan hệ.

1.2. Các khả năng của hệ quản trị CSDL

Có hai khả năng chính cho phép phân biệt các hệ quản trị cơ sở dữ liệu với các kiểu hệ thống lập trình khác:

i. Khả năng quản lý dữ liệu tồn tại lâu dài: đặc điểm này chỉ ra rằng có một cơ sở dữ liệu tồn tại trong một thời gian dài, nội dung của cơ sở dữ liệu này là các dữ liệu mà hệ quản trị CSDL truy nhập và quản lý.

ii. Khả năng truy nhập các khối lượng dữ liệu lớn một cách hiệu quả.

Ngoài hai khả năng cơ bản trên, hệ quản trị CSDL còn có các khả năng khác mà có thể thấy trong hầu hết các hệ quản trị CSDL đó là:

iii. Hỗ trợ ít nhất một mô hình dữ liệu hay một sự trừu tượng toán học mà qua đó người sử dụng có thể quan sát dữ liệu.

iv. Đảm bảo tính độc lập dữ liệu hay sự bất biến của chương trình ứng dụng đối với các thay đổi về cấu trúc trong mô hình dữ liệu.

v. Hỗ trợ các ngôn ngữ cao cấp nhất định cho phép người sử dụng định nghĩa cấu trúc dữ liệu, truy nhập dữ liệu và thao tác dữ liệu.

vi. Quản lý giao dịch, có nghĩa là khả năng cung cấp các truy nhập đồng thời, đúng đắn đối với CSDL từ nhiều người sử dụng tại cùng một thời điểm.

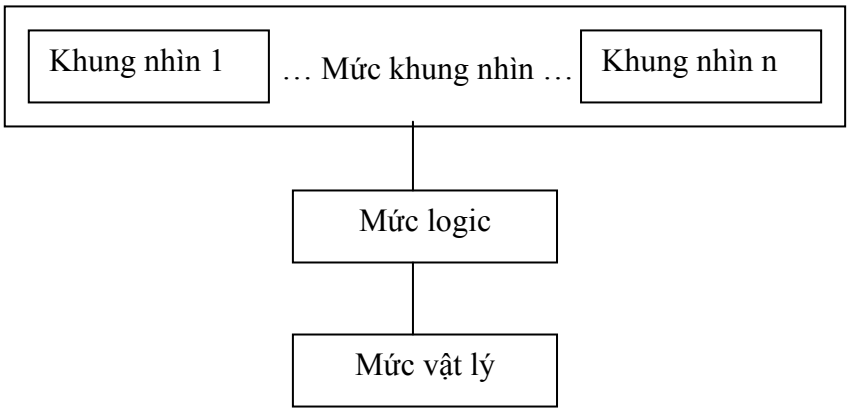
vii. Điều khiển truy nhập, có nghĩa là khả năng hạn chế truy nhập đến các dữ liệu bởi những người sử dụng không được cấp phép và khả năng kiểm tra tính đúng đắn của CSDL.

viii. Phục hồi dữ liệu, có nghĩa là có khả năng phục hồi dữ liệu, không làm mất mát dữ liệu với các lỗi hệ thống.

1.3. Đặc điểm của một hệ quản trị CSDL

1.3.1. Sự trừu tượng hoá dữ liệu:

Để cho hệ thống có thể sử dụng được, hệ quản trị CSDL phải tra cứu hay tìm kiếm dữ liệu một cách có hiệu quả. Điều này dẫn đến việc thiết kế các cấu trúc dữ liệu phức tạp để biểu diễn dữ liệu trong CSDL này. Người phát triển che dấu tính phức tạp này thông qua một số mức trừu tượng để đơn giản hoá các tương tác của người sử dụng đối với hệ thống.



Hình 1.1. Ba mức trừu tượng dữ liệu

- ***Mức vật lý:*** Mức thấp nhất của sự trừu tượng mô tả dữ liệu được lưu trữ một cách thực sự như thế nào. Tại mức vật lý, các cấu trúc dữ liệu mức thấp phức tạp được mô tả chi tiết.
- ***Mức logic:*** Mức cao tiếp theo của sự trừu tượng hoá mô tả những dữ liệu nào được lưu trữ và các mối quan hệ nào tồn tại giữa các dữ liệu này. Mức logic của sự trừu tượng được xác định người quản trị CSDL, cụ thể phải quyết định những thông tin gì được lưu trữ trong CSDL.

- ***Mức khung nhìn:*** Mức cao nhất của sự trừu tượng mô tả chỉ một phần của toàn bộ CSDL. Mặc dù sử dụng các cấu trúc đơn giản mức logic, một số phức tập vẫn còn tồn tại do kích thước lớn của CSDL. Thực chất những người sử dụng chỉ cần truy nhập đến một phần CSDL, do vậy sự tương tác của họ với hệ thống này là đơn giản hoá và mức khung nhìn của sự trừu tượng được xác định. Hệ thống có thể được cung cấp nhiều khung nhìn đối với cùng một cơ sở dữ liệu.

1.3.2. Ngôn ngữ cơ sở dữ liệu

Một hệ quản trị cơ sở dữ liệu thường cung cấp hai kiểu ngôn ngữ khác nhau đó là: ngôn ngữ mô tả sơ đồ cơ sở dữ liệu và ngôn ngữ biểu diễn các truy vấn và các cập nhật cơ sở dữ liệu.

- Ngôn ngữ định nghĩa dữ liệu (Data Definition Language - DDL)
  + Một sơ đồ CSDL đặc tả bởi một tập các định nghĩa được biểu diễn bởi một ngôn ngữ đặc biệt được gọi là ngôn ngữ định nghĩa dữ liệu. Kết quả của việc dịch các ngôn ngữ này là một tập các bảng được lưu trữ trong một tệp đặc biệt được gọi là từ điển dữ liệu hay thư mục dữ liệu.
  + Một từ điển dữ liệu là một tệp chứa các siêu dữ liệu có nghĩa là các dữ liệu về dữ liệu. Tệp này được tra cứu trước khi dữ liệu thực sự được đọc hay được sửa đổi trong hệ CSDL.
  + Cấu trúc và các phương pháp truy nhập được sử dụng bởi hệ CSDL được đặc tả bởi một tập các định nghĩa trong một kiểu đặc biệt của DDL là ngôn ngữ định nghĩa và lưu trữ dữ liệu.
- Ngôn ngữ thao tác dữ liệu (Data Manipulation Language - DML):
  + Các yêu cầu về thao tác dữ liệu bao gồm:
    - Tìm kiếm thông tin được lưu trữ trong CSDL.

- Thêm thông tin mới vào CSDL.
- Xoá thông tin từ CSDL.
- Thay đổi thông tin được lưu trữ trong CSDL.

+ Một ngôn ngữ thao tác dữ liệu (DML) là một ngôn ngữ cho phép người sử dụng truy nhập hay thao tác dữ liệu được tổ chức bởi mô hình dữ liệu thích hợp. Có hai kiểu ngôn ngữ thao tác dữ liệu cơ bản:

  - Các DML thủ tục đòi hỏi người sử dụng phải đặc tả dữ liệu nào cần tìm kiếm và tìm kiếm những dữ liệu này như thế nào.
  - Các DML phi thủ tục đòi hỏi người sử dụng đặc tả dữ liệu nào cần tìm kiếm mà không phải đặc tả tìm kiếm những dữ liệu này như thế nào.

1.3.3. Xử lý câu hỏi

Công việc của bộ xử lý câu hỏi là biến đổi một truy vấn hay một thao tác CSDL có thể được biểu diễn ở các mức cao thành một dãy các yêu cầu đối với các dữ liệu lưu trữ trong CSDL.

Thường phần khó nhất của nhiệm vụ xử lý câu hỏi là tối ưu hoá câu hỏi, có nghĩa là lựa chọn một kế hoạch tốt nhất đối với hệ thống lưu trữ để trả lời truy vấn này nhanh nhất.

1.3.4. Quản trị giao dịch

Thông thường một số thao tác trên CSDL hình thành một đơn vị logic công việc. Điều này có nghĩa là hoặc tất cả các thao tác được thực hiện hoặc không thao tác nào được thực hiện. Hơn nữa sự thực hiện các thao tác này phải đảm bảo tính nhất quán của CSDL.

Một giao dịch là một tập hợp các thao tác mà xử lý như một đơn vị không chia cắt được. Các hệ quản trị CSDL điển hình cho phép người sử dụng một hay nhiều nhóm thao tác tra cứu hay thay đổi CSDL thành một giao dịch.

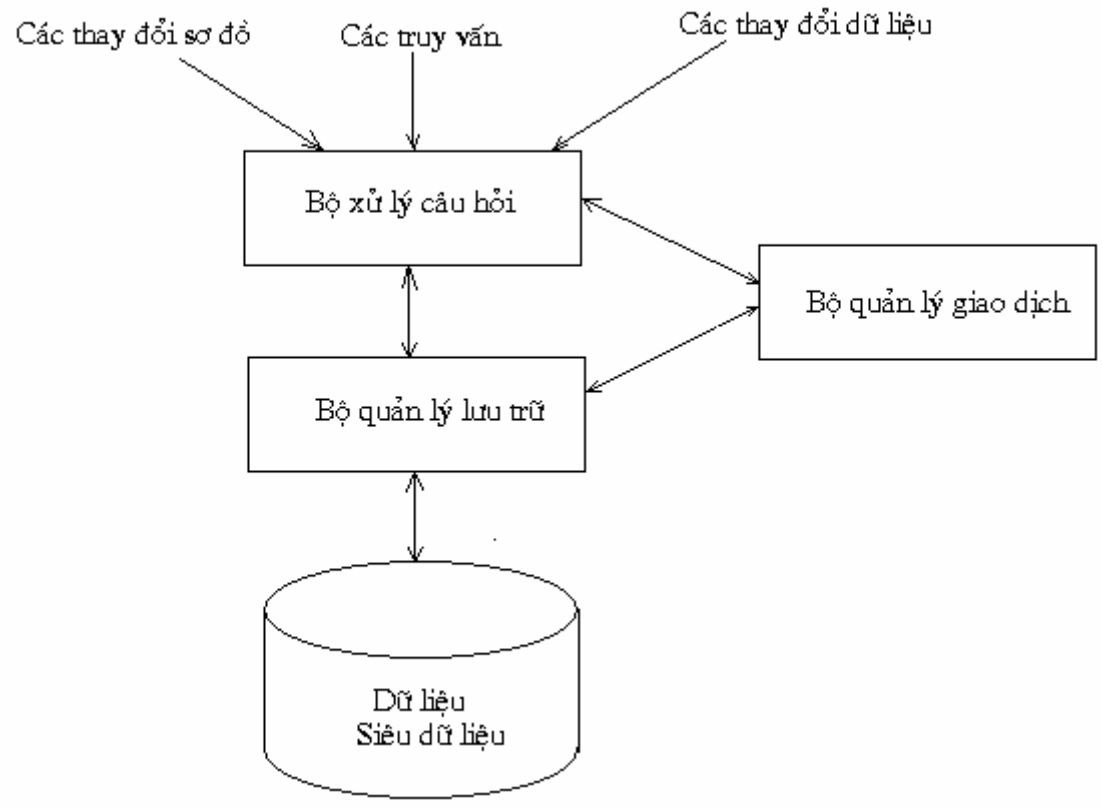
1.3.5. Quản lý lưu trữ

Các CSDL thường đòi hỏi một khối lượng lớn không gian lưu trữ. Do bộ nhớ chính của máy tính không thể lưu trữ nhiều thông tin như vậy, các thông tin này được lưu trữ ở các thiết bị nhớ ngoài như đĩa cứng, đĩa mềm,.v.v…

Khi xử lý, dữ liệu cần phải được di chuyển từ đĩa từ vào bộ nhớ chính; sự di chuyển này là khá chậm so với tốc độ xử lý của bộ nhớ trung tâm, do vậy các hệ CSDL phải tổ dữ liệu vật lý sao cho tốt, tối thiểu hoá số yêu cầu chuyển dữ liệu giữa đĩa từ vào bộ nhớ chính.

1.4. Kiến trúc của một hệ quản trị CSDL

Chúng ta sẽ phác thảo kiến trúc và thấy cách thức của một hệ quản trị CSDL điển hình. Ta có sơ đồ kiến trúc hình 1.2:



Hình 1.2. Các thành phần chính của hệ quản trị CSDL

- Dữ liệu, siêu dữ liệu: Đáy kiết trúc là thiết bị nhớ ngoài lưu trữ dữ liệu và siêu dữ liệu. Trong phần này không chỉ chứa dữ liệu được trữ trong CSDL mà chứa cả các siêu dữ liệu, tức là thông tin cấu trúc của CSDL. *Ví dụ:* Trong

hệ quản trị cơ sở dữ liệu quan hệ, các siêu dữ liệu bao gồm các tên của các quan hệ, tên các thuộc tính của các quan hệ, và các kiểu dữ liệu đối với các thuộc tính này.

- Bộ quản lý lưu trữ: Nhiệm vụ của bộ quản lý lưu trữ là lấy ra các thông tin được yêu cầu từ những thiết bị lưu trữ dữ liệu và thay đổi những thông tin này khi được yêu cầu bởi các mức trên nó của hệ thống.

- Bộ xử lý câu hỏi: Bộ xử lý câu hỏi điều khiển không chỉ các câu hỏi mà cả các yêu cầu thay đổi dữ liệu hay siêu dữ liệu. Nhiệm vụ của nó là tìm ra cách tốt nhất một thao tác được yêu cầu và phát ra lệnh đối với bộ quản lý lưu trữ và thực thi thao tác đó.

- Bộ quản trị giao dịch: Bộ quản trị giao dịch có trách nhiệm đảm bảo tính toàn vẹn của hệ thống. Nó phải đảm bảo rằng một số thao tác thực hiện đồng thời không cản trở mỗi thao tác khác và hệ thống không mất dữ liệu thậm chí cả khi lỗi hệ thống xảy ra.

+ Nó tương tác với bộ xử lý câu hỏi, do vậy nó phải biết dữ liệu nào được thao tác bởi các thao tác hiện thời để tránh sự đụng độ giữa các thao tác và cần thiết nó có thể làm trễ một số truy vấn nhất định hay một số thao tác cập nhật để đụng độ không thể xảy ra.

+ Nó tương tác với bộ quản lý lưu trữ bởi vì các sơ đồ đối với việc bảo vệ dữ liệu thường kéo theo việc lưu trữ một nhật ký các thay đổi đối với dữ liệu. Hơn nữa, việc sắp thứ tự các thao tác một cách thực sự được nhật ký này sẽ chứa trong một bản ghi đối với mỗi thay đổi khi gặp lỗi hệ thống, các thay đổi chưa được ghi vào đĩa có thể được thực hiện lại.

- Các kiểu thao tác đối với hệ quản trị CSDL: Tại đỉnh kiến trúc, ta thấy có 3 kiểu thao tác:

+ *Các truy vấn:* Đây là các thao tác hỏi đáp về dữ liệu được lưu trữ trong CSDL. Chúng được sinh ra theo hai cách sau:

    - Thông qua giao diện truy vấn chung. Ví dụ: Hệ quản trị CSDL quan hệ cho phép người sử dụng nhập các câu

lệnh truy vấn SQL mà nó được chuyển qua bộ xử lý câu hỏi và được trả lời.

- Thông qua các giao diện chương trình ứng dụng: Một hệ quản trị CSDL điển hình cho phép người lập trình viết các chương trình ứng dụng gọi đến hệ quản trị CSDL này và truy vấn CSDL.

+ *Các cập nhật dữ liệu:* Đây là các thao tác thay đổi dữ liệu như xoá, sửa dữ liệu trong CSDL. Giống như các truy vấn, chúng có thể được phát ra thông qua giao diện chung hoặc thông qua giao diện của chương trình.

+ *Các thay đổi sơ đồ:* Các lệnh này thường được phát bởi một người sử dụng được cấp phép, thường là những người quản trị CSDL mới được phép thay đổi sơ đồ của CSDL hay tạo lập một CSDL mới.

1.5. Các chức năng của hệ quản trị CSDL quan hệ

1.5.1. Các khái niệm trong mô hình dữ liệu quan hệ

- *Miền (domain):* là một tập các giá trị hoặc các đối tượng.

- *Thực thể:* Thực thể là một đối tượng cụ thể hay trừu tượng trong thế giới thực mà nó tồn tại và có thể phân biệt được với các đối tượng khác.

Ví dụ: Bạn Nguyễn Văn A là một thực thể cụ thể. Hay Sinh viên cũng là một thực thể, thực thể trừu tượng.

- *Thuộc tính (Attribute):* Là tính chất của thực thể.

+ Các thực thể có các đặc tính, được gọi là các thuộc tính. Nó kết hợp với một thực thể trong tập thực thể từ miền giá trị của thuộc tính. Thông thường, miền giá trị của một thuộc tính là một tập các số nguyên, các số thực, hay các xâu ký tự.

+ Một thuộc tính hay một tập thuộc tính mà giá trị của nó xác định duy nhất mỗi thực thể trong tập các thực thể được gọi là khoá đối với tập thực thể này.

+ Mỗi một thuộc tính nhận tập số các giá trị nhất định được gọi là domain của thuộc tính đó.

- *Một quan hệ (Relation):* Định nghĩa một cách đơn giản, một quan hệ là một bảng dữ liệu có các cột là các thuộc tính và các hàng là các bộ dữ liệu cụ thể của quan hệ.

- *Các liên kết:* Một liên kết là một sự kết hợp giữa một số thực thể (hay quan hệ). Ví dụ: Mối liên kết giữa phòng ban và nhân viên thể hiện: Một nhân viên A sẽ thuộc một phòng ban B nào đó.

+ Các liên kết một – một: đây là dạng liên kết đơn giản, liên kết trên hai thực thể là một – một, có nghĩa là mỗi thực thể trong tập thực thể này có nhiều nhất một thực thể trong tập thực thể kia kết hợp với nó và ngược lại.

+ Các liên kết một – nhiều: Trong một liên kết một – nhiều, một thực thể trong tập thực thể A được kết hợp với không hay nhiều thực thể trong tập thực thể B. Nhưng mỗi thực thể trong tập thực thể B được kết hợp với nhiều nhất một thực thể trong tập thực thể A.

+ Các liên kết nhiều – nhiều: Đây là dạng liên kết mà mỗi thực thể trong tập thực thể này có thể liên kết với không hay nhiều thực thể trong tập thực thể kia và ngược lại.

Ví dụ 1.1. Các mối liên kết giữa các thực thể:

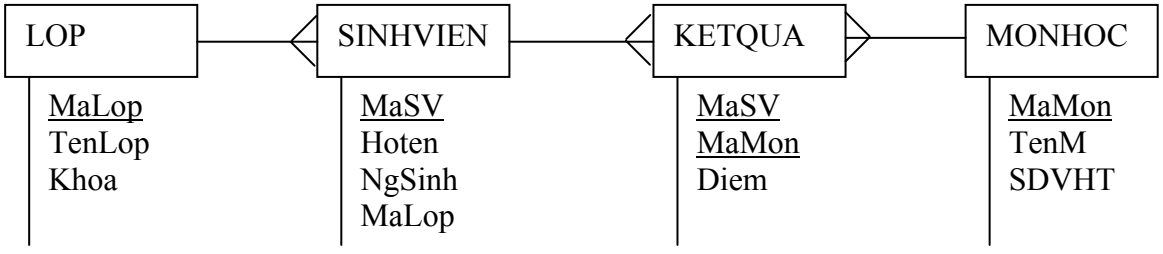
LOP(MaLop, TenLop, Khoa),

SINHVIEN(MaSV, Hoten, NgSinh, MaLop),

MONHOC(MaMon, TenM, SDVHT) và

KETQUA (MaSV, MaMon, Diem)

Ta có mối quan hệ giữa các thực thể đó là:



- *Mô hình dữ liệu quan hệ:* Làm việc trên bảng hay trên quan hệ trong đó: Mỗi cột là một thuộc tính, mỗi dòng là một bộ (một bản ghi).

+ Các ưu điểm của mô hình dữ liệu quan hệ
    - Cấu trúc dữ liệu dễ dùng, không cần hiểu biết sâu về kỹ thuật cài đặt.
    - Cải thiện tính độc lập dữ liệu và chương trình.
    - Cung cấp ngôn ngữ thao tác phi thủ tục.
    - Tối ưu hoá cách truy xuất dữ liệu.
    - Tăng tính bảo mật và toàn vẹn dữ liệu.
    - Cung cấp các phương pháp thiết kế có hệ thống. Và mở ra cho nhiều loại ứng dụng (lớn và nhỏ).

+ *Khoá của quan hệ:*
    - Khoá của quan hệ (key): Là tập các thuộc tính dùng để phân biệt hai bộ bất kỳ trong quan hệ.
    - Khoá ngoại của quan hệ (Foreign Key): Một thuộc tính được gọi là khoá ngoại của quan hệ nếu nó là thuộc tính không khoá của quan hệ này nhưng là thuộc tính khoá của quan hệ khác.

1.5.2. Các chức năng của hệ quản trị CSDL quan hệ

Các chức năng của hệ quản trị CSDL quan hệ có thể được phân thành các tầng chức năng như hình 1.3:

- *Tầng giao diện (Interface layer):* Quản lý giao diện với các ứng dụng. Các chương trình ứng dụng CSDL được thực hiện trên các khung nhìn (view) của CSDL. Đối với một ứng dụng, khung nhìn rất có ích cho việc biểu diễn một hình ảnh cụ thể về CSDL (được dùng chung bởi nhiều ứng dụng).

Khung nhìn quan hệ là một quan hệ ảo, được dẫn xuất từ các quan hệ cơ sở (base relation) bằng cách áp dụng các phép toán đại số quan hệ.

Quản lý khung nhìn bao gồm việc phiên dịch câu vấn tin người dùng trên dữ liệu ngoài thành dữ liệu khái niệm. Nếu câu vấn tin của người dùng được diễn tả bằng các phép toán quan hệ, câu vấn tin được áp dụng cho dữ liệu khái niệm vẫn giữ nguyên dạng này.

- *Tầng điều khiển (Control Layer):* chịu trách nhiệm điều khiển câu vấn tin bằng cách đưa thêm các vị từ toàn vẹn ngữ nghĩa và các vị từ cấp quyền.

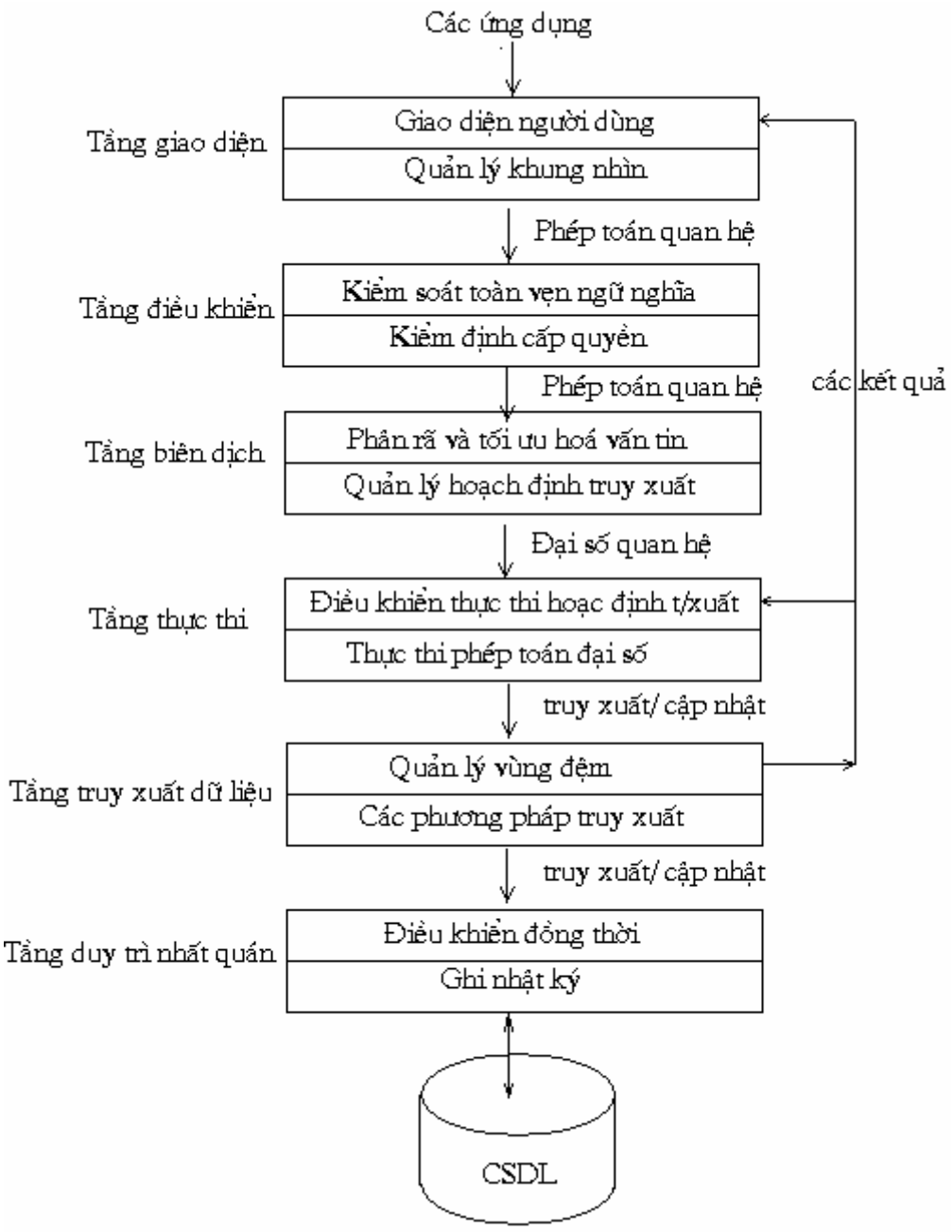
- *Tầng xử lý vấn tin (Query processing layer):* chịu trách nhiệm ánh xạ câu vấn tin thành chuỗi thao tác đã được tối ưu ở mức thấp hơn.

Tầng này liên quan đến vấn đề hiệu năng. Nó phân rã câu vấn tin thành một cây biểu thị các phép toán đại số quan hệ và thử tìm ra một thứ tự "tối ưu" cho các phép toán này. Kết xuất của tầng này là câu vấn tin được diễn tả bằng đại số quan hệ hoặc một dạng mã ở mức thấp.

- *Tầng thực thi (Execution layer):* Có trách nhiệm hướng dẫn việc thực hiện các hoạch định truy xuất, bao gồm việc quản lý giao dịch (uỷ thác, tái khởi động) và động bộ hoá các phép đại số quan hệ. Nó thông dịch các phép toán đại số quan hệ bằng cách gọi tầng truy xuất dữ liệu qua các yêu cầu truy xuất và cập nhật.

- *Tầng truy xuất dữ liệu (data access layer):* Quản lý các cấu trúc dữ liệu dùng để cài đặt các quan hệ (tập tin, chỉ mục). Nó quản lý các vùng đệm bằng cách lưu tạm các dữ liệu thường được truy xuất đến nhiều nhất. Sử dụng tầng này làm giảm thiểu việc truy xuất đến đĩa.

- *Tầng duy trì nhất quán (Consistency layer):* chịu trách nhiệm điều khiển các hoạt động đồng thời và việc ghi vào nhật ký các yêu cầu cật nhật. Tầng này cũng cho phép khôi phục lại giao dịch, hệ thống và thiết bị sau khi bị sự cố.



Hình 1.3. Các chức năng của hệ quản trị CSDL quan hệ

Chương 2: CÁC CÂU LỆNH SQL CƠ BẢN

Ngôn ngữ SQL (Structured Query Language) là ngôn ngữ truy vấn có cấu trúc, dùng để thao tác với dữ liệu trong cơ sở dữ liệu cũng như tạo và thay đổi cấu trúc của các cơ sở dữ liệu. Trong chương này ta sẽ trình bày một số câu lệnh SQL cơ bản.

2.1. CÁC CÂU LỆNH ĐỊNH NGHĨA DỮ LIỆU

2.1.1. Lệnh CREATE

- *Ý nghĩa:* Lệnh CREATE dùng để tạo các đối tượng cơ sở dữ liệu như các bảng, các view, các tệp chỉ số .v.v…

- *Cú pháp:*

+ CREATE TABLE <Tên bảng>(<Danh sách: Tên_cột Kiểu_cột> <Điều_kiện_kiểm_soát_dl >)
+ CREATE VIEW <Tên View>(<Danh sách: Tên_cột Kiểu_cột> <Điều_kiện_kiểm_soát_dl >) AS Q; với Q là một khối câu lệnh SELECT định nghĩa khung nhìn (view).
+ CREATE [UNIQUE] INDEX <tên chỉ số> ON <Ten bảng>(Tên cột [ASC|DESC])

- *Một số kiểu dữ liệu:* Integer - số nguyên; float- dấu phảy động; char - ký tự, datetime- ngày tháng, boolean,…

Ví dụ 2.1. Sử dụng câu lệnh CREATE.

+ CREATE TABLE S (S# Integer NOT NULL, SNAME Char(30), STATUS Integer, CITY Char(50)) PRIMARY KEY (S#);
+ CREATE VIEW vieS (S# Integer NOT NULL, SNAME Char(30)) AS SELECT S#, SNAME FROM S;
+ CREATE TABLE PHONGBAN (MaPB Char(5) NOT NULL, TenPB Char(30)) PRIMARY KEY (MaPB);

+ CREATE TABLE NHANVIEN (MaNV Char(5) NOT NULL, TenNV Char(30), Ngaysinh date, MaPB Char(5)) PRIMARY KEY (MaNV) FOREIGN KEY (MaPB) REFERENCES TO PHONGBAN;

+ CREATE INDEX index1 ON SP(S# ASC, P# DESC)

2.1.2. Lệnh thay thế sửa đổi ALTER

- *Ý nghĩa:* Dùng để thay đổi cấu trúc lược đồ của các đối tượng CSDL.

- *Cú pháp:*

+ ALTER TABLE <Tên bảng> <Thực hiện các lệnh trên cột>

Các lệnh trên cột có thể là:

- Xóa một cột: Delete <tên cột>
- Thêm một cột: Add <Tên cột>
- Thay đổi tên cột: Change column <Tên cột>To<Tên cột>
- Xóa khóa chính: Drop PRIMARY KEY
- Xóa khóa ngoại: Drop FOREIGN KEY
- Thiết lập khóa chính: PRIMARY KEY (Tên cột)
- Thiết lập khóa ngoại:

FOREIGN KEY (Tên cột) REFERENCES TO <tên bảng ngoài>

+ ALTER VIEW <Tên View>(<Danh sách: Tên_cột Kiểu_cột> <Điều_kiện_kiểm_soát_dl >) AS Q; với Q là một khối câu lệnh SELECT định nghĩa khung nhìn (view).

Ví dụ 2.2. Thay đổi cấu trúc của bảng NHANVIEN

ALTER TABLE NHANVIEN Add Quequan char(50);

ALTER TABLE NHANVIEN Delete Ngaysinh;

Ví dụ 2.3. Thay đổi khung nhìn vieS

CREATE VIEW vieS AS SELECT S#, SNAME, CITY FROM S

2.1.3. Xoá cấu trúc DROP

- *Ý nghĩa:* Dùng để xóa các đối tượng cơ sở dữ liệu như Table, View, Index, .v.v…

- *Cú pháp:*

DROP TABLE <Tên bảng>

DROP VIEW <Tên view>

DROP INDEX <Tên index>

2.2. CÁC CÂU LỆNH CẬP NHẬT DỮ LIỆU

2.2.1. Lệnh Insert

- *Ý nghĩa:* Dùng để chèn một hàng hoặc một số hàng cho bảng.

- *Cú pháp:*

+ INSERT INTO <Tên bảng> (Danh sách các cột) VALUES (Danh sách các giá trị) hoặc
+ INSERT INTO <Tên bảng> (Danh sách các cột) (Các câu hỏi con);

Ví dụ 2.4. Chèn dữ liệu vào bảng S.

```
INSERT INTO S (S#, SNAME, STATUS)
    VALUES (s1, Smith, 20, Paris);

INSERT INTO S
    SELECT * FROM W WHERE CITY="Paris";
```

2.2.2. Lệnh Update

- *Ý nghĩa:* Dùng để sửa đổi dữ liệu.

- *Cú pháp:*

```
UPDATE  <Tên bảng>
    SET <Tên_cột_1=Biểu_thức_1, Tên_cột_2=Biểu_thức_2,… >
```

[WHERE <điều kiện>]

Ví dụ 2.5. Sử dụng lệnh Update

```
UPDATE SINHVIEN SET TenSV="Nguyễn Thị Hạnh"
    Where MaSV="20042390"

UPDATE HangHoa SET Dongia=Dongia*1.1
    Where TenHH IS LIKE 'Bia %' and Dongia<3500
```

2.2.2. Lệnh Delete

- *Ý nghĩa:* Xoá một số hàng trong bảng.

- *Cú pháp:*

DELETE FROM <Tên bảng> WHERE <Điều kiện>

Ví dụ 2.6. Xoá tất cả các hàng trong bảng KETQUA có trường Diem<5

```
DELETE FROM KETQUA WHERE Diem<5;
```

2.3. KIỂM SOÁT DỮ LIỆU

2.3.1. Trao quyền GRANT

- *Ý nghĩa:* Dùng để trao quyền cho một acount nào đó.

- *Cú pháp:*

GRANT <Quyền> ON <Tên bảng/ Tên View> TO <user>
[WITH GRANT OPITION]

Các quyền có thể trao là: All, Select, update, delete, insert, index, alter, read, write,…

User có thể là: Public, tên một user cụ thể,…

- *Chú ý:* Nếu được trao quyền với chỉ định WITH GRANT OPITION thì anh ta có thể trao lại quyền ấy cho người khác.

Ví dụ 2.7. Trao quyền Select cho acount Lannt

```
GRANT Select ON SINHVIEN TO Lannt WITH GRANT OPITION
```

2.3.2. Thu hồi quyền REVOTE

- *Ý nghĩa:* Dùng để thu hồi quyền của một acount nào đó.

- *Cú pháp:*

REVOTE <Quyền> ON <Tên bảng/ Tên View> FROM <user>

Ví dụ 2.8. Thu hồi quyền Select của acount Lannt

REVOTE Select ON SINHVIEN FROM Lannt;

2.4. TRUY VẤN DỮ LIỆU

Khối câu lệnh phổ dụng: SELECT - FROM – WHERE. Ta có thể sử dụng theo cú pháp chung như sau:

SELECT [*| DISTINCT] <Danh sách các cột [AS <Bí danh>]>
FROM <Danh sách Tên bảng/Tên View>
[WHERE <Biểu thức điều kiện>]
[GROUP BY <Danh sách cột>]
[HAVING <Điều kiện>]
[ORDER BY <Tên cột/ Số thứ tự cột/Biểu thức> [ASC/DESC]]

2.4.1. Tìm kiếm theo câu hỏi đơn giản

- *Tìm kiếm đơn giản:*

+ Nếu xuất hiện giá trị * nghĩa là xem toàn bộ các cột của bảng.

Select * From SINHVIEN;

+ Nếu sử dụng DISTINCT thì sẽ lấy giá trị đại diện.

Select Distinct S#, P# From SP;

- *Xử lý xâu:* dùng toán tử [NOT] LIKE <Mẫu so sánh>

+ Dùng dấu gạch dưới để thay cho một ký tự.

+ Dùng dấu % để thay cho một dãy các ký tự tuỳ ý.

Ví dụ 2.9. Cho bảng hồ sơ sinh viên HOSOSV(MaSV, Hodem, TenSV, Ngaysinh, MaLop). Hãy cho biết mã và họ tên sinh viên có hai chữ đầu là 'Ba'

Select MaSV, Hodem+TenSV as Hoten
From HOSOSV

```
Where TenSV like 'Ba%'
```

Ví dụ 2.10. Cho bảng hồ sơ sinh viên HOSOSV(MaSV, Hodem, TenSV, Ngaysinh, MaLop). Hãy cho biết mã và họ tên sinh viên có không có hai chữ đầu là 'Ba'

```
Select MaSV, Hodem+TenSV as Hoten
    From HOSOSV
    Where TenSV like 'Ba%'
```

- Sử dụng Between và IN để xác định phạm vi:

Ví dụ 2.11. Cho bảng thông tin sách mượn SACHMUON(MaBD, MaSach, NgayMuon, NgayTra). Hãy cho biết mã các bạn đọc mượn sách của thư viện trong khoảng ngày {1/1/2008} và {31/3/2008}

```
Select MaBanĐoc
    From SACHMUON
    Where NgayMuon Between {1/1/2000} and {31/3/2000}
```

2.4.2. Sử dụng các hàm thư viện

Các hàm thư viện thực hiện các thao tác như thống kê dữ liệu, tính toán dữ liệu có sẵn như:

- Count(): Dùng để đếm các bảng ghi,
- Max(): Trả về giá trị lớn nhất của một tập hợp các giá trị,
- Min(): Trả về giá trị lớn nhất của một tập hợp các giá trị,
- Sum(): Trả về tổng giá trị của một tập hợp các giá trị,
- Avg():Trả về giá trị trung bình của một tập hợp các giá trị,

Ví dụ 2.12.: Cho bảng DIEM(MaSV, MaMH, DiemL1, DiemL2). Hãy xem sinh viên có mã SV061001 đã tham gia thi bao nhiêu môn:

```
Select Count(MaMH)  AS Tongso
    From DIEM
    where MaSV='SV061001';
```

Ví dụ 2.13: Cho biết điểm thi cao nhất lần 1 của môn có mã '03AB'

```
Select Max(DiemL1) as DiemCN
    From DIEM
    Where MaMH='03AB';
```

Ví dụ 2.14. Cho biết chênh lệnh giữa điểm thi cao nhất và thấp nhất của môn có mã môn học là '03AB'

```
Select (Max(DiemL1) - Min (DiemL1)) As Chenh_Lenh
From DIEM
where MaMT='03AB';
```

2.4.3. Tìm kiếm nhờ các mệnh đề

- *Sử dụng phân nhóm GROUP BY:* Mệnh đề GROUP BY được sử dụng để tạo hiệu quả sắp xếp và tính toán theo từng phân nhóm.

Ví dụ 2.15. Cho biết tình hình thi của từng sinh viên:

```
Select MaSV, MaMH, DiemL1, DiemL2
    From DIEM
    Group By MaSV
```

- *Sử dụng HAVING:* Mệnh đề HAVING dùng để đặt điều kiện lọc cho các phân nhóm con.

Ví dụ 2.16. Cho bảng mặt hàng đã được cung cấp SP(S#, P#, QTY). Tìm mã những nhà cung cấp cung cấp ít nhất 2 mặt hàng:

```
Select S#  From SP
    Group By S#
    Having Count(Distinct P#)>=2;
```

Ví dụ 2 .17. Tìm mã các sinh viên không có môn thi nào dưới 5

```
Select MaSV From DIEM
    Group By MaSV
     Having Min(Diem)>=5;
```

Chú ý: Having đi sau Group By để đặt điều kiện chọn lọc ra những phân nhóm thoả mãn điều kiện sau Having. Nếu không có từ khoá Group By thì Having sẽ tác động trên toàn bảng coi như một phân nhóm duy nhất.

- *Sử dụng Order By:* Được sử dụng để tạo hiệu quả sắp xếp dữ liệu. Ta có thể sắp xếp theo chiều tăng (ASC) hoặc giảm (DESC).

+ Ta có thể tác động sắp xếp lại trên từng phân nhóm bởi Order By.

Ví dụ 2.18. Cho biết tình hình thi lần 1 của mỗi sinh viên sao cho kết quả điểm thi được sắp xếp giảm dần.

```
SELECT MaSV, MaMT, ĐiemL1
    GROUP BY MaSV
    ORDER BY Diem DESC
```

- *Chú ý:* Tương tự HAVING, nếu trước Order by không có Group By thì hiệu quả sắp xếp dữ liệu sẽ tác động trên toàn bảng và bảng được coi như một phân nhóm chính.

2.4.4. Câu hỏi phức tạp

Khi thực hiện các truy vấn làm việc với dữ liệu từ 2 bảng trở nên thì điều kiện xử lý phức tạp hơn.

- *Tự kết nối:*

Ví dụ 2.19. Kiểm tra bảng kết quả thi KETQUA(SoBD, MaMT, Diem) có bị nhập trùng hay không? Nghĩa là nhập trùng MaSV, MaMT nhưng điểm thi lại khác nhau (nhập 2 lần).

```
SELECT  a.SoBD, a.MaMT, a.Diem, b.Diem
From KETQUA a, KETQUA b
    Where (a.SoBD=b.SoBD) and (a.MaMT=b.MaMT) and (a.Diem>b.Diem)
```

- *Kết nối nhiều bảng:*

Ví dụ 2.20. Cho biết kết quả thi môn Toán của các sinh viên.

```
+ CREATE VIEW MaToan AS
        SELECT MaMT From MONTHI
            Where TenMon='Toán'

+ CREATE VIEW TAM AS
        SELECT a.SoBD, a.Diem
            From KETQUA a, MaToan b
```

```
Where a.MaMT=b.MaMT;
```

+ SELECT a.SoBD, TenSV, a.Diem
From Tam a, THISINH b
Where a.SoBD=b.SoBD

Ví dụ 2.21. Liên kết nhiều bảng authors, titleauthor và title.

```
SELECT au_lname, au_fname, title, price FROM authors
    JOIN titleauthor ON authors.au_id = titleauthor.au_id
    JOIN titles ON titleauthor.title_id = titles.title_id
    ORDER BY au_lname, au_fname
```

- *Ánh xạ lồng*

Ví dụ 2.22. Cho các quan hệ:

S(S#, SNAME, STATUS, CITY)
SP(S#, P#, QTY)
P(P#, PNAME, COLOR, WEIGH)

Hãy cho biết mã và tên các hãng có bán sản phẩm màu đỏ.

```
SELECT S#, SNAME From S Where S# IN
    (SELECT S# From SP Where P# IN
        (Select P# From P Where COLOR='Red'));
```

- *Sử dụng các lượng từ: EXISTS, ANY, ALL,…*

Ví dụ 2.23. Cho các bảng trong ví dụ 2.22. Tìm các nhà cung cấp đã cung cấp ít nhất một mặt hàng nào đó.

```
SELECT * From S Where EXISTS
    (SELECT * From SP Where SP.S# =S.S#);
```

Ta có thể thay thế bằng câu lệnh:

```
SELECT * From S
    Where 0 < (SELECT Count(*) From SP Where SP.S#=S.S#);
```

Ví dụ 2.24. Tìm tên những mặt hàng có mã số mặt hàng mà mặt hàng nào đó mà hãng S1 đã bán.

```
SELECT PNAME From P Where S# = ANY
```

(SELECT P# From SP Where S#='S1');

Ví dụ 2.25. Tìm những hãng cung cấp số lượng một lần một mặt hàng nào đó > số lượng mỗi lần của các hãng cung cấp.

```
SELECT S From SP
    Where QTY>= ALL (SELECT QTY From SP ); hay
SELECT S From SP Where QTY=
    (SELECT Max(QTY) From SP );
```

Chương 3: HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER

3.1. TỔNG QUAN VỀ HỆ QUẢN TRỊ SQL SERVER

3.1.1. Giới thiệu hệ quản trị SQL Server

Microsoft SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (Relation Database Management System - RDBMS), cung cấp cách tổ chức dữ liệu bằng cách lưu chúng vào các bảng. Dữ liệu quan hệ được lưu trữ trong các bảng và các quan hệ đó được định nghĩa giữa các bảng với nhau.

Người dùng truy cập dữ liệu trên Server thông qua ứng dụng. Người quản trị CSDL truy cập Server trực tiếp để thực hiện các chức năng cấu hình, quản trị và thực hiện các thao tác bảo trì CSDL.

Ngoài ra, SQL Server là một CSDL có khả năng mở rộng, nghĩa là chúng có thể lưu một lượng lớn dữ liệu và hỗ trợ tính năng cho phép nhiều người dùng truy cập dữ liệu đồng thời.

Các phiên bản của SQL Server phổ biến hiện này trên thị trường là SQL Server 7.0, SQL Server 2000, SQL Server 2005, SQL Server 2008. Trong giáo trình này, tác giả giới thiệu với các bạn trên hai phiên bản SQL Server 2000, SQL Server 2005.

3.1.2.Các thành phần của SQL Server

3.1.2.1. Các thành phần của SQL Server 2000

SQL Server cung cấp một số loại thành phần khác nhau:

- Nhân của nó là các thành phần server, các thành phần này được thực hiện như windows 32 bit.
- Các công cụ đồ họa dựa trên client và các dòng tiện ích phục vụ cho công tác quản trị. Các công cụ và tiện ích này sử dụng sử dụng các thành phần giao tiếp client do SQL Server 2000 cung cấp. Các thành phần giao tiếp cung cấp các cách khác nhau mà

trong đó ứng dụng client có thể truy cập dữ liệu thông qua các thành phần server.

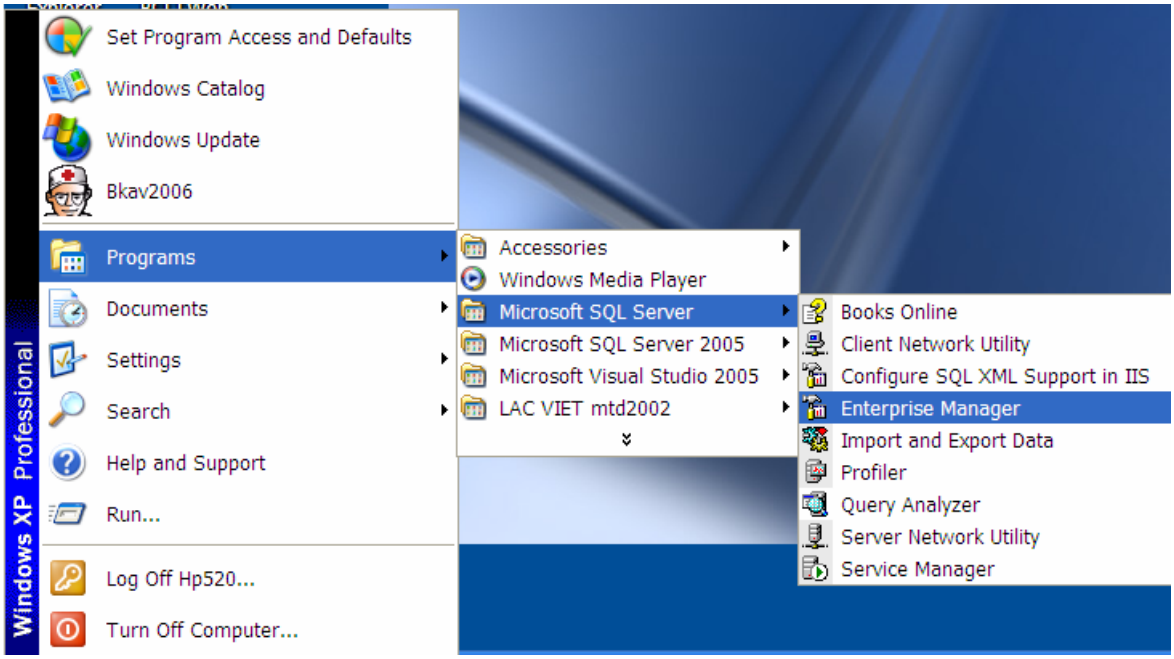
a) Các thành phần server

Các thành phần server của SQL Server 2000 thường được thực hiện như các dịch vụ Windows 32 bit. Do đó các dịch vụ của SQL Server và SQL Server Agent có thể chạy như các ứng dụng độc lập trên bất kỳ nền tảng nào được hỗ trợ hệ điều hành Windows. Các thành phần Server được mô tả trong bảng 3.1:

Bảng 3.1. Các thành phần server của SQL Server 2000

Thành phần server	***Chức năng***
Dịch vụ SQL Server	Dịch vụ MSSQLServer thực thi cỗ máy CSDL SQL Server 2000. Có một dịch vụ này cho mỗi thể hiện của SQL Server 2000.
Dịch vụ về các dịch vụ phân tích của SQL Server 2000	MSSQLServerOLAPService thực thi các dịch vụ phân tích của SQL Server 2000. Chỉ có một dịch vụ không liên quan đến số thể hiện của SQL Server 2000.
Dịch vụ SQL Server Agent	Dịch vụ SQLServerAgent thực thi các tác nhân chạy các tác vụ quản trị đã được định thời gian biểu của SQL Server 2000.
Dịch vụ tìm kiếm	Dịch vụ tìm kiếm Microsoft thực thi cỗ máy tìm kiếm toàn văn bản. Chỉ có một dịch vụ, không liên quan đến số thể hiện của SQL Server 2000.
Dịch vụ MS DTC - Distributed Transaction Coordinator	Thành phần điều phối giao dịch phân tán, quản lý các giao dịch phân tán giữa các thể hiện của SQL Server 2000. Chỉ có một dịch vụ, không liên quan đến số thể hiện của SQL Server 2000.

SQL Server 2000 cung cấp các giao diện đồ họa giúp cho người sử dụng sử dụng các dịch vụ của SQL Server 2000 (Hình 3.1. và Bảng 3.2).



Hình 3.1. Các giao diện đồ họa của SQL Server 2000.

Bảng 3.2 Các công cụ giao diện đồ họa của SQL Server 2000.

Công cụ	***Chức năng***
SQL Server Enterprise Manager	Đây là công cụ quản trị CSDL server chính, nó cung cấp một giao tiếp với người dùng Microsoft Management Console (MMC).
SQL Query Analyzer	Dùng để tạo và quản lý các đối tượng CSDL và kiểm tra các phát biểu Transact-SQL, các bó lệnh và các script một cách tương tác.
SQL Profiler	Giám sát và ghi nhận các sự kiện SQL Server 2000 đã chọn để phân tích và xem lại.
SQL Server Service Manager	Ứng dụng nằm trên thanh task bar của Windows được dùng để chạy, tạm dừng hoặc thay đổi các dịch vụ SQL Server 2000.
Client Network	Được dùng để quản lý Net-Libraries của client và

Utility	định nghĩa các bí danh server.
Server Network Utility	Dùng để quản lý Net-Libraries của server bao gồm thiết lập mã hóa SSL.

b) Các thành phần giao tiếp Client.

Người dùng truy cập SQL Server 2000 thông qua các ứng dụng client, SQL Server 2000 cung cấp hai kiểu ứng dụng client chính:

- Các ứng dụng CSDL quan hệ, là kiểu ứng dụng truyền thống dùng môi trường client/server 2 lớp. Các ứng dụng này gửi các phát biểu T-SQL đến cỗ máy CSDL quan hệ và nhận kết quả trả về như tập kết quả quan hệ.
- Các ứng dụng Internet, chúng là thành phần của nền tảng Microsoft.NET. Chúng gửi các phát biểu T-SQL hoặc các truy vấn Xpath tới cỗ máy CSDL quan hệ và nhận về kết quả dạng XML.

Các tiện ích dòng lệnh thường được sử dụng do SQL Server 2000 cung cấp cho trong bảng 3.3.

Bảng 3.3 Các tiện ích dòng lệnh của SQL Server 2000.

Tiện ích	***Chức năng***
Osql	Tiện ích này cho phép truy vấn tương tác một thể hiện của SQL Server 2000 bằng các phát biểu T-SQL, các thủ tục và các script.
Scm (Server Control Manager)	Dùng để chạy, dừng, tạm dừng, cài đặt, xóa hoặc thay đổi các dịch vụ SQL Server 2000.
Sqldiag	Tiện ích này thu thập và lưu trữ các thông tin chuẩn đoán để xử lý và đơn giản hóa thông tin thu thập bởi dịch vụ hỗ trợ sản phẩm Microsoft.
Bcp	Tiện ích này sao chép dữ liệu giữa một thể hiện của SQL Server 2000 và tập tin dữ liệu theo định dạng của người dùng.

Dtsrun	Tiện ích này thực thi các gói được tạo bởi DTS.
Sqlmaint	Tiện ích này thực thi các hoạt động bảo trì trên một hoặc nhiều CSDL. Những hoạt động bao gồm việc kiểm tra tính nhất quán dữ liệu, sao lưu tập tin dữ liệu và tập tin giao dịch, cập nhật các thống kê phân tán, xây dựng lại các chỉ mục.

3.1.2.2. Các thành phần của SQL Server 2005

SQL Server 2005 nâng cao hiệu năng, độ tin cậy, khả năng lập trình đơn giản và giao diện dễ sử dụng hơn so với SQL Server 2000. SQL Server 2005 tập trung vào khả năng xử lý giao dịch trực tuyến (online transaction processing - OLTP), ứng dụng thương mại điện tử (e- ecommerce) và kho dữ liệu (data warehousing). Ngoài ra những cải tiến quan trọng trong SQL Server 2005 là thêm các dịch vụ mới như: dịch vụ báo cáo (reporting service), service broker và sự thay đổi đáng kể trong cỗ máy cơ sở dữ liệu.

a) Các phiên bản của SQL Server 2005:

Trước khi đi vào các thành phần của SQL Server 2005, ta xét các phiên bản của SQL Server 2005. SQL Server 2005 được sử dụng rộng rãi cho nhiều đối tượng khác nhau nên Microsoft cung cấp nhiều phiên bản khác nhau cho phù hợp với các yêu cầu về chi phí, thời gian thực hiện, của các tổ chức, cá nhân. Năm phiên bản của SQL Server 2005 là:

+ Microsoft SQL Server 2005 Enterprise Edition
+ Microsoft SQL Server 2005 Standard Edition
+ Microsoft SQL Server 2005 Workgroup Edition
+ Microsoft SQL Server 2005 Developer Edition
+ Microsoft SQL Server 2005 Express Edition

Hầu hết các tổ chức đều chọn trong ba phiên bản SQL Server 2005 Enterprise Edition, SQL Server 2005 Standard Edition, và SQL Server 2005 Workgroup Edition. Các tổ chức chọn một trong ba phiên bản này với lý do là

chỉ có các phiên bản Enterprise, Standard, và Workgroup được cài đặt và sử dụng trong môi trường server phục vụ cho hoạt động thực tế.

+ ***SQL Server 2005 Enterprise Edition (32-bit và 64-bit)***

  Enterprise Edition được sử dụng trong các doanh nghiệp, tổ chức có các mức yêu cầu xử lý giao dịch trực tuyến trên diện rộng (online transaction processing - OLTP), khả năng phân tích dữ liệu phức tạp cao, hệ thống kho dữ liệu (data warehousing systems) và web sites. Enterprise Edition phù hợp cho các tổ chức lớn và các yêu cầu phức tạp.

+ ***SQL Server 2005 Standard Edition (32-bit và 64-bit)***

  Standard Edition là phiên bản phục vụ cho việc quản trị và phân tích dữ liệu phù hợp cho các doanh nghiệp, tổ chức vừa và nhỏ. Nó bao gồm các giải pháp cần thiết cho thương mại điện tử (e-commerce), kho dữ liệu (data warehousing) và dòng doanh nghiệp (line-of-business).

+ ***SQL Server 2005 Workgroup Edition (32-bit only)***

  Workgroup Edition là giải pháp quản trị dữ liệu phù hợp cho các doanh nghiệp, tổ chức nhỏ chỉ cần một cơ sở dữ liệu không giới hạn kích thước hoặc số người sử dụng. Workgroup Edition là lý tưởng cho các mức cơ sở dữ liệu tin cậy, mạnh mẽ và dễ quản trị.

+ ***SQL Server 2005 Developer Edition (32-bit và 64-bit)***

  Developer Edition có tất cả các tính năng của phiên bản SQL Server 2005 Enterprise Edition, nhưng nó chỉ là phiên bản sử dụng cho phát triển và kiểm tra ứng dụng. Phiên bản này phù hợp cho các cá nhân, tổ chức xây dựng và kiểm tra ứng dụng.

+ ***SQL Server 2005 Express Edition (32-bit only)***

  SQL Server Express, dễ sử dụng và quản trị cơ sở dữ liệu đơn giản. Được tích hợp với Microsoft Visual Studio 2005, SQL Server Express trở nên dễ dàng để phát triển các ứng dụng dữ liệu giầu khả năng, an toàn trong lưu trữ, và nhanh chóng triển khai.

SQL Server Express là phiên bản miễn phí, có thể dùng như một cơ sở dữ liệu máy khách hoặc cơ sở dữ liệu máy chủ đơn giản. SQL Server Express là lựa chọn tốt cho những người dùng chỉ cần một phiên bản SQL Server 2005 nhỏ gọn, dùng trên máy chủ có cấu hình thấp, những nhà phát triển ứng dụng không chuyên hay những người yêu thích xây dựng các ứng dụng nhỏ.

b) Các thành phần Server của SQL Server 2005:

Các thành phần server của SQL Server 2005 được cho trong bảng 3.4.

Bảng 3.4 Các thành phần server của SQL Server 2005.

Thành phần Server	***Chức năng***
SQL Server Database	Cỗ máy cơ sở dữ liệu bao gồm Database Engine, lõi dịch vụ cho việc lưu trữ, xử lý và bảo mật dữ liệu, sao lưu và đồng bộ (Replication), tìm kiếm toàn văn (Full-Text Search), và các công cụ cho việc quản trị dữ liệu quan hệ và XML.
Analysis Services	Analysis Services bao gồm các công cụ cho việc tạo và quản lý tiến trình phân tích trự tuyến (online analytical processing - OLAP) và các ứng dụng khai thác dữ liệu.
Reporting Services	Reporting Services bao gồm các thành phần server và client cho việc tạo, quản lý và triển khai các báo cáo. Reporting Services cũng là nền tảng cho việc phát triển và xây dựng các ứng dụng báo cáo.
Notification Services	Dịch vụ thông báo Notification Services là nền tảng cho sự phát triển và triển khai các ứng dụng tạo và gửi thông báo. Notification Services có thể gửi thông báo theo địch thời đến hàng ngàn người đăng ký sử dụng nhiều loại thiết bị khác nhau.

Integration Services	Integration Services là một tập hợp các công cụ đồ họa và các đối tượng lập trình cho việc di chuyển, sao chép và chuyển đổi dữ liệu.

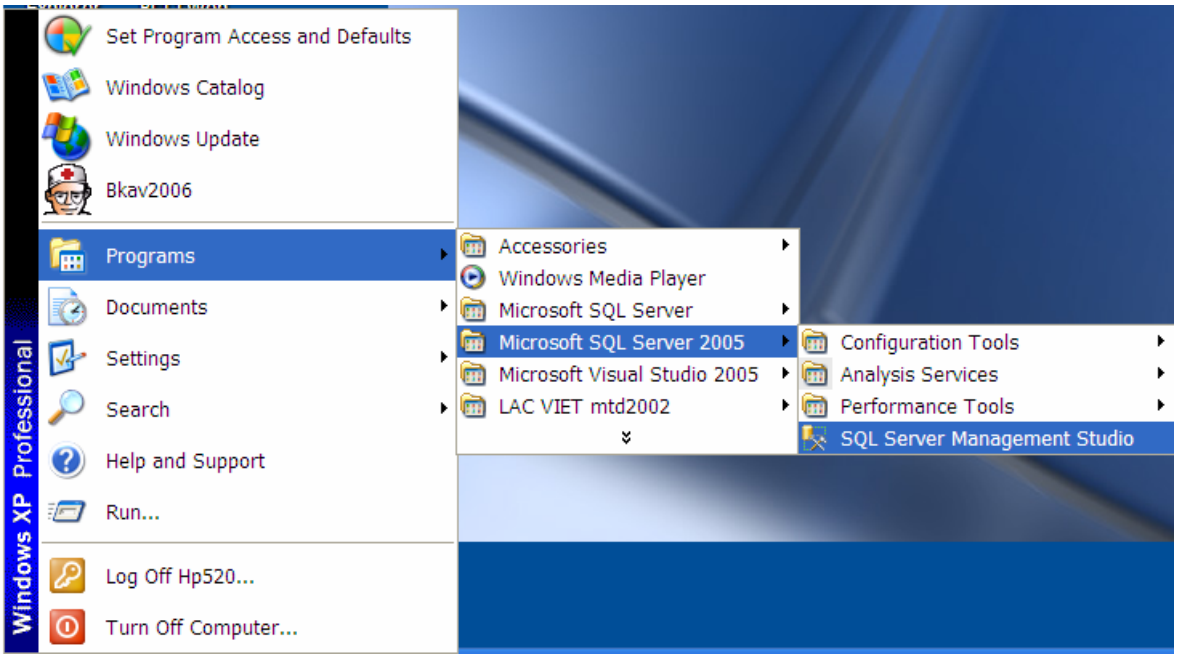
c) Các thành phần Client

Bảng 3.5 Các thành phần client của SQL Server 2005.

Thành phần Client	***Chức năng***
Connectivity Components	Là các thành phần cho việc truyền thông giữa clients và servers, và các thư viện mạng như DB-Library, ODBC, and OLE DB.

c) Các công cụ đồ họa

Các công cụ giao diện đồ họa giúp cho việc truy xuất và quản trị SQL Server được thay đổi khá nhiều so với các phiên bản trước đó, các công cụ quản trị đó được cho trong bảng 3.6. và hình 3.2.



Hình 3.2. Các giao diện đồ họa của SQL Server 2005.

Bảng 3.5 Các công cụ quản trị trên SQL Server 2005.

Management tools	*Chức năng*
SQL Server Management Studio	SQL Server Management Studio (SSMS), là công cụ mới trên Microsoft SQL Server 2005, nó là một môi trường được tích hợp cho việc truy xuất, cấu hình, quản trị và phát triển tất cả các thành phần của SQL Server. SSMS kết hợp các tính năng của Enterprise Manager, Query Analyzer, và Analysis Manager, được bao hàm trong các phiên bản trước của SQL Server, thành một môi trường đơn mà cung cấp truy xuất SQL Server để phát triển và quản trị tất cả các mức kỹ năng trên.
SQL Server Configuration Manager	SQL Server Configuration Manager cung cấp các quản trị cấu hình cơ sở cho các dịch vụ SQL Server (SQL Server services), các giao thức server (server protocols), các giao thức client (client protocols) và các bí danh client (client aliases).
SQL Server Profiler	SQL Server Profiler cung cấp giao diện người dùng đồ họa cho việc giám sát thể hiện của Database Engine hoặc thể hiện của Analysis Services.
Database Engine Tuning Advisor	Database Engine Tuning Advisor cố vấn, giúp tạo các tập tối ưu các chỉ số (indexes), indexed views, và các phân vùng (partitions).

3.1.3. Quản lý các dịch vụ của SQL Server.

3.1.3.1. Quản lý các dịch vụ của SQL Server 2000

a) Sử dụng SQL Server Service Manager

Dịch vụ là một chương trình hay một tiến trình thực hiện một chức năng cụ thể để hỗ trợ chương trình khác. SQL Server 2000 cung cấp các dịch vụ:

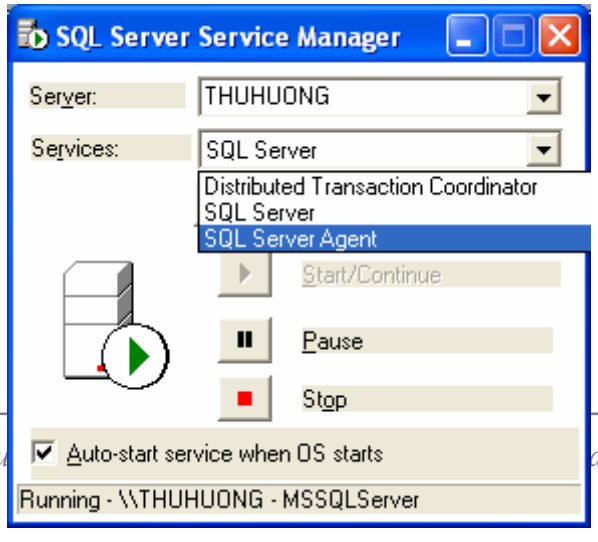
- *Dịch vụ SQL Server:* Khi khởi động SQL Server thì dịch vụ SQL Server được khởi động trên Windows. Dịch vụ này quản lý các

tập tin CSDL, xử lý các phát biểu Transaction – SQL, cấp phát tài nguyên giữa các kết nối người dùng đồng thời, đảm bảo tính nhất quán dữ liệu,.v.v…

- *Dịch vụ SQL Server Agent:* Dịch vụ này hỗ trợ lập các chương trình, thực thi tác vụ, cảnh báo, thông báo và kế hoạch bảo trì CSDL. Nó cho phép ta thực hiện tự động hóa các tác vụ bảo trì CSDL.
- *Dịch vụ Distributed Transaction Coordinator:* Là trình quản lý giao dịch cung cấp các khả năng bao gồm nhiều nguồn dữ liệu khác nhau kể cả các CSDL từ xa trong các giao dịch ứng dụng.

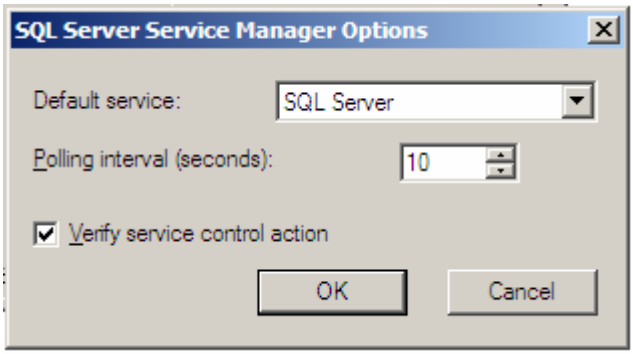
Khởi động hoặc dừng các dịch vụ SQL Server sử dụng trình SQL Server Service Manager thực hiện theo các bước sau:

1. Vào *start/Programs/Microsoft SQL Server/Service Manager*; Hoặc dưới góc phải của màn hình trên thanh task bar hệ thống của windows, double–click vào biểu tượng Service Manager. Khi đó cửa sổ SQL Server Service Manager xuất hiện như hình 3.3.
2. Trong danh sách Server chọn tên server và danh sách Service chọn dịch vụ SQL Server.
3. Click nút *Start/Continue* để khởi chạy dịch vụ; Click nút *Pause* để tạm dừng dịch vụ, tạm dừng dịch vụ để ngăn chặn người dùng đăng nhập vào SQL Server và có thời gian cho người dùng đang kết nối có thời gian hoàn tất các tác vụ và thoát khỏi SQL Server trước khi ta đóng SQL Server; Click vào nút *Stop* để dừng dịch vụ.



Hình 3.3. Ứng dụng SQL Server Service Manager

4. Trong khi chạy *Service Manager*, trạng thái hiển thị của các dịch vụ được mặc định là 5giây, để thay đổi thay đổi thời gian cập nhật, ta click chuột vào biểu tượng ở góc trên bên trái của SQL Server Service Manager, chọn Options xuất hiện hộp thoại SQL Server Service Manager Options nhập vào khoảng thời gian kiểm soát vòng mới cho các dịch vụ, chẳng hạn là 10 (Hình 3.4).



Hình 3.4. SQL Server Service Manager Options

b) Sử dụng Enterprise Manager

Enterprise Manager là thành phần Microsoft Management Console (MMC). MMC là ứng dụng trung tâm dùng để quản lý tất cả các giao tiếp của hệ thống. Nó cho phép thực hiện các tác vụ sau:

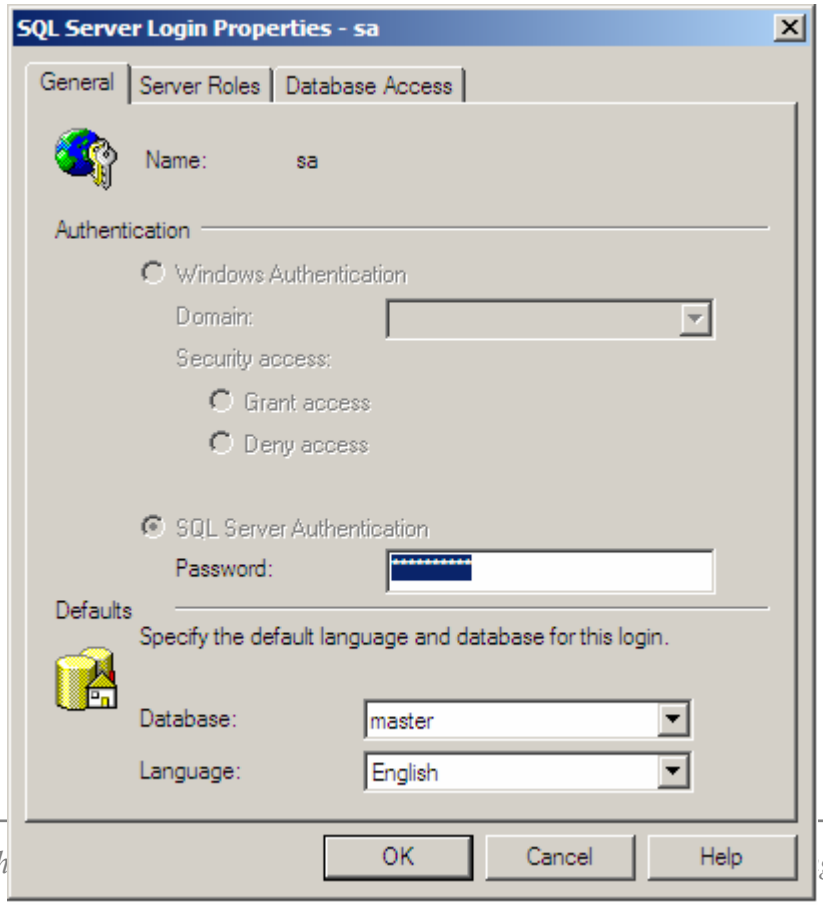
+ Khởi động, dừng, tạm dừng Server.
+ Đăng ký server,
+ Cấu hình server cục bộ và từ xa
+ Cấu hình và quản lý các thể hiện của server,
+ Thiết lập bảo mật đăng nhập, thêm người dùng, người quản trị hệ thống.
+ Gán mật khẩu cho người quản trị hệ thống,

+ Tạo và lập thời gian biểu thực thi công việc,
+ Thiết lập và quản lý CSDL, bảng, chỉ mục, view, stored proceduce, trigger, …
+ Quản lý các dịch vụ SQL server khác,…

** Thay đổi mật khẩu mặc định*

Tất cả các SQL server đều có một tài khoản quản trị mặc định sẵn là sa (system administrator). Lúc mới cài tài khoản này chưa được gán mật khẩu. Để đảm bảo mức bảo mật cao nhất cho SQL server ta phải gán cho tài khoản sa một mật khẩu. Khi gán mật khẩu ta thực hiện theo các bước sau:

1. Trong *Enterprise Manager* chọn tên server
2. Chọn *Security/Logins* để hiển thị tất cả các tài khoản người dùng.
3. Right click chuột lên tài khoản *sa*, và chọn *Properties*, xuất hiện cửa sổ SQL Server Login Properties như hình 3.5.
4. Nhập mật khẩu mới vào hộp Password sau đó click OK để hiểm hị hộp thoại Confirm Password.
5. Trong hộp thoại Confirm Password nhập lại mật khẩu trên để xác nhận lại mật khẩu và chọn OK.



Hình 3.5. Thay đổi mật khẩu tài khoản sa.

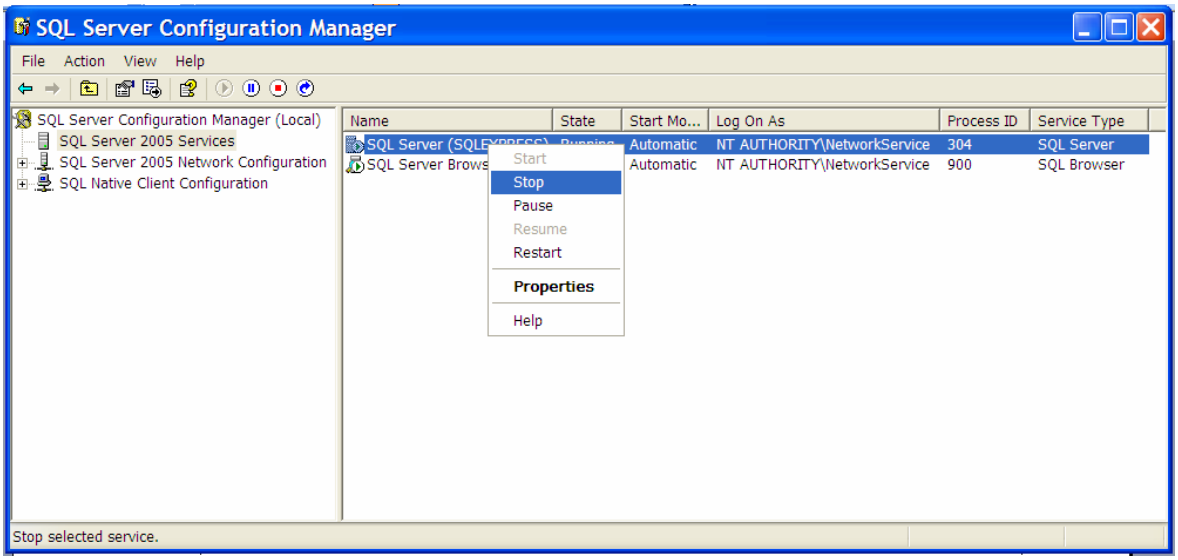
3.1.3.2. Quản lý các dịch vụ của SQL Server 2005

a) Sử dụng SQL Server Configuration Manager

SQL Server Configuration Manager là công cụ để quản lý các dịch vụ kết hợp với SQL Server, để cấu hình các giao thức mạng được sử dụng bởi SQL Server, và để quản lý cấu hình kết nối mạng từ các máy tính trạm SQL Server. SQL Server Configuration Manager kết hợp các chức năng của các công cụ trong phiên bản SQL Server 2000 là: Server Network Utility, Client Network Utility, và Service Manager.

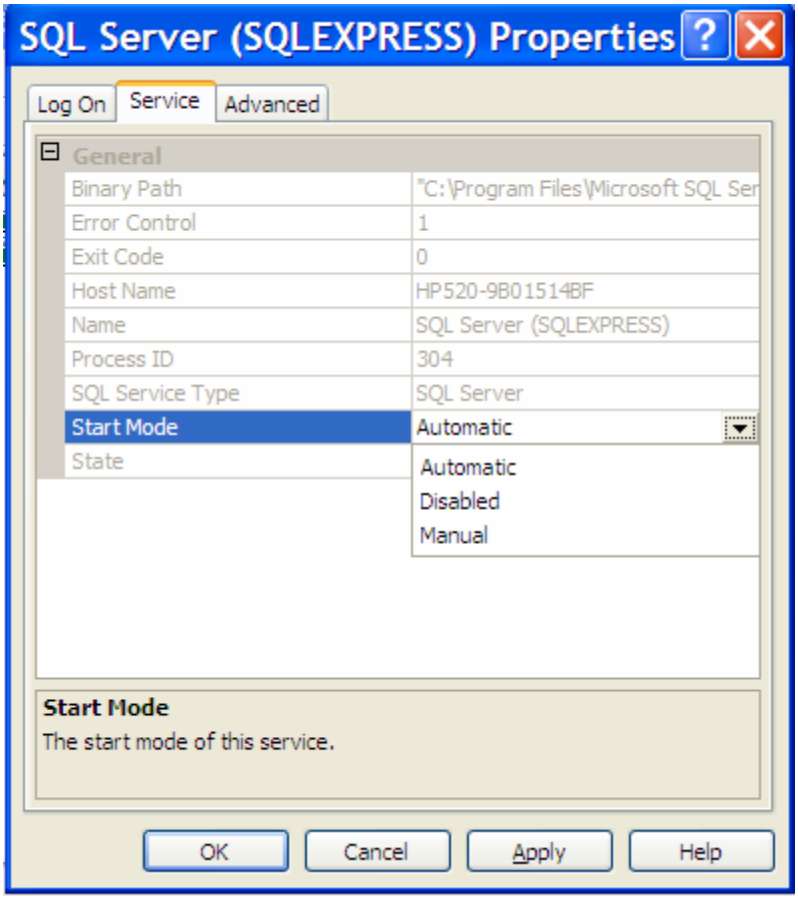
Khởi động hoặc dừng các dịch vụ SQL Server sử dụng SQL Server Configuration Manager ta thực hiện theo các bước sau:

1. Vào *start/Programs/Microsoft SQL Server 2005/Configuration Tools/SQL Server Configuration Manager*, xuất hiện cửa sổ SQL Server Configuration Manager
2. Chọn SQL Server 2005 Services, trong khu vực chi tiết bên phải, right click lên thể hiện của SQL Server mà ta muốn khởi chạy hoặc dừng. Giả sử SQL Server (SQLEXPRESS) như hình 3.6.
    + Start: Khởi chạy thể hiện của SQL Server
    + Stop: Dừng hoạt động của thể hiện SQL Server.
    + Pause: Tạm dừng hoạt động của thể hiện SQL Server
    + Restart: Khởi động lại thể hiện của SQL Server



Hình 3.6. Cửa sổ SQL Server Configuration Manager.

3. Muốn khởi chạy tự động dịch vụ SQL Server, trong cửa sổ trên chọn Properties. Trong hộp thoại SQL Server Properties, chọn tab Service và chọn thuộc tính Start Mode là Automatic.



Hình 3.7. Cửa sổ SQL Server Properties.

b) Sử dụng SQL Server Management Studio

Microsoft SQL Server Management Studio là môi trường tích hợp cho việc truy cập, cấu hình, quản lý, quản trị và phát triển tất cả các thành phần của SQL Server. SQL Server Management Studio kết hợp một nhóm rộng lớn các công cụ đồ họa giầu trình biên tập (script editors) cung cấp các truy xuất đến SQL Server để phát triển và quản trị tất cả các mức kỹ năng. Và có thể dùng nó để quản trị SQL Server 2000.

SQL Server Management Studio kết hợp các tính năng của Enterprise Manager, Query Analyzer, và Analysis Manager trong phiên bản trước. Thêm vào đó, SQL Server Management Studio làm việc với tất cả các thành phần cuả SQL Server như là: Reporting Services, Integration Services, SQL Server Mobile, và Notification Services.

Microsoft SQL Server Management Studio bao gồm các tính năng tổng quát sau:

+ Cung cấp hầu hết các tác vụ quản trị cho SQL Server 2005 và SQL Server 2000.
+ Là môi trường đơn, tích hợp cho việc quản trị và trao quyền SQL Server Database Engine.
+ Các hộp thoại mới cho việc quản lý các đối tượng trong SQL Server Database Engine, Analysis Services, Reporting Services, Notification Services, và SQL Server Mobile, cho phép ta thực thi các hành động ngay lập tức, gửi chúng tới Code Editor, hoặc tạo tập lệnh cho lần thực thi tiếp theo.
+ Các hộp thoại cho phép truy cập đến nhiều điều khiển trong khi hộp thoại đó đang được mở.
+ Lập lịch cho phép ta thực thi các hành động của các hộp thoại quản trị.
+ Export và import đăng ký server SQL Server Management Studio từ một môi trường Management Studio này đến môi trường khác.

+ Save hoặc in file XML Showplan hoặc Deadlock files được sinh bởi SQL Server Profiler, xem lại, hoặc gửi chúng tới administrators để phân tích.v.v…

Để truy cập vào SQL Server Management Studio:

1. Để chạy SQL Server Management Studio, trên thanh taskbar, click *Start/ Programs/Microsoft SQL Server 2005*, và sau đó click SQL Server Management Studio.
2. Khi khởi chạy SQL Server Management Studio, một hộp thoại "Connect to Server" (Hình 3.8) xuất hiện. Ta có thể chọn một thể hiện của Server để kết nối hoặc không chọn một thể hiện nào cả.

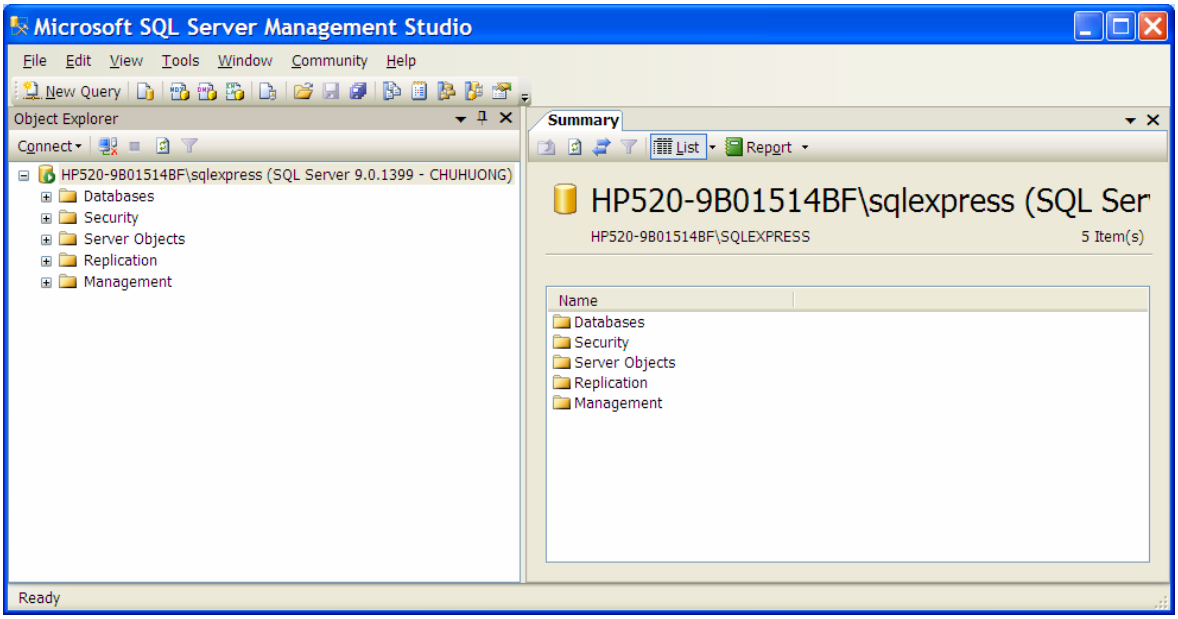
Hình 3.8. Cửa sổ Connect to Server.

+ Server type: Chọn Database Engine để kết nối đến cỗ máy cơ sở dữ liệu.
+ Server name: chọn hoặc nhập tên server
+ Authentication: Chọn chế độ xác thực là Windows Authentication hoặc SQL Server Authentication, nếu chọn SQL Server Authentication thì ta phải cung cấp thông tin cho các mục Login và Password.
+ Login: Nhập tên đăng nhập

+ Password: Mật khẩu của tên đăng nhập

+ Remember password: Tùy chọn được chọn để là đăng nhập sau không phải đánh mật khẩu.

Sau hộp thoại "Connect to Server "cho vào cửa sổ SQL Server Management Studio (Hình 3.9)



Hình 3.9. Cửa sổ SQL Server Management Studio.

** Đăng ký Server*

Cửa sổ Registered Servers trong SQL Server Management Studio chứa danh sách các thể hiện danh sách các server đã đăng ký. Ta sử dụng cửa sổ Registered Servers nhằm mục đích:

+ Lưu thông tin kết nối cho các thể hiện của SQL Server trên mạng.

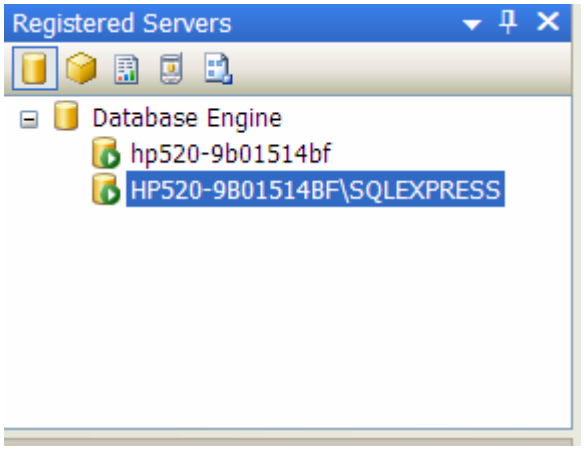
+ Hiển thị một thể hiện đang chạy hay không chạy

+ Kết nối tới một thể hiện trong cửa sổ Object Explorer hoặc Query Editor

+ Nhóm các server

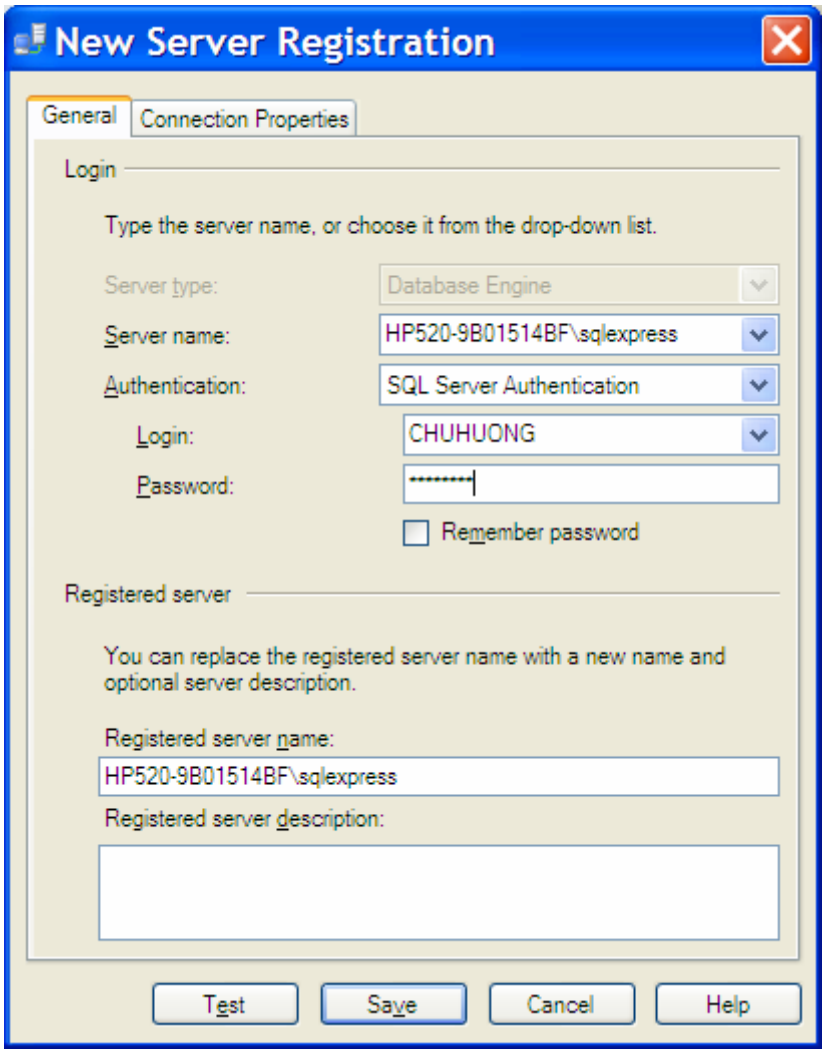
Để đăng ký một thể hiện của SQL Server ta thực hiện các bước sau

1. Vào View\chọn Registered Servers để xuất hiện cửa sổ Registered Servers hình 3.10.
2. Chọn biểu tượng kiểu thể hiện muốn đăng ký, theo thứ tự từ trái sang phải là Database Engine, Analysis Services, Reporting Services, SQL Server Mobile và Integration Services. Ta chọn Database Engine.



Hình 3.10. Cửa sổ Registered Servers.

3. Right click lên biểu tượng hoặc vùng trống chọn New\Server Registration xuất hiện cửa sổ New Server Registration (Hình 3.11)



Hình 3.11. Cửa sổ New Server Registration.

Trong cửa sổ này ta thực hiện các lựa chọn sau:

+ Server Name: Nhập hoặc chọn tên thể hiện của SQL Server từ hộp danh sách Server Name, có thể chọn <Browser for more …> để tự tìm các thể hiện của SQL Server trên mạng.
+ Authentication: Chọn chế độ xác thực, tương tự như trong cửa sổ Connect to Server.
+ Registered server name: Tên của thể hiện trên cửa sổ Registered Servers.
+ Nút Test: Dùng để kiểm tra xem kết nối có thành công hay không?

+ Nút Save: Dùng để lưu thông tin đăng ký.

Để kết nối tới thể hiện của SQL Server và hiển thị nó trong cửa sổ Object Explorer hoặc tạo truy vấn mới ta thực hiện các bước sau:

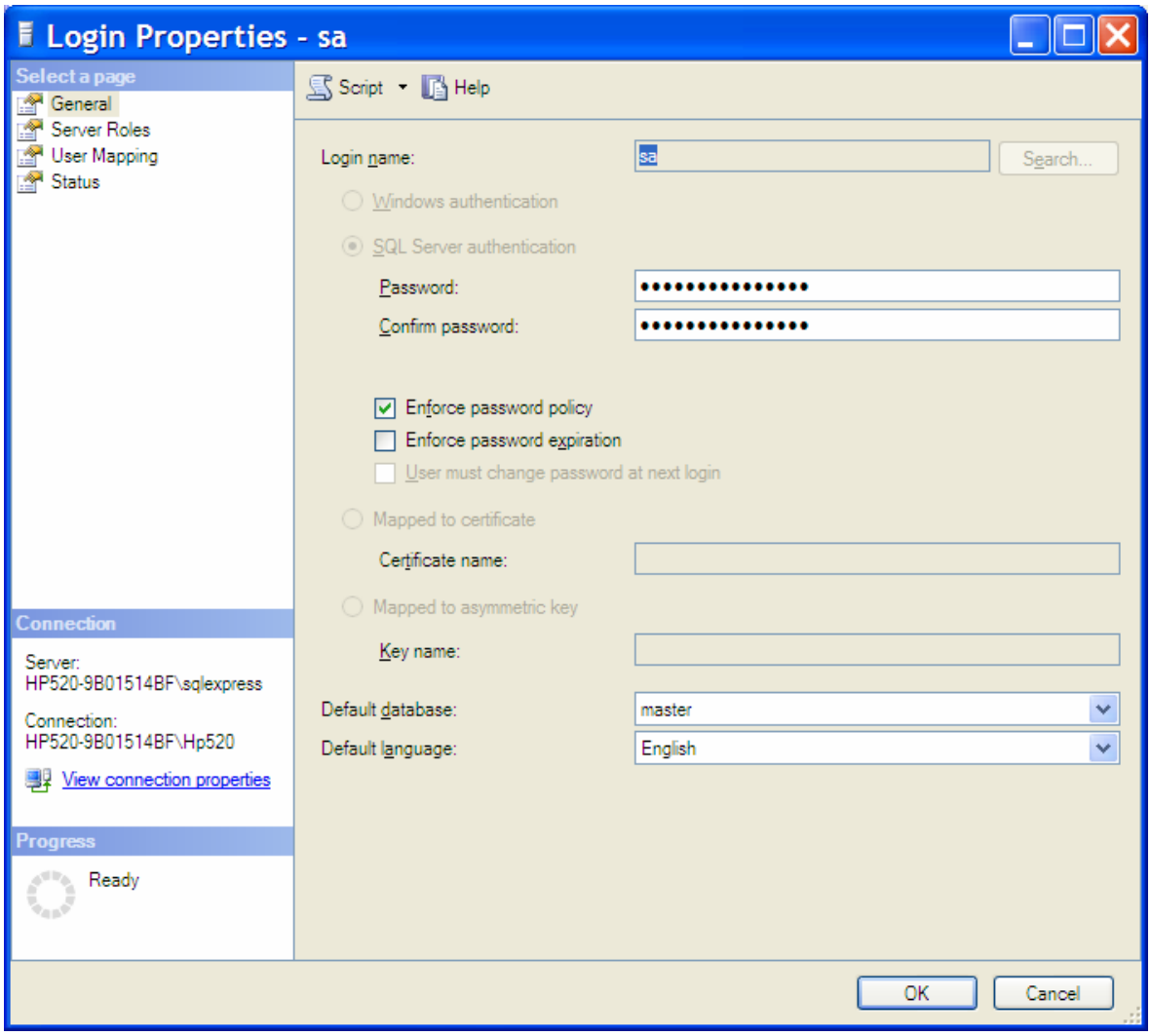
1. Trong cửa sổ Registered Servers chọn biểu tượng kiểu thể hiện muốn kết nối. Ta chọn Database Engine.
2. Right click lên thể hiện muốn kết nối, chọn Connect và chọn:
   + New Query: để tạo một truy vấn đã kết nối tới thể hiện này.
   + Object Explorer: Để hiển thị nó trong cửa sổ Object Explorer.

** Thay đổi mật khẩu mặc định*

Thay đổi mật khẩu cho tài khoản đăng nhập sa ta thực hiện theo các bước như sau:

1. Trong cửa sổ Object Explorer của SQL Server Management Studio chọn tên thể hiện của server
2. Chọn *Security/Logins* để hiển thị tất cả các tài khoản người dùng.
3. Right click chuột lên tài khoản *sa*, và chọn *Properties*, xuất hiện cửa sổ Login Properties như hình 3.12.

Hình 3.12. Cửa sổ Login Properties.



+ Trong tab General: Nhập mật khẩu mới vào hộp Password và xác nhận mật khẩu vào ô Confirm Password.

+ Trong tab Status, mục Login chọn Enabled để cho phép có thể login vào theo tài khoản sa, hoặc Disabled để không thể login vào theo tài khoản sa.

3.2. LÀM VIỆC VỚI CÁC ĐỐI TƯỢNG TRONG SQL SERVER

3.2.1. Cơ sở dữ liệu - Database

a) Các Database hệ thống

Khi cài đặt SQL Server có 4 Database hệ thống được cài đặt, đó là:

- *master:* Ghi nhận thông tin cấp hệ thống, thông tin khởi tạo SQL Server và thiết lập cấu hình SQL Server. Database này cũng ghi nhận tất cả các tài khoản đăng nhập, sự tồn tại của các Database khác, vị trí tập tin chính cho tất cả Database người dùng.
- *tempdb:* Giữ các bảng tạm, các stored procedure tạm,.v.v... Được dùng cho các nhu cầu lưu trữ tạm của SQL Server.
- *model*: là khuôn mẫu cho tất cả các CSDL khác được tạo trên hệ thống kể cả tempdb. Database model phải được tồn tại trên hệ thống, bởi vì nó được dùng để tạo lại tempdb mỗi khi SQL server được khởi động.
- *msdb:* Giữ các bảng mà SQL Server Agent dùng để lập thời gian biểu thực thi các công việc, các cảnh báo và các operator.

b) Tạo Database

Để tạo Database trong SQL Server 2000 , ta có 3 cách khác nhau để tạo:

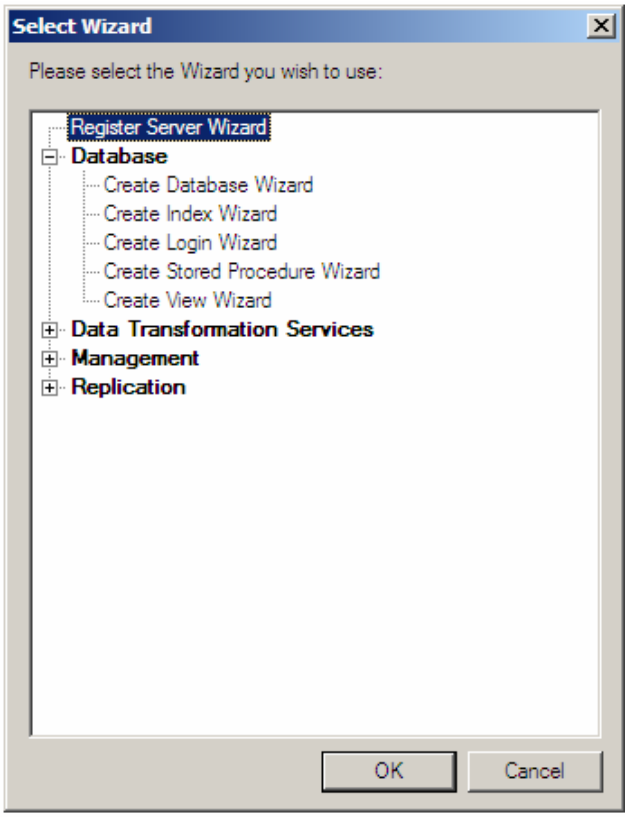
+ Sử dụng Create Database Wizard
+ SQL Server Enterprise Manager
+ Dùng T-SQL

Trong SQL Server 2005 , ta có 2 cách khác nhau để tạo:

+ SQL Server Management Studio
+ Dùng T-SQL

**** Sử dụng Create Database Wizard trên SQL Server 2000 :***

1. Khởi động SQL Server Enterprise Manager, chọn tên server cục bộ và vào menu *Tools\Wizards* → Xuất hiện cửa sổ *Select Wizard* hình 3.13.

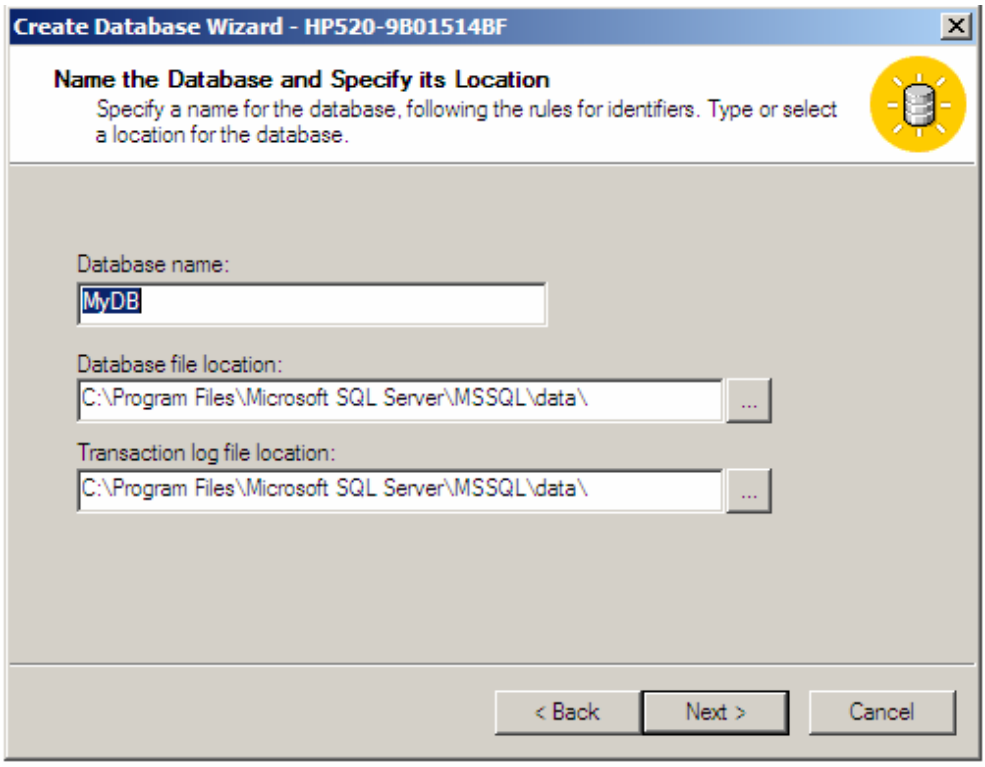


Hình 3.13. Cửa sổ Select Wizard

Hình 3.14. Cửa sổ Welcome to Create Database Wizard

2. Chọn *Create Database Wizard* và chọn *OK* → Xuất hiện cửa sổ Cửa sổ Welcome to Create Database Wizard hình 3.14.

3. Click *Next* → Xuất hiện cửa sổ 3.15. *Name the Database And Specify its Location*. Nhập tên Database và vị trí lưu các file cơ sở dữ liệu và file log → và chọn Next.



Hình 3.15. Cửa sổ Name the Database And Specify its Location

4. Màn hình Name the Database files (hình 3.16), nhập tên tập tin dữ liệu chính (có thể dùng tên mặc) và kích thước khởi tạo cho tập tin đó. Sau đó chọn Next → Xuất hiện cửa sổ 3.17.

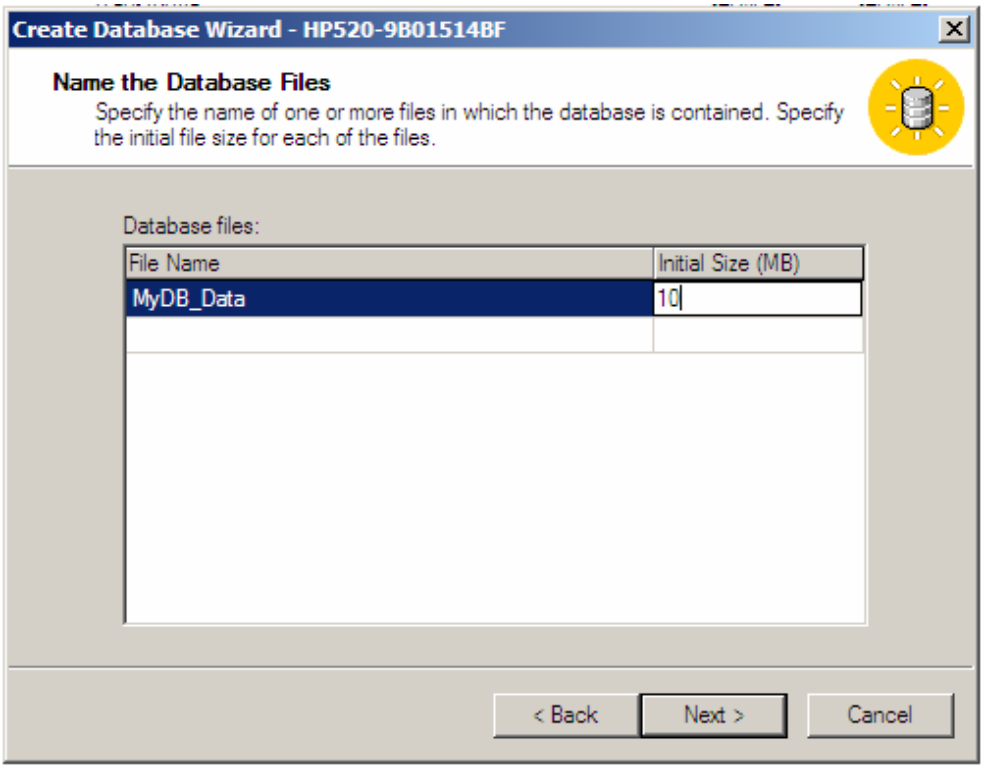
5. Theo tùy chọn mặc định: Tập tin CSDL tự động gia tăng và gia tăng 10%, không giới hạn dung lượng tối đa. Ta có thể thay đổi các tham số đó thông qua các tùy chọn:

   + Do not automatically grow the data file: Không tự động gia tăng kích thước của file dữ liệu.
   + Automatically grow the data file: Tự động gia tăng kích thước của file dữ liệu.

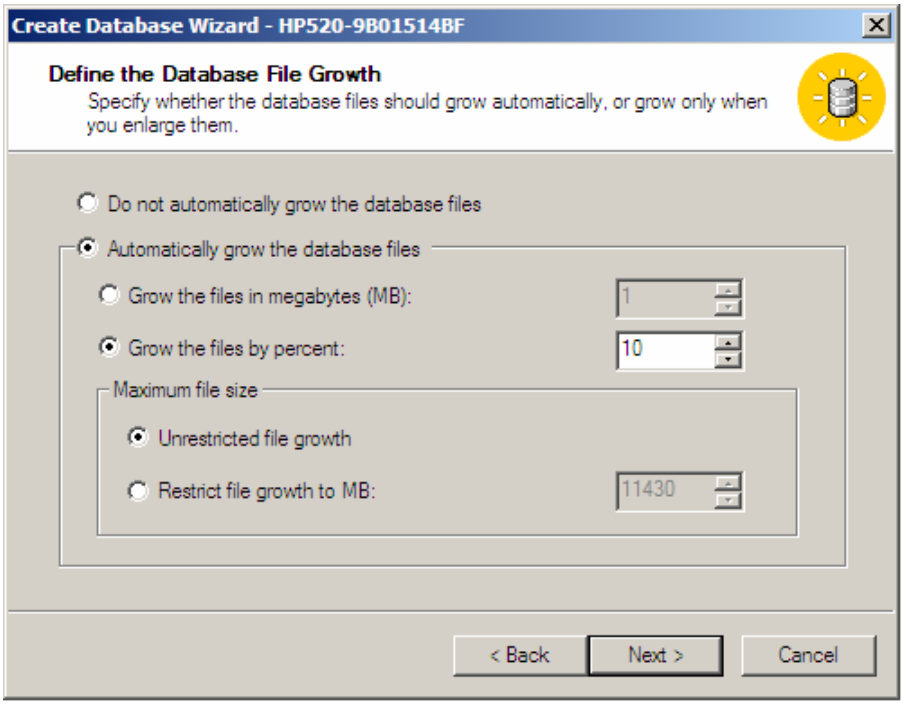
+ Grow the file in megabytes (MB): Gia tăng theo MB.
+ Grow the file by percent: Gia tăng theo phần trăm.
+ Unrestricted file growth: Gia tăng không giới hạn cận trên của file.
+ Restricted file growth to MB: Giới hạn cận trên của file theo MB.

Chọn Next. Xuất hiện của sổ Name the Transaction Log Files hình 3.18.

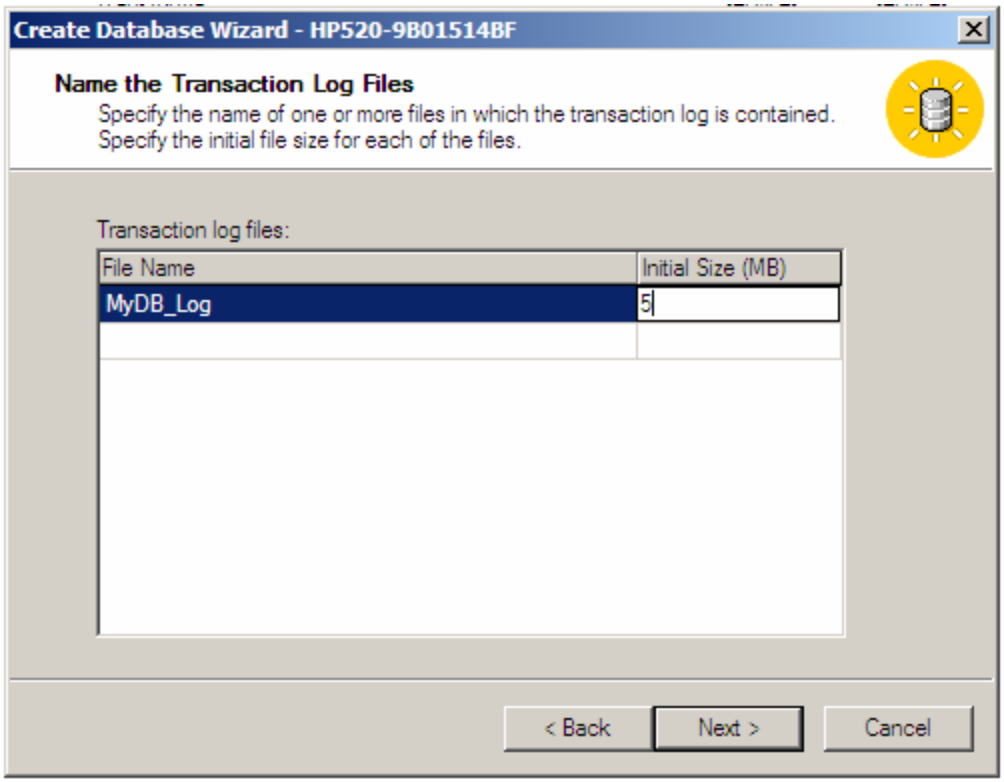
6. Nhập tên tập tin bản ghi giao dịch và kích thước khởi tạo chúng. Chọn Next.
7. Xuất hiện cửa sổ Define the Transaction Log File Growth. Các thông tin điền tương tự như cửa sổ Define the Database file Growth. Chọn Next để tiếp tục.
8. Chọn Finish để kết thúc.



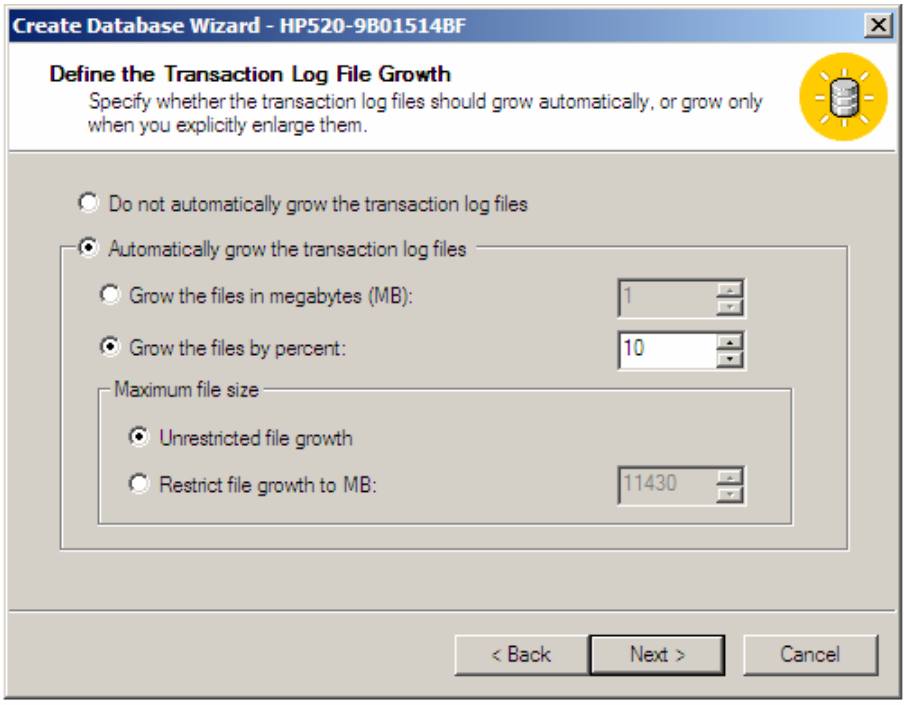
Hình 3.16. Cửa sổ Name the Database files



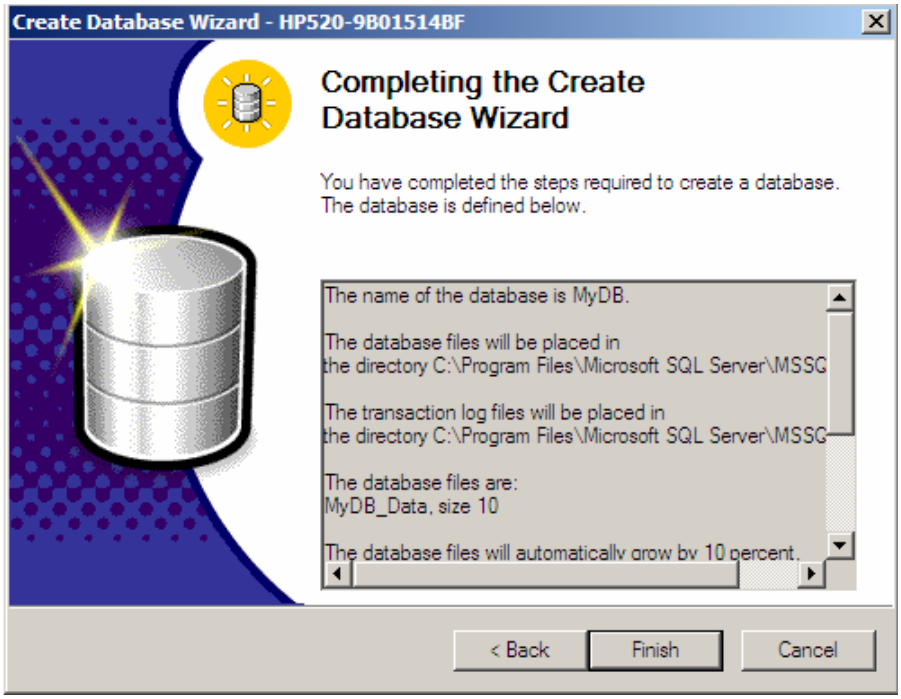
Hình 3.17. Cửa sổ Define the Database file Growth



Hình 3.18. Cửa sổ Name the Transaction Log Files



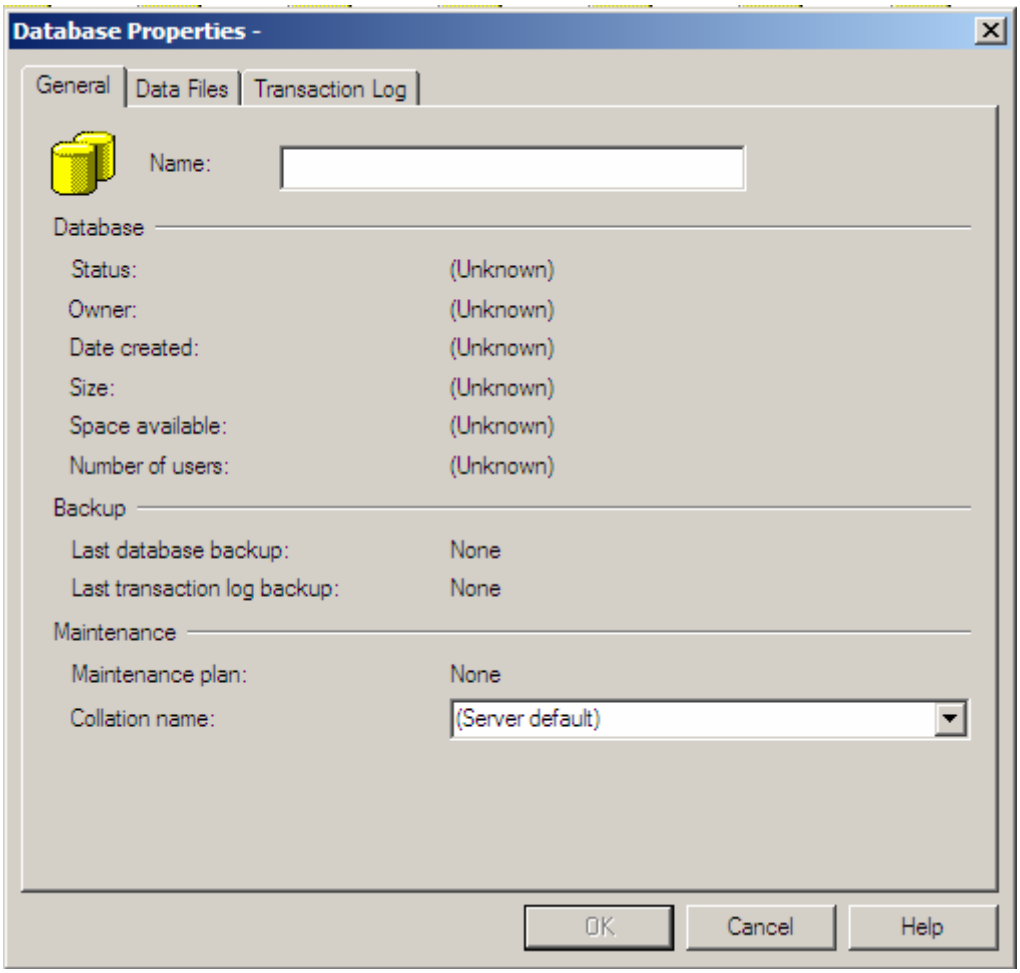
Hình 3.19. Cửa sổ Define the Transaction Log File Growth



Hình 3.20. Cửa sổ hoàn thành tạo Database.

**** Sử dụng SQL Server Enterprise Manager trên SQL Server 2000***

Khởi động SQL Server Enterprise Manager. Chọn tên *Server* và chọn *Database*. Rigth click lên mục Database và chọn New Database. Xuất hiện cửa sổ Database Properties (hình 3.21).



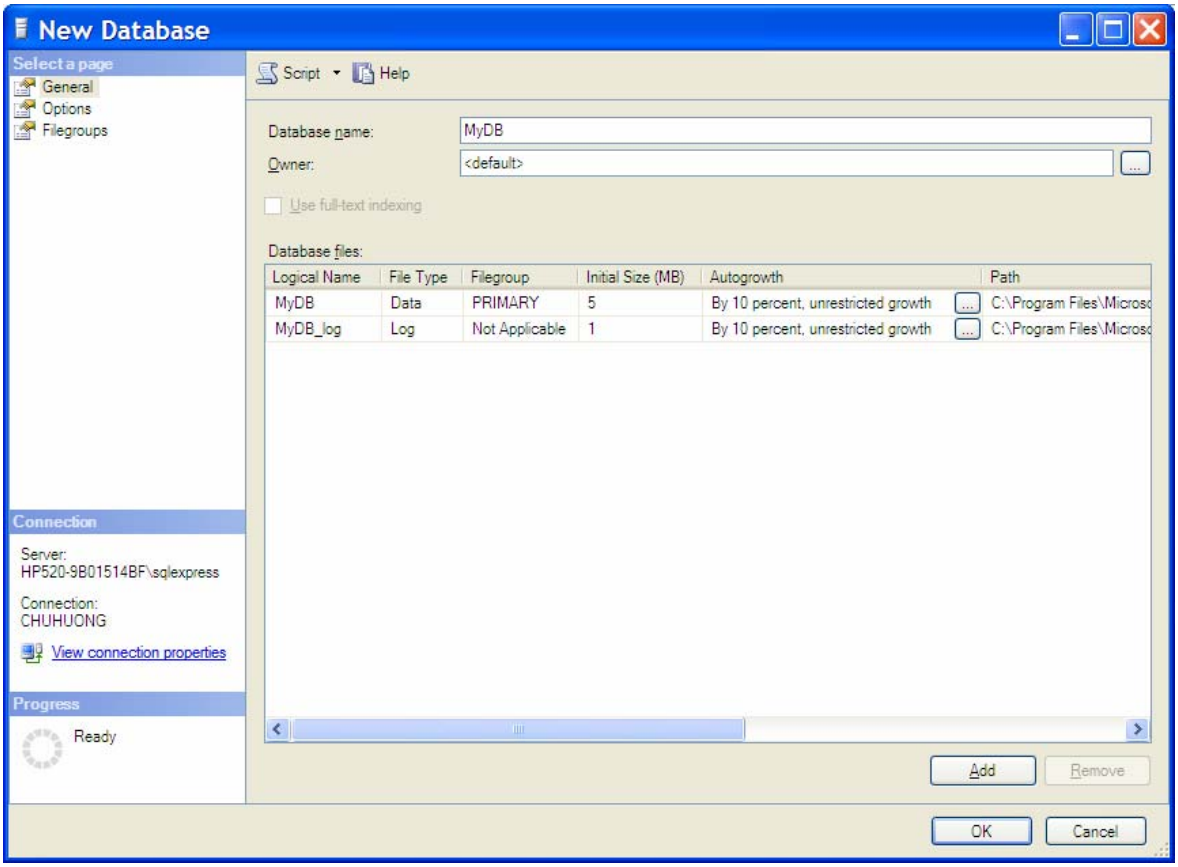
Hình 3.21. Cửa sổ Database Properties.

+ *Tab General:* Nhập tên Database chẳng hạn QLDiemSV
+ *Tab Data files:* Đặt tên các file dữ liệu, vị trí lưu trữ chúng và các tham số growth. Chú ý, SQL Server Enterprise Manager tự động tạo tập tin dữ liệu chính QLDiemSV_Data (có thể thay đổi được tên) thuộc nhóm PRIMARY không thể thay đổi nhóm. Ta có thể tạo các tập tin dữ liệu phụ trên các nhóm khác nhau.
+ *Tab Transaction log:* Tương tự như trên nhưng đối với tập tin ghi các bản ghi giao dịch.

**** Sử dụng SQL Server Management Studio của SQL Server 2005:***

Như đã giới thiệu ở trên, SQL Server Management Studio là công cụ tích hợp SQL Server Enterprise Manager của phiên bản 2000 nên ta có thể sử dụng để tạo database tương tự như phiên bản 2000. Cách tạo như sau:

1. Mở rộng các đối tượng trên thể hiện của SQL trong cửa sổ Object Explorer muốn tạo Database.
2. Right click lên mục Database\ chọn New Database xuất hiện cửa sổ New Database (Hình 3.22).

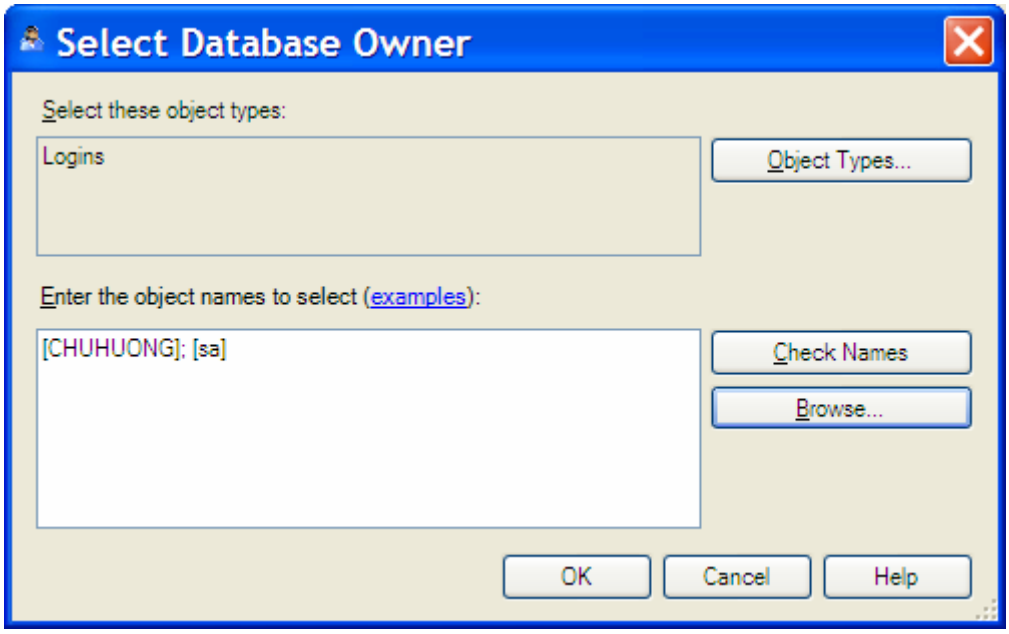


Hình 3.22. Cửa sổ New Database.

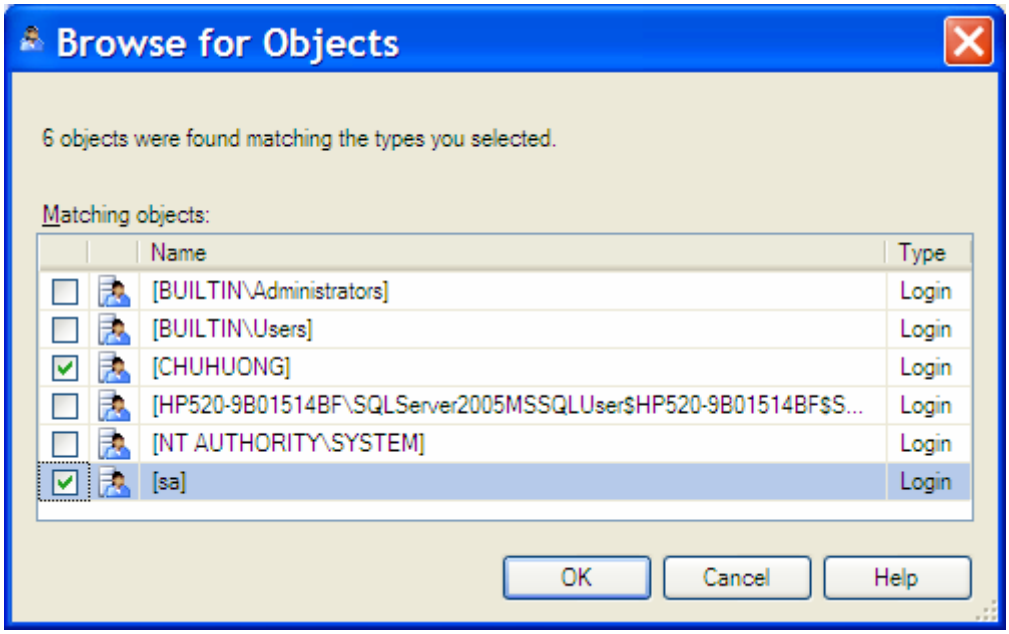
Trong cửa sổ New Database, tại tab General ta thực hiện điền các thông số cho các mục như sau:

+ Database name: Điền tên Database muốn tạo
+ Owner: Chỉ định tên các Logins sở hữu Database đang tạo. Để chọn các logins ta click vào nút … xuất hiện cửa sổ 'Select Database Owner' (Hình 3.23). Trong mục 'Enter the Object names to select' ta nhập tên các logins hoặc chọn nút

Browse để liệt kê và chọn các logins trong cửa sổ 'Browse for Objects' (Hình 3.24)



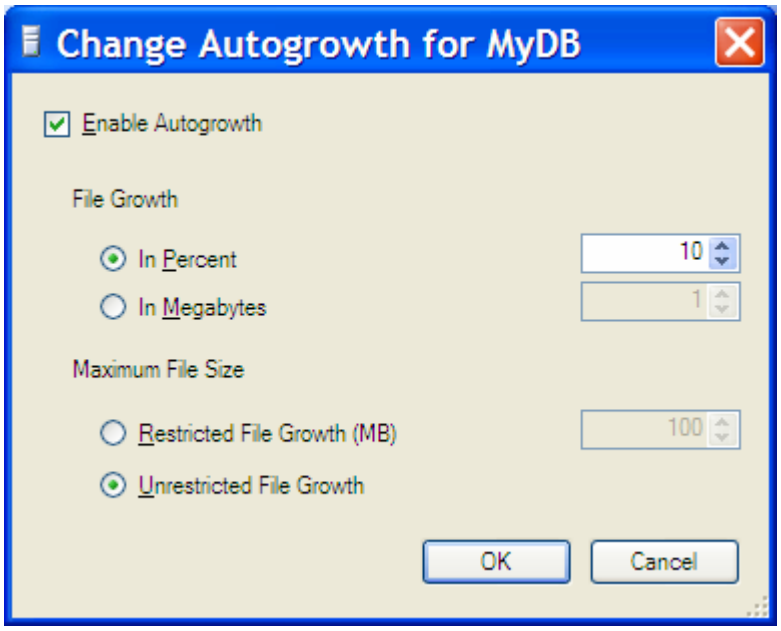
Hình 3.23. Cửa sổ Select Database Owner.



Hình 3.24. Cửa sổ Browse for Objects.

+ Database files: Thực hiện chọn các thuộc tính cho các files database, bao gồm:
  - Logical name: Tên logic của file database.
  - File Type: Kiểu file (Data, Log)

- Filegroup: Chỉ định nhóm file.
- Initial Size (MB): Kích thước khởi tạo dung lượng các file, tính theo MB.
- Autogrowth: Chỉ định các tham số tự động gia tăng dung lượng cho các file. Click vào nút … xuất hiện cửa sổ 'Change Autogrowth for MyDB' (Hình 3.25) cho phép khai báo các tham số Autogrowth:
  - Tùy chọn Enable Autogrowth cho phép tự động gia tăng;
  - File Growth chỉ định độ tự động gia tăng theo phần trăm (In Percent) hoặc theo dung lượng chỉ định (In Megabytes);
  - Maximum File Size định nghĩa độ gia tăng của file đến dung lượng lớn nhất.



Hình 3.25. Cửa sổ Change Autogrowth for MyDB

- Path: Chỉ định đường dẫn vật lý lưu trữ các files database. Click nút … để thay đổi đường dẫn mặc định.

**** Sử dụng T-SQL***

Ta có thể sử dụng T-SQL hay script để tạo Database, với cách này ta cần hiểu rõ về các cú pháp câu lệnh trong SQL Server để tạo Database.

```
CREATE DATABASE database_name
    [ ON
        [ PRIMARY ] [ <filespec> [ ,...n ]
        [ , <filegroup> [ ,...n ] ]
    [ LOG ON { <filespec> [ ,...n ] } ]
    ]
    ]
[;]
```

Để attach một database:

```
CREATE DATABASE database_name
    ON <filespec> [ ,...n ]
    FOR { ATTACH | ATTACH_REBUILD_LOG }
[;]
```

```
<filespec> ::=
{
(
    NAME = logical_file_name ,
    FILENAME = 'os_file_name'
        [ , SIZE = size [ KB | MB | GB | TB ] ]
        [ , MAXSIZE = { max_size [ KB | MB | GB | TB ]
| UNLIMITED } ]
        [ , FILEGROWTH = growth_increment [KB | MB |
GB | TB | % ] ]
) [ ,...n ]
}
```

```
<filegroup> ::=
{
FILEGROUP filegroup_name [ DEFAULT ]
    <filespec> [ ,...n ]
}
```

Các tham số trong đó:

database_name:

Là tên của CSDL. Nếu tên của file dữ liệu (data file) không được chỉ định thì SQL Server sử dụng *database_name* là tên cho *logical_file_name* và *os_file_name*.

ON

Chỉ định định nghĩa các file trên đĩa được sử dụng để lưu trữ các phần dữ liệu của database, *data files*.

PRIMARY

Chỉ định này liên quan đến danh sách định nghĩa primary file <filespec>. File đầu tiên được chỉ định trong <filespec> của nhóm *filegroup primary* filegroup trở thành *file primary*. Một database chỉ có thể duy nhất một *file primary*.

Nếu từ khóa PRIMARY không được chỉ định thì file đầu tiên trong danh sách các fie của câu lệnh CREATE DATABASE sẽ trở thành *file primary*.

LOG ON

Chỉ định định nghĩa file log lưu trữ trên đĩa, log files.

FOR ATTACH

Chỉ định database được tạo băng việc attach tập các file hệ thống đã tồn tại.

FOR ATTACH có các yêu cầu sau:

- Các files data (MDF và NDF) phải đã tồn tại.
- Nếu nhiều files log tồn tại, thì tất cả phải sẵn có.

FOR ATTACH_REBUILD_LOG

Chỉ định database được tạo băng việc attach tập các file hệ thống đã tồn tại. Nếu một hoặc nhiều files log giao dịch bị lỗi thì file log sẽ được xây dựng lại.

<filespec>

Điều khiển các thuộc tính của file.

- NAME logical_file_name: Chỉ định tên logical cho file. NAME được yêu cầu khi FILENAME được chỉ định.
- FILENAME *os_file_name*: Chỉ định tên, đường dẫn file hệ điều hành (file vật lý).
- SIZE size: Chỉ định kích thước file.

- MAXSIZE *max_size:* Chỉ định kích thước lớn nhất mà file có thể phát triển đến. Từ khóa UNLIMITED chỉ định file được phát triển cho đến khi đĩa bị đầy.
- FILEGROWTH *growth_increment*: Chỉ định độ tự đông gia tăng của file.

<filegroup>

Điều khiển các thuộc tính của filegroup.

- FILEGROUP filegroup_name: Chỉ định tên logical của filegroup.
- DEFAULT: Chỉ định tên filegroup là filegroup mặc định trên database.

Ví dụ 3.1. Đưa ra một cách tạo Database MyDB với tập tin dữ liệu chính là MyDB_Data.mdf, dung lượng khởi tạo là 1MB và tối đa là 10MB và độ gia tăng kích thước là 10%. Tập tin bản ghi giao dịch là MyDB_Log.ldf với dung lượng ban đầu là 2MB và kích thước tối đa không giới hạn, độ gia tăng dung lượng là 10MB.

Để tạo script (dùng T- SQL) ta mở cửa sổ query để soạn thảo:

- Đối với SQL Server 2000: Vào *Start/Programs/Microsoft SQL Server/Query Analyzer*, xuất hiện hộp thoại *Connect to SQL Server*, trong danh sách server chọn server cục bộ máy mình và nhập đoạn mã lệnh sau.
- Đối với SQL Server 2005: Trong cửa sổ SQL Server Management Studio, trên thanh Standard chọn nút New Query để tạo cửa sổ truy vấn kết nối đến thể hiện SQL đang được kết nối hoặc nút Database Engine Query để xuất hiện cửa sổ kết nối (Hình 3.8) ta thực hiện kết nối đến thể hiện SQL mà muốn thực hiện trên đó. Xuất hiện cửa sổ truy vấn ta thực hiện nhập đoạn mã lệnh sau:

```
Use master
go
create database MyDBT
On
(
Name= MyDBT_Data,
```

```
FileName='E:\Tempt\MyDBT_Data.mdf',
Size=10MB,
MaxSize=100MB,
FileGrowth=10%
)
Log On
(
Name= MyDBT_log,
FileName='E:\Tempt\MyDBT_Log.ldf',
Size=2MB,
MaxSize=UNLIMITED,
FileGrowth=10%
)
```

- Click nút *Execute* hoặc *F5* để chạy.

c) Xóa Database

** Dùng Enterprise Manager trong SQL Server 2000:*

Trong Enterprisse Manager, chọn nhóm server cục bộ và mở rộng mục Database. Sau đó right click lên CSDL muốn xóa và chọn Delete Database. Xuất hiện hộp thoại xác nhận xóa và chọn Yes.

** Sử dụng SQL Server Management Studio của SQL Server 2005:*

Để xóa một database ta đăng nhập vào SQL Server Management Studio với một login có quyền xóa database đó và thực hiện các bước sau:

1. Mở rộng các đối tượng trên thể hiện của SQL trong cửa sổ Object Explorer muốn xóa Database.
2. Mở rộng mục Database và Right click lên database muốn xóa và chọn Delete xuất hiện cửa sổ 'Delete Object' chọn OK để xóa.

** Dùng T-SQL*

Để xóa Database ta sử dụng cú pháp sau:

```
DROP DATABASE database_name [ ,...n ]
```

Ví dụ: nhập đoạn T-SQL sau để xóa Database MyDB.

```
Use master
Go
Drop Database MyDB
Go
```

3.2.2.Bảng - Table

Trước khi đi vào các thao tác đối với bảng, ta xét các kiểu dữ liệu được sử dụng trong SQL Server.

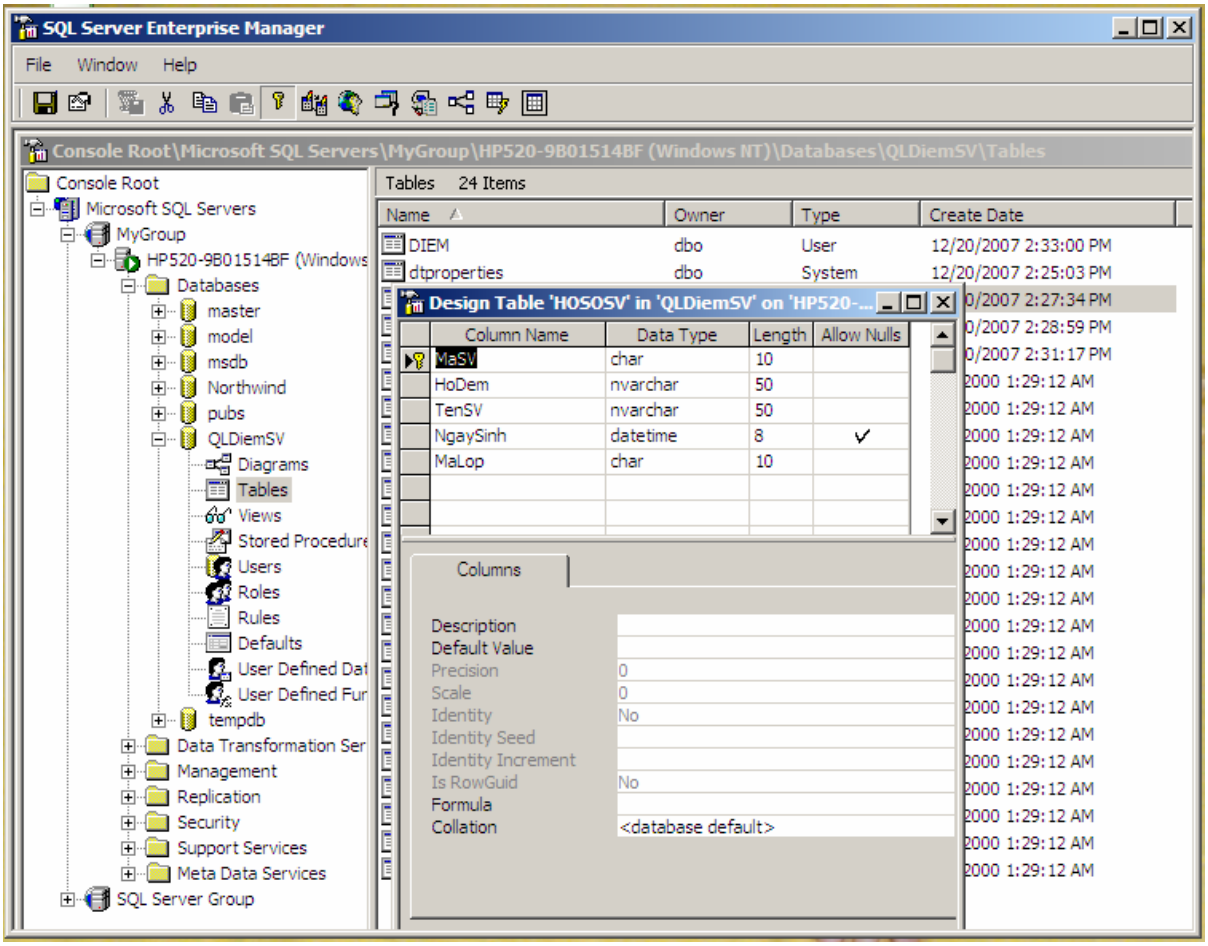
a) Các kiểu dữ liệu

Kiểu dữ liệu	***Ý nghĩa***
bigint	Số nguyên 8 byte.
binary[(n)]	Dữ liệu nhị phân có kích thước cố định n byte. Kích thước lưu trữ là n+4 byte. Với 1<n<8000
bit	Dữ liệu số nguyên nhận các giá trị 0, 1 hoặc NULL
char[(n)]	Dữ liệu ký tự, có chiều dài n ký tự, không hỗ trợ Unicode. Với độ dài 1<n<8000.
cursor	Tham chiếu tới một con trỏ. Chỉ dùng cho các biến và tham số trong stored procedure.
datetime	Dữ liệu ngày giờ. Giá trị nhận từ 1/1/1753 đến 31/12/9999
decimal[(p,[s])] numeric[(p,[s])]	Dữ liệu số thập phân. p là tổng số ký tự số có thể được lưu, s là số chữ số thập phân.
float[(n)]	Dữ liệu số động có phạm vi từ -1.79E+308 đến 1.79E+308
image	Dữ liệu nhị phân có chiều dài thay đổi, dài hơn 8000 byte và tối đa là 2^31-1 byte.
integer hoặc int	Dữ liệu số nguyên từ -2^31 đến 2^31-1. Chiếm 4 byte.
money	Dữ liệu kiểu tiền tệ từ -2^63 đến 2^63. Chiếm 8byte.
nchar[(n)]	Dữ liệu ký tự, có chiều dài cố định n ký tự. Có hỗ trợ Unicode. Với 1<n<4000. Chiếm 2n byte.
ntext	Dữ liệu ký tự có chiều dài thay đổi, với chiều dài tối đa là

	2^30-1 ký tự. Hỗ trợ unicode.
nvarchar[(n)]	Dữ liệu ký tự có chiều dài thay đổi với n ký tự. Có hỗ trợ Unicode. Với 1<n<4000.
real	Dữ liệu số thực động, phạm vi từ - 3.4E+38 đến 3.4E+38. Chiếm 4 byte.
smalldatetime	Dữ liệu ngày giờ từ 1/1/1900 đến 6/6/2079. Chiếm 4 byte.
smallint	Dữ liệu số nguyên từ -2^15 đến 2^15-1. Chiếm 2 byte.
smallmoney	Dữ liệu kiểu tiền tệ từ -2^31 đến 2^31. Chiếm 4 byte.
sql_variant	Cho phép giữ các giá trị của các kiểu dữ liệu khác nhau (tất cả các kiểu dữ liệu hệ thống khác).
sysname	Đây là kiểu dữ liệu đặc biệt, do SQL Server định nghĩa là kiểu nvarchar(128). Chiếm 256 byte.
table	Tương tự như bảng tạm, khai báo gồm danh sách các cột và các kiểu dữ liệu.
text	Dữ liệu ký tự có chiều dài thay đổi, dài hơn 8000byte. Có thể lưu tới 2^31 ký tự. Không hỗ trợ Unicode.
timestamp	Cột timestamp được cập nhật tự động mỗi khi dòng được thêm hoặc được cập nhật. Mỗi bảng chỉ có thể có 1 cột timestamp. Kích thước lưu trữ là 8 byte.
tinyint	Dữ liệu số nguyên từ 0 đến 255. Chiếm 1 byte.
unique-identifier	Giá trị nhị phân 16 byte, là số định danh duy nhất toàn cục.
varbinary[(n)]	Dữ liệu nhị phân n có chiều dài thay đổi. Với 1<n<8000
varchar[(n)	Dữ liệu ký tự có chiều dài thay đổi với n ký tự. Không hỗ trợ Unicode. Với 1<n<4000.

b) Tảo bảng.

** Dùng SQL Server Enterprise Manager trong SQL Server 2000:*



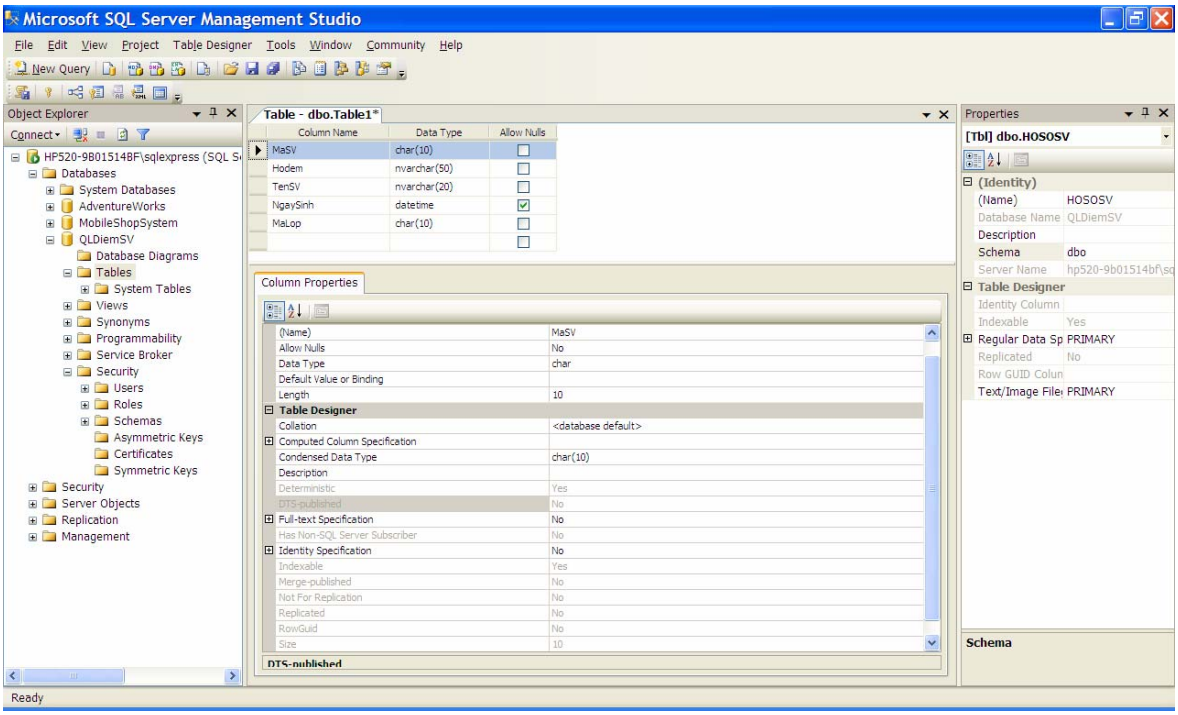
Hình 3.26. Cửa sổ thiết Table.

- Trong Enterprise Manager, mở rộng danh mục Database và mở rộng cơ sở dữ liệu QLDiemSV.
- Right Click lên danh mục Tables, chọn New Table.
- Xuất hiện cửa sổ thiết kế bảng như hình 3.26. Ta thực hiện thiết kế bảng HOSOSV gồm các trường (trong cột Column Name), kiểu trường (Data Type) và độ dài dữ liệu (Length) và cho phép trường đó nhận giá trị NULL hay không (Allow Nulls) hình 3.26.
- Chọn dòng MaSV, sau đó click vào nút Set primary key để thiết lập trường MaSV là khóa.
- Click vào nút Save, xuất hiện cửa sổ *Choose Name* nhập tên bảng là HOSOSV để lưu lại bảng.

** Dùng SQL Server Management Studio trong SQL Server 2005:*

Để tạo bảng trong SQL Server Management Studio ta thực hiện:

1. Mở rộng các đối tượng trên thể hiện của SQL trong cửa sổ Object Explorer muốn tạo bảng.
2. Mở rộng mục Database và chọn cơ sở dữ liệu muốn tạo bảng. Right click lên mục Table và chọn New Table xuất hiện cửa sổ thiết kế table (Hình 3.27):
   + Column Name: Nhập tên các cột trong bảng.
   + Data Type: Chọn kiểu dữ liệu và độ dài cho kiểu.
   + Allow Nulls: Cho phép chấp nhận dữ liệu NULL hay không?
   + Column Properties: Thiết lập các thuộc tính cho cột đang được chọn.



Hình 3.27. Cửa sổ thiết Table trong SQL Server Management Studio .

3. Để thiết lập khóa, chọn các trường khóa của bảng sau đó click vào nút biểu tượng khóa (Set Primary Key) hoặc right click vào các trường đó và chọn Set Primary Key.

4. Dùng tổ hợp phím Ctrl+S hoặc click vào nút Save để lưu cấu trúc bảng vừa tạo trong cơ sở dữ liệu.

** Dùng T-SQL*

Để tạo bảng dùng T – SQL ta sử dụng cú pháp sau:

```
CREATE TABLE table_name
   ( { < column_definition > | < table_constraint > }
[ ,...n ]
   )

< column_definition > ::=
   { column_name data_type }
   [ { DEFAULT constant_expression
      | [ IDENTITY [ ( seed , increment ) ]
      ]
    } ]
   [ < column_constraint > [ ...n ] ]

< column_constraint > ::=
   [ CONSTRAINT constraint_name ]
   { [ NULL | NOT NULL ]
      | [ PRIMARY KEY | UNIQUE ]
      | REFERENCES ref_table [ ( ref_column ) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
   }

< table_constraint > ::=
   [ CONSTRAINT constraint_name ]
   { [ { PRIMARY KEY | UNIQUE }
      { ( column [ ,...n ] ) }
      ]
   | FOREIGN KEY
     ( column [ ,...n ] )
      REFERENCES ref_table [ ( ref_column [ ,...n ] )
]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
   }
```

Ví dụ 3.2. Tạo bảng

a) Tạo bảng LOP dùng T-SQL.

```
Use QLDiemSV
Go
Create Table DMLop
(MaLop char(10) NOT NULL,
TenLop nvarchar(50) NOT NULL,
Khoa char(10)
CONSTRAINT PK_DMLOP Primary Key (MaLop))
```

b) Tạo hai bảng MyCustomers và MyOrders. Bảng MyCustomers có hai cột, cột CustID vừa là cột nhận dạng `IDENTITY[(seed,increment )]` có giá trị khởi tạo ban đầu là 100 và bước nhảy là 1 vừa là khóa. Bảng MyOrders có khóa là OrderID và khóa ngoại là CustID tham chiếu đến bảng MyCustomers.

```
CREATE TABLE MyCustomers (CustID int IDENTITY (100,1)
PRIMARY KEY, CompanyName nvarchar (50))
CREATE TABLE MyOrders (OrderID int PRIMARY KEY,
CustID int REFERENCES MyCustomers(CustID))
```

c) Thay đổi cấu trúc bảng

** Dùng SQL Server Enterprise Manager trong SQL Server 2000:*

+ Trong Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu có bảng cần thay đổi cấu trúc, chẳng hạn QLDiemSV và chọn mục Table.
+ Right Click lên bảng thuộc CSDL QLDiemSV cần thay đổi cấu trúc, chọn Design Table. Xuất hiện cửa sổ thiết kế bảng như hình 3.26. Ta thực hiện các thay đổi về cấu trúc: đổi kiểu dữ liệu, thêm trường, xóa trường, thiết lập khóa .v.v…với bảng đã chọn đó.

** Dùng SQL Server Management Studio trong SQL Server 2005:*

Để thay đổi cấu trúc bảng trong SQL Server Management Studio ta thực hiện các bước sau:

+ Mở rộng các đối tượng trên thể hiện của SQL trong cửa sổ Object Explorer muốn thay đổi cấu trúc bảng.

+ Mở rộng mục Database và chọn cơ sở dữ liệu và bảng cần thay đổi cấu trúc. Right click lên bảng đó và chọn Modify xuất hiện cửa sổ thiết kế table (Hình 3.27) và ta thực hiện các thay đổi đối với cấu trúc bảng như cách tạo bảng.

** Dùng T-SQL*

Để thay đổi cấu trúc bảng bằng T-SQL ta có cú pháp câu lệnh sau:

```
ALTER TABLE table_name
{
    ALTER COLUMN column_name
    {
        type_name [ ( { precision [ , scale ]
            | max } ) ]
        [ NULL | NOT NULL ]
    }

    | [ WITH { CHECK | NOCHECK } ] ADD
    {
        <column_definition>
      | <computed_column_definition>
      | <table_constraint>
    } [ ,...n ]

    | DROP
    {
        [ CONSTRAINT ] constraint_name
        | COLUMN column_name
    } [ ,...n ]
    | [ WITH { CHECK | NOCHECK } ] { CHECK | NOCHECK }
CONSTRAINT
        { ALL | constraint_name [ ,...n ] }
    | { ENABLE | DISABLE } TRIGGER
        { ALL | trigger_name [ ,...n ] }
 [ ; ]
```

Ví dụ 3.3. Thay đổi cấu trúc bảng MyCustomers

```
ALTER TABLE MyCustomers ADD CustType CHAR(10) NULL
```

d) Nhập dữ liệu cho bảng

** Dùng Enterprise Manager*

- Trong Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu QLDiemSV và chọn mục Table.
- Right Click lên bảng thuộc CSDL QLDiem cần nhập dữ liệu, chọn *Open Table\Return all row*. Xuất hiện cửa sổ nhập dữ liệu cho bảng đó.

** Dùng SQL Server Management Studio trong SQL Server 2005:*

+ Mở rộng các đối tượng trên thể hiện của SQL trong cửa sổ Object Explorer muốn nhập dữ liệu cho bảng.
+ Mở rộng mục Database, chọn cơ sở dữ liệu và bảng cần nhập dữ liệu. Right click lên bảng cần nhập dữ liệu và chọn Open Table xuất hiện cửa sổ nhập dữ liệu cho bảng.

** Dùng T-SQL*

Ví dụ 3.3. Chèn dữ liệu vào bảng LOP

- Chèn một bản ghi:

```
Insert Into LOP(MaLop, TenLop, Khoa)
        VALUES ('TH6A',N'Tin học 6A', '6')
```

- Chèn tất cả các bản ghi từ bảng DMLOP vào bảng LOP.

```
Insert Into LOP(MaLop, TenLop, Khoa)
    Select MaLop, TenLop, Khoa From DMLOP
```

e) Xóa bảng

** Dùng Enterprise Manager*

- Trong Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu QLDiemSV và chọn mục Table.
- Right Click lên bảng cần xóa, chọn Delete.

** Dùng SQL Server Management Studio trong SQL Server 2005:*

+ Mở rộng các đối tượng trên thể hiện của SQL trong cửa sổ Object Explorer muốn xóa bảng.

+ Mở rộng mục Database, chọn cơ sở dữ liệu chứa bảng muốn xóa. Right click lên bảng muốn xóa và chọn Delete xuất hiện cửa sổ xác nhận thông tin xóa, chọn OK.

** Dùng T-SQL*

```
DROP TABLE DMLOP
```

3.2.3. View

a) Khái niệm View

View là một bảng ảo được định nghĩa bởi một truy vấn với phát biểu SELECT. View được hình thành dữ liệu từ một hoặc nhiều bảng thật. Đối với người sử dụng thì view giống như một bảng thật.

b) Tạo view

Để tạo View trong SQL Server 2000 , ta có 3 cách khác nhau để tạo:

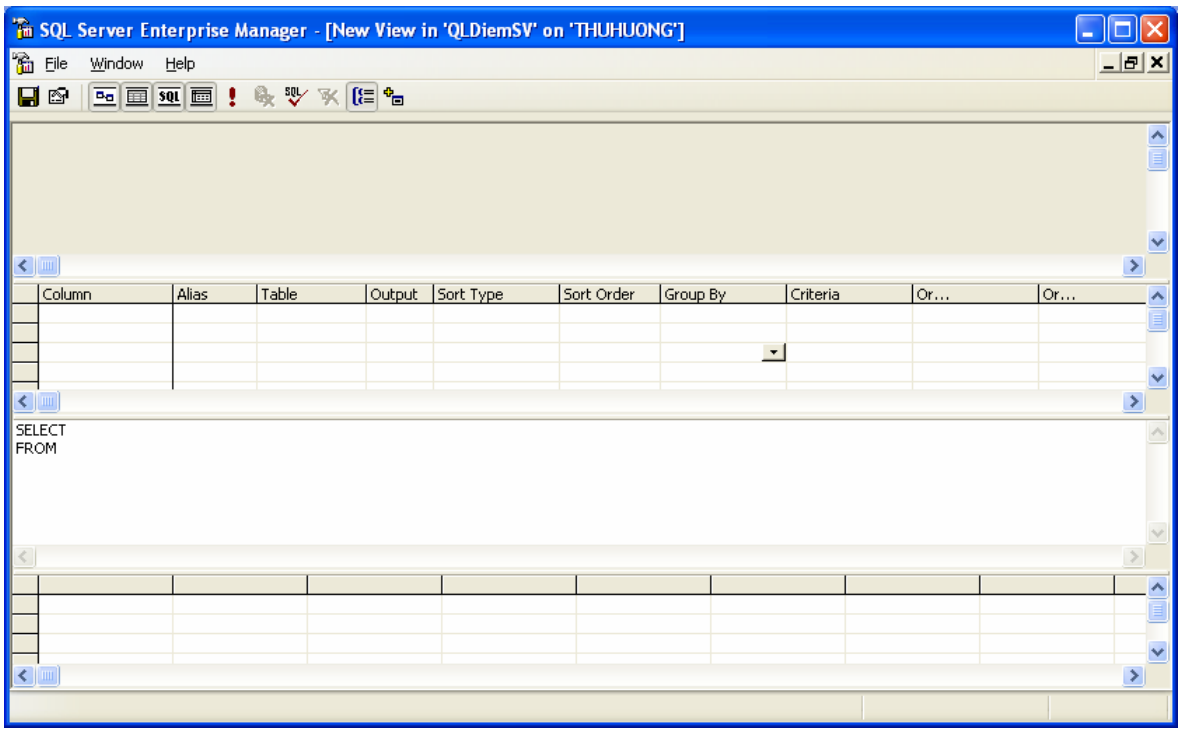
+ Sử dụng Create View Wizard
+ SQL Server Enterprise Manager
+ Dùng T-SQL

Trong SQL Server 2005 , ta có 2 cách khác nhau để tạo:

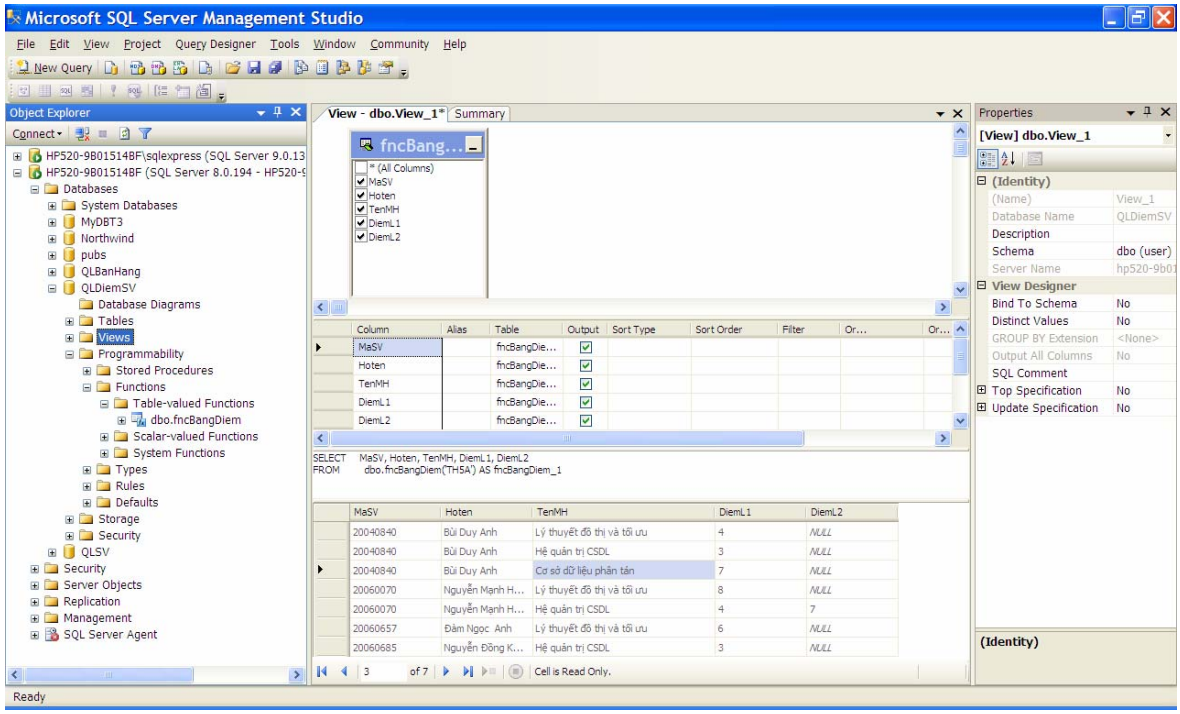
+ SQL Server Management Studio
+ Dùng T-SQL

**** Dùng Enterprise Manager hoặc Management Studio***

- Trong Enterprise Manager hoặc Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn tạo view, chẳng hạn CSDL QLDiemSV và chọn mụcViews.
- Right Click lên danh mục Views, chọn New View. Xuất hiện cửa sổ thiết kế view như hình 3.28 đối với SQL Server 2000 và hình 3.29 đối với SQL Server 2005, gồm 4 vùng sau:



Hình 3.28. *Cửa sổ thiết kế view trên SQL Server 2000.*



Hình 3.29. *Cửa sổ thiết kế view trên SQL Server 2005.*

+ *Diagram pane*: Hiển thị dữ liệu nguồn có thể là các bảng, các view khác, functions để tạo view. Các cột dữ liệu của view được chọn từ vùng này.

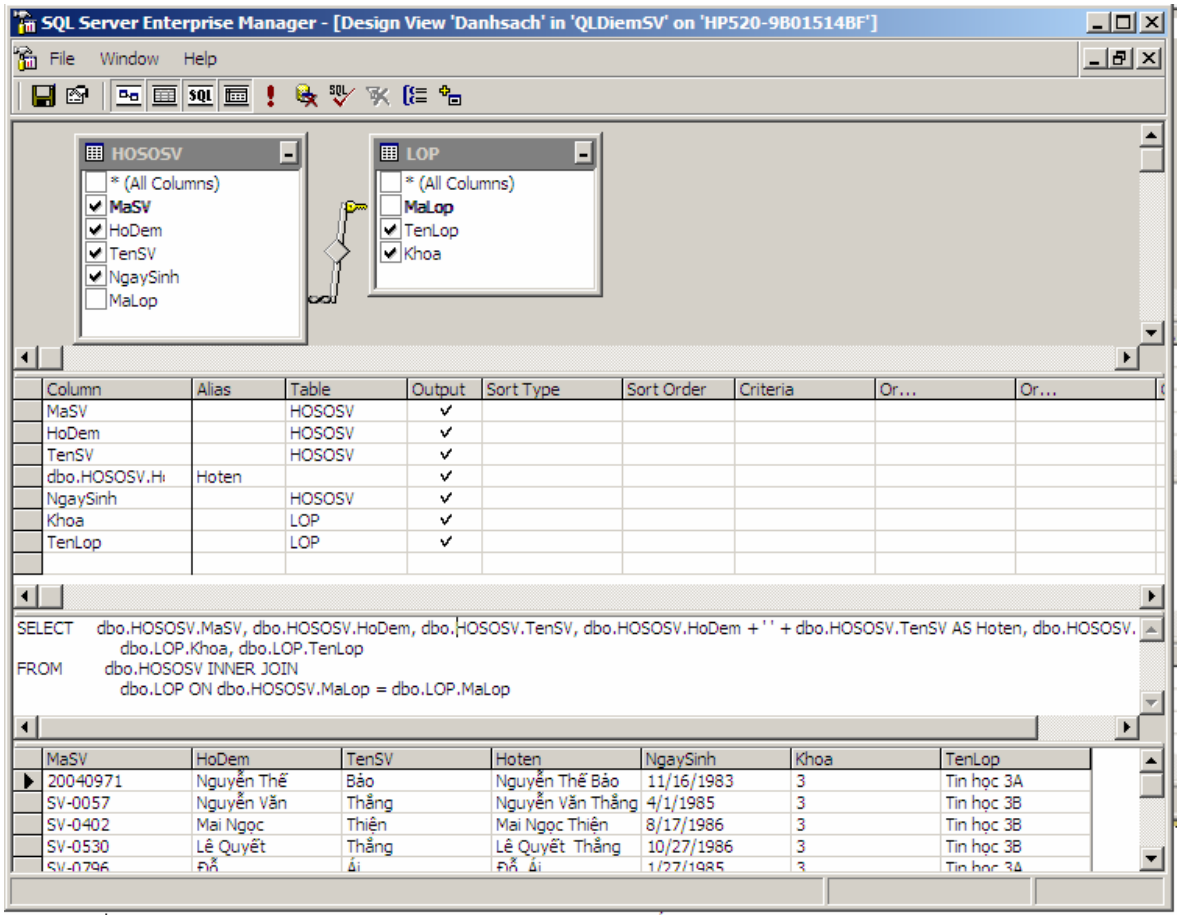
+ *Grid pane (Criteria pane)*: Hiển thị các cột của view đã được chọn từ vùng diagram.
+ *SQL pane*: Hiển thị phát biểu SQL dùng để định nghĩa view.
+ *Results pane*: Hiển thị kết quả nhận được từ view.

Ta có thể ẩn hoặc hiện các cửa sổ này bằng cách click vào các nút tương ứng trên thanh công cụ của cửa sổ thiết kế view. Danh sách sau thể hiện ý nghĩa của các nút trên thanh công cụ tính từ trái sang phải:

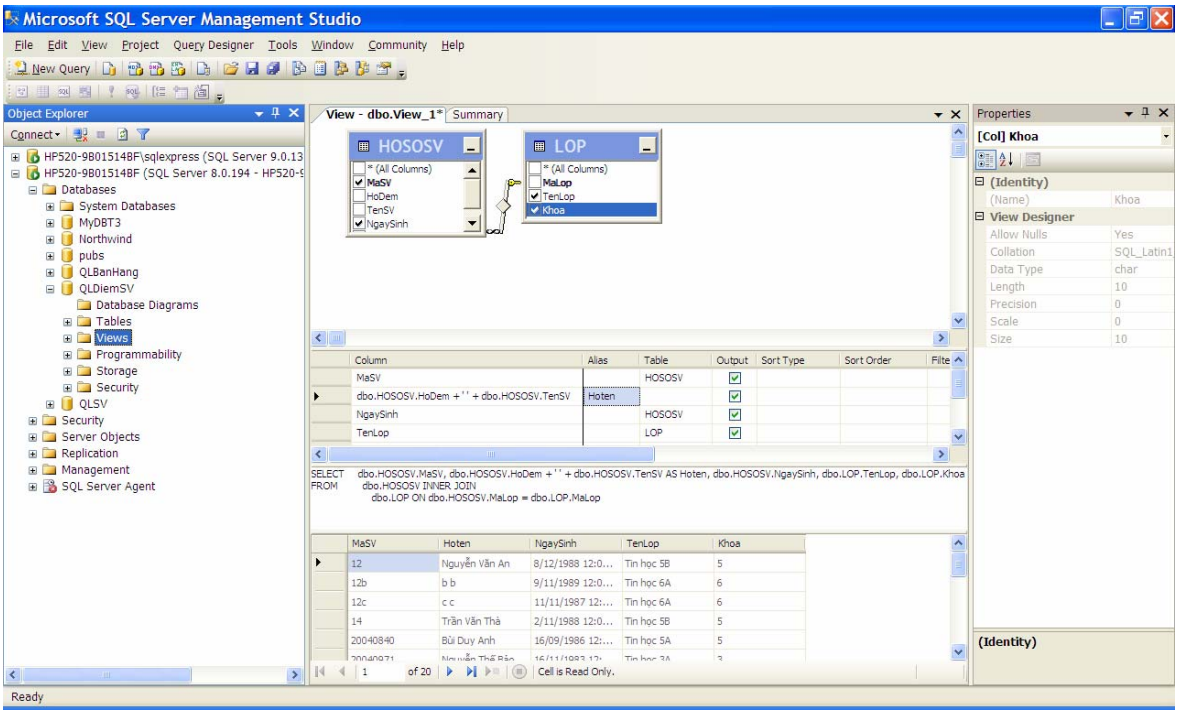
- Đối với SQL Server 2000:
  + *Save*: Lưu view.
  + *Properties*: Cho phép thay đổi thuộc tính của view.
  + *Show hide pane*: các nút ẩn hoặc hiện các cửa sổ trên.
  + *Run*: Thực thi và hiển thị kết quả của view trong Results pane.
  + *Cancel Execution And Clear Results*: Xóa Results pane.
  + *Verify SQL*: Kiểm tra cú pháp của phát biểu SQL.
  + *Remove Filter*: loại bỏ tất cả các lọc dữ liệu đã được định nghĩa.
  + *Use GROUP BY*: Thêm mệnh đề group by trong câu lệnh SQL.
  + *Add Table*: Thêm các bảng, view làm nguồn dữ liệu cho view mới.
- Đối với SQL Server 2005:
  + *Show Diagram pane*: Ẩn hoặc hiện cửa sổ Diagram pane.
  + *Show Criteria pane*: Ẩn hoặc hiện cửa sổ Criteria pane.
  + *Show SQL pane*: Ẩn hoặc hiện cửa sổ SQL pane.
  + *Show Results pane*: Ẩn hoặc hiện cửa sổ kết quả.
  + *Execute SQL*: Thực hiện truy vấn.
  + *Verify SQL Syntax*: Kiểm tra cú pháp câu lệnh SQL.
  + *Add Group By*: Thêm mệnh đề group by trong câu lệnh SQL.

+ *Add Table*: Thêm các bảng, view, hàm làm nguồn dữ liệu cho view mới.

- Click vào nút Add Table, xuất hiện hộp thoại Add Table chọn các bảng các view làm nguồn dữ liệu cho view mới. Trong ví dụ ta chọn bảng HOSOSV và LOP.
- Sau khi chọn nguồn dữ liệu ta thực hiện chọn các trường làm dữ liệu trên view như hình 3.30 (SQL Server 2000) và 3.31 (SQL Server 2005)
- Click vào nút Save, xuất hiện hộp thoại Save lưu với tên Danhsach.



Hình 3.30. *Cửa sổ thiết kế view Danhsach trong SQL Server 2000 .*



Hình 3.31. *Cửa sổ thiết kế view Danhsach trong SQL Server 2005.*

Sử dụng vùng Grid Pane (Criteria) để hỗ trợ cho việc thiết kế View:

+ *Cột Column*: Chứa tên các trường trong bảng\view làm nguồn dữ liệu cho truy vấn này hoặc chứa một biểu thức định nghĩa một trường mới cho view.
+ *Cột Alias*: Chỉ định bí danh cho trường trong view mới.
+ *Cột Table*: Chỉ định nguồn dữ liệu.
+ *Cột Output*: Chỉ định hiển thị hay không hiển thị trường đó.
+ *Cột Sort Type và Sort Order*: Dùng để sắp xếp dữ liệu.
+ *Cột Criteria*: Dùng để đặt điều kiện lọc cho các bản ghi.
+ *Các cột Or*: Dùng kết hợp với cột Criteria để tạo các điều kiện lọc dữ liệu phức tạp.

Tính tổng trong truy vấn: Để tính tổng trong truy vấn dùng vùng Grid Pane ta có thể tiến hành phân nhóm các bản ghi và thực hiện tính toán trên từng phân nhóm đó. Để tính tổng trong truy vấn ta chọn nút Use Group By

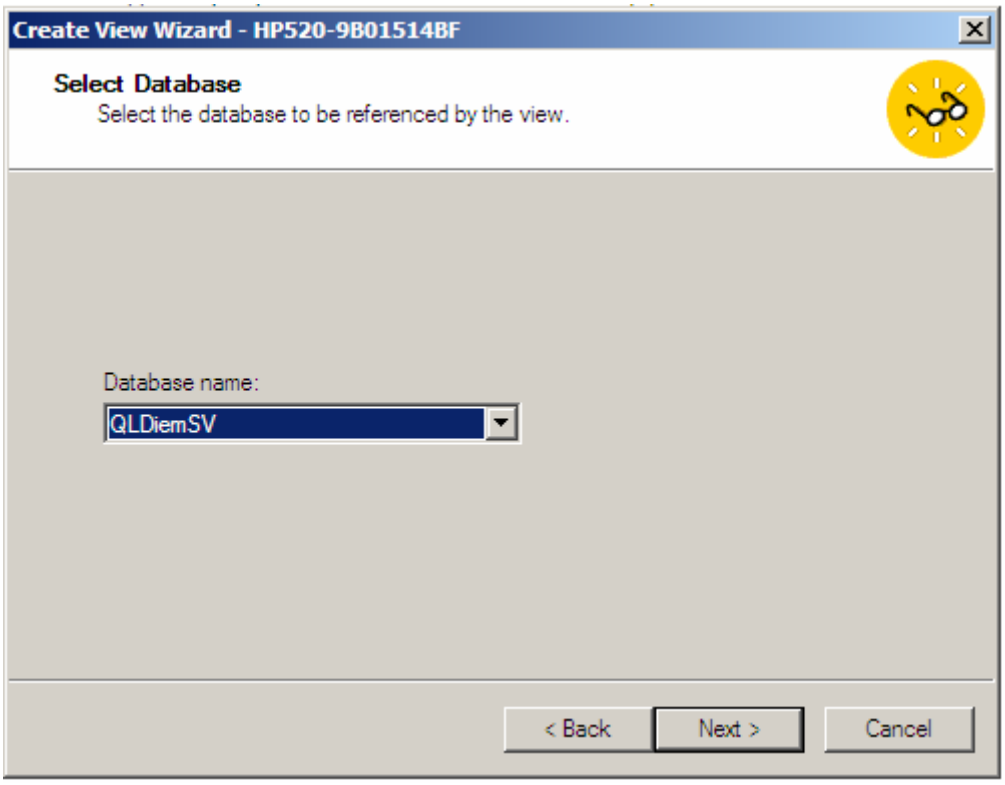
xuất hiện cột Group By trong vùng Grid Pane. Ta thực hiện tiến hành phân các nhóm trường như sau:

+ Trường làm điều kiện, tiêu chuẩn tham gia phân nhóm và tính tổng: Chọn Where trong cột Group By và đặt biểu thức điều kiện trong cột Creteria.
+ Trường phân nhóm: Chọn Group by trong cột Group By.
+ Trường tính toán: Chọn một hàm có sẵn (sum, count, avg, max, min, .v.v...) trong cột Group By hoặc xây dựng một biểu thức tính toán trong cột Column.
+ Định tiêu chuẩn hiển thị kết quả: Đặt điều kiện ở cột Criteria tại các trường phân nhóm và trường tính toán.
+ Chọn thứ tự hiển thị: Dùng cột Sort Type và Sort Order tại các các trường phân nhóm và các trường tính toán.

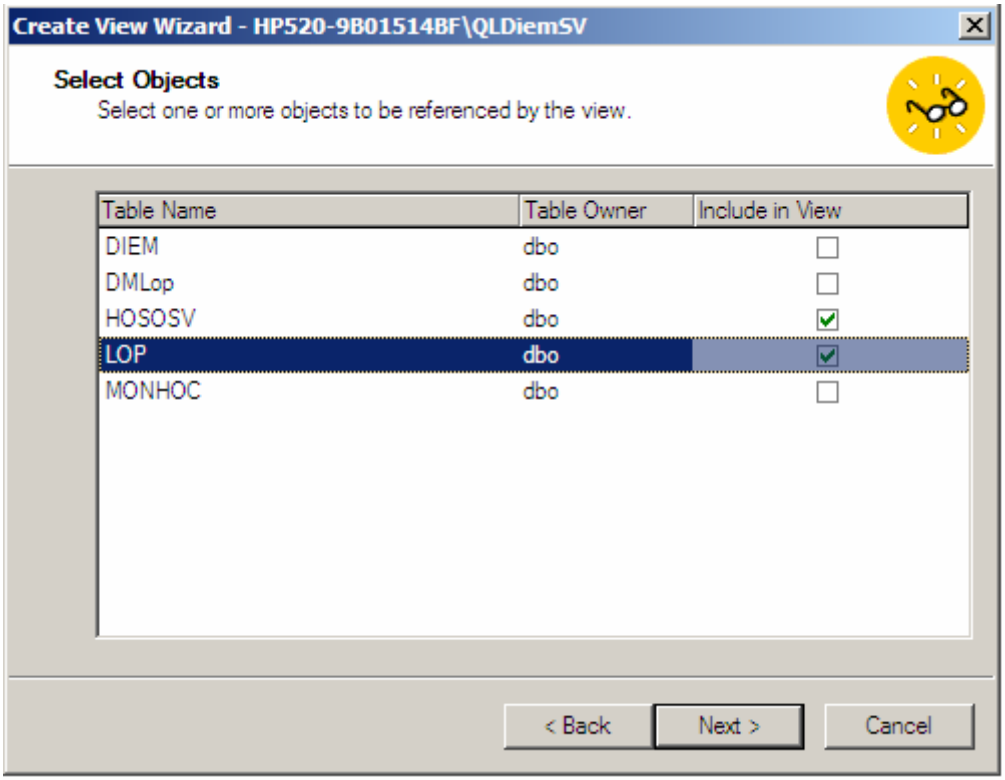
** Dùng Create view wizard*

- Khởi động SQL Server Enterprise Manager, chọn tên server cục bộ và vào menu Tools\Wizards. Xuất hiện cửa sổ Select Wizard hình 3.5.
- Chọn Create View Wizard xuất hiện cửa sổ Welcometo the Create View Wizard, như hình 3.32. Chọn Next.
- Xuất hiện cửa sổ Select Database (Hình 3.33) chọn CSDL mà view được tạo trên đó. Chọn Next.
- Trong cửa sổ Select Objects chọn các bảng làm nguồn dữ liệu cho View mới (Hình 3.34). Chọn Next.
- Chọn các cột dữ liệu của view (hình 3.35).
- Cửa sổ tiếp theo (hình 3.36): Đặt điều kiện lọc tương tự như mệnh đề WHERE trong khối câu lệnh SELECT. Chọn Next.
- Chọn tên view (hình 3.37) và chọn Finish (hình 3.38) để kết thúc.

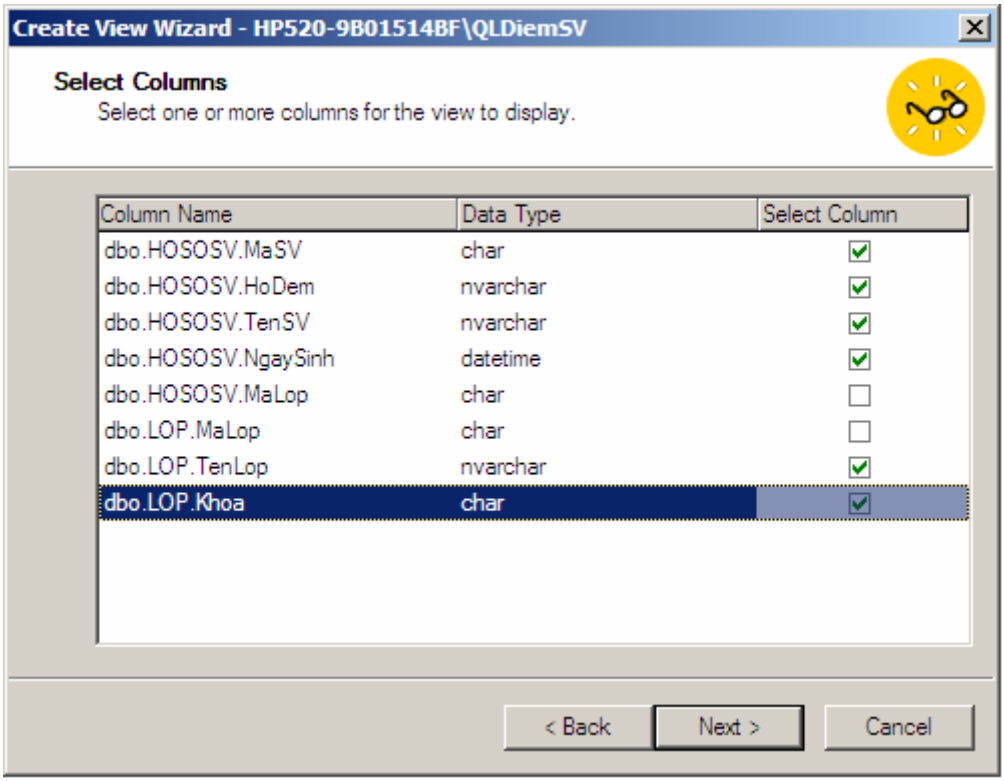
Hình 3.32.*Cửa sổ Welcometo the Create View Wizard.*



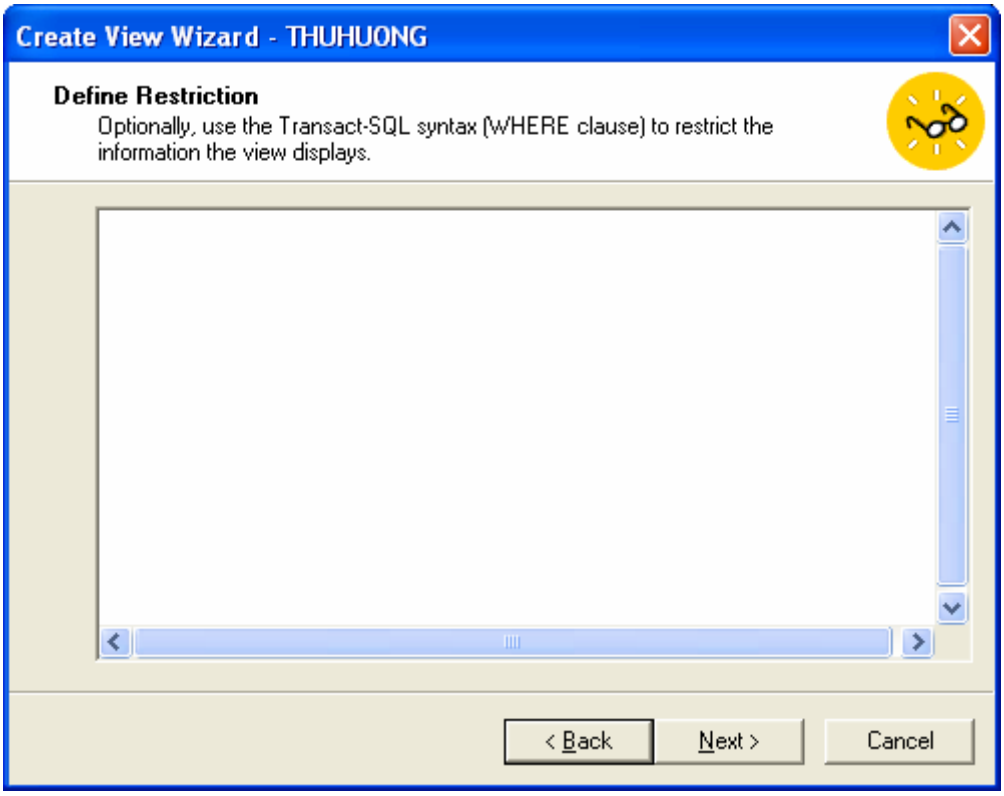
Hình 3.33.*Cửa sổ Select Database.*



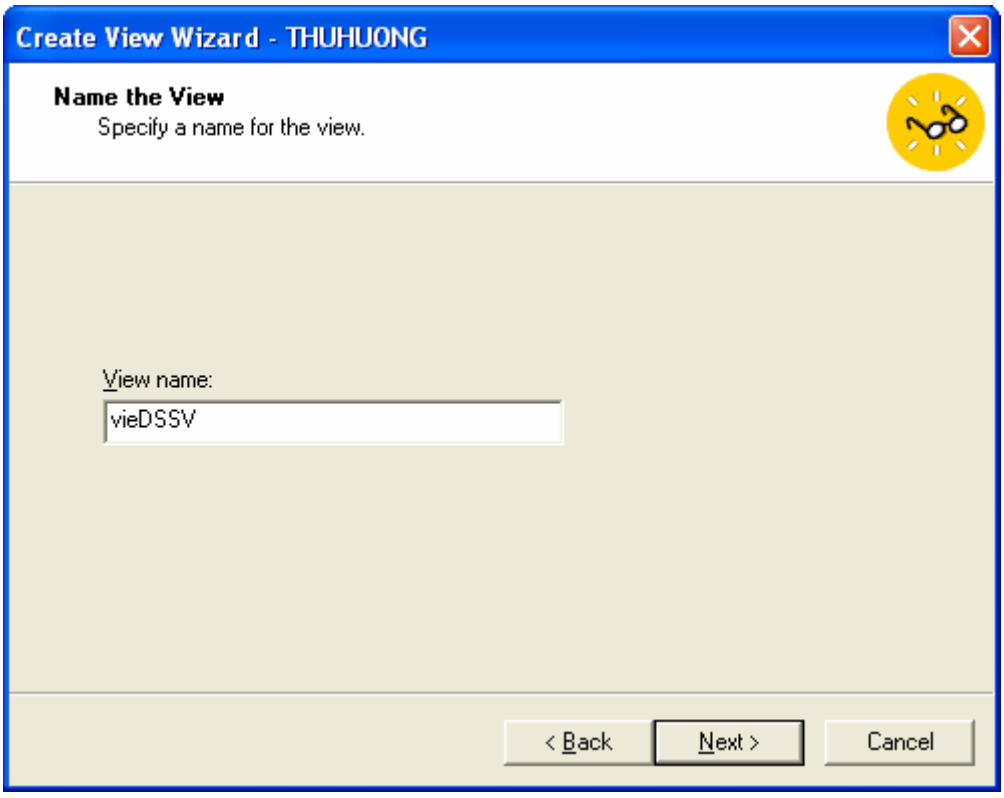
Hình 3.34.*Cửa sổ Select Objects.*



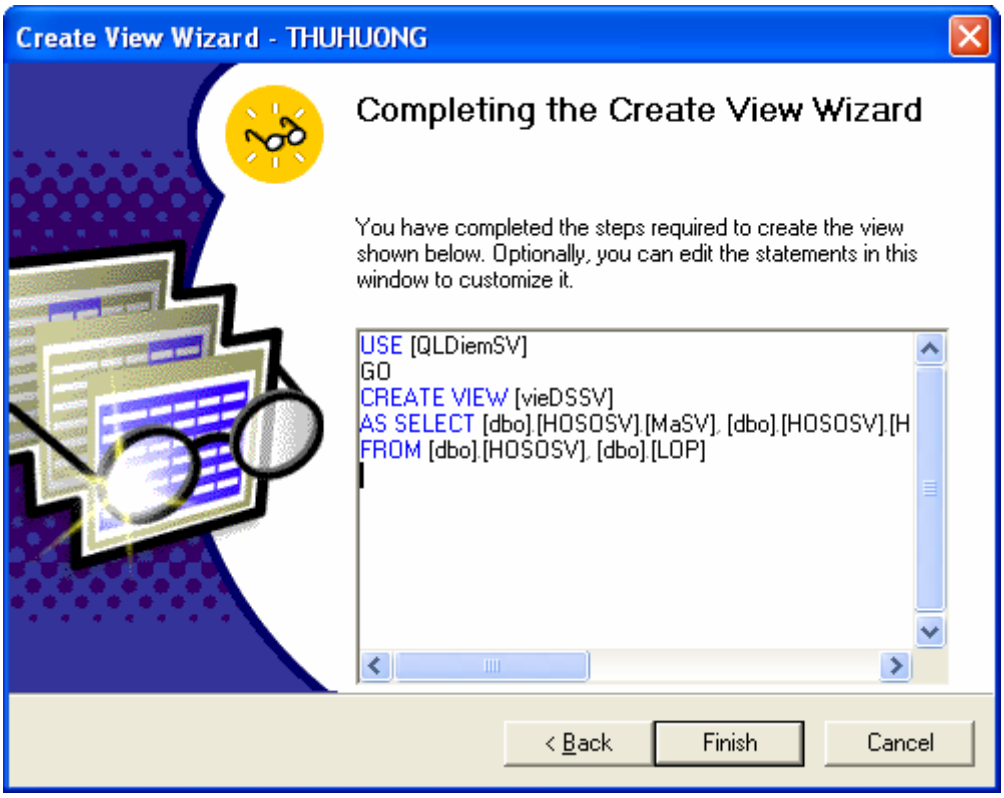
Hình 3.35.*Cửa sổ Select Columns.*



Hình 3.36.*Cửa sổ Define Restriction*



Hình 3.37.*Cửa sổ Define Restriction*



Hình 3.38.*Cửa sổ Define Restriction*

** Dùng T-SQL:* Ta dùng cú pháp câu lệnh sau:

```
CREATE VIEW [schema_name.]view_name[(column[,...n])]
[ WITH <view_attribute> [ ,...n ] ]
AS select_statement [ ; ]
[ WITH CHECK OPTION ]

<view_attribute> ::=
{    [ ENCRYPTION ]
     [ SCHEMABINDING ]
     [ VIEW_METADATA ]      }
```

Trong đó:

schema_name

Là tên của lược đồ mà view thuộc lược đồ đó.

view_name

Là tên của view.

column

Là tên được sử dụng cho một cột trong view. Tên một cột chỉ yêu cầu khi cột đó được sinh ra từ một biểu thức đại số, một hàm, một hằng; khi hai hoặc nhiều cột khác nhau có cùng tên, điển hình là trong liên kết; hoặc khi một cột trong view được chỉ định tên khác với tên từ nguồn dữ liệu. Các tên cột có thể được gán trong câu lệnh SELECT. Nếu cột không được chỉ định trong view thì các cột trong view có tên cùng tên với các cột câu lệnh SELECT.

select_statement

Là các khối câu lệnh SELECT định nghĩa view. Các khối câu lệnh SELECT được phân cách nhau bởi mệnh đề UNION hoặc UNION ALL.

Một Index được định nghĩa trên view đòi hỏi view phải được xây dựng từ một bảng đơn hoặc nhiều bảng liên kết nhau trong tổ hợp tùy chọn.

CHECK OPTION

Bắt buộc tất các câu lệnh sửa đổi dữ liệu thực hiện trên view phải tuân theo các điều kiện trong khối câu lệnh select_statement.

ENCRYPTION

Mã hóa bản text của câu lệnh CREATE VIEW trong sys.syscomments. Việc sử dụng WITH ENCRYPTION ngăn cản view bị công bố như là một phần bản sao SQL Server.

SCHEMABINDING

Buộc view vào một lược đồ dưới một bảng hoặc nhiều bảng. Khi SCHEMABINDING được chỉ định, bảng hoặc các bảng cơ cở không được sửa trong những cách ảnh hưởng đến định nghĩa view.

VIEW_METADATA

Chỉ định các thể hiện SQL Server sẽ trả về cho DB-Library, ODBC, và OLE DB APIs thông tin siêu dữ liệu về view, thay cho bảng và các bảng cơ sở khi trình duyệt siêu dữ liệu được yêu cầu cho một truy vấn tham chiếu đến view.

Ví dụ 3.4. Trong CSDL *Northwind* có hai bảng *Orders* và *Order Details*. Ta xây dựng view tính tổng giá trị cho từng hóa đơn (SQL Server 2000).

```
Create View TongGT
as
SELECT dbo.Orders.OrderID,
       SUM(dbo.[Order Details].UnitPrice *
dbo.[Order Details].Quantity) AS Tolal
FROM   dbo.Orders INNER JOIN
       dbo.[Order Details] ON dbo.Orders.OrderID
= dbo.[Order Details].OrderID
GROUP BY dbo.Orders.OrderID
Go
```

c) Thay đổi view

** Dùng Enterprise Manager*

- Trong Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn tạo view, chẳng hạn CSDL QLDiemSV và chọn mụcViews.
- Right Click lên View, chọn Design View. Xuất hiện cửa sổ thiết kế view.
- Thực hiện các thay đổi trên view và ghi lại các thay đổi đó.

** Dùng SQL Server Management Studio*

- Trong SQL Server Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn tạo view, chẳng hạn CSDL QLDiemSV và chọn mụcViews.
- Right Click lên View, chọn Modify. Xuất hiện cửa sổ thiết kế view.
- Thực hiện các thay đổi trên view và ghi lại các thay đổi đó.

** Dùng T-SQL:* Ta dùng cú pháp câu lệnh sau

```
ALTER VIEW [schema_name.]view_name
[(column[,...n])]
[ WITH <view_attribute> [ ,...n ] ]
AS select_statement [ ; ]
[ WITH CHECK OPTION ]

<view_attribute> ::=
{[ENCRYPTION][SCHEMABINDING][VIEW_METADATA]
}
```

Ví dụ 3.5. Sửa đổi view DSSV như sau:

```
ALTER VIEW DSSV
AS
    Select  MaSV, Hodem + ' '+ TenSV AS Hoten, Ngaysinh
    From HosoSV
GO
```

d) Xóa view

- Trong cửa sổ Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn xóa view, chẳng hạn CSDL QLDiemSV và chọn mụcViews. Sau đó right click lên View muốn xóa, chọn Delete.

- Trong cửa sổ SQL Server Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn xóa view, chẳng hạn CSDL QLDiemSV và chọn mụcViews. Sau đó right click lên View muốn xóa, chọn Delete xuất hiện cửa sổ xác nhận xóa chọn OK.

- Dùng T-SQL dùng lệnh DROP VIEW theo cú pháp:

```
DROP VIEW view_name
```

Ví dụ 3.6. Xóa view DSSV như sau:

```
DROP VIEW DSSV
```

3.2.4. Chỉ mục - Index

a) Khái niệm chỉ mục

Chỉ mục là một trong những công cụ mạnh có sẵn đối với người thiết kế CSDL. Một chỉ mục là một cấu trúc phụ cho phép cải thiện hiệu suất thực thi các truy vấn bằng cách giảm thiểu các hoạt động nhập/xuất dữ liệu cần thiết để được dữ liệu yêu cầu. Tùy thuộc vào kiểu của nó, mà chỉ mục được lưu với dữ liệu hoặc tách biệt với dữ liệu.

- *Chỉ mục khóa:* Chỉ mục khóa chỉ dõ cột hoặc các cột được dùng để sinh ra chỉ mục. Nó cho phép tìm nhanh chóng dòng dữ liệu muốn tìm. Để truy cập dòng dữ liệu dùng chỉ mục, ta chỉ cần đưa ra giá trị khóa của chỉ mục hoặc đưa các giá trị vào mệnh đề WHERE trong khối câu lệnh SELECT.

- *Chỉ mục duy nhất*: Là chỉ mục chỉ chứa một dòng dữ liệu cho mỗi khóa chỉ mục. Một chỉ mục là duy nhất nếu bản thân dữ liệu là duy nhất, nếu không duy nhất ta có thể tạo chỉ mục kết hợp trên nhiều cột để đạt được chỉ mục duy nhất.

- *Các kiểu chỉ mục:* Các chỉ mục được lưu trữ dữ liệu dưới dạng cây nhị phân B-Tree. Có hai kiểu chỉ mục

- Chỉ mục liên cung (clustered): Là chỉ mục lưu trữ các dòng dữ liệu thực sự của bảng trong nút lá, theo thứ tự đã được sắp xếp.
- Chỉ mục phi liên cung (Nonclustered): Không chứa dữ liệu trong nút lá, mà nó chứa thông tin về vị trí của dòng dữ liệu: nếu không có chỉ mục liên cung trên bảng thì nó chứa số nhận dạng dòng (Row ID); nếu có chỉ mục liên cung thì trong nút lá này sẽ chứa giá trị khóa chỉ mục liên cung cho dữ liệu đó.

Chú ý: Indexes được tạo tự động khi các ràng buộc PRIMARY KEY và UNIQUE được định nghĩa trên các cột của bảng

b) Tạo chỉ mục

Để tạo Indexes trong SQL Server 2000 , ta có 3 cách khác nhau để tạo:

+ Sử dụng Create View Wizard
+ SQL Server Enterprise Manager
+ Dùng T-SQL

Trong SQL Server 2005, ta có 2 cách khác nhau để tạo:

+ SQL Server Management Studio
+ Dùng T-SQL

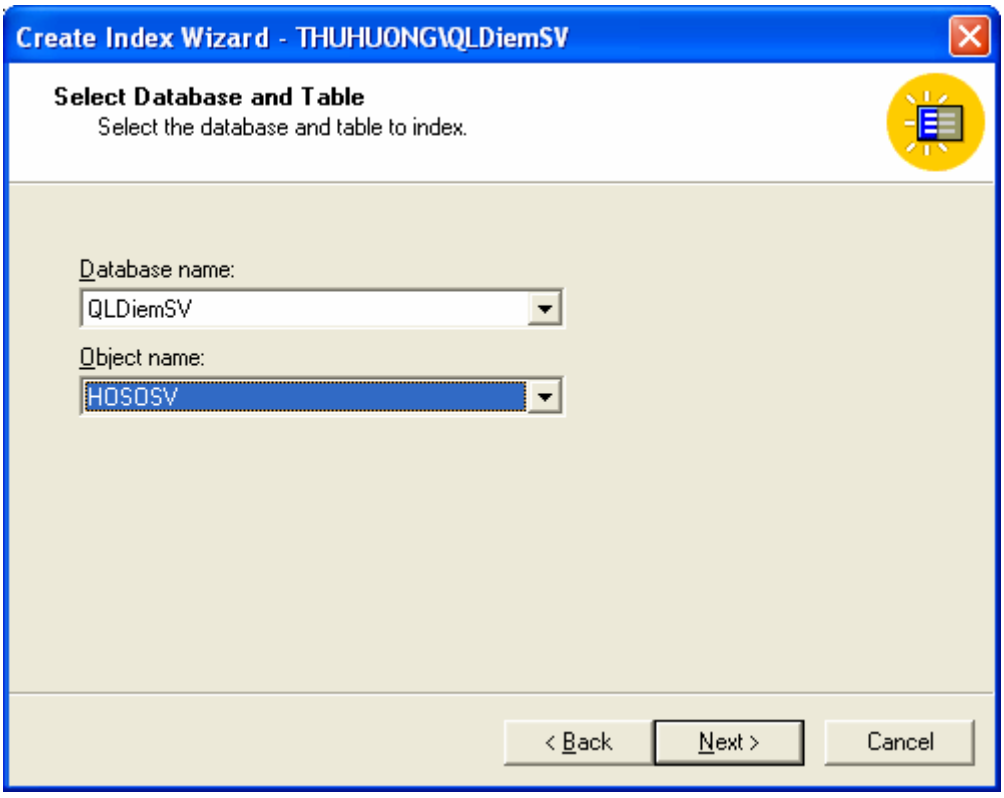
** Dùng wizard*

- Khởi động SQL Server Enterprise Manager, chọn tên server cục bộ và vào menu *Tools\Wizards*. Xuất hiện cửa sổ Select Wizard hình 3.5.
- Chọn Create Index Wizard xuất hiện cửa sổ *Welcometo the Create Index Wizard*, như hình 3.39. Chọn Next.
- Xuất hiện cửa sổ *Select Database and Table (hình 3.40)*. Chọn CSDL và bảng dữ liệu cần tạo index.
    + Mục Database name: Chọn cơ sở dữ liệu
    + Mục Object name: Chọn bảng dữ liệu

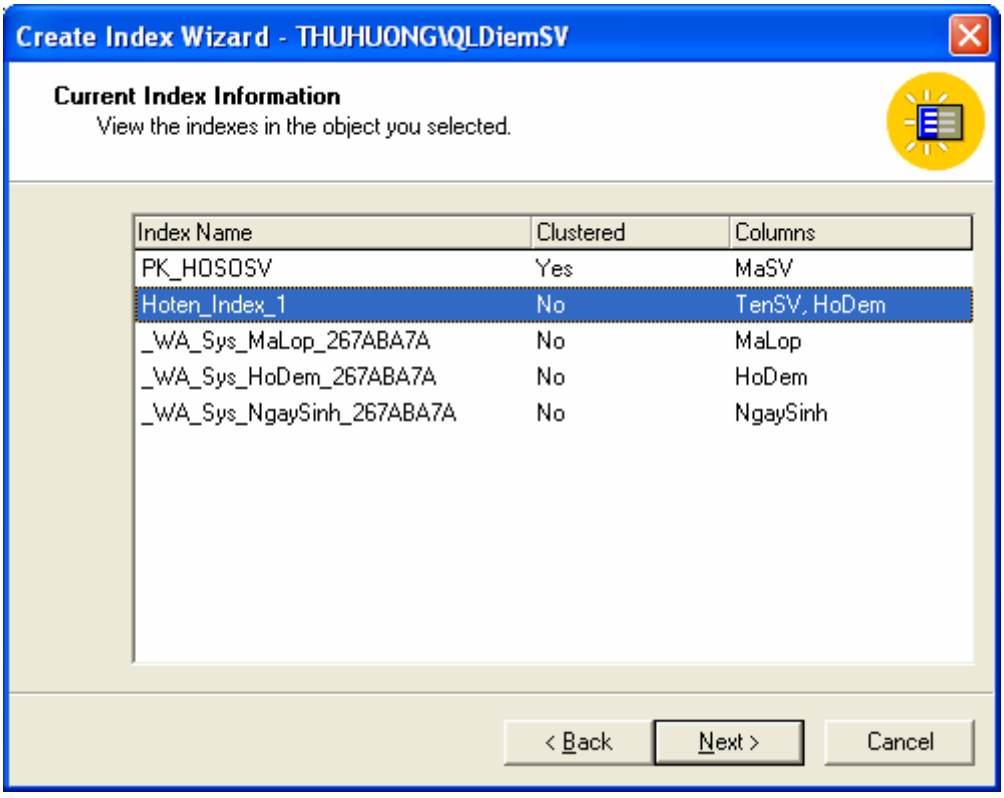
Sau đó chọn Next.

- Xuất hiện cửa sổ *Current index information (hình 3.41)* đưa thông tin các Index đã được tạo trước đó.
- Cửa sổ tiếp theo là cửa sổ *Select columns (hình 3.42).* Trong cửa sổ này ta chọn các cột để tạo chỉ mục. Chọn Next.
- Xuất hiện cửa sổ *Specify Index Option (hình 3.43).* Chỉ định các tham số cho Index như: Tạo chỉ mục duy nhất, tối ưu,.v.v…. Chọn Next.
- Xuất hiện cửa sổ Completing the Create Index Wizard (hình 3.44). Chọn thứ tự, đặt tên cho Index. Cuối cùng chọn Finish để kết thúc.

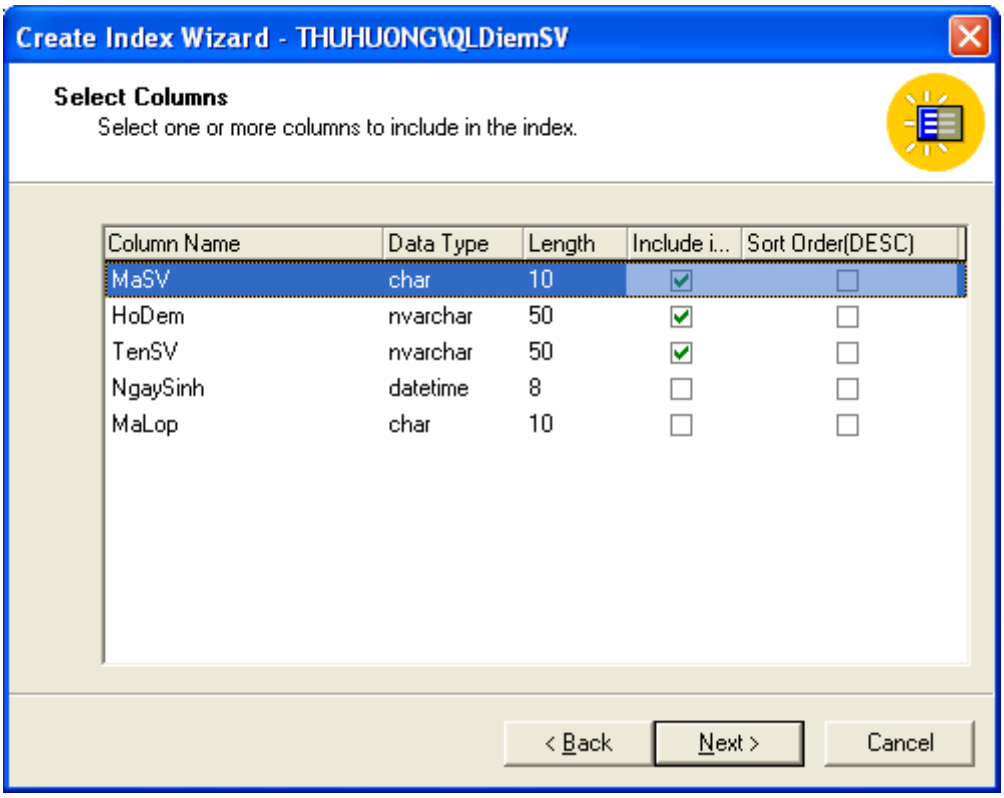
Hình 3.39.*Cửa sổ Welcometo the Create Index Wizard.*



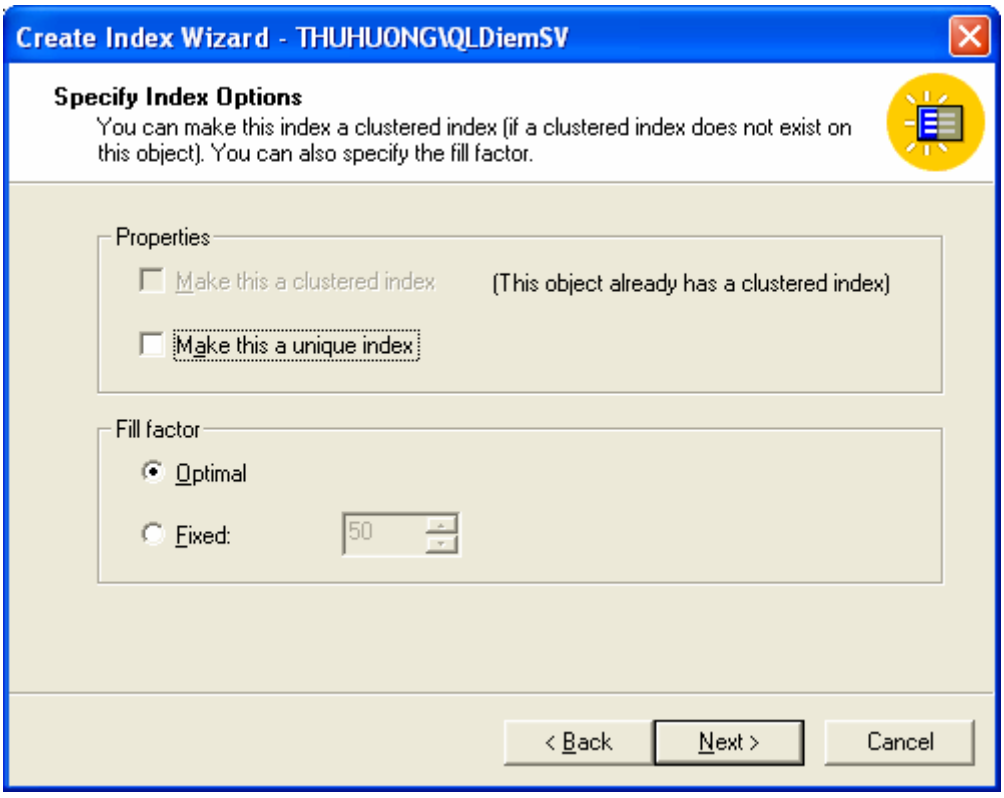
Hình 3.40.*Cửa sổ Select Database and Table.*



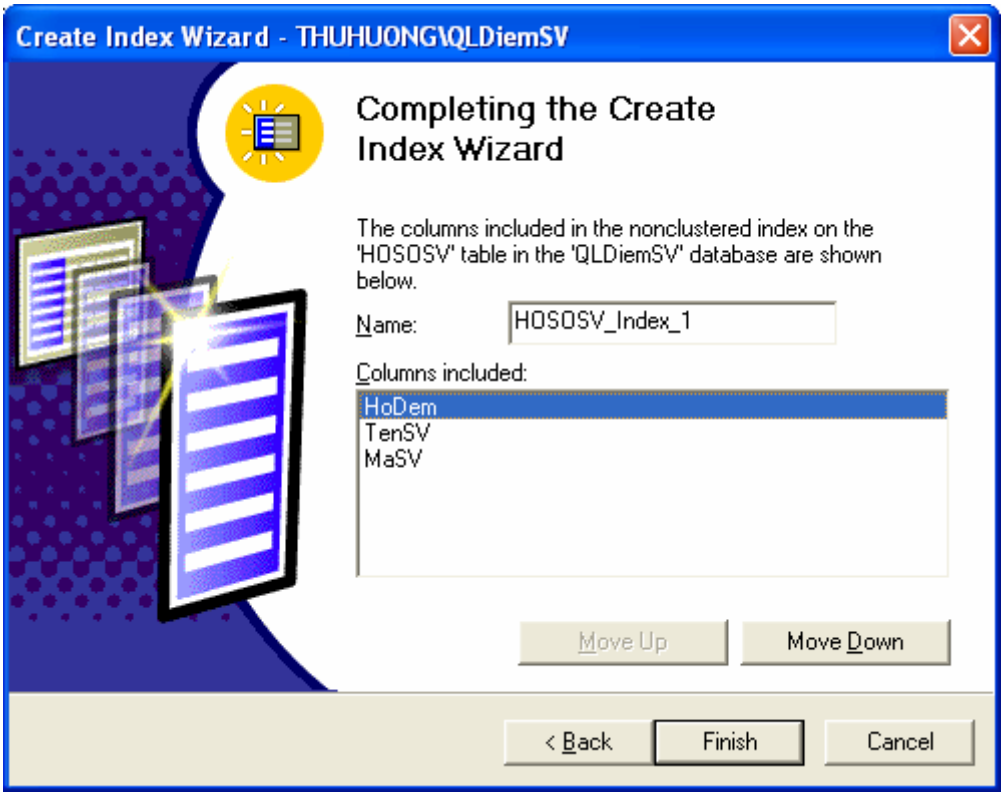
Hình 3.41.*Cửa sổ Current index information.*



Hình 3.42.*Cửa sổ select columns.*



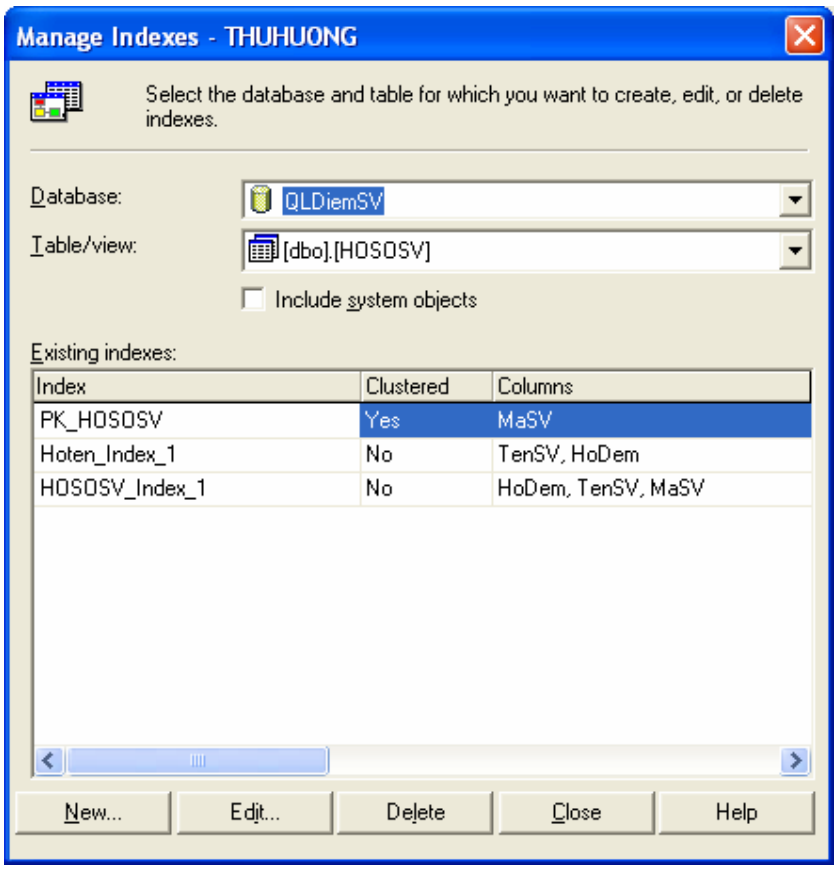
Hình 3.43. *Cửa sổ Specify Index Option.*



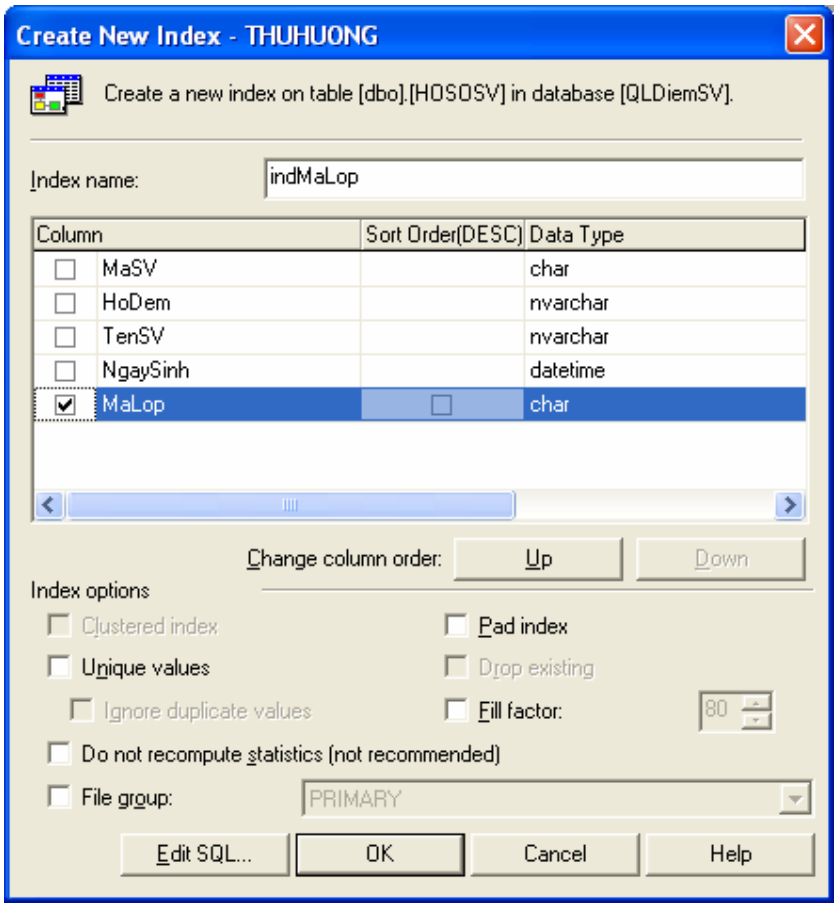
Hình 3.44. *Cửa sổ Completing the Create Index Wizard*

** Dùng Enterprise Manager:*

- Trong cửa sổ Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn tạo Index, chẳng hạn CSDL QLDiemSV và chọn mục Tables. Sau đó right click lên bảng muốn tạo chỉ mục. Chọn All Task\Manage Indexes. Xuất hiện của sổ Manage Indexes như hình 3.45.
- Trong cửa sổ này chọn CSDL trong mục Database, chọn bảng hoặc View trong mục Table/view và sau đó click nút New xuất hiện cửa sổ Create New Index hình 3.46.
- Trong cửa sổ này ta nhập tên của Index trong mục Index name và chọn các trường của Index trong cột Column.



Hình 3.45. *Cửa sổ Manage Indexes*



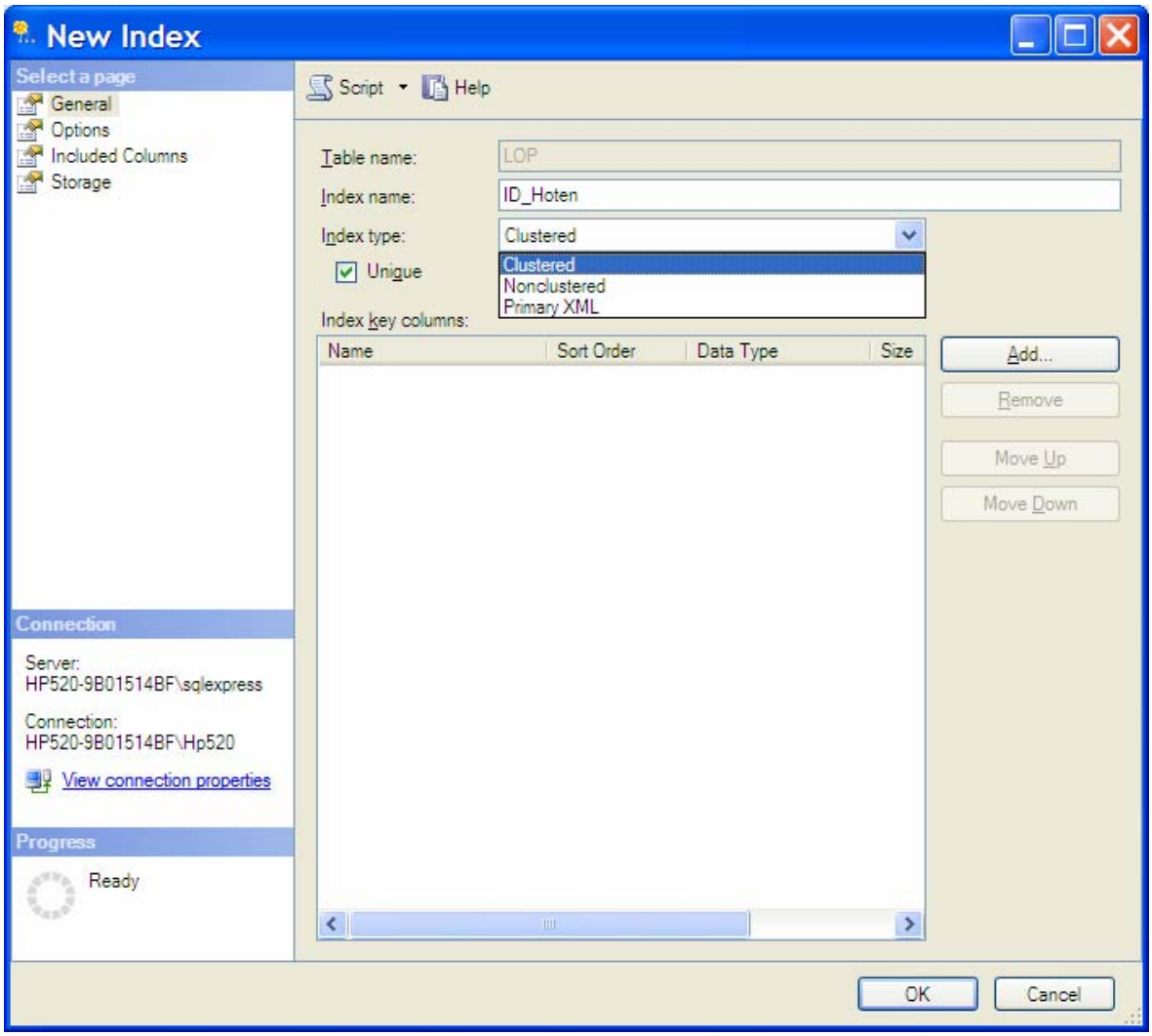
Hình 3.46. *Cửa sổ Create New Index*

Ngoài ra, trong cửa sổ 3.45 Manage Indexes ta có các nút thao tác quản lý chỉ mục (Index):

- Nút New: Dùng để thêm một Index mới.
- Nút Edit: Chọn Index đã có trong danh sách và chọn nút để Edit để thực hiện chỉnh sửa chỉ mục đã có.
- Nút Delete: Chọn Index đã có trong danh sách và chọn nút để Delete để thực hiện xóa chỉ mục đã chọn.

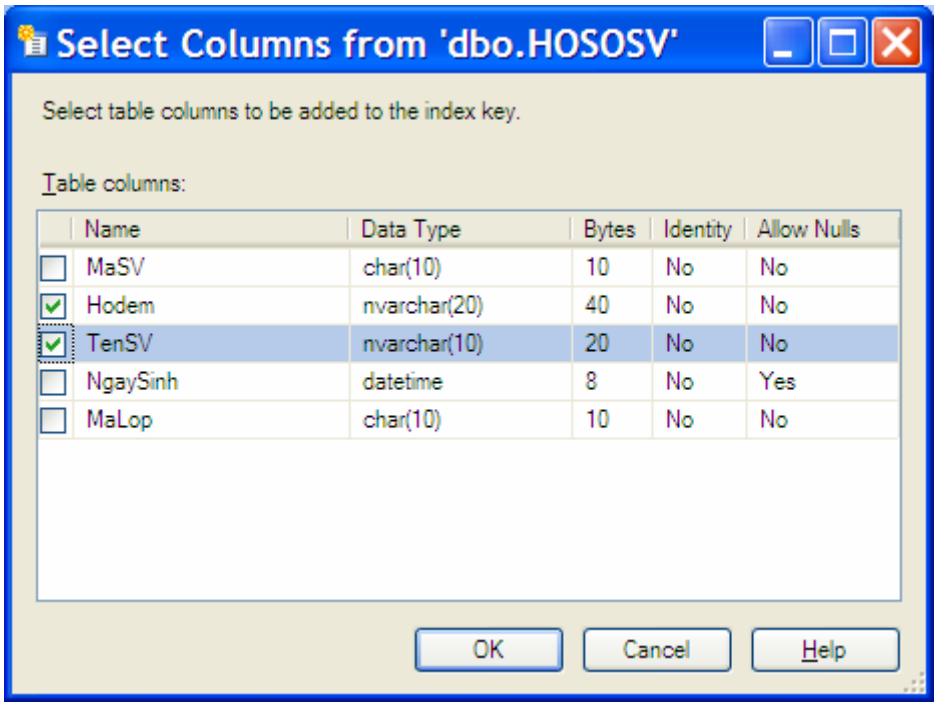
** Dùng SQL Server Management Studio*

- Trong SQL Server Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu chứa bảng hoặc view muốn tạo chỉ số.



Hình 3.47. *Cửa sổ New Index*

- Mở rộng mục Table hoặc View, mở rộng bảng hoặc view muốn tạo Index. Right Click lên thư mục Indexes chọn New Index, xuất hiện cửa sổ New Index như hình 3.47. gồm các tham số:
  + Table name (hoặc View name đối với view): Tên bảng (hoặc tên view) mà tệp chỉ số được xây dựng trên đó.
  + Index name: Đặt tên tệp chỉ số muốn xây dựng.
  + Index type: Kiểu của index (Clustered, Nonclustered)
  + Unique: Tùy chọn cho phép tạo tệp chỉ số này có là tệp chỉ số duy nhất hay không?
  + Index key columns: Xác định các trường khóa của index bằng cách click nút Add xuất hiện cửa sổ Select Columns Hình 3.48, ta chọn các cột làm khóa cho Index.



Hình 3.48. *Cửa sổ Select Columns*

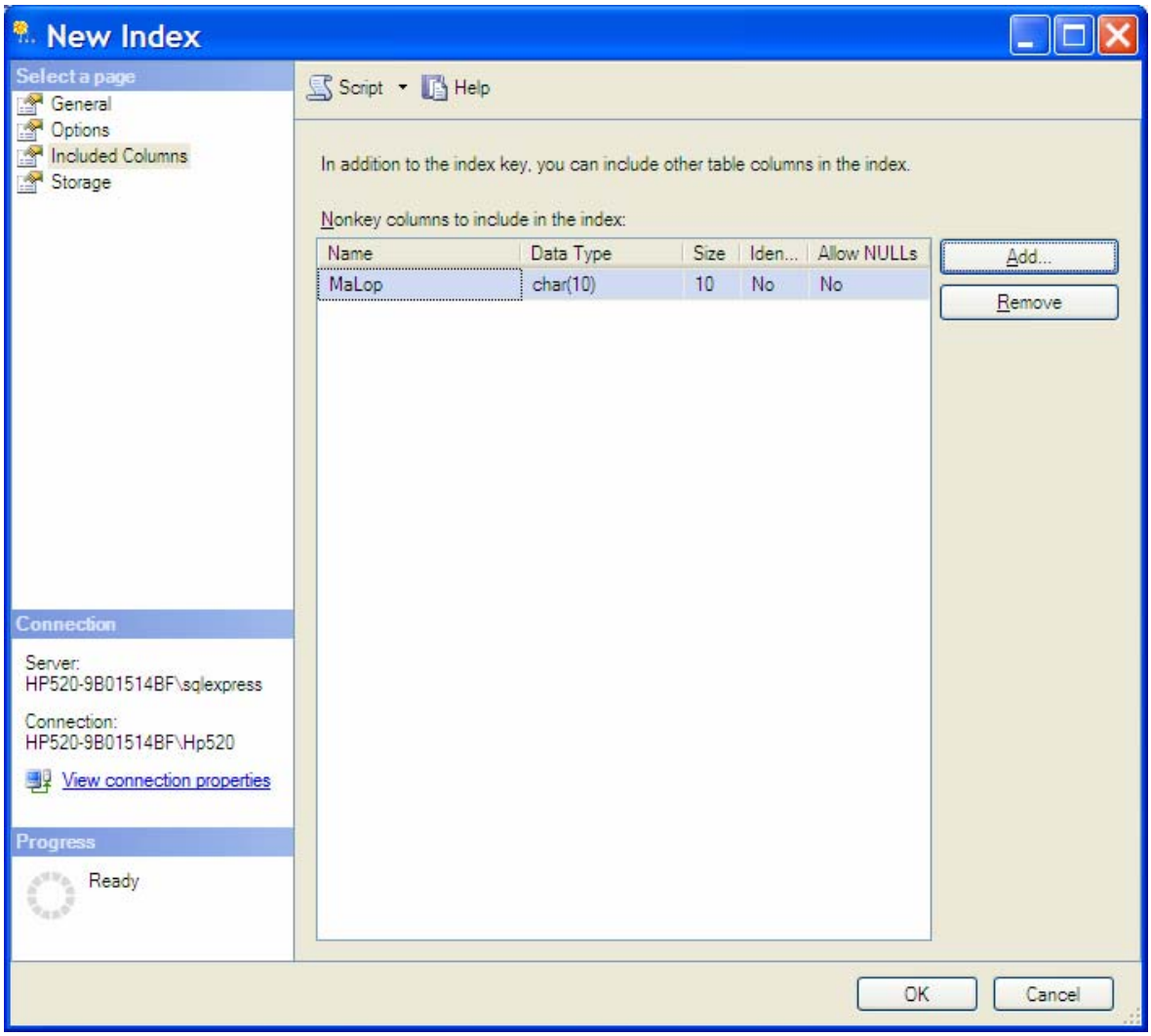
Chú ý: Trong SQL Server 2005, chúng ta có thể mở rộng chức năng của các tệp chỉ số phi liên cung (nonclustered indexes) bằng cách thêm các cột không là khóa (nonkey columns) vào các nút lá của cây nonclustered index. Các cột khóa được lưu trữ tại tất cả các mức còn các cột không khóa chỉ lưu trữ tại mức lá của index. Bằng việc thêm các cột không khóa, ta có thể tạo các

tệp chỉ số phi liên cung phủ nhiều truy vấn hơn bởi vì các cột không khóa có các lợi ích sau:

+ Chúng có thể là các cột có kiểu dữ liệu không được phép làm các cột khóa trong index.
+ Chúng không được Database Engine xét khi tính đến số các cột khóa của Index hay kích thước khóa của index.

Một index được bao gồm tất cả các cột không khóa có thể cải thiện đáng kể sự thực thi truy vấn khi tất cả các cột trong được bao gồm trong index (cả các cột khóa và không khóa của index).

Để thêm các cột không khóa ta chọn trang Included Columns (Hình 3.49). Sau đó click nút Add để xuất hiện cửa sổ Select Columns (Hình 3.48).



Hình 3.49. *Cửa sổ New Index*

** Dùng T - SQL*

- Trên SQL Server 2000:

```
CREATE [ UNIQUE][CLUSTERED|NONCLUSTERED]INDEX
index_name
    ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )
[ WITH < index_option > [ ,...n] ]
[ ON filegroup ]

< index_option > :: =
    { PAD_INDEX |
        FILLFACTOR = fillfactor |
        IGNORE_DUP_KEY |
        DROP_EXISTING |
    STATISTICS_NORECOMPUTE |
    SORT_IN_TEMPDB
    }
```

- Trên SQL Server 2005:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX
index_name
    ON <object> ( column [ ASC | DESC ] [ ,...n ] )
    [ INCLUDE ( column_name [ ,...n ] ) ]
    [ WITH ( <relational_index_option> [ ,...n ] ) ]
    [ ON { filegroup_name|default }
    ]
[ ; ]

<object> ::=
{
    [ database_name. [ schema_name ] . | schema_name. ]
        table_or_view_name
}

<relational_index_option> ::=
{
    PAD_INDEX  = { ON | OFF }
  | FILLFACTOR = fillfactor
  | SORT_IN_TEMPDB = { ON | OFF }
  | IGNORE_DUP_KEY = { ON | OFF }
  | STATISTICS_NORECOMPUTE = { ON | OFF }
  | DROP_EXISTING = { ON | OFF }
  | ONLINE = { ON | OFF }
  | ALLOW_ROW_LOCKS = { ON | OFF }
  | ALLOW_PAGE_LOCKS = { ON | OFF }
  | MAXDOP = max_degree_of_parallelism
 }
```

Các tham số trong đó:

UNIQUE

Chỉ định tạo một unique index trên bảng hoặc trên view. Một clustered index trên view buộc phải là unique.

CLUSTERED

Chỉ định tạo chỉ mục liên cung.

NONCLUSTERED

Chỉ định tạo chỉ mục phi liên cung. Mặc định là chỉ mục NONCLUSTERED.

index_name

Là tên của tệp chỉ số.

column

Là tên cột hoặc các cột mà index dựa trên đó.

INCLUDE (column [,... n])

Chỉ định các cột không khóa được thêm vào mức lá của chỉ mục phi liên cung.

ON filegroup_name

Tạo index trên filegroup chỉ định. Nếu không có chỉ định này thì index sử dụng cùng filegroup mà table hoặc view dựa trên.

ON "default"

Tạo index dựa trên filegroup mặc định.

Ví dụ 3.7. Tạo Index trên bảng LOP của CSDL QLDiemSV

```
Use QLDiemSV
create Unique index indTenLopind On LOp(TenLop)
```

Ví dụ 3.8. Xây dựng lại Index `TenLop_ind` trên bảng LOP của CSDL QLDiemSV

```
Use QLDiemSV
create Unique index TenLop_ind
    On LOp(TenLop)
    With DROP_EXISTING
```

Ví dụ 3.9. Dùng từ khóa `DBCC DBREINDEX`

```
Use QLDiemSV
Go
DBCC DBREINDEX(HOSOSV)
```

c) Loại bỏ chỉ mục

+ *Dùng Enterprise Manager:* Trong cửa sổ Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn xóa Index, chẳng hạn CSDL QLDiemSV và chọn mục Tables. Sau đó right click lên bảng muốn tạo chỉ mục. Chọn All Task\Manage Indexes. Xuất hiện của sổ Manage Indexes như hình 3.45. Chọn Index muốn xóa và chọn Delete.
+ *Dùng SQL Server Management Studio:* Trong SQL Server Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu chứa bảng hoặc view muốn xóa chỉ số. Mở rộng mục Indexes của bảng hoặc view đó, right click lên Index muốn xóa và chọn Delete.
+ *Dùng lệnh* T-SQL:

```
DROP INDEX <table.index> | <view.index>
```

Chú ý: Đối với SQL Server 2005 ta có thể sử dụng cú pháp sau:

```
DROP INDEX index_name ON <Table|View>
```

Ví dụ 3.10. Loại bỏ chỉ mục `TenLop_ind`

```
Use QLDiemSV
Go
DROP INDEX Lop. indTenLop
Go
```

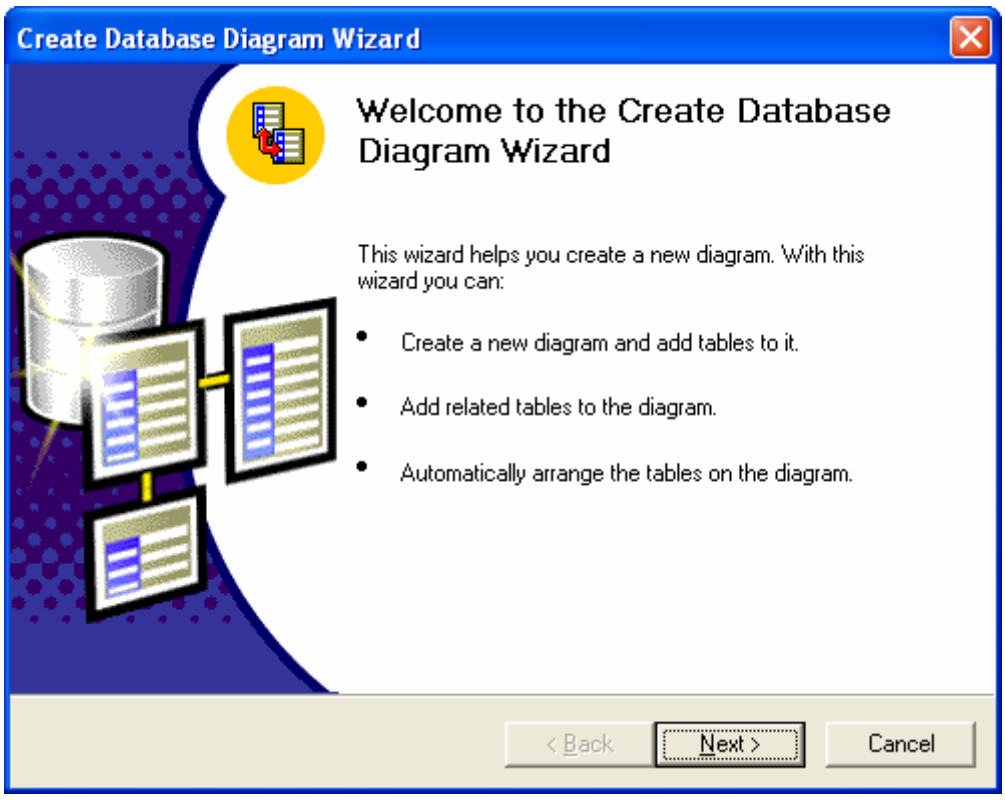
3.2.5. Lược đồ - Diagrams

a) Tạo lược đồ

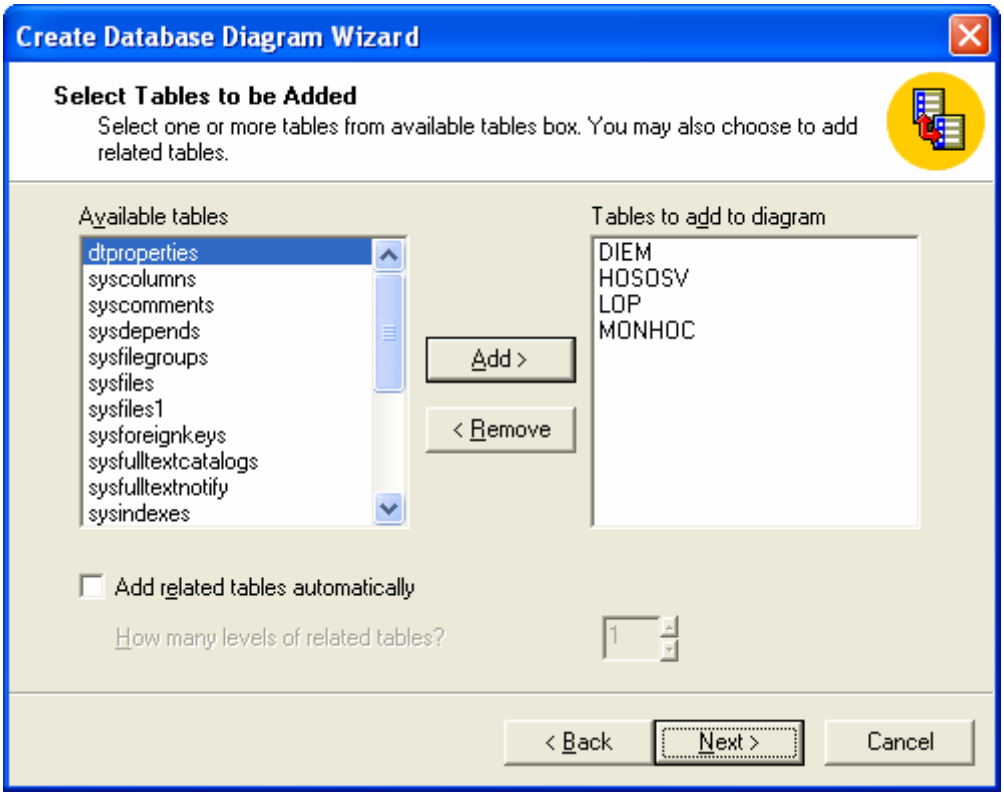
** Dùng Enterprise Manager:*

- Trong Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn tạo lược đồ, chẳng hạn CSDL QLDiemSV và chọn mục Diagrams.

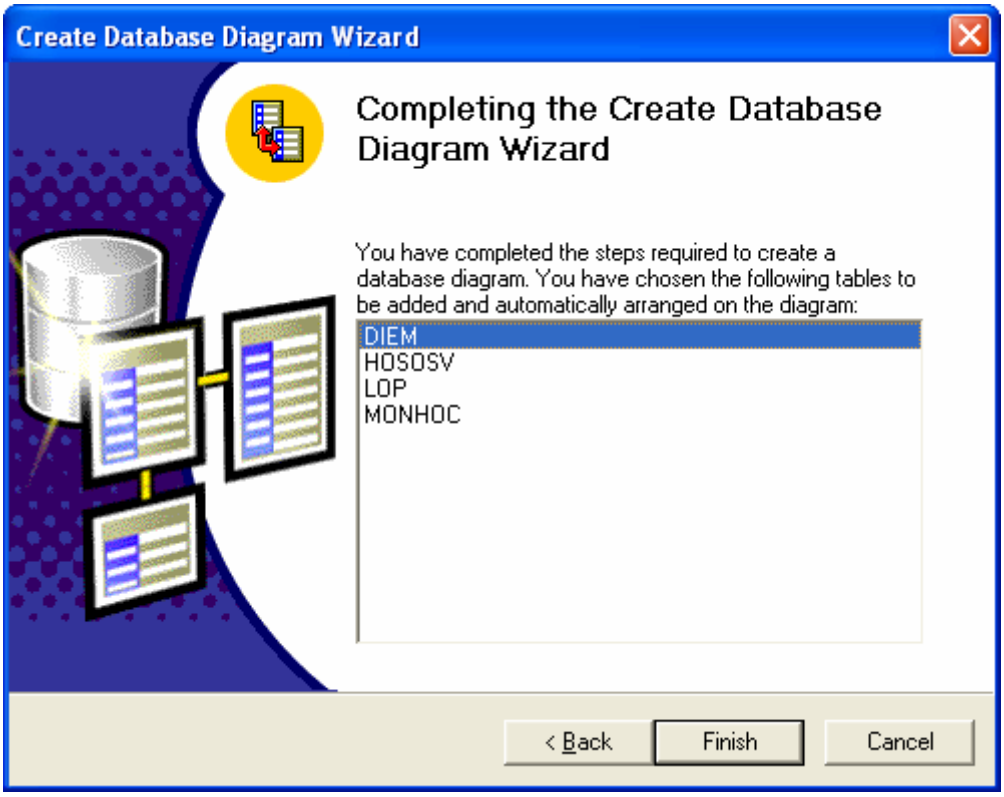
- Right Click lên Diagrams, chọn New Database Design. Xuất hiện cửa sổ Wellcome to Create Database Wizard (Hình 3.50). Chọn Next.
- Xuất hiện cửa sổ *Cửa sổ Select Tables to be Added* (hình 3.51). Chọn các bảng sẽ dùng để xây dựng lược đồ quan hệ. Chọn Next.
- Cửa sổ *Completing the Create Database Wizard* (hình 3.52) chọn Finish.
- Xuất hiện cửa sổ thiết kế Diagrams (hình 3.53). Trong cửa sổ này ta thực hiện thiết kế các mối quan hệ giữa các bảng bằng cách: Kéo và giữ chuột trên trường quan hệ của bảng này sau đó nhả vào trường quan hệ của bảng kia.
- Xuất hiện cửa sổ Relationship (hình 3.54). Trong cửa sổ này ta chọn trường quan hệ giữa hai bảng và các điều kiện cho mối quan hệ đang tạo giữa hai bảng này.



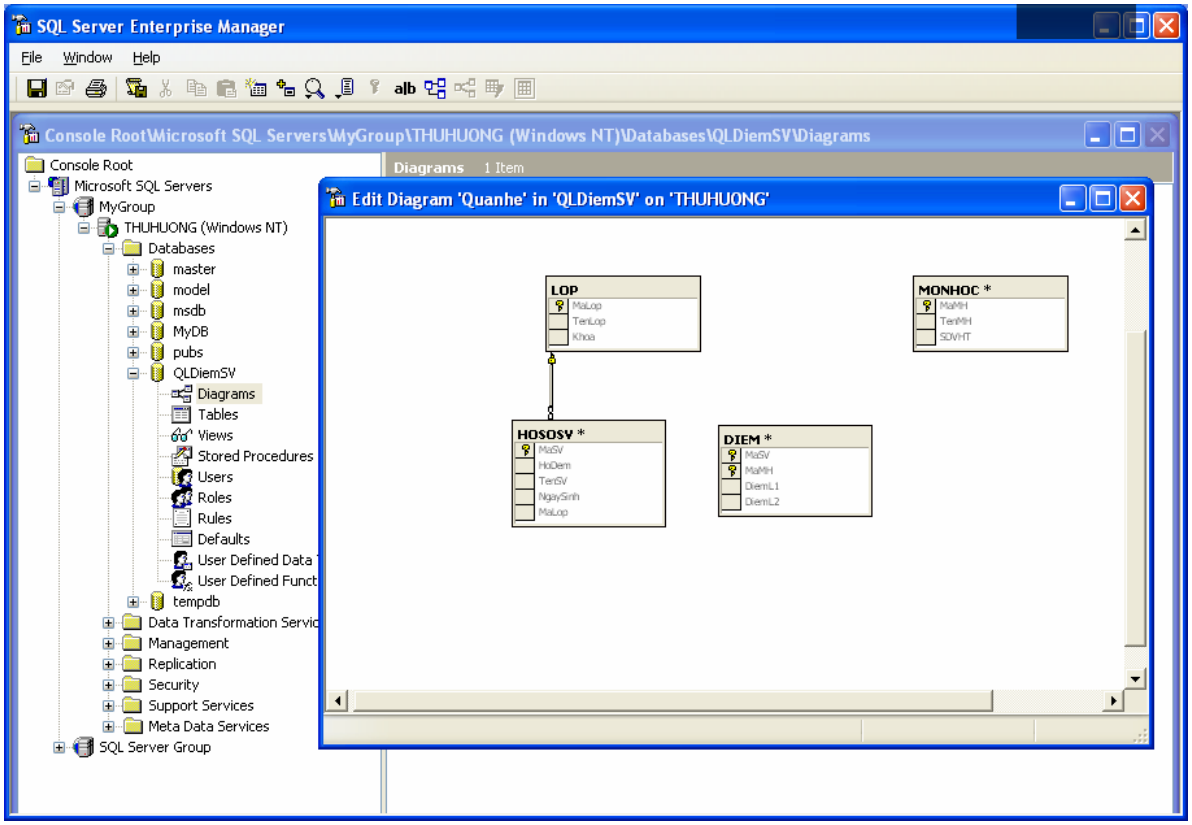
Hình 3.50. *Cửa sổ Wellcome to Create Database Wizard*



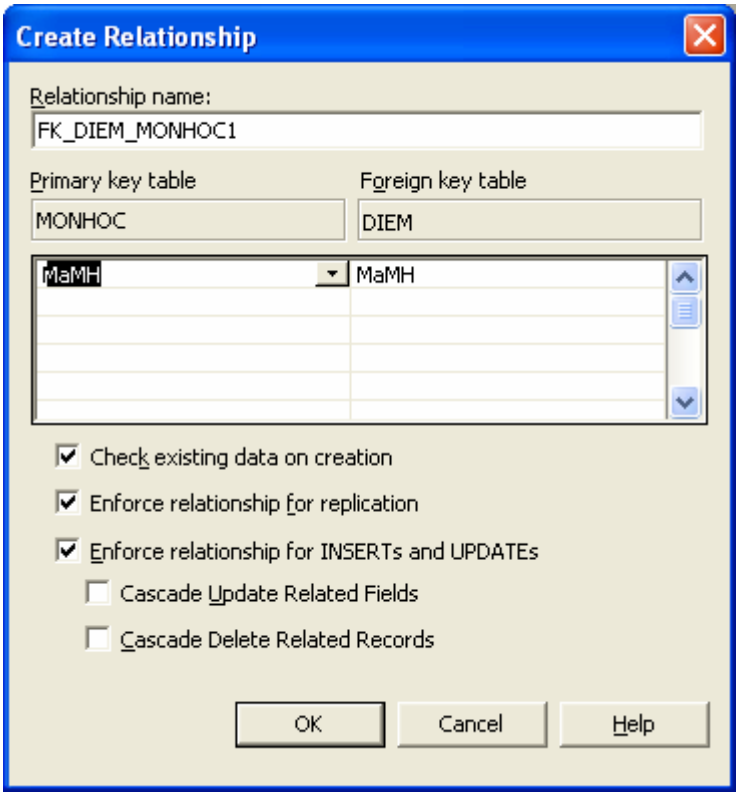
Hình 3.51. *Cửa sổ Select Tables to be Added*



Hình 3.52. *Cửa sổ Completing the Create Database Wizard*



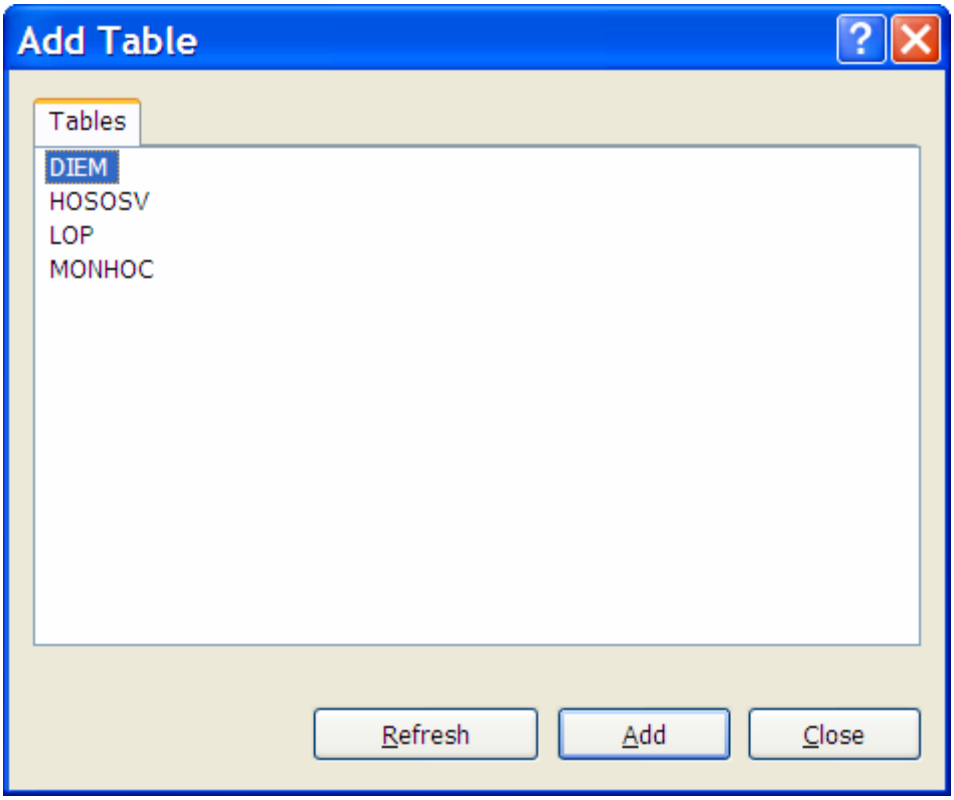
Hình 3.53. *Cửa sổ thiết kế Diagrams*



Hình 3.54. *Cửa sổ Create Relationship*

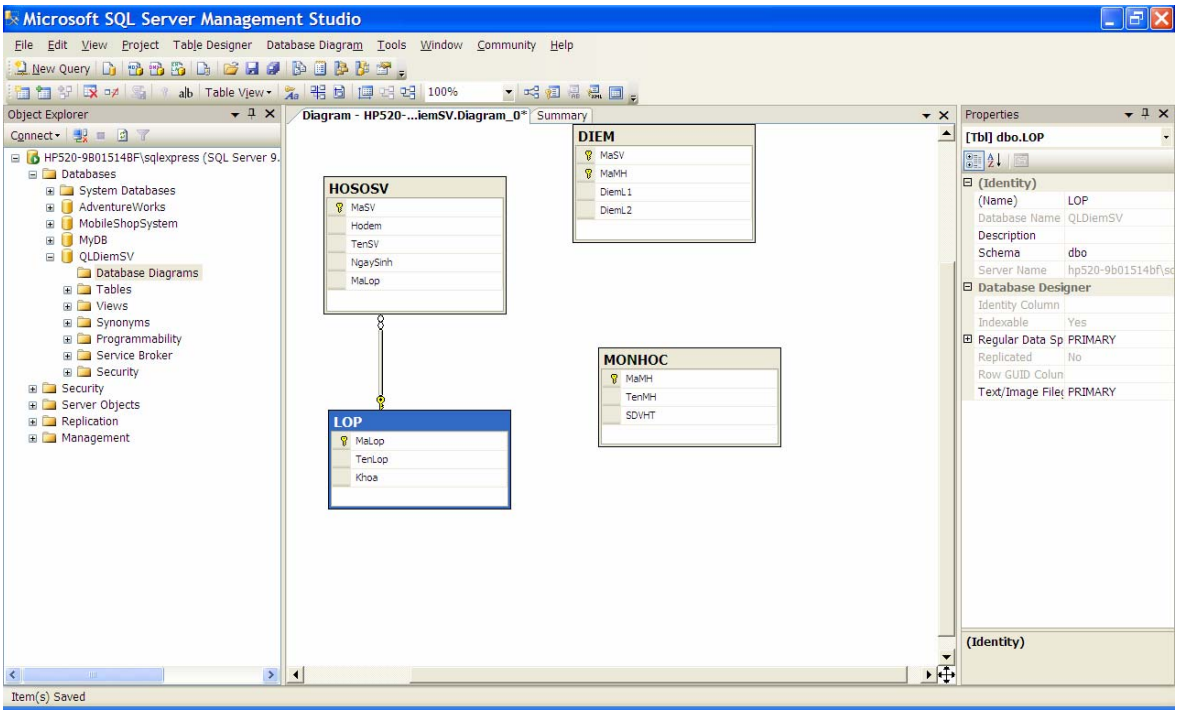
** Dùng SQL Server Management Studio*

- Trong SQL Server Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn tạo Database Diagrams. Right click và chọn New Database Diagram xuất hiện cửa sổ Add Table hình 3.55.

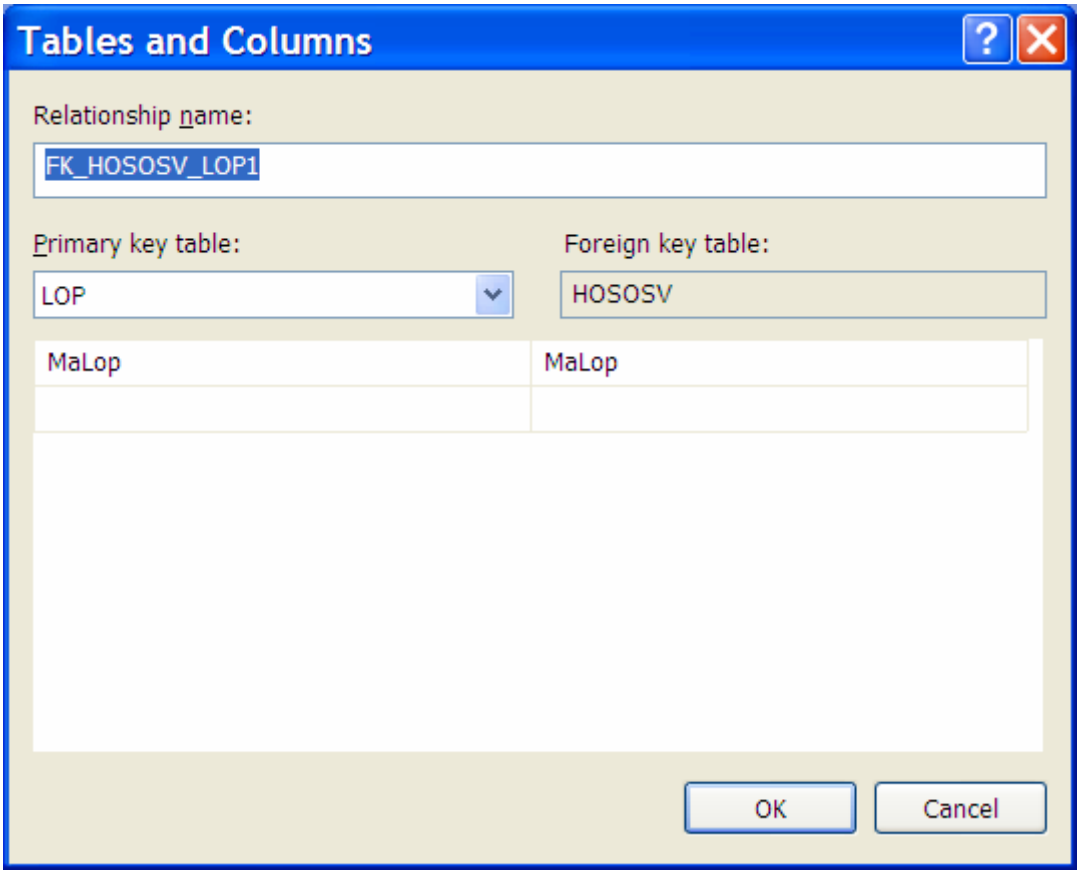


***Hình 3.55.** Cửa sổ Add Table*

- Chọn các bảng xây dựng lược đồ thông qua nút Add. Xuất hiện cửa sổ thiết kế Diagram (Hình 3.56).
- Ta thực hiện thiết kế mối quan hệ giữa các bảng trong cơ sở dữ liệu bằng việc kéo và giữ trường của bảng này thả sang trường tương ứng của bảng khác xuất hiện cửa sổ Table and Columns (Hình 3.57). Ta thực hiện điều chỉnh các tham số cho mối quan hệ đó.



Hình 3.56. *Cửa sổ thiết kế Diagram*



Hình 3.57. *Cửa sổ thiết kế Table and Columns*

b) Chỉnh sửa lược đồ

** Sử dụng Enterprise Manager:*

- Trong Enterprise Manager, mở rộng danh mục Database, mở rộng cơ sở dữ liệu muốn tạo lược đồ, chẳng hạn CSDL QLDiemSV và chọn mục Diagrams.
- Double Click lên lược đồ muốn chỉnh sửa. Xuất hiện cửa sổ Edit Diagram (Hình 3.53). Ta thực hiện các thao tác chỉnh sửa lược đồ trên cửa sổ này:
    + Thêm bảng mới: Right click và chọn Add table
    + Xóa bảng: Right click lên bảng xóa và chọn Remove Table from Diagram.
    + Xóa quan hệ: Right click lên quan hệ muốn xóa và chọn Delete Relationship from Database.
    + Chỉnh sửa lại quan hệ: Right click lên quan hệ muốn chỉnh sửa và chọn Properties. Thực hiện các chỉnh sửa trên cửa sổ này.

** Dùng SQL Server Management Studio*

- Trong SQL Server Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu sửa đổi Database Diagrams. Right click vào Diagram muốn sửa đổi và chọn Modify xuất hiện cửa sổ thiết kế Database Diagram như hình 3.56 ta thực hiện sửa đổi trên cửa sổ này.

c) Xóa lược đồ

** Sử dụng Enterprise Manager:*

Trong Enterprise Manager, mở rộng danh mục Database, mở rộng CSDL và chọn mục Diagrams. Right click lên lược đồ muốn xóa và chọn Delete.

** Dùng SQL Server Management Studio*

Trong SQL Server Management Studio, mở rộng danh mục Database, mở rộng cơ sở dữ liệu sửa đổi Database Diagrams. Right click vào Diagram muốn sửa đổi và chọn Delete xuất hiện cửa sổ xác nhận xóa và chọn OK.

3.3. BẢO ĐẢM DỮ LIỆU TRONG SQL SERVER

3.3.1. Phân quyền và bảo mật trong SQL Server

3.3.1.1. Tạo và quản lý người dùng

a) Các chế độ xác thực

Để truy cập vào SQL Server ta dùng hai chế độ xác thực:

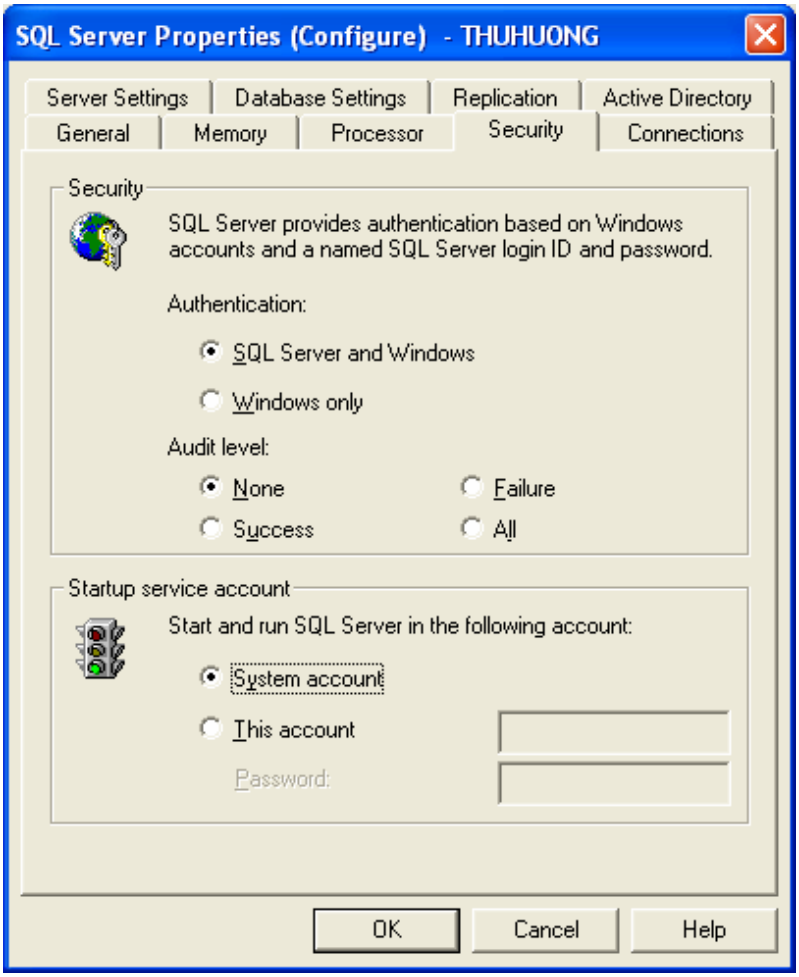
- *Xác thực thông qua hệ điều hành*: Với kiểu xác thực này, SQL Server dựa vào Windows NT/2000 để cấp bảo mật đăng nhập. Khi người dùng đăng nhập vào Windows NT/2000, số định danh tài khoản người dùng được kiểm tra sự hợp lệ. SQL Server tích hợp quá trình bảo mật đăng nhập của nó với quá trình bảo mật đăng nhập của Windows để cung cấp những dịch vụ này. Khi người dùng xác thực bằng hệ điều hành thì không cần thêm chế độ xác thực nào nữa để truy cập SQL Server.
- *Chế độ xác thực hỗn hợp*: Với chế độ xác thực hỗn hợp người dùng có thể truy cập vào SQL Server bằng xác thực thực Windows hoặc bằng xác thực SQL Server. Khi chế độ xác thực kết hợp được dùng, kết nối được tạo từ hệ thống không bảo mật, SQL Server xác thực đăng nhập bằng cách kiểm tra tài khoản đăng nhập có được thiết lập để truy cập hay không. SQL Server thực hiện xác thực tài khoản này bằng cách so sánh tên tài khoản và mật khẩu do người dùng cung cấp để thực hiện kết nối với thông tin tài khoản được lưu trong CSDL. Nếu tài khoản đăng nhập chưa được thiết lập hoặc người dùng nhập sai thì SQL Server từ chối kết nối.

***Thiết lập chế độ xác thực*:**

** Đối với SQL Server 2000:*

+ Trong cửa sổ Enterprise Manager, Right click lên tên server quản lý CSDL mà muốn thiết lập chế độ xác thực và chọn Properties. Xuất hiện cửa sổ Properties như hình 3.58. Chọn tab Security.

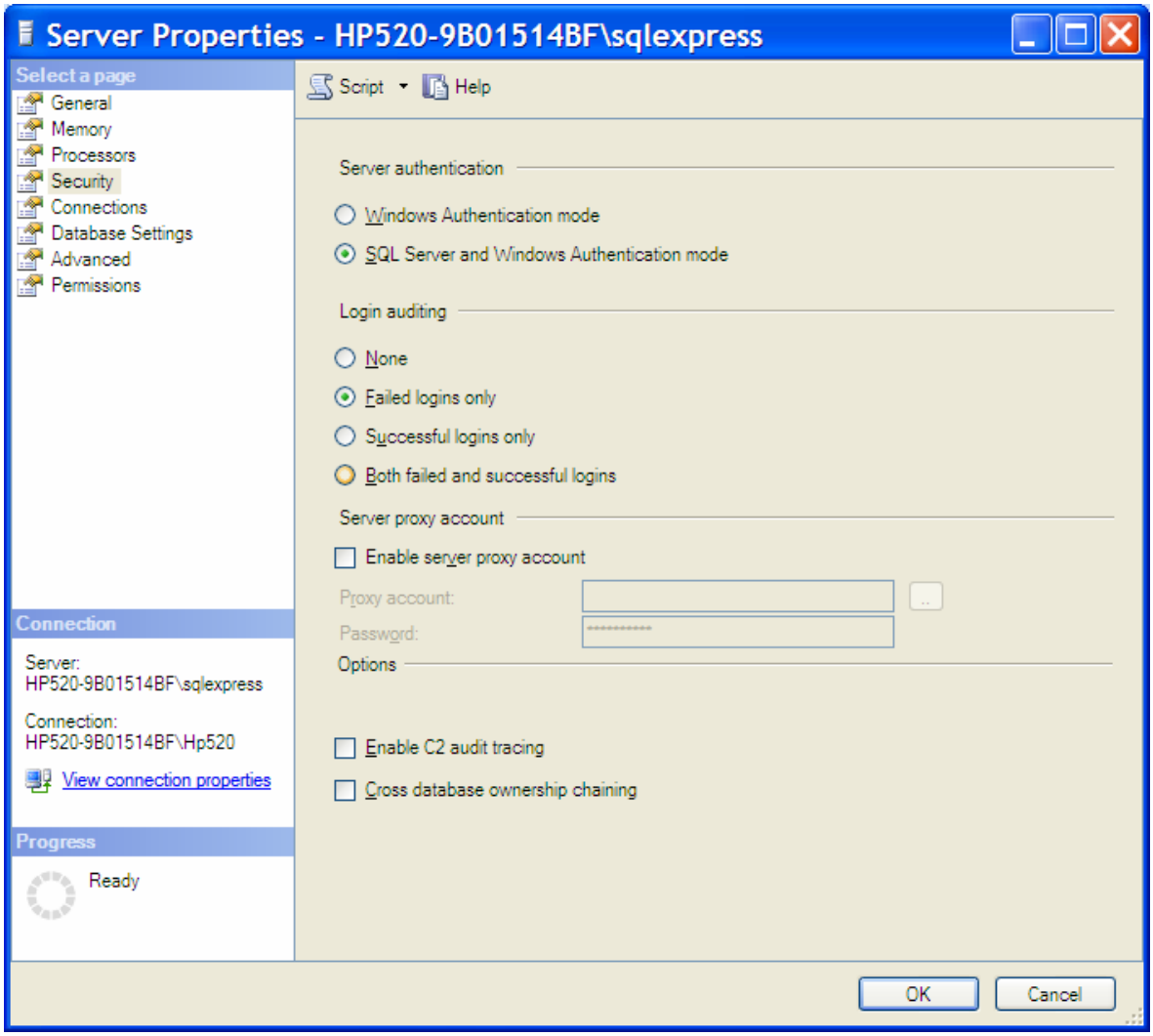
+ Vùng Security ta chọn chế độ xác thực: SQL Server and Windows hoặc Windows Only. Và các mức dò xét hành vị đăng nhập, tùy thuộc vào yêu cầu ảo mật. Có 4 mức cho sẵn đó là:
    - *None*: Không thực hiện dò xét hành vi,
    - *Success*: Ghi nhận tất cả các đăng nhập thành công,
    - *Failure*: Ghi nhận những đăng nhập không thành công,
    - *All*: ghi nhận tất cả các lần đăng nhập.
+ Trong vùng Startup service account: Chỉ ra tài khoản Windows được dùng khi SQL Server khởi động.
    + System account: các tài khoản hệ thống cục bộ được xây dựng sẵn.
    + This account: Chỉ ra một tài khoản cụ thể.



Hình 3.58. *Cửa sổ Properties*

** Đối với SQL Server 2005:*

+ Trong cửa sổ SQL Server Management Studio, right click vào thể hiện của SQL Server muốn thiết lập chế độ xác thực và chọn Properties xuất hiện cửa sổ Server Properties (Hình 3.59), chọn trang Security.



Hình 3.59. *Cửa sổ Server Properties*

+ Mục Server Authentication: Chọn chế độ xác thực Window (Windows Authentication mode) hoặc chế độ xác thực hỗn hợp (SQL Server and Windows Authentication mode)

+ Login Auditing: Kiểu ghi nhận thông tin login (None, Failed logins only, Successful login only, …)

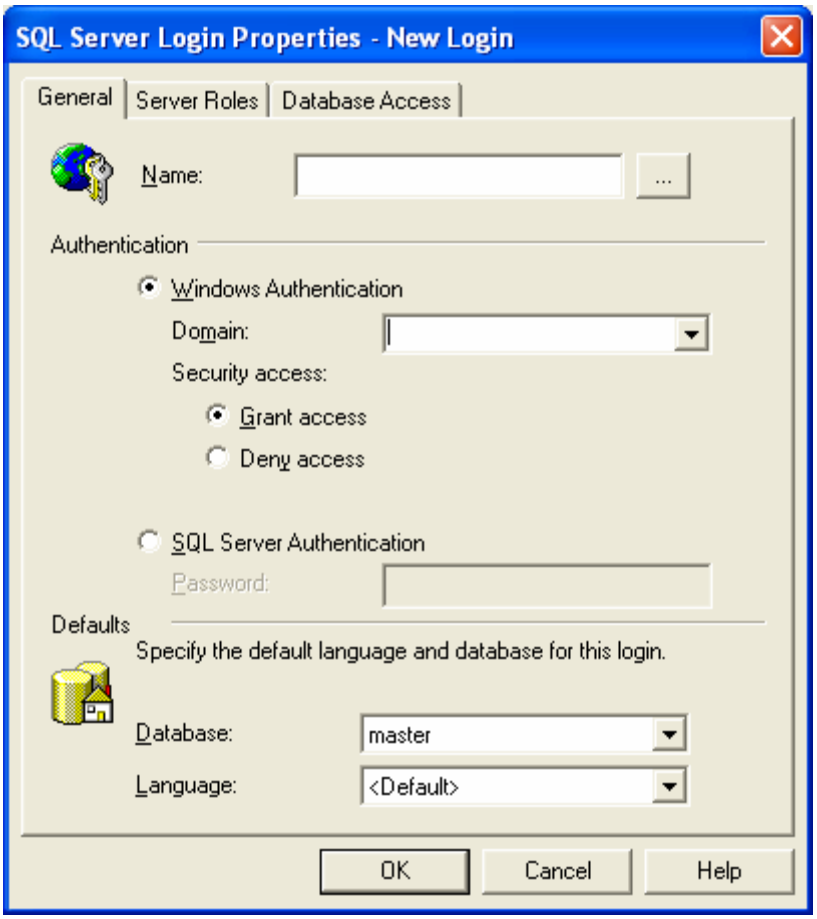
b) Người dùng và đăng nhập

Tài khoản dùng để kết nối tới SQL Server được gọi là tài khoản đăng nhập SQL Server. Cùng với tài khoản đăng nhập SQL Server, mỗi CSDL có một tài khoản người dùng ảo được gán với nó. Những tài khoản ảo này cung cấp một bí danh tới tài khoản đăng nhập SQL Server được gọi là tài khoản người dùng CSDL.

*** Tạo tài khoản đăng nhập SQL Server:**

➢ ***Dùng Enterprise manager:***

+ Trong cửa sổ Enterprise Manager, mở rộng server muốn tạo tài khoản đăng nhập và mở rộng mục Security. Right click lên Logins và chọn New Login để xuất hiện cửa sổ SQL Server Login Properties (hình 3.60).



Hình 3.60. *Cửa sổ New Login*

+ Trên tab General:

+ Name: Nhập tên tài khoản đăng nhập. nếu chọn chế độ xác thực bằng Window thì tên tài khoản đăng nhập phải là tài khoản đã tồn tại trong Windows.

+ Authentication: Chọn chế độ xác thực của Windows là Windows Authentication hay chế độ xác hỗn hợp SQL Server Authentication.

+ Default: Chọn CSDL và ngôn ngữ mặc định sẽ được dùng.

+ Tab Server Roles: Ở đây ta chọn nhóm quyền server cho đăng nhập mới bằng cách chọn các nhóm quyền trong danh sách. Click vào nút Properties để xem và sửa đổi nhóm quyền đã chọn. Nếu là tài khoản đăng nhập thường thì không cần cấp quyền server.

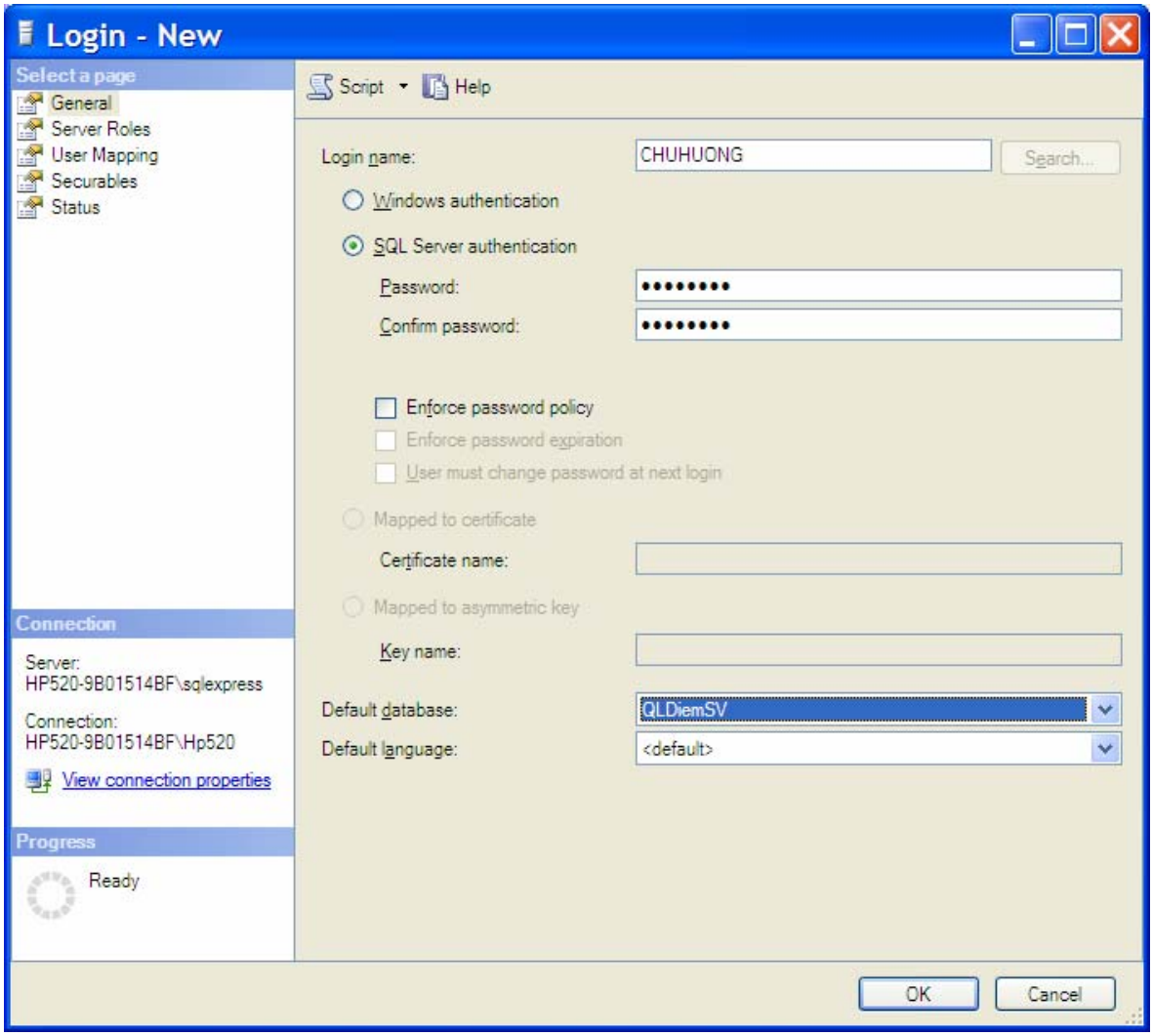
+ Tab Database Access: Cho phép chọn CSDL mà người dùng được phép truy cập.

➢ ***Dùng SQL Server Management Studio:***

+ Trong cửa sổ SQL Server Management Studio, mở rộng thể hiện server muốn tạo tài khoản đăng nhập và mở rộng mục Security. Right click lên Logins và chọn New Login để xuất hiện cửa sổ Login – New (hình 3.61).

+ Trang General có các lựa chọn:

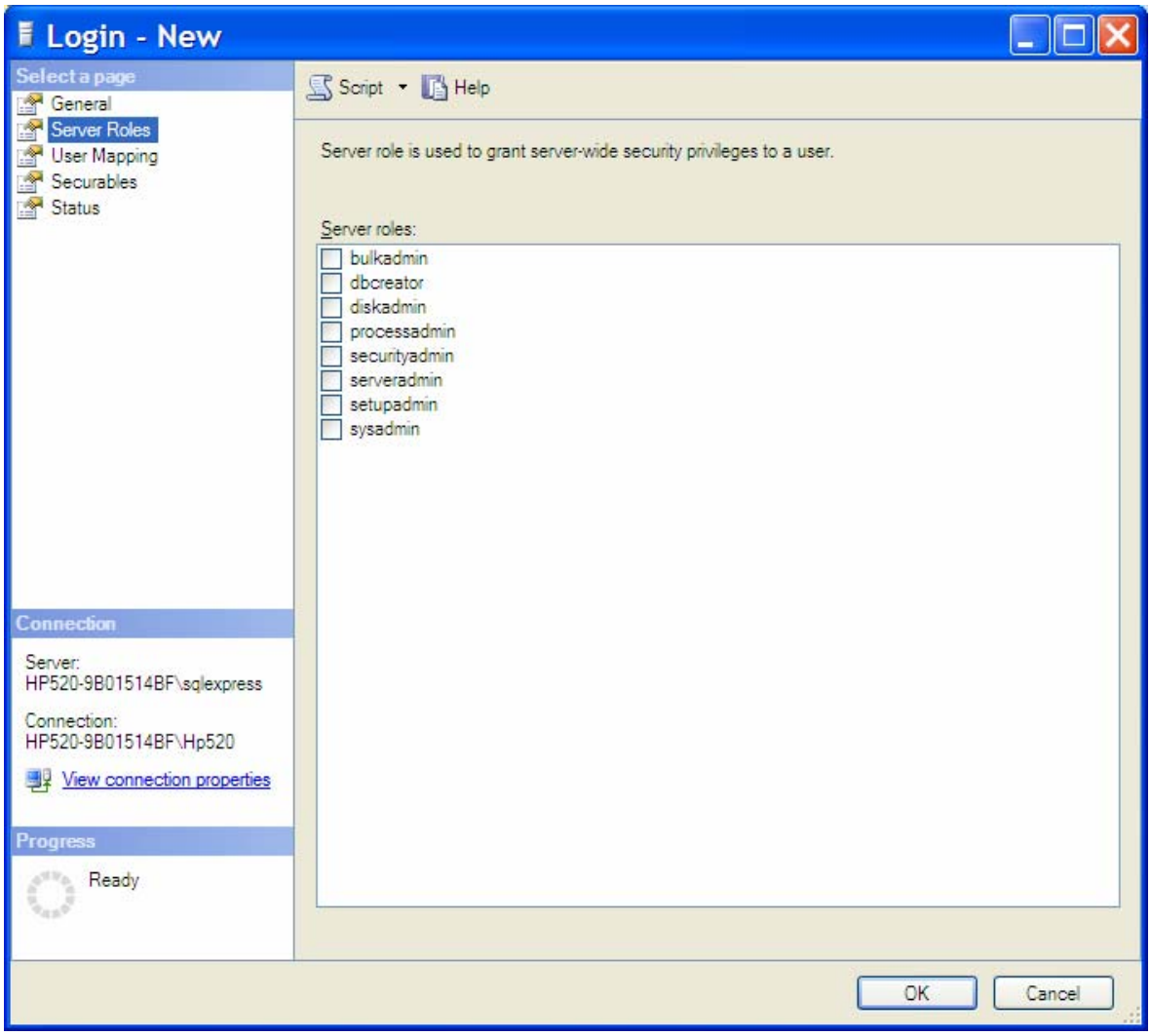
- Login Name: Nhập tên tài khoản đăng nhập. Nếu chọn chế độ xác thực bằng Window thì tên tài khoản đăng nhập phải là tài khoản đã tồn tại trong Windows.
- Default database: Chọn CSDL mặc định được sử dụng.
- Default language: Chọn ngôn ngữ mặc định.



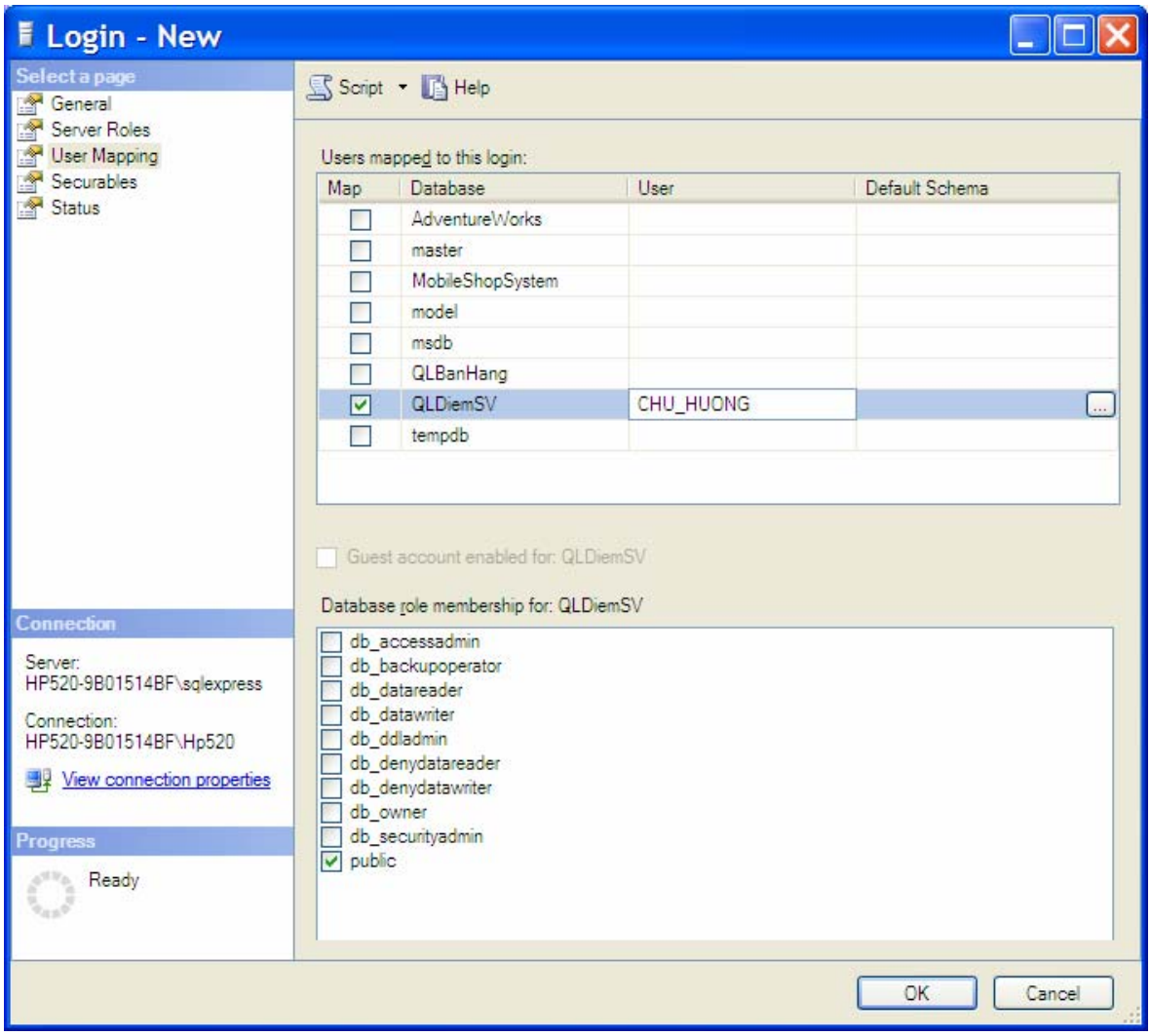
Hình 3.61. *Cửa sổ Login - New*

+ Trang Server Roles có các lựa chọn: Chọn nhóm quyền server cho đăng nhập mới bằng cách chọn các nhóm quyền trong danh sách (Hình 3.62).

+ Trang User Mapping (Hình 3.63): Cho phép chọn CSDL mà người dùng được phép truy cập.



Hình 3.62. *Cửa sổ Login - New*



Hình 3.63. *Cửa sổ Login - New*

➢ ***Dùng T-SQL:***

- Ta có thể tạo tài khoản đăng nhập bằng thủ tục sp_addlogin hoặc sp_grandlogin.

  + Thủ tục sp_addlogin chỉ có thể thêm người dùng được xác thực bằng SQL Server.

  + Thủ tục sp_grandlogin có thể thêm người dùng được xác thực bằng Windows.

Ví dụ 3.11. Tạo tài khoản đăng nhập *Huongct* với Password là *'abcd'* và CSDL mặc định là *'QLDiemSV'*

```
EXEC sp_addlogin  'Huongct1','abcd','QLDiemSV'
```

Ví dụ 3.12. Tạo tài khoản đăng nhập *Huongct* với chế độ xác thực Windows.

```
EXEC sp_grantlogin 'THUHUONG\Huongct'
```

- Ngoài ra, đối với SQL Server 2005 ta có thể sử dụng cú pháp T-SQL sau:

+ Tạo Login

```
CREATE LOGIN login_name { WITH PASSWORD =
'password' [,<option_list>[ ,... ]] | FROM
WINDOWS [ WITH <windows_options> [ ,... ]]}

<option_list> ::=
    DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
    | CHECK_EXPIRATION = { ON | OFF}
    | CHECK_POLICY = { ON | OFF}
    [ CREDENTIAL = credential_name ]

<windows_options> ::=
    DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
```

+ Sửa Login

```
ALTER LOGIN login_name
    {
    <status_option>
    | WITH <set_option> [ ,... ]
    }

<status_option> ::=
        ENABLE | DISABLE

<set_option> ::=
    PASSWORD = 'password'
    [
            OLD_PASSWORD = 'oldpassword'
    ]
    | DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
    | NAME = login_name
    | CHECK_POLICY = { ON | OFF }
    | CHECK_EXPIRATION = { ON | OFF }
```

+ Xóa Login

```
DROP LOGIN login_name
```

Ví dụ 3.13. Tạo các login

- Tạo Login HUONGCT dùng chế độ xác thực SQL

```
USE  master
CREATE LOGIN HUONGCT WITH PASSWORD = '123456';
GO
```

- Tạo Login [HP520-9B01514BF\Hp520] từ domain account Windows.

```
CREATE LOGIN [HP520-9B01514BF\Hp520] FROM WINDOWS;
GO
```

- Sửa Login HUONGCT

```
USE  master
ALTER LOGIN HUONGCT WITH PASSWORD =
'12102006',DEFAULT_DATABASE = QLDiemSV;
```

- Xóa Login HUONGCT

```
USE  master
DROP LOGIN HUONGCT
```

➢ ***Dùng Wizard trong SQL Server 2000:***

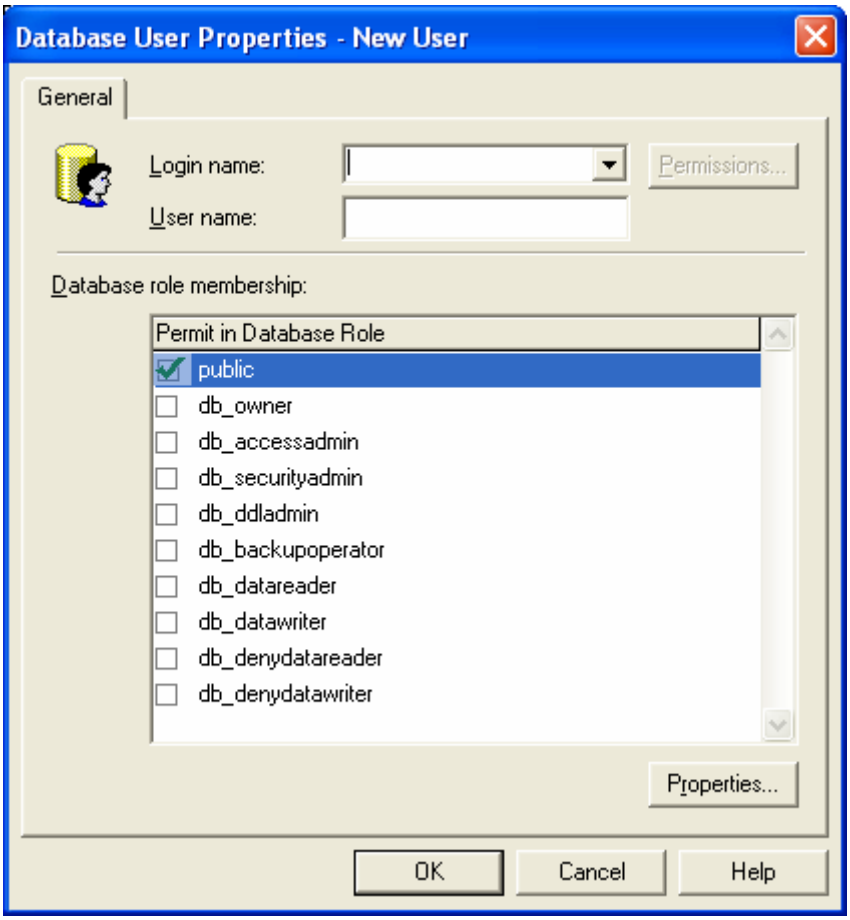
Vào Tools\Wizard xuất hiện cửa sổ hình 3.5. Chọn mục Database và chọn Create Login Wizard. Sau đó thực hiện theo sự chỉ dẫn của trình Wizard.

*** Tạo người dùng SQL Server:**

Để tạo người dùng SQL Server, trước hết ta phải tạo đăng nhập SQL Server cho người dùng đó vì tên người dùng tham chiếu đến tên đăng nhập.

➢ ***Dùng Enterprise manager:***

+ Trong cửa sổ Enterprise Manager, mở rộng mục Database. Right click CSDL muốn tạo người dùng và chọn New\Database User xuất hiện cửa sổ New User (hình 3.64).



Hình 3.64. *Cửa sổ New User*

+ Nhập tên đăng nhập hợp lệ trong danh sách các tên đăng nhập của hộp combo Login Name và nhập tên người dùng mới vào hộp User Name (Mặc định SQL Server tự điền tên User Name trùng tên Login Name, ta có thể thay đổi tên này)

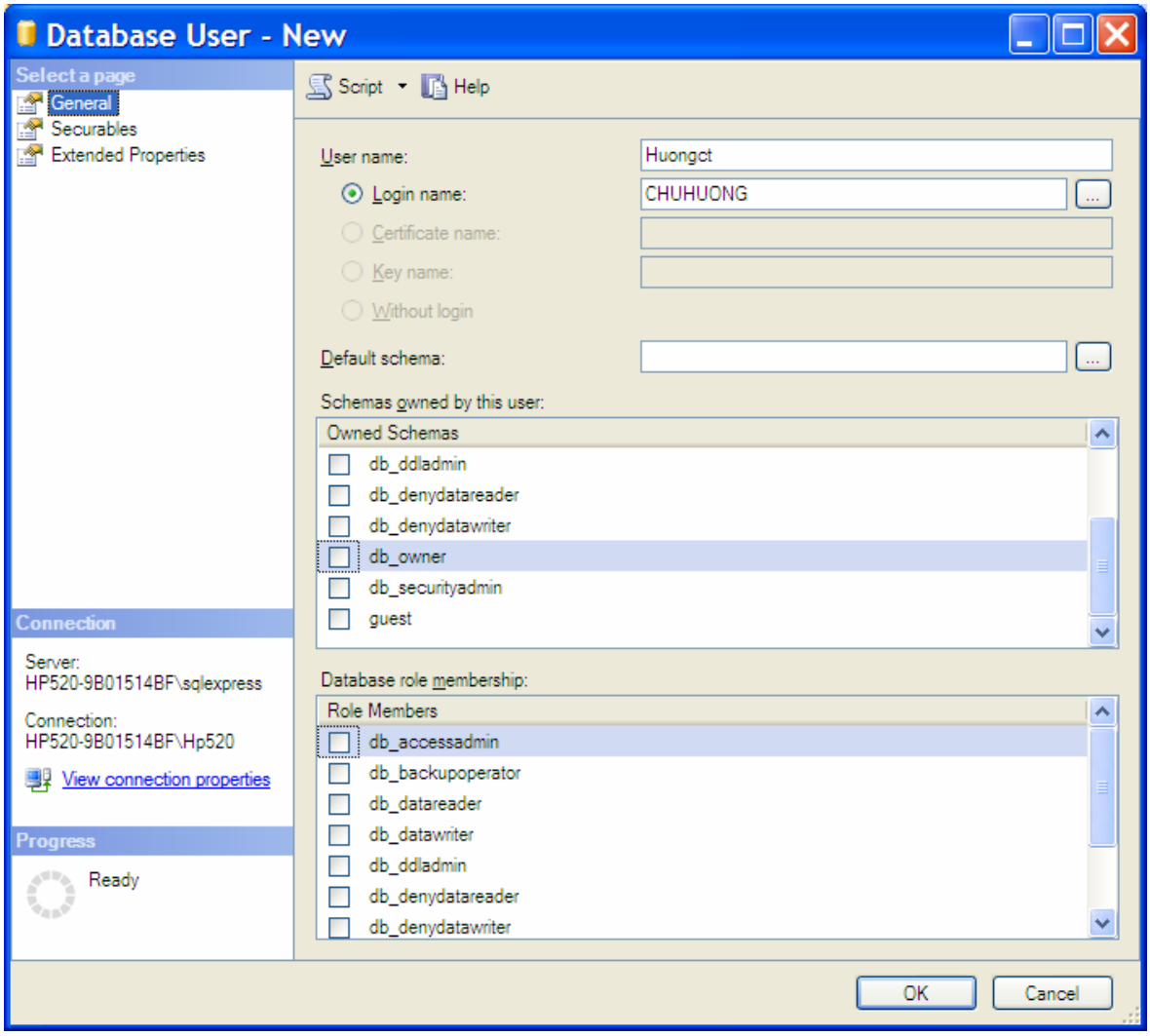
+ Chọn nhóm quyền CSDL mà người dùng mới này là thành viên, sau đó chọn OK.

➢ ***Dùng SQL Server Management Studio:***

+ Trong cửa sổ SQL Server Management Studio, mở rộng thể hiện server và mục Database. Mở rộng mục Security của cơ sở dữ liệu muốn tạo người dùng, Right click lên Users và chọn New User để xuất hiện cửa sổ Database User – New (hình 3.65).

+ Trang General có các lựa chọn:

- User name: Nhập tên người dùng
- Login name: Nhập tên hoặc chọn Login mà người dùng này ánh xạ đến.



Hình 3.65. *Cửa sổ Database User - New*

➢ ***Dùng T-SQL:***

- Ta có thể tạo người dùng mới bằng thủ tục sp_adduser.
- Đối với SQL Server 2005 ta có thể dùng cú pháp sau:

+ Tạo User

```
CREATE USER user_name
    { { FOR | FROM } LOGIN login_name |
WITHOUT LOGIN }
    [ WITH DEFAULT_SCHEMA = schema_name ]
```

+ Sửa User

```
ALTER USER user_name
     WITH <set_item> [ ,...n ]
<set_item> ::=
     NAME = new_user_name
     | DEFAULT_SCHEMA = schema_name
```

+ Xóa User

```
DROP USER user_name
```

Ví dụ 3.14. Tạo các người dùng mới là *Huong* với tên đăng nhập *Huongct* trên CSDL QLDiemSV

```
USE QLDiemSV
Go
sp_adduser 'Huongct','Huong'
```

Ví dụ 3.15. Tạo các người dùng mới trùng với tên đăng nhập *Huongct* trên CSDL QLDiemSV sử dụng xác thực của Windows

```
USE QLDiemSV
Go
sp_adduser 'THUHUONG\Huongct'
```

Ví dụ 3.16. Tạo các User

```
CREATE USER Huongct WITHOUT LOGIN
CREATE USER ChuHuong FOR LOGIN CHUHUONG
```

3.3.1.2. Quản lý nhóm quyền CSDL

Các nhóm quyền CSDL được thiết kế cho phép các nhóm những người dùng nhận các quyền CSDL giống nhau mà không cần phải cấp quyền một cách riêng biệt cho từng người dùng.

a) Các nhóm quyền Server cố định

Một số nhóm quyền ở cấp server đã được định nghĩa trước tại thời điểm cài đặt SQL Server. Những nhóm quyền cố định này được dùng để cấp quyền cho người quản trị CSDL. Các nhóm quyền server cố định được liệt kê trong danh sách sau:

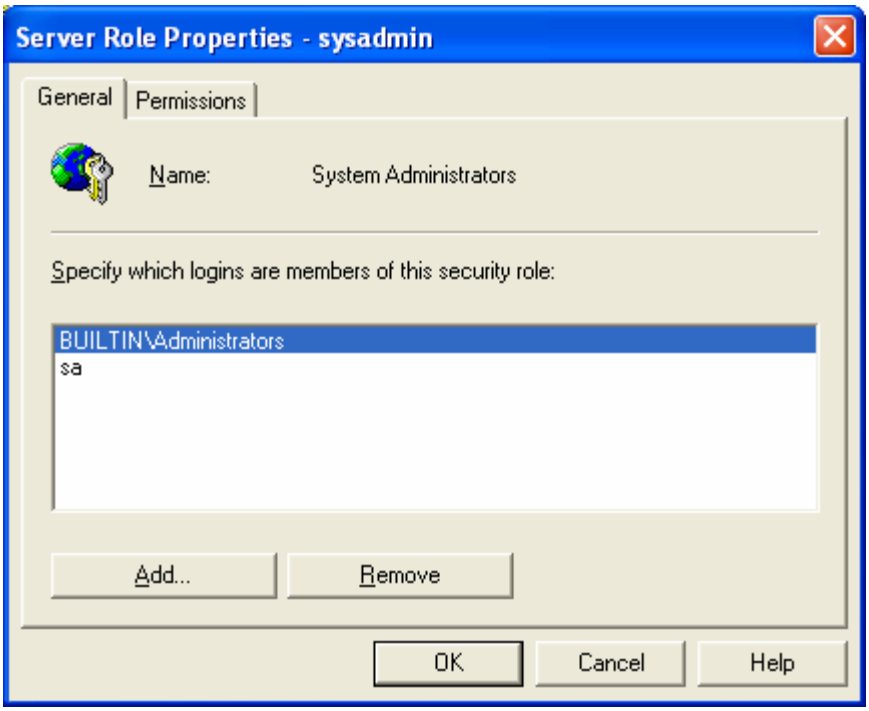
+ bulkadmin: Có thể thực thi lệnh BULK INSERT để thêm lượng lớn dữ liệu vào bảng.

+ dbcreator: Có thể tạo và sửa đổi CSDL.
+ diskadmin: Có thẻ quản lý các tập tin trên đĩa.
+ processadmin: Có thể quản lý các quá trình của SQL Server.
+ securityadmin: Có thể quản lý đăng nhập và tạo các quyền CSDL.
+ serveradmin: Có thể thiết lập bất kỳ tùy chọn server nào và có thể đóng CSDL.
+ setupadmin: Có thể quản lý các server liên kết và có thể đóng CSDL.
+ sysadmin: Có thể thực hiện bất kỳ hoạt động server nào.

Thêm người dùng vào các nhóm quyền server cố định.

➢ **Dùng Enterprise Manager**

+ Trong cửa sổ Enterprise Manager, mở rộng server và mở rộng mục Security. Sau đó chọn server roles và right click lên nhóm quyền server muốn thêm người dùng, chẳng hạn chọn System Administrators xuất hiện cửa sổ hình 3.66.

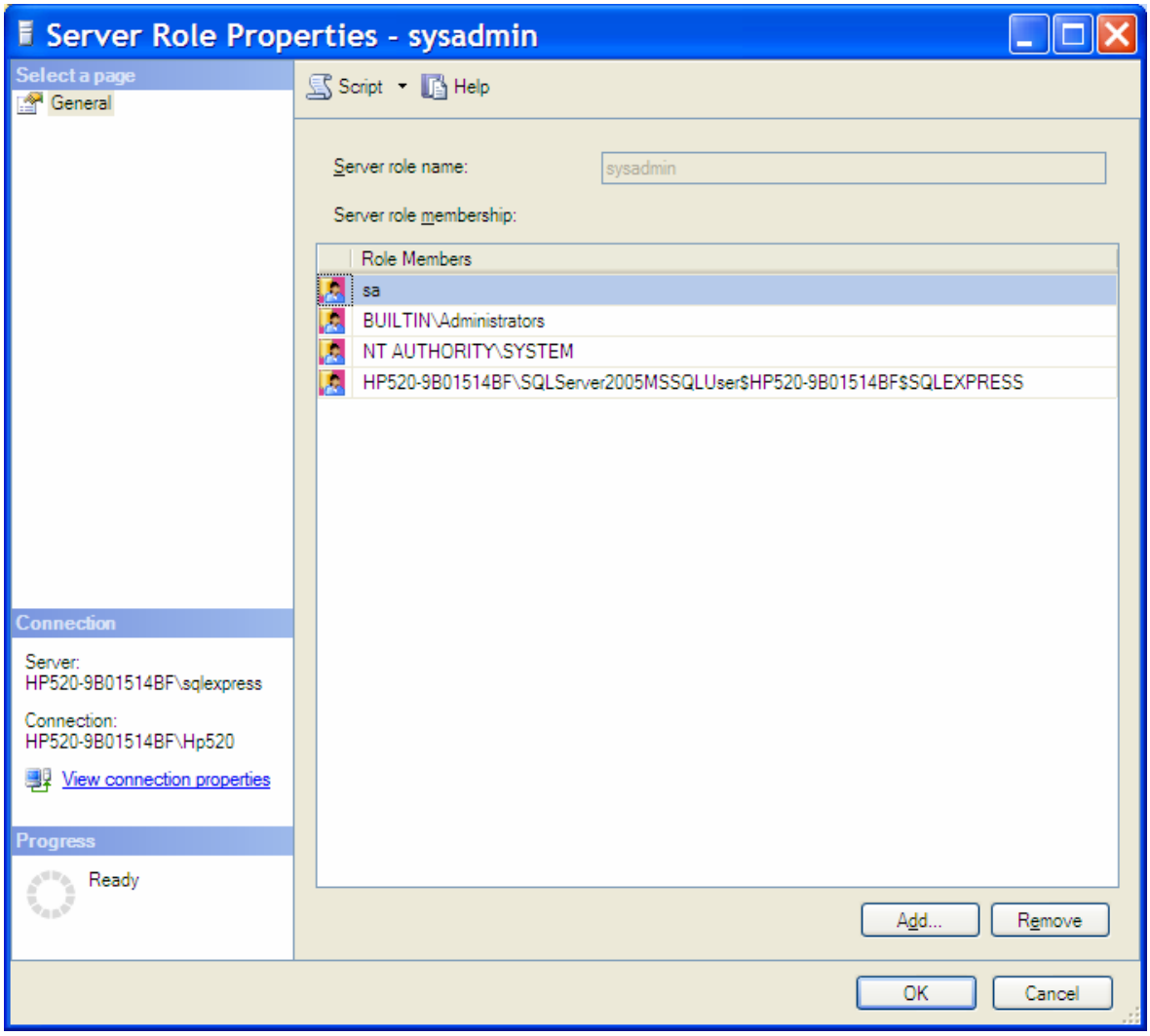


Hình 3.66. *Cửa sổ Server Role Propeties*

+ Click nút Add để thêm người dùng vào trong nhóm.

➢ **Dùng SQL Server Management Studio**

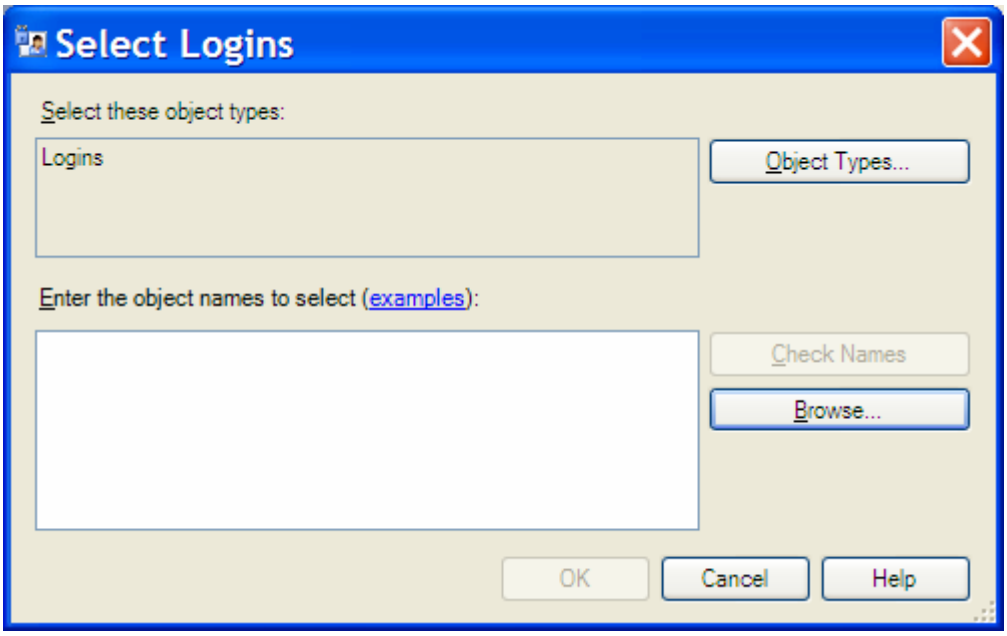
+ Trong cửa sổ SQL Server Management Studio, mở rộng mục Security ở cấp Server. Sau đó chọn Server roles và right click lên lên nhóm quyền server muốn thêm người dùng, chẳng hạn chọn System Administrators xuất hiện cửa sổ hình 3.67.



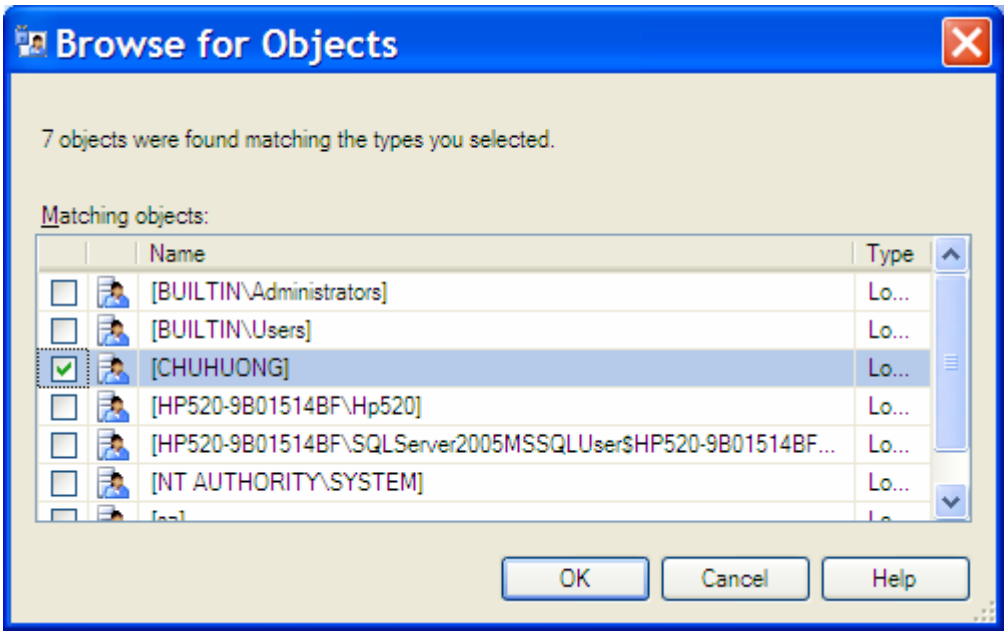
Hình 3.67. *Cửa sổ Server Role Propeties*

+ Click vào nút Add để thêm người dùng vào nhóm. Hộp thoại Select Login xuất hiện (Hình 3.68).

+ Click vào nút Browse để chọn đăng nhập ta muốn thêm vào nhóm. Xuất hiện hộp thoại Browse for Objects (Hình 3.69)



Hình 3.68. *Cửa sổ* Select Login



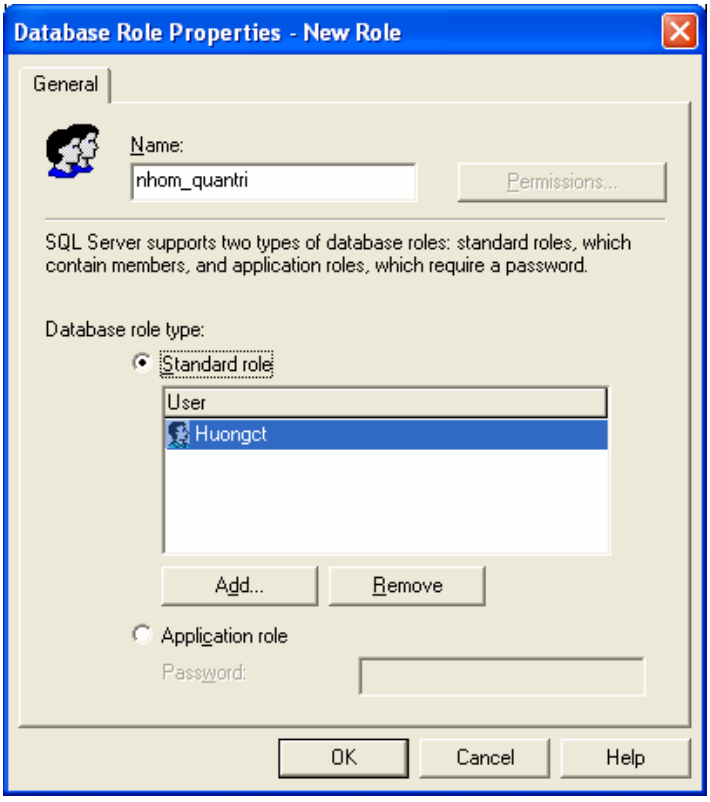
Hình 3.69. *Cửa sổ* Browse for Objects

+ Chọn các Login và click OK để đóng cửa sổ Browse for Objects.
+ Click OK để đóng cửa sổ Select Logins.
+ Click OK để đóng cửa sổ Server Role Propeties để thêm người dùng đã chọn vào nhóm.

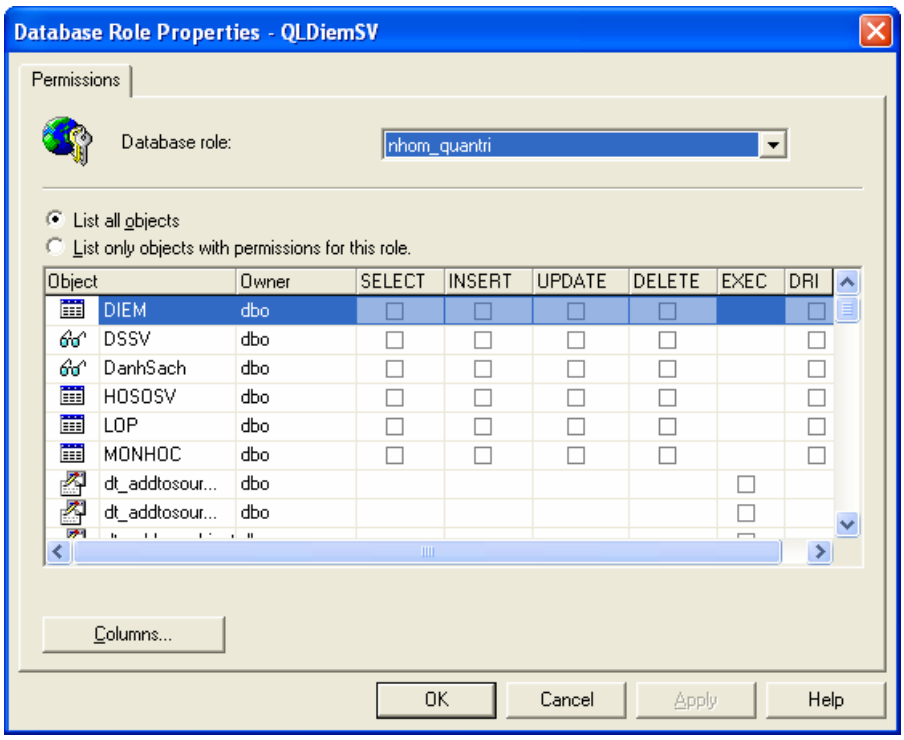
b) Tạo và sửa đổi nhóm quyền

➢ **Dùng Enterprise manager:**

+ Trong cửa sổ Enterprise Manager, mở rộng mục Database. Right click CSDL muốn tạo nhóm người dùng và chọn New\Database Role xuất hiện cửa sổ (hình 3.70).
+ Trong cửa sổ này ta nhập tên mô tả nhóm và Add các tài khoản người dùng thuộc nhóm bằng cách click vào nút Add và chọn các người dùng trong danh sách. Sau đó click nút OK để tạo nhóm.
+ Trở về Enterprise Manage, chọn mục Roles ta sẽ thấy nhóm quyền mới vừa tạo ở danh mục bên phải.
+ Ta gán quyền cho nhóm bằng cách right click lên nhóm quyền và chọn Propeties. Xuất hiện cửa sổ Database Role Propeties và chọn Permissions xuất hiện cửa sổ hình 3.71.



Hình 3.70. *Cửa sổ New Role*



Hình 3.71. *Cửa sổ Database Role Propeties*

+ Ta thực hiện gán các quyền cho các nhóm quyền này trên các đối tượng của CSDL và click OK.

➢ **Dùng SQL Server Management Studio:** SQL Server 2005 cung cấp hai loại nhóm quyền do người dùng định nghĩa

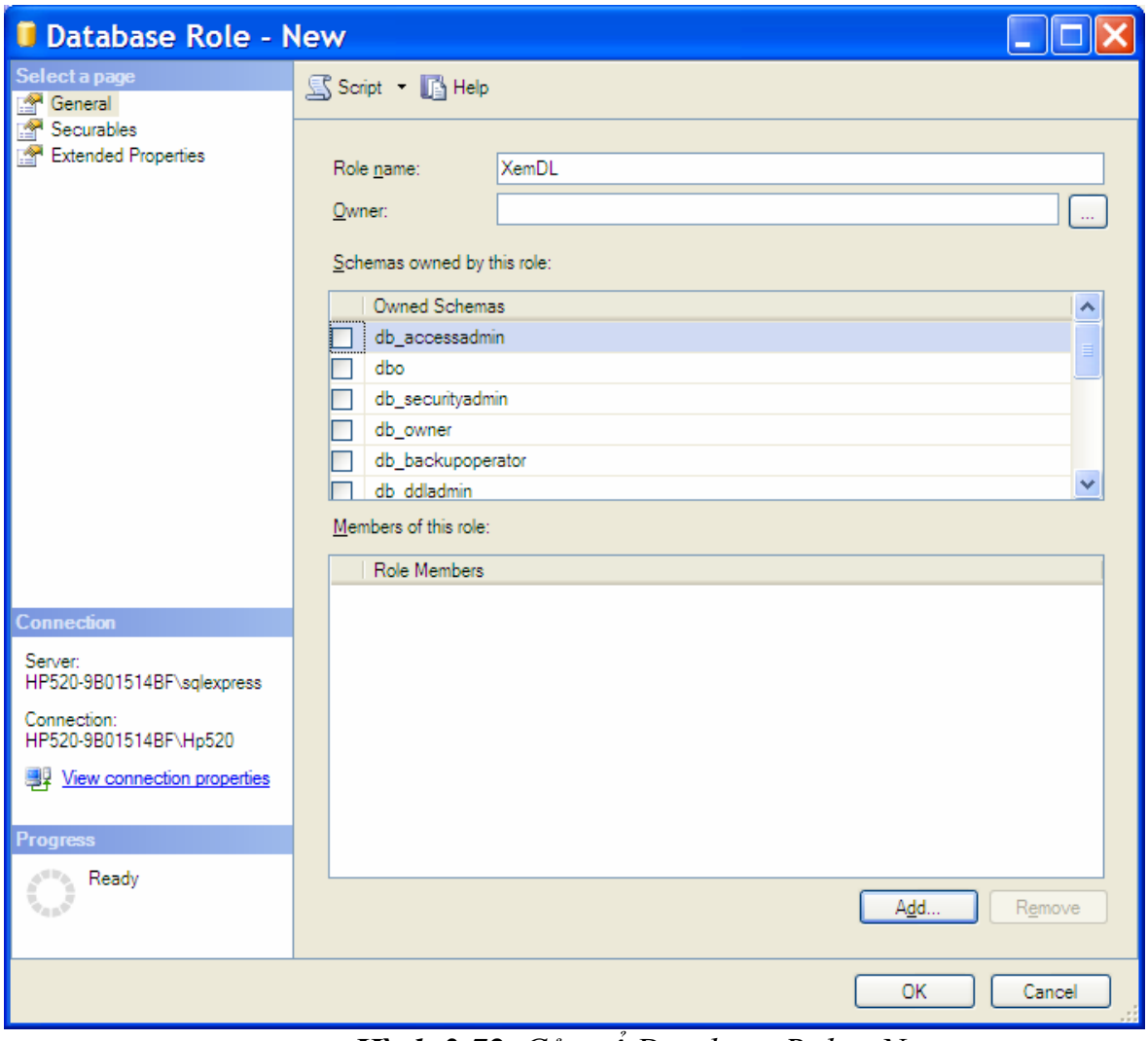
+ Database Roles: Đây là nhóm quyền chuẩn, dùng cho các tác vụ gán quyền tới cơ sở dữ liệu.

+ Application Roles: Dùng cho các quyền liên quan đến ứng dụng.

Xây dựng nhóm quyền Database Role ta tiến hành thực hiện theo các bước sau:

+ Trong SQL Server Management Studio, mở rộng mục Security cấp CSDL của CSDL ta muốn tạo nhóm quyền.

+ Mở rộng mục Roles, right click lên mục Database Roles và chọn New Database Role xuất hiện cửa sổ Database Role – New (Hình 3.72)



Hình 3.72. *Cửa sổ Database Role - New*

- Role name: Nhập tên nhóm
- Owner: Chọn danh sách các người dùng.
- Members of this role: Click vào nút Add để thêm các người dùng vào nhóm.

➢ **Dùng T-SQL:**

+ Ta thực hiện thông qua 2 bước.

1. *Tạo nhóm quyền:* Ta có thể tạo nhóm quyền bằng thủ tục sp_addrole. Đối SQL Server 2005 ta có thể dùng cú pháp sau:

```
CREATE ROLE role_name [AUTHORIZATION owner_name ]
```

Trong đó:

- *role_name:* Là tên của nhóm sẽ được tạo.
- AUTHORIZATION *owner_name* : Là người dùng CSDL hoặc các nhóm mà sở hữu nhóm mới này. Nếu không có người dùng nào chỉ định thì nhóm sẽ được sở hữu bởi người dùng thực hiện câu lệnh CREATE ROLE này.

Ví dụ 3.21. Tạo nhóm quyền 'xem_dl' trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
Sp_addrole 'xem_dl'
```

2. *Thêm quyền vào nhóm:* Ta có thể thêm quyền vào nhóm bằng việc sử dụng lệnh GRANT và thu hồi quyền của nhóm sử dụng lệnh REVOKE.

   Ví dụ 3.22. Ta thêm quyền SELECT bảng HOSOSV vào nhóm 'xem_dl' trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
GRANT SELECT
ON HOSOSV
TO xem_dl
```

3. *Thêm người dùng vào nhóm quyền:* Để thêm người dùng vào nhóm quyền ta sử dụng thủ tục sp_addrolemember

   Ví dụ 3.23. Thêm người dùng Guest vào nhóm quyền 'xem_dl' trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
Sp_addrolemember 'xem_dl','Guest'
Go
```

3.3.1.3. Quản lý quyền CSDL

Xác thực người dùng là quá trình đảm bảo chỉ có những người dùng hợp lệ mới được phép làm việc với cơ sở dữ liệu. Sau khi người dùng truy cập được vào CSDL thì họ có các quyền cụ thể với các đối tượng trong CSDL.

Các quyền trên các đối tượng là:

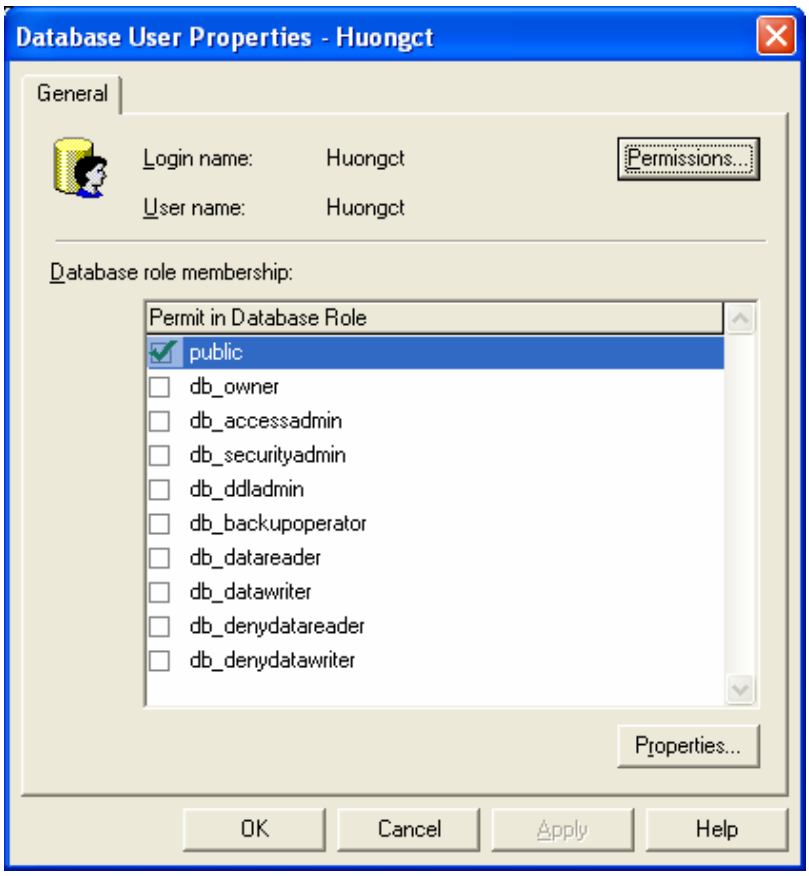
+ Đối tượng database: BACKUP DATABASE, BACKUP LOG, CREATE DATABASE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE TABLE, và CREATE VIEW.
+ Đối tượng scalar function: EXECUTE và REFERENCES.
+ Đối tượng table-valued function: DELETE, INSERT, REFERENCES, SELECT, và UPDATE.
+ Đối tượng stored procedure: DELETE, EXECUTE, INSERT, SELECT, và UPDATE.
+ Đối tượng table: DELETE, INSERT, REFERENCES, SELECT, và UPDATE.
+ Đối tượng view: DELETE, INSERT, REFERENCES, SELECT, and UPDATE.

Cấp quyền trên các đối tượng CSDL ta tiến hành thực hiện như sau:

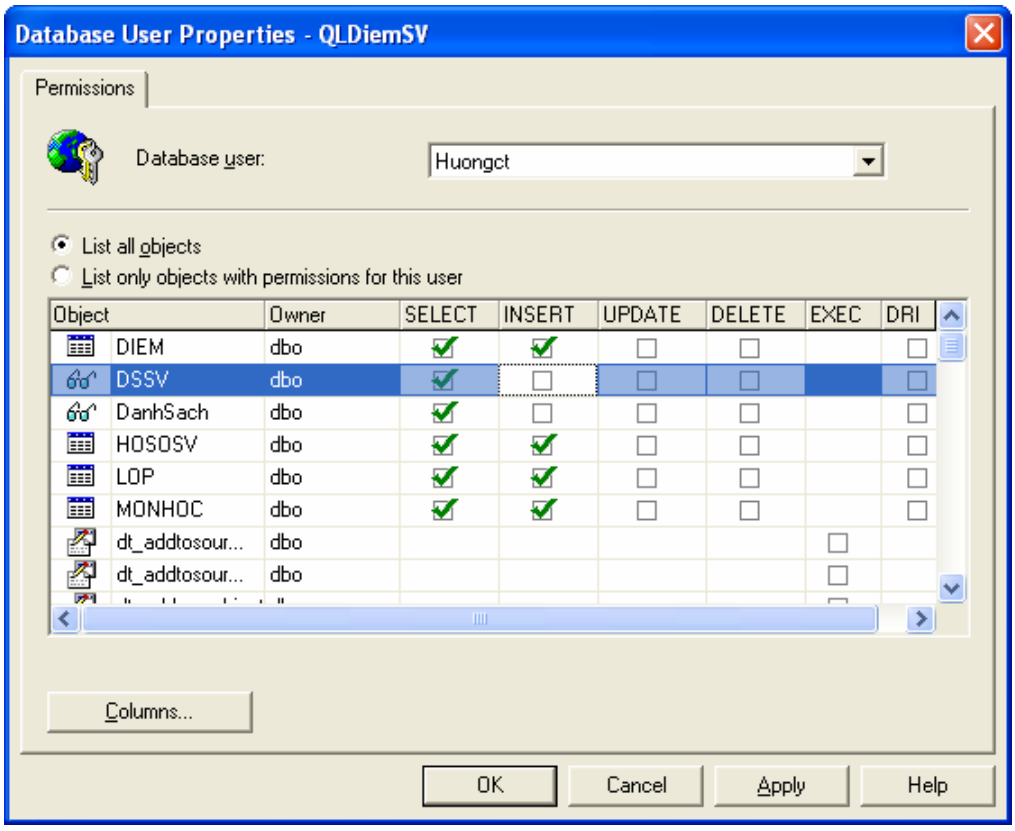
➢ **Dùng Enterprise manager để cấp quyền trên các đối tượng:** Có hai cách quản lý quyền trên các đối tượng.

Cách 1: Gán quyền cho tất cả các đối tượng cho một người dùng hoặc một nhóm người dùng.

+ Trong cửa sổ Enterprise Manager, mở rộng mục Database. Chọn CSDL muốn cấp quyền và mở rộng mục Users. Sau đó right click lên tên người dùng muốn cấp quyền và chọn Properties, xuất hiện cửa sổ như hình 3.73.
+ Click vào nút Permissionss để hiển thị cửa sổ Database Users Properties như hình 3.74. Ta gán các quyền cho người dùng này trên các đối tượng bằng cách chọn hộp kiểm tương ứng. Các tùy chọn *List all Objects* dùng để liệt kê tất cả các đối tượng còn chọn *List only objects with permissions for this user* thì chỉ liệt kê các đối tượng mà người dùng này có quyền truy cập.



Hình 3.73. *Cửa sổ* Database Users Properties

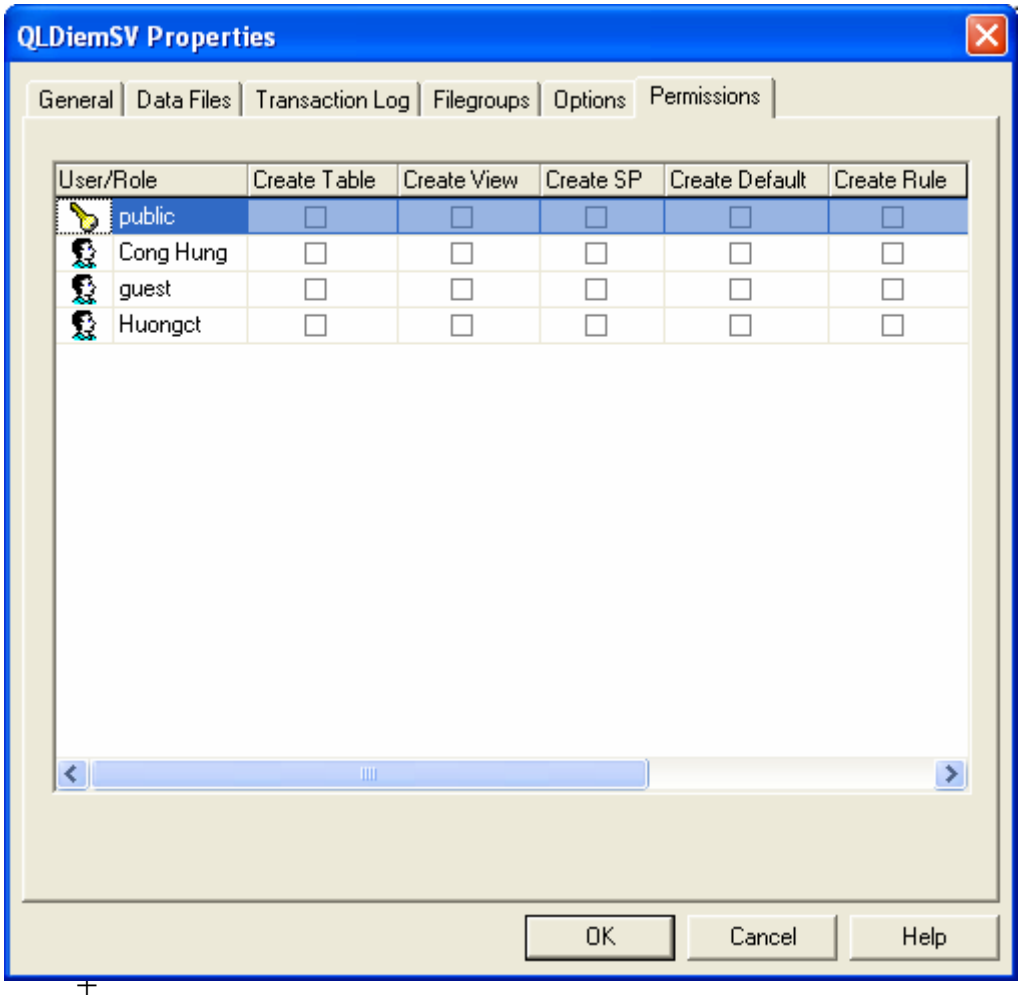


Hình 3.74. *Cửa sổ* Database Users Properties

Cách 2: Gán các quyền trên một đối tượng cho tất cả các người dùng hoặc các nhóm người dùng.

+ Trong cửa sổ Enterprise Manager, mở rộng mục Database. Sau đó right click lên CSDL muốn cấp quyền, ví dụ QLDiemSV và chọn Properties, xuất hiện cửa sổ như hình 3.75. Chọn tab Permissions.

+ Thực hiện cấp quyền thực thi cho các người dùng bằng cách tích vào ô tương ứng với các quyền đó.



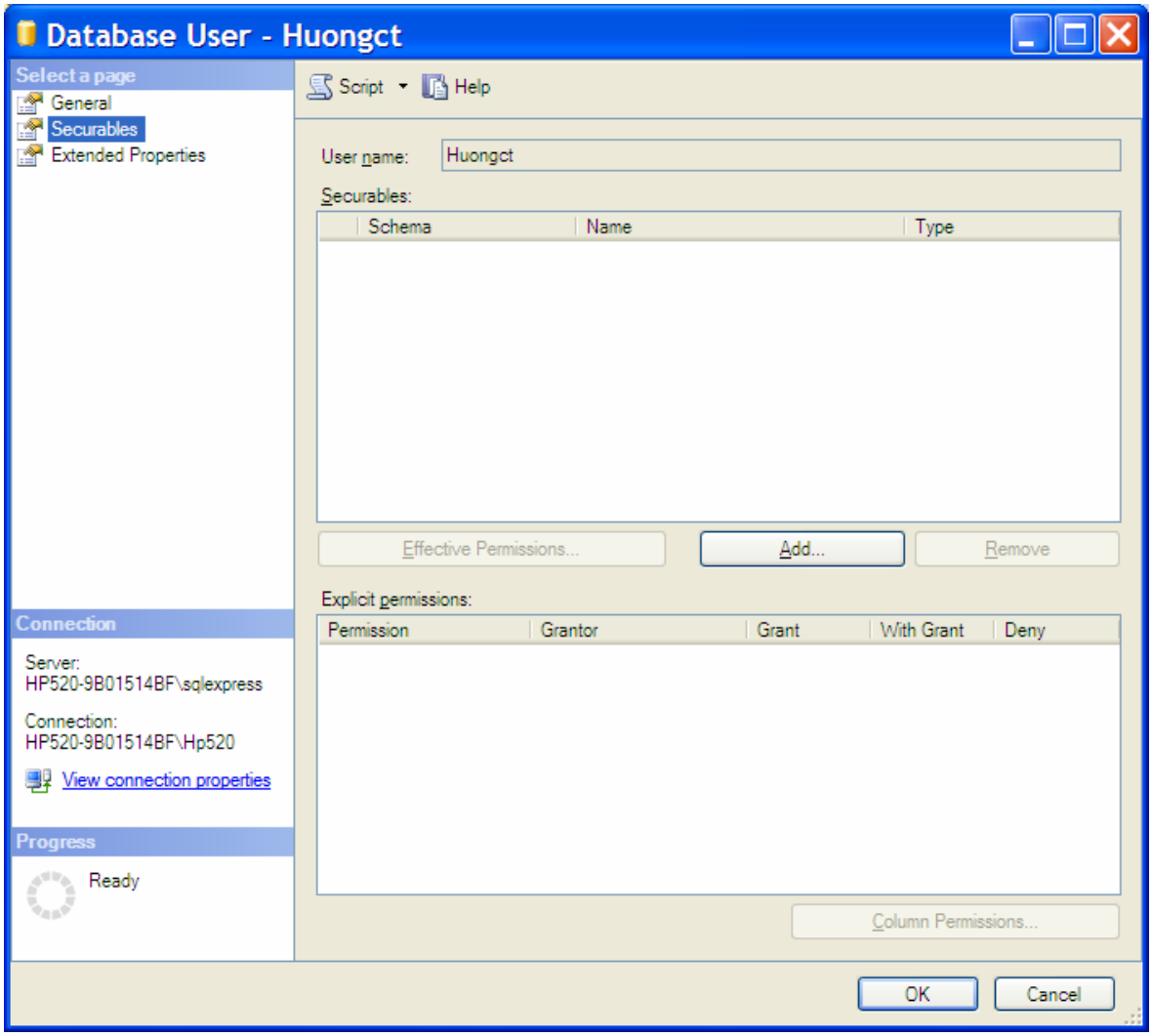
Hình 3.75. Cửa sổ QLDiemSV Properties

➢ **Dùng SQL Server Management Studio để cấp quyền trên các đối tượng:** Có hai cách quản lý quyền trên các đối tượng.

Cách 1: Gán quyền cho tất cả các đối tượng cho một người dùng hoặc một nhóm người dùng.

+ Trong cửa sổ Object Explorer của SQL Server Management Studio, mở rộng mục Database. Chọn CSDL muốn cấp quyền và mở rộng mục Users hoặc Roles\Database Roles.

+ Right click lên người dùng hoặc nhóm người dùng muốn và chọn Properties. Hộp thoại Properties xuất hiện chọn trang Securables (Hình 3.76)

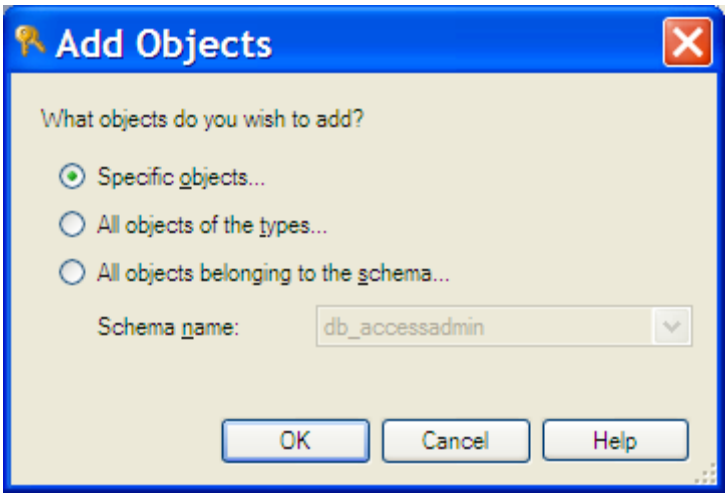


Hình 3.76. Cửa sổ Database User

+ Click nút Add để thêm các đối tượng muốn bảo mật. Hộp thoại Add Objects xuất hiện (Hình 3.77), ta chỉ các đối tượng muốn bảo mật.

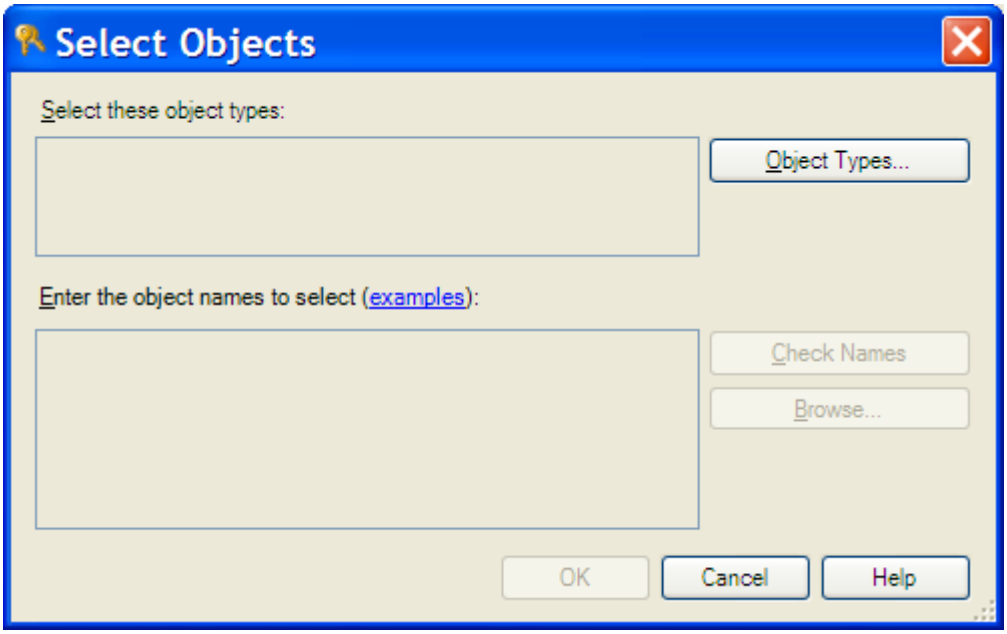
  - Specific objects: Chỉ định các đối tượng cụ thể.

- All objects of the types: Tất cả các đối tượng của các kiểu cụ thể.
- All objects belong to schema: Các đối tượng thuộc giản đồ.



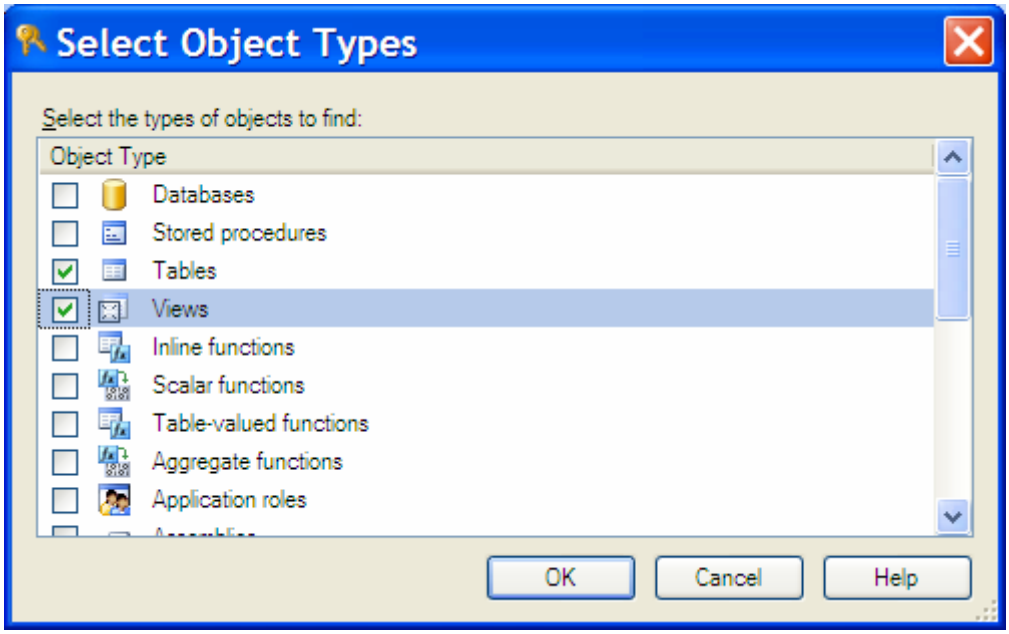
Hình 3.77. Cửa sổ Add Objects

+ Nếu chọn tùy chọn Specific objects, click OK hộp thoại Select Object xuất hiện (Hình 3.78).



Hình 3.78. Cửa sổ Select Objects

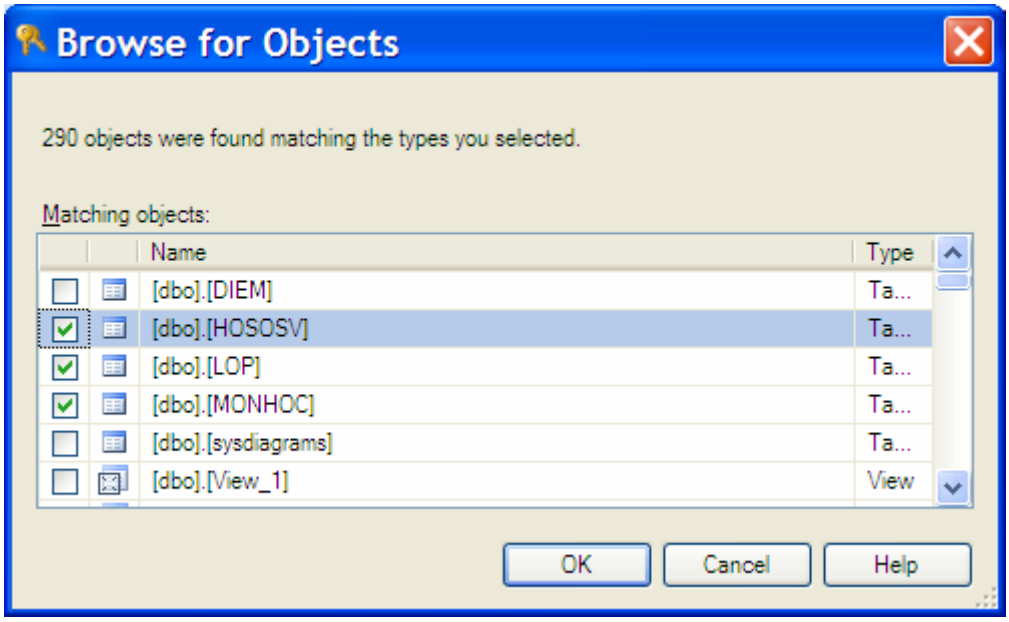
+ Click vào nút Object Types, hộp thoại Select Object Types xuất hiện như hình 3.79. Ta chọn các kiểu đối tượng mà muốn bảo mật cho người dùng đó và click OK.



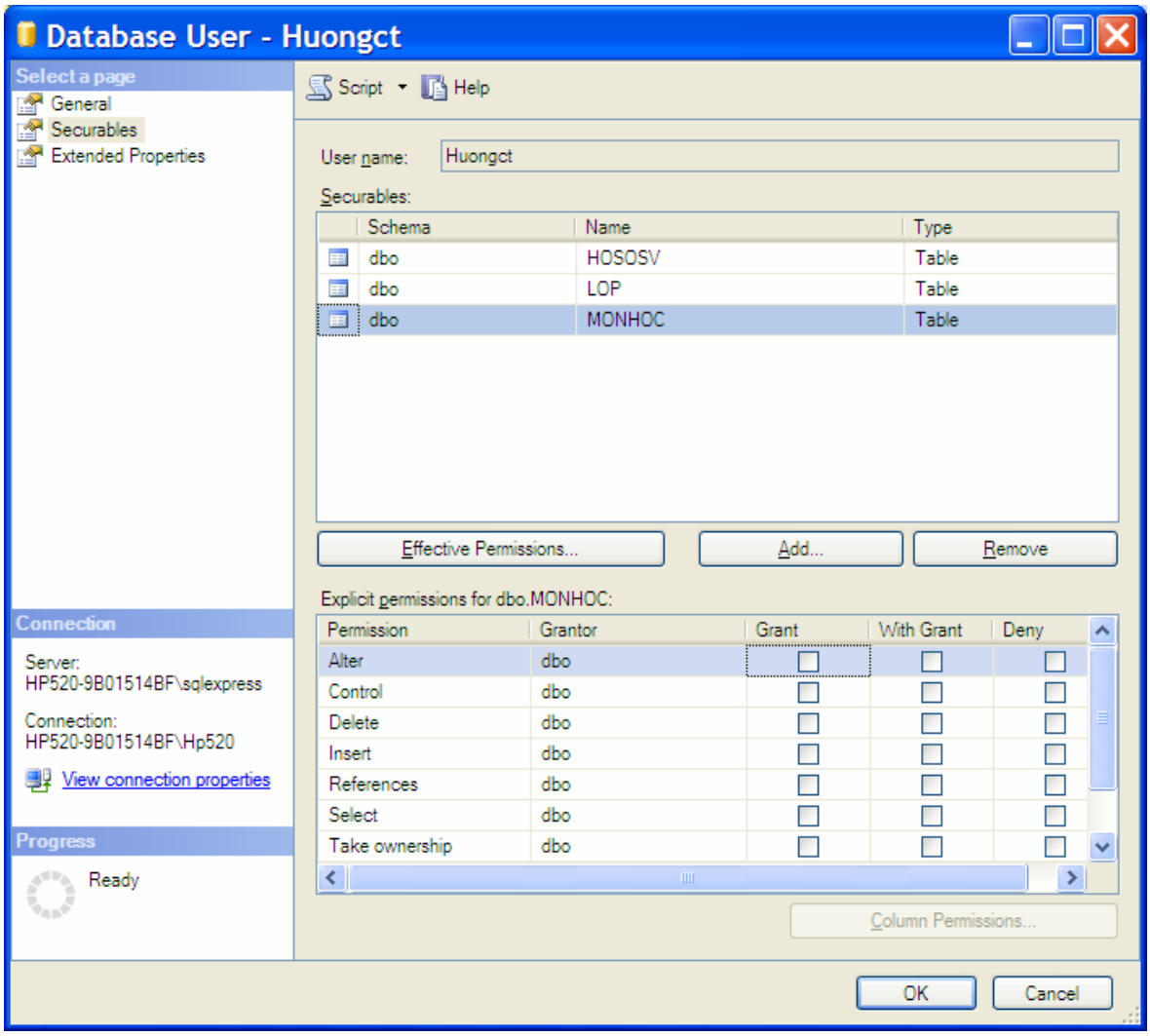
Hình 3.79. Cửa sổ Select Object Type

+ Click vào nút Browse trên hộp thoại Select Objects (Hình 3.78) xuất hiện cửa sổ Browse for Objects hình 3.80. Ta chọn các đối tượng mà muốn bảo mật cho người dùng đó và click OK.

+ Click OK trên cửa sổ Select Objects để quay trở lại trang Securable của hộp thoại Database User như hình 3.81. Ta chọn từng đối tượng và cấp quyền cho người dùng này.



Hình 3.80. Cửa sổ Browse for Objects



Hình 3.81. Cửa sổ Database User

➢ **Dùng T-SQL để cấp quyền:** Ta dùng lệnh các lệnh

+ GRANT: để cấp quyền cho người dùng;

+ DENY: dùng để ngăn cản quyền của người dùng nào đó. Ngăn cản các người dùng từ việc kế thừa các quyền trong nhóm.

+ REVOKE: để thu hồi lại quyền đã cấp.

Có cú pháp như sau:

```
GRANT { ALL }
      | permission [( column [ ,...n ] ) ] [ ,...n ]
      [ ON securable ] TO principal [ ,...n ]
      [ WITH GRANT OPTION ]
```

```
DENY { ALL }
      | permission [( column [ ,...n ] ) ] [ ,...n ]
      [ ON securable ] TO principal [ ,...n ]
      [ CASCADE]
REVOKE [ GRANT OPTION FOR ]
      {
        [ ALL ]
        |permission[( column [ ,...n ] ) ] [ ,...n ]
      }
      [ ON securable ]
      { TO | FROM } principal [ ,...n ]
      [ CASCADE]
```

Trong đó:

+ ALL: Cấp tất cả các quyền ;
+ permission: Tên các quyền cụ thể được cấp;
+ column: Tên các cột của bảng mà các quyền đó được được cấp;
+ securable: Chỉ định đối tượng đang cấp quyền trên đó.
+ Principal: Chỉ định người được cấp quyền.
+ WITH GRANT OPTION: Chỉ định người được cấp quyền có thể cấp quyền này cho người khác.
+ CASCADE: Chỉ định ngăn cản (đối với DENY) hoặc thu hồi (đối với REVOKE) theo dây truyền đối với các người dùng được cấp quyền với từ khóa WITH GRANT OPTION.

Ví dụ 3.15. Trao các quyền INSERT, SELECT, UPDATE cho *Huongct* trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
GRANT INSERT,SELECT, UPDATE
ON HOSOSV
TO Huongct
```

Ví dụ 3.16. Trao tất cả các quyền cho *Huongct* trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
```

```
GRANT ALL
ON LOP
TO Huongct
```

Ví dụ 3.17. Lấy các quyền INSERT, UPDATE của *Huongct* trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
REVOKE INSERT,UPDATE
ON HOSOSV
TO Huongct
```

Ví dụ 3.18. Lấy tất cả các quyền của *Huongct* trên bảng LOP của CSDL QLDiemSV.

```
Use QLDiemSV
Go
REVOKE ALL
ON LOP
TO Huongct
```

Ví dụ 3.19. Cấp các quyền CREATE TABLE, CREATE VIEW cho *Huongct* trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
GRANT CREATE TABLE, CREATE VIEW
TO Huongct
```

Ví dụ 3.20. Thu hồi tất cả các quyền của *Huongct* trên CSDL QLDiemSV.

```
Use QLDiemSV
Go
REVOKE ALL
TO Huongct
```

3.3.2. Sao lưu - phục hồi CSDL

3.3.2.1.Sao lưu CSDL

Sao lưu là hoạt động dữ liệu được sao chép từ CSDL và lưu ở một nơi khác. Có nhiều phương pháp sao lưu CSDL khác nhau, đó là:

- ***Sao lưu đầy đủ (Full Database):*** Là sao lưu toàn bộ CSDL, tất cả các nhóm tập tin và tập tin CSDL là một phần của CSDL đều

được sao lưu. Đây là kỹ thuật phổ biến dùng cho các CSDL có kích thước vừa và nhỏ.

- ***Sao lưu những thay đổi (Differential Database):*** Cho phép chỉ sao lưu những dữ liệu thay đổi kể từ lần sao lưu gần nhất. Kỹ thuật này nhanh hơn và ít tốn không gian lưu trữ hơn so với sao lưu đầy đủ. Nhưng dùng phương pháp này khó khăn hơn và tốn nhiều thời gian gian hơn để khôi phục dữ liệu.
- ***Sao lưu tập tin log giao dịch (Transaction Log):*** Cho phép sao lưu transaction log, sao lưu này rất quan trọng cho phục hồi CSDL.
- ***Sao lưu nhóm tập tin (Full File Group):*** bao gồm sao lưu tất cả các tập tin dữ liệu kết hợp với tập tin đơn trong CSDL. Dùng phương pháp này để sao lưu các nhóm tập tin riêng biệt tùy thuộc vào cách hệ thống được cấu hình
- ***Sao lưu tập tin dữ liệu (Full File):*** Cho phép sao lưu một tập tin đơn trong nhóm tập tin. Phương pháp này kết hợp với khả năng của SQL Server để khôi phục một tập tin dữ liệu đơn riêng biệt.

Thực hiện sao lưu dữ liệu: Ta có thể thực hiện bằng các phương pháp như dùng:

Đối với SQL Server 2000: Enterprise manager, T-SQL, Create Database Backup Wizard.

Đối với SQL Server 2005: Phương pháp sử dụng SQL Server Management Studio, T-SQL

Physical and Logical Devices

SQL Server Database Engine nhận dạng các thiết bị back up (backup devices) hoặc là tên thiết bị vật lý (physical device) hoặc tên logic (logical device) :

- Thiết bị sao lưu vật lý là tên được sử dụng bởi hệ điều hành cho việc nhận dạng thiết bị back up. Ví dụ: C:\Backups\Accounting\Full.bak.

- Thiết bị sao lưu logic là bí danh do người dùng định nghĩa, được sử dụng để nhận dạng một thiết bị sao lưu vật lý. Tên thiết bị logic được lưu trữ thường trực trên các bảng hệ thống trong SQL Server. Tiện lợi của việc sử dụng thiết bị sao lưu logic là tên của nó đơn giản hơn so với tên thiết bị vật lý. Ví dụ, ta sử dụng tên logic là Accounting_Backup nhưng tên vật lý có thể là E:\Backups\Accounting\Full.bak.

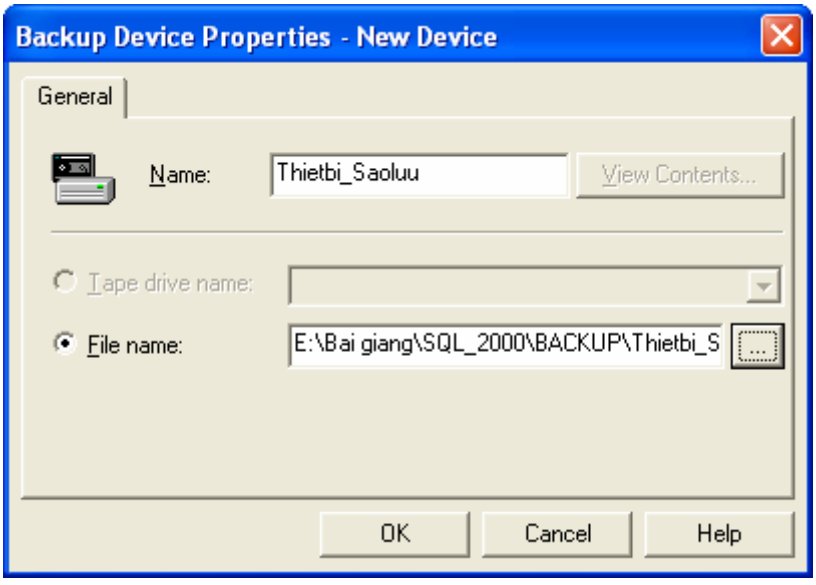
Hoạt động sao lưu có thể hướng tới thiết bị vật lý hoặc thiết bị logic. Thiết bị vật lý là đĩa cứng, băng từ,.v.v… còn thiết bị logic chỉ tồn tại trong SQL Server và chỉ được dùng cho SQL Server thực hiện sao lưu. Để sao lưu tới thiết bị logic, ta phải tạo thiết bị đó trước.

➢ **Dùng Enterprise manager:**

- ***Tạo thiết bị sao lưu logic:***

+ Trong cửa sổ Enterprise Manager, mở rộng server muốn thực hiện sao lưu và mở rộng mục Management.

+ Right click lên danh mục Backup và chọn New Backup Device xuất hiện cửa sổ *Backup Device Propeties (Hình 3.82)*. Ta nhập tên cho thiết bị sao lưu vào hộp Name và đường dẫn lưu file vào hộp thoại File Name. Sau đó click OK để lưu lại.

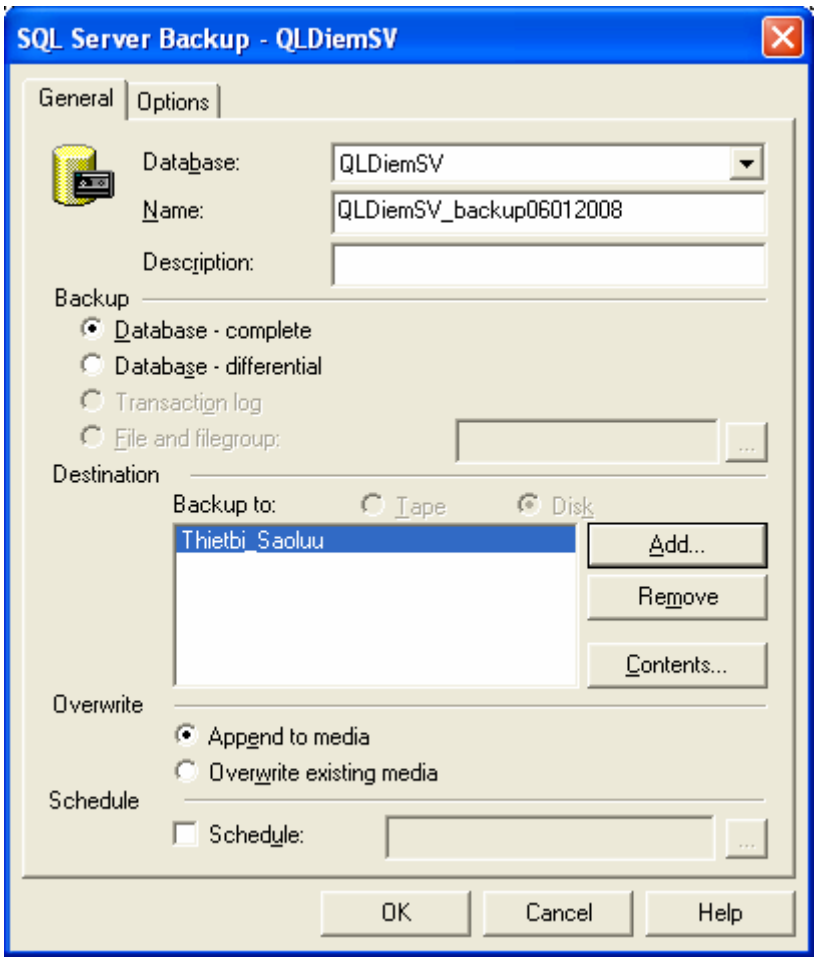


Hình 3.82. *Cửa sổ Backup Device Propeties*

- ***Thực hiện sao lưu:***

  + Trong cửa sổ Enterprise Manager, mở rộng server muốn thực hiện sao lưu và mở rộng mục Management.

  + Right click lên danh mục Backup và chọn New Backup a Database xuất hiện cửa sổ SQL Server Backup *(Hình 3.83)*.



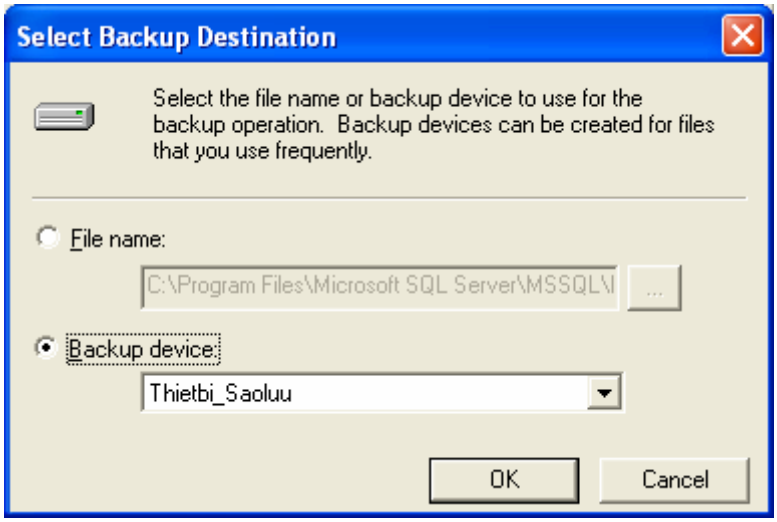
Hình 3.83. *Cửa sổ SQL Server Backup*

  + Trong danh sách Database chọn CSDL muốn thực hiện sao lưu, chẳng hạn chọn QLDiemSV. Tên sao lưu được điền tự động vào hộp Name, ta có thể thay đổi tên này. Nhập mô tả vào hộp Description.

  + Trong vùng BACKUP chỉ ra kiểu sao lưu. Có các tùy chọn có sẵn như là:

- *Database Complete*: Thực hiện sao lưu đầy đủ CSDL.
- *Database Differential*: Thực hiện sao lưu phần thay đổi.
- *Transaction log*: Thực hiện sao lưu tập tin log giao dịch.
- *File and filegroup*: Thực hiện sao lưu tập tin và nhóm tập tin.

+ Trong vùng Destination: Click vào nút Add xuất hiện cửa sổ *Select Backup Destination* hình 3.84.
  - *File Name:* Chọn vào tùy chọn này để thực hiện sao lưu thành một tập tin lưu trực tiếp xuống hệ điều hành (dùng thiết bị vật lý)
  - *Backup device:* Chọn thiết bị sao lưu logic. Chọn Thietbi_Saoluu vừa tạo ở mục trên và chọn OK.



Hình 3.84. *Cửa sổ Select Backup Destination*

+ Trong vùng Overwrite: Có 2 tùy chọn
  - *Append to media*: Ghi nối tiếp vào thiết bị
  - *Overwrite existing media*: để ghi đè lên thiết bị đang tồn tại.

+ Trong vùng Shedule: Dùng khi muốn thiết lập biểu sao lưu tự động.

+ Click OK để thực hiện sao lưu.

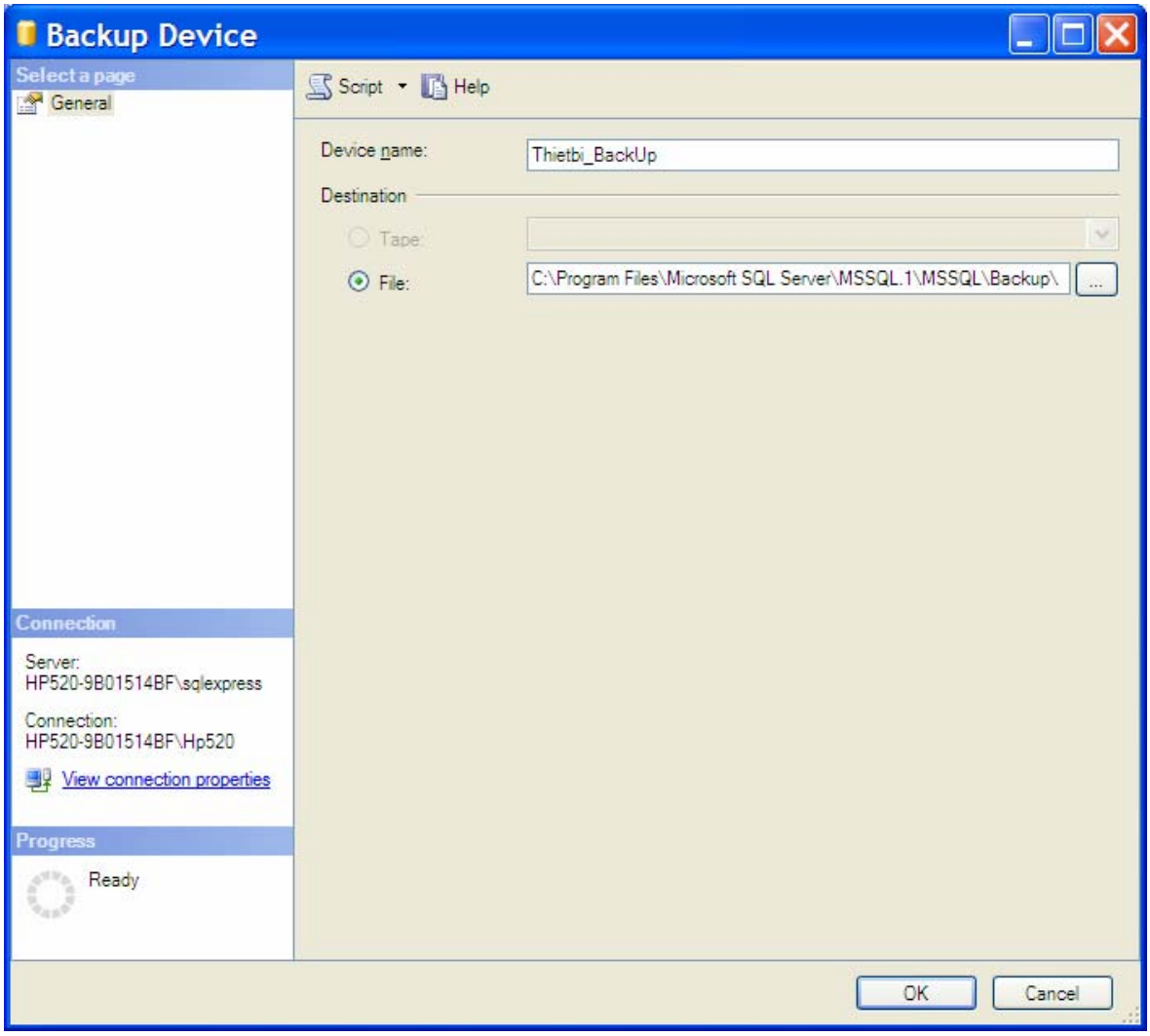
➢ **Dùng SQL Server Management Studio:**

- ***Tạo thiết bị sao lưu logic:***

  + Trong cửa sổ Object Explorer, click vào tên server để mở rộng cây server.

  + Mở rộng mục Server Objects, và right-click Backup Devices và chọn New Backup Device. Xuất hiện hộp thoại Backup Device (Hình 3.85). Trong hộp thoại mục:

    - Device name: Nhập tên thiết bị logic.
    - Destination, click File và chỉ định đường dẫn đầy đủ của file.

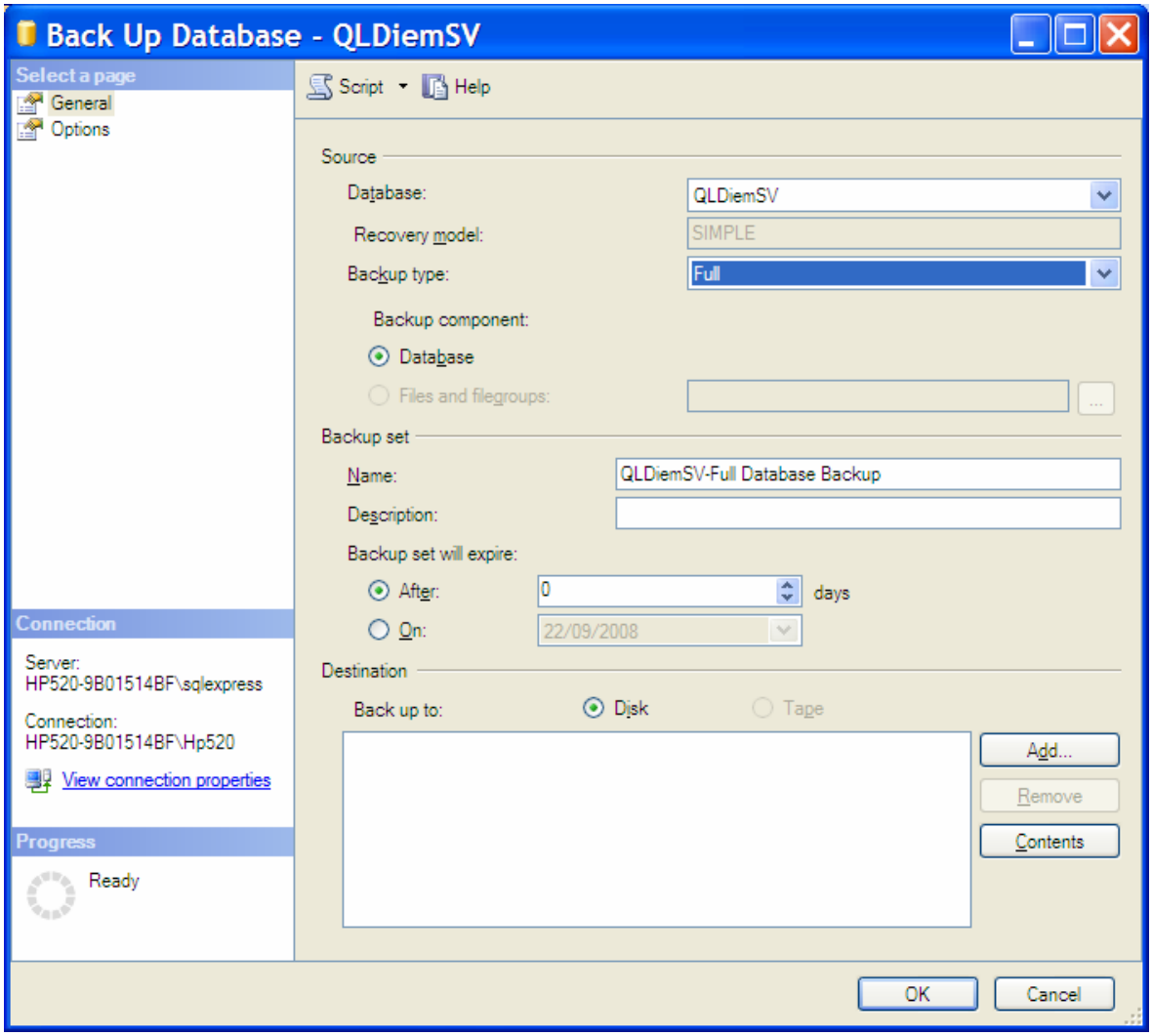


Hình 3.85. *Cửa sổ Backup Device*

- ***Thực hiện sao lưu:***

  + Trong cửa sổ Object Explorer, click vào tên server để mở rộng cây server.

  + Mở rộng mục Databases, và right-click lên cơ sở dữ liệu muốn tạo file Backup và chọn Tasks\Back Up. Xuất hiện hộp thoại Back Up Database (Hình 3.86).

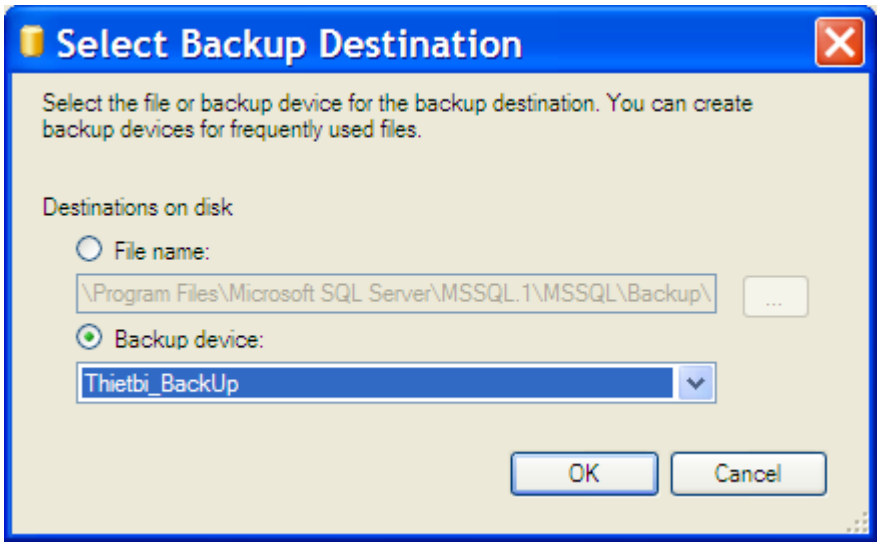


Hình 3.86. *Cửa sổ Back Up Database*

  + Trong hộp thoại ta thực hiện các lựa chọn:

    - Vùng Source: Database: Chọn Database muốn tạo file Back Up; Backup type: Chọn kiểu Back Up.

- Vùng Backup set: Thiết lập tên, mô tả,… file BackUp.
- Vùng Destination: Mục Backup to chọn Disk và click vào nút Add để chọn thiết bị sao lưu. Xuất hiện cửa sổ Select Backup Destination (Hình 3.87) ta sẽ chọn kiểu thiết bị sao lưu vật lý (chọn File name) hoặc logic (Backup device) và click OK.



Hình 3.87. *Cửa sổ* Select Backup Destination

+ Ta click vào nút OK trên cửa sổ Back Up Database để thực hiện quá trinhd sao lưu.

3.3.2.2. Phục hồi dữ liệu

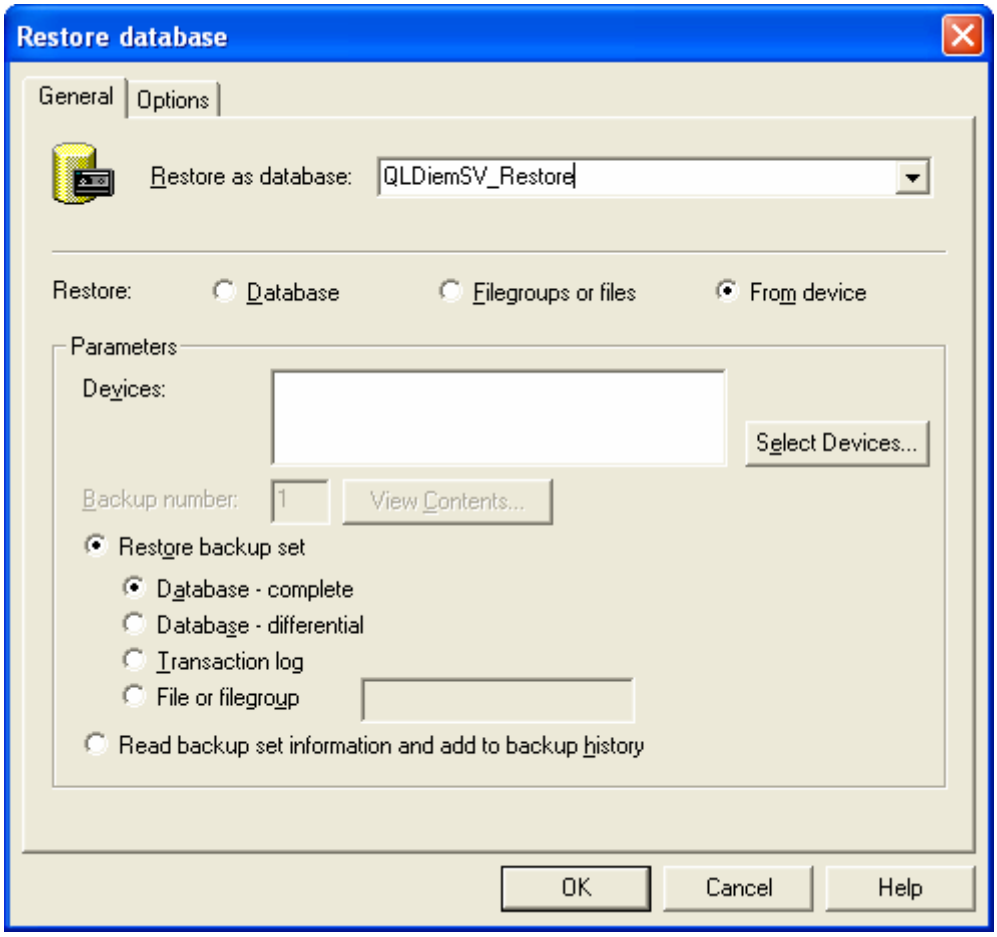
Phục hồi dữ liệu có thể coi là một quá trình ngược với quá trình sao lưu, dữ liệu sao chép được sao chép trở lại CSDL. Loại sao lưu dữ liệu sẽ ảnh hưởng đến cách khôi phục dữ liệu. Ta thực hiện các bước sau để khôi phục dữ liệu.

➢ **Dùng Enterprise manager:**

- Trong cửa sổ Enterprise Manager, right click lên mục Database. Chọn *All Task/Restore Database*. Khi đó xuất hiện hộp thoại Restore Database hình 3.88.

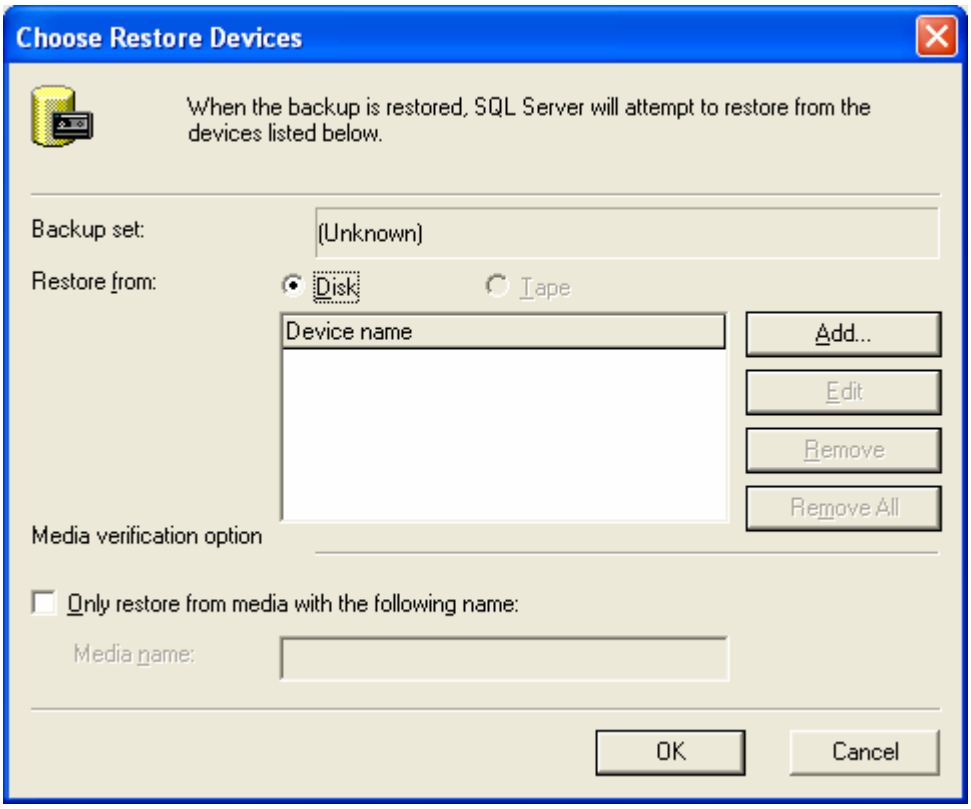
- Chọn tab General, mục Restore As Database cho phép chỉ ra CSDL sẽ được phục hồi từ dữ liệu sao lưu. Ví dụ ta chọn CSDL QLDiemSV thì việc khôi phục dữ liệu sao lưu sẽ thay thế toàn bộ dữ liệu hiện thời trong QLDiemSV. Khi đó ta phải chọn hộp check box Force Restore Over Existing Database trong tab Options.

  SQL Server không đòi hỏi phải khôi phục CSDL trực tiếp mà cho phép ta khôi phục dữ liệu với một cái tên khác. Ví dụ, người dùng xóa nhầm bảng dữ liệu. Như vậy, nếu ta khôi phục toàn bộ dữ liệu thì dữ liệu cũ sẽ thay thế toàn bộ dữ liệu đang có trên các bảng khác. Thay vào đó ta khôi phục với một cái tên khác sau đó trích bảng đã bị xóa và thêm vào CSDL đã bị mất. Ví dụ ta thay với tên QLDiemSV_Restore.



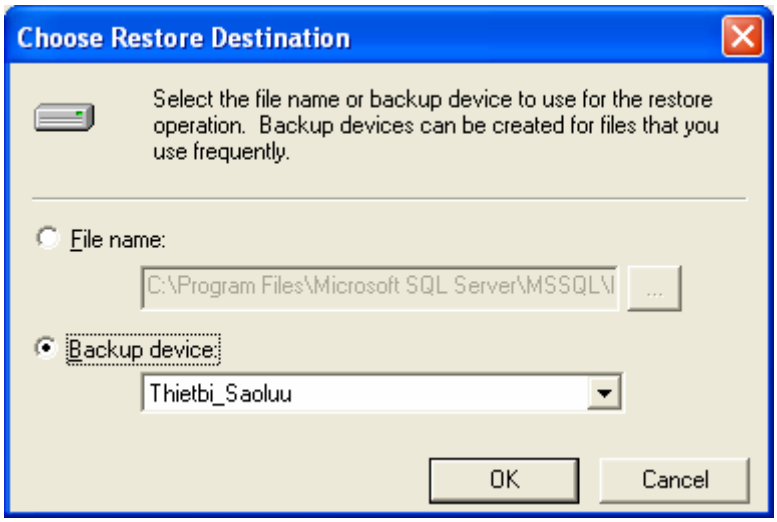
Hình 3.88. *Cửa sổ Restore Database.*

- Trong vùng Restore chọn From Device. Trong vùng Paramaters chọn Restore backup set và chọn Database complete.
- Click vào nút Select Device xuất hiện hộp thoại Choose Restore Device như hình 3.89.



Hình 3.89. *Cửa sổ Choose Restore Device.*

- Và chọn nút Add xuất hiện hộp thoại Choose Restore Destination (Hình 3.90).



Hình 3.90. *Cửa sổ Choose Restore Destination.*

- Trong đó có hai lựa chọn:
  + File name: Chọn tập tin sao lưu trên thiết bị vật lý.
  + Backup Device: Chọn tên thiết bị logic.

  Sau đó chọn OK để trở về cửa sổ hình 3.89. Trong cửa sổ 3.89 click OK để trở về cửa sổ 3.88.
- Chọn tab Options:
  + Force Restore Over Existing Database: Chọn nếu ta thực hiện khôi phục trực tiếp CSDL với dữ liệu cũ sẽ đè lên dữ liệu mới.
  + Mục Move to physical file name: Ta có thể sửa đổi lại thành tên khác với tên tập tin ban đầu. Ví dụ tập tin QLDiemSV_Restore.mdf chuyển thành QLDiemSV_Restore_Data.mdf
- Click OK để bắt đầu thực hiện phục hồi dữ liệu.

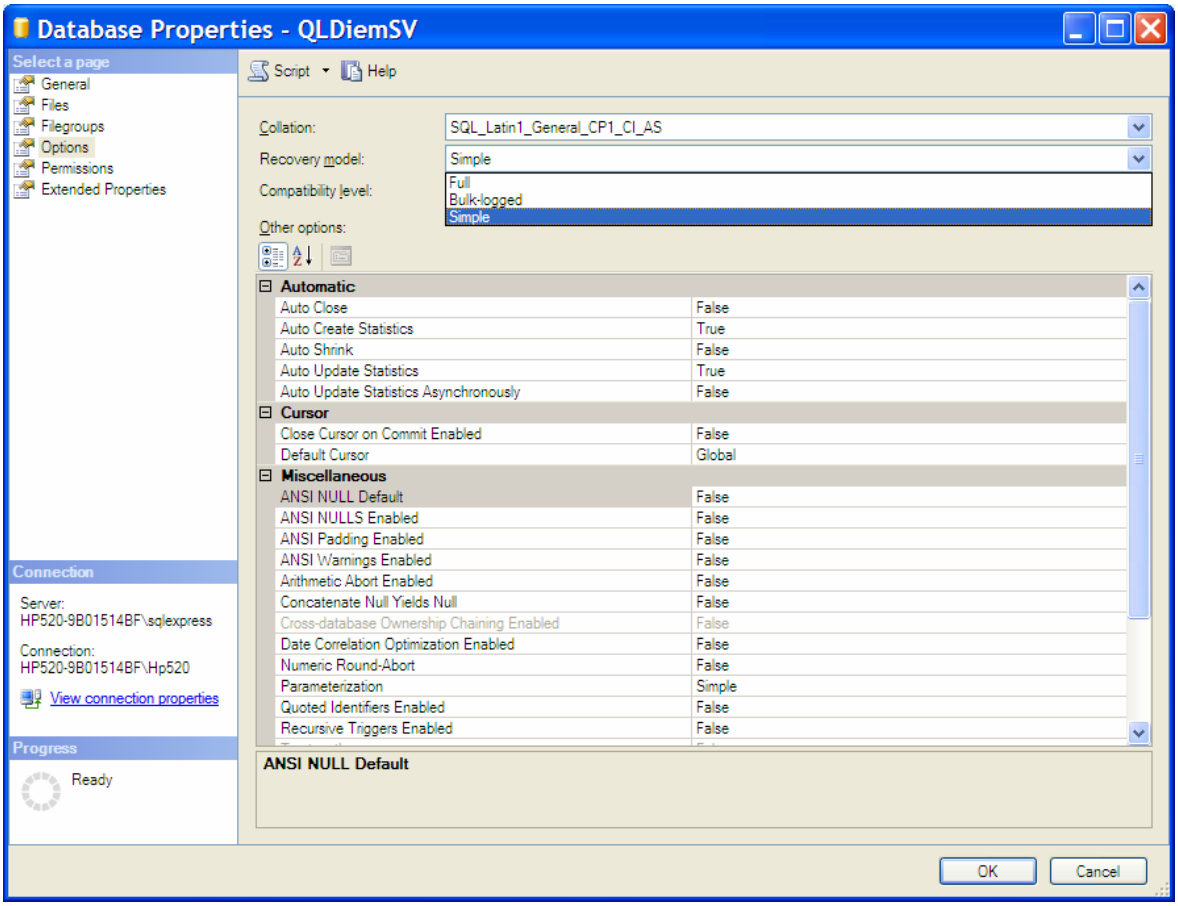
➢ **Dùng SQL Server Management Studio:**

- ***Các mô hình khôi phục dữ liệu:***
  + Full Recovery Model: Đây là mô hình cho phép phục hồi dữ liệu với ít rủi ro nhất. Nếu một database ở trong mô hình này thì tất cả các hoạt động không chỉ insert, update, delete mà kể cả insert bằng Bulk Insert, hay bcp đều được log vào transaction log file. Khi có sự cố thì ta có thể phục hồi lại dữ liệu ngược trở lại tới một thời điểm trong quá khứ. Khi data file bị hư nếu ta có thể backup được transaction log file thì ta có thể phục hồi database đến thời điểm transaction gần nhất được commited.
  + Bulk-Logged Recovery Model: Ở mô hình này này các hoạt động mang tính hàng loạt như Bulk Insert, bcp, Create Index, WriteText, UpdateText chỉ được log minimum vào transaction log file đủ để cho biết là các hoạt động này có diễn ra mà không log toàn bộ chi tiết như trong Full Recovery Mode. Các hoạt động khác như Insert, Update, Delete vẫn được log đầy đủ để dùng cho việc phục hồi sau này.

+ Simple Recovery Model: Ở mô hình này thì Transaction Log File được truncate thường xuyên. Với mo hình này bạn chỉ có thể phục hồi tới thời điểm backup gần nhất mà không thể phục hồi tới một thời điểm trong quá khứ.

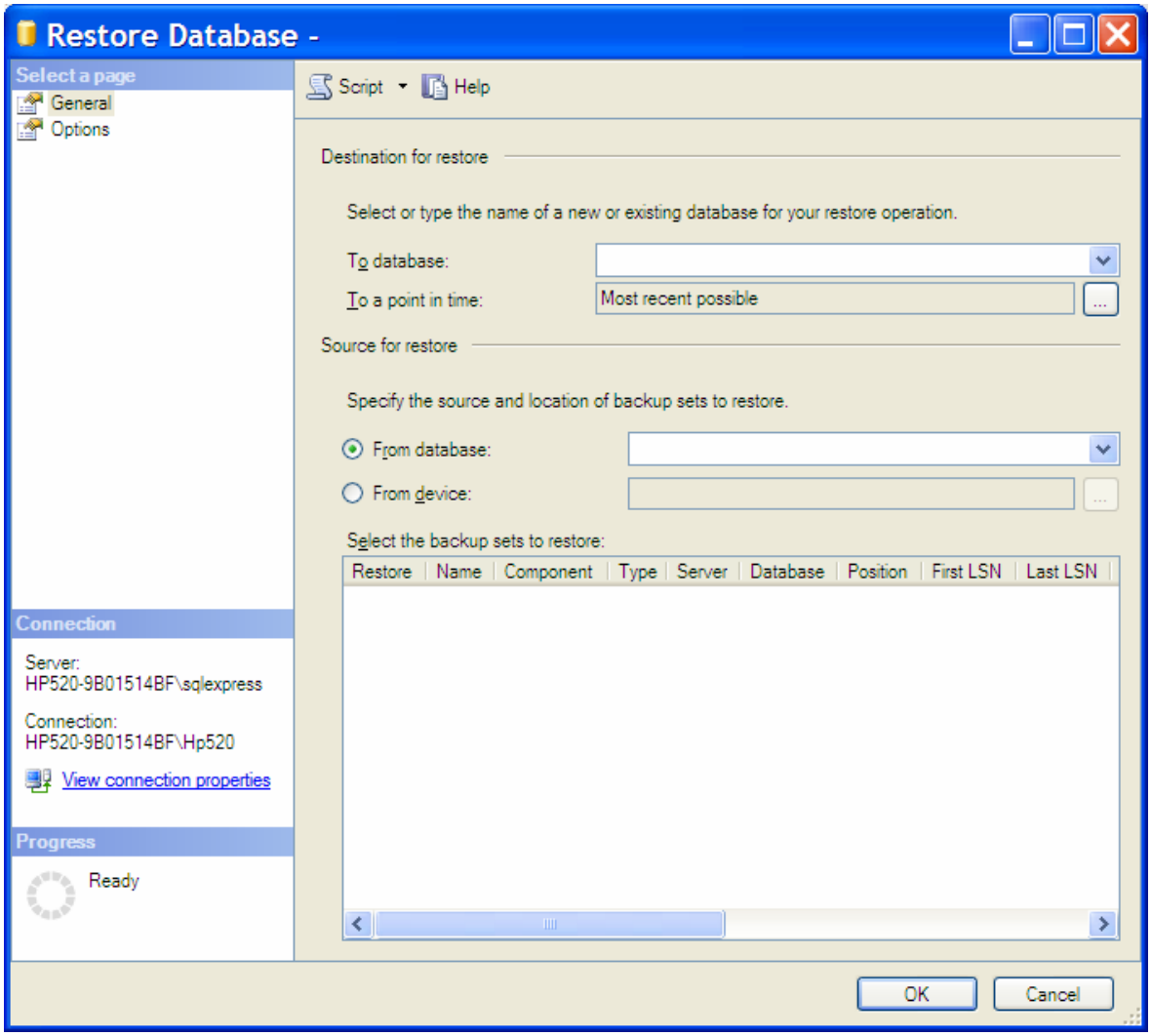
Để thay đổi mô hình Recovery, ta tiến hành thực hiện như sau:

+ Trong cửa sổ Object Explorer, click vào tên server để mở rộng cây server.
+ Mở rộng mục Databases, và right-click lên cơ sở dữ liệu muốn thay đổi mô hình Recovery và chọn Properties. Xuất hiện hộp thoại Database Properties, chọn trang Options (Hình 3.91).
+ Mục Recovery model: Ta chọn mô hình khôi phục dữ liệu.



Hình 3.91. *Cửa sổ Database Properties*

- ***Phục hồi dữ liệu:***
  - Trong cửa sổ Object Explorer, click vào tên server để mở rộng cây server.
  - Right click lên mục Databases, và chọn Restore Database xuất hiện cửa sổ Restore Database (Hình 3.92). Có các lựa chọn.

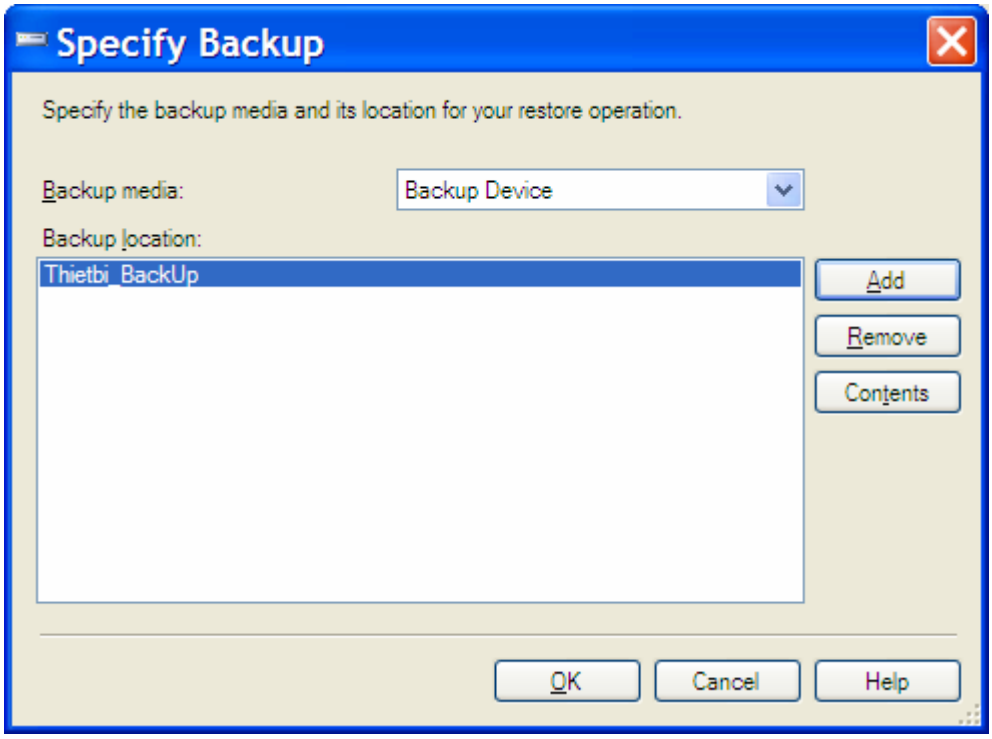


Hình 3.92. *Cửa sổ Restore Database*

+ *To database*: Điền tên Database sẽ được phục hồi. Tên này có thể là một tên Database mới hoặc chọn trong danh sách các Database có trong Server.

+ *To a point in time*: Restore database đến thời điểm sẵn có backup gần đây nhất hoặc đến một thời điểm được chỉ định.
+ *From database*: Chọn database để restore từ danh sách. Danh sách này chỉ chứa các database đã được backed up theo nhật ký của msdb backup.
+ *From device*: Chọn nguồn từ tập các file backup. Click vào nút browse để xuất hiện cửa sổ Specify Backup (Hình 3.93). Tong hộp thoại này ta chọn kiểu thiết bị (Backup media) là thiết bị vật lý hay thiết bị logic. Click nút Add để lấy các file Backup dùng để phục hồi dữ liệu.

o Trong cửa sổ Restore Database, sau khi thiết lập các tham số click OK để thực hiện quá trình phục hồi dữ liệu.



Hình 3.93. *Cửa sổ Restore Database*

Chương 4. LẬP TRÌNH TRÊN SQL SERVER

4.1. Giới thiệu ngôn ngữ T-SQL

4.1.1. Khái niệm

Transaction SQL (T-SQL) là ngôn ngữ phát triển nâng cao của ngôn ngữ SQL chuẩn. Nó là ngôn ngữ dùng để giao tiếp giữa ứng dụng và SQL Server. T-SQL các khả năng của ngôn ngữ định nghĩa dữ liệu - DDL và ngôn ngữ thao tác dữ liệu – DML của SQL chuẩn cộng với một số hàm mở rộng, các store procedure hệ thống và cấu trúc lập trình (như IF, WHILE,…) cho phép lập trình trên SQL Server được linh động hơn.

Trong các chương trước ta đã giới thiệu ngôn ngữ SQL chuẩn và làm quen với các câu lệnh T-SQL dùng để định nghĩa dữ liệu, thao tác dữ liệu như: Tạo CSDL, tạo bảng, tạo View, tạo Index, chèn dữ liệu,.v.v… Trong chương này ta sẽ tìm hiểu thêm về T-SQL.

4.1.2. Phát biểu truy vấn dữ liệu nâng cao

a) Mệnh đề TOP

Mệnh đề TOP chỉ định tập hợp các dòng đầu tiên được trả về trong truy vấn. Tập hợp các dòng đó có thể là một con số hoặc theo tỷ lên phần trăm (PERCENT) các dòng dữ liệu. Mệnh đề TOP được sử dụng trong các khối câu lệnh Select, Insert, Update và Delete. Cú pháp:

```
[     TOP (expression) [PERCENT]
      [ WITH TIES ]
]
```

Trong đó:

- `expression`: Là biểu thức trả về giá trị kiểu số.
- `PERCENT`: Chỉ định số dòng trả về là expression phần trăm trong tập kết quả.
- `WITH TIES`: TOP ...WITH TIES chỉ được chỉ định trên khối câu lệnh SELECT và có mệnh đề ORDER BY. Chỉ định thêm các dòng từ tập kết quả cơ sở có cùng giá trị với các cột

trong mệnh đề ORDER BY xuất hiện như là dòng cuối cùng của TOP n (PERCENT).

Ví dụ 4.1. Sử dụng mệnh đề TOP

- Trong câu lệnh Insert

```
INSERT TOP (2) INTO LOP
     SELECT * FROM DMLOP ORDER BY Khoa
```

- Trong câu lệnh Select

```
INSERT INTO LOP
     SELECT TOP (2) WITH TIES * FROM DMLOP
             ORDER BY Khoa
```

b) Điều kiện kết nối - JOIN

Trong khối câu lệnh SELECT, ở mệnh đề FROM ta có thể sử dụng phát biểu JOIN để kết nối các bảng có quan hệ với nhau.

Mệnh đề kết nối Join được phân loại như sau:

- Inner joins (toán tử thường dùng để kết nối thường là các toán tử so sánh = hoặc <>). Inner joins sử dụng một toán tử so sánh để so khớp các dòng từ hai bảng dựa trên các giá trị của các cột so khớp của mỗi bảng. Kết quả trả về của Inner Join là các dòng thỏa mãn điều kiện so khớp.
- Outer joins. Outer joins có thể là left, right, hoặc full outer join.
  - LEFT JOIN hoặc LEFT OUTER JOIN : Kết quả của left outer join không chỉ bao gồm các dòng thỏa mãn điều kiện so khớp giữa hai bảng mà còn gồm tất cả các dòng của bảng bên trái trong mệnh đề LEFT OUTER. Khi một dòng ở bảng bên trái không có dòng nào của bảng bên phải so khớp đúng thì các giá trị NULL được trả về cho tất cả các cột ở bảng bên phải.
  - RIGHT JOIN or RIGHT OUTER JOIN: Right outer join là nghịch đảo của left outer join. Tất cả các dòng của bảng bên phải được trả về. Các giá trị Null cho bảng bên trái khi

bất cứ một dòng nào bên phải không có một dòng nào bảng bên trái so khớp đúng.

+ FULL JOIN or FULL OUTER JOIN: full outer join trả về tất cả các dòng trong cả hai bảng bên trái và phải. Bất kỳ một dòng không có dòng so khớp đúng của bảng còn lại thì bảng còn lại nhận các giá trị NULL. Khi có sự so khớp đúng giữa các bảng thì tập kết quả sẽ chứa dữ dữ liệu các bảng cơ sở đó.

➢ Cross joins: Trả về tất cả các dòng của bảng bên trái và mỗi dòng bên trái sẽ kết hợp với tất cả các dòng của bảng bên phải. Cross joins còn được gọi là tích Đề các (Cartesian products).

Ví dụ 4.2. Sử dụng Join

- Inner Joins:

```
SELECT MONHOC.MaMH, MONHOC.TenMH, MONHOC.SDVHT,
       DIEM.MaSV, DIEM.DiemL1
FROM DIEM INNER JOIN MONHOC
       ON DIEM.MaMH = MONHOC.MaMH
```

- Left Joins:

```
SELECT MONHOC.MaMH, MONHOC.TenMH, MONHOC.SDVHT,
       DIEM.MaSV, DIEM.DiemL1
FROM  MONHOC LEFT JOIN  DIEM
       ON MONHOC.MaMH= DIEM.MaMH
```

- Right Joins:

```
SELECT MONHOC.MaMH, MONHOC.TenMH, MONHOC.SDVHT,
       DIEM.MaSV, DIEM.DiemL1
FROM  DIEM Right JOIN MONHOC
      ON DIEM.MaMH= MONHOC.MaMH
```

- Full Joins:

```
SELECT MONHOC.MaMH, MONHOC.TenMH, MONHOC.SDVHT,
       DIEM.MaSV, DIEM.DiemL1
FROM  DIEM Full JOIN MONHOC
      ON DIEM.MaMH= MONHOC.MaMH
```

- Cross Joins:

```
SELECT MONHOC.MaMH, MONHOC.TenMH, MONHOC.SDVHT,
       DIEM.MaSV, DIEM.DiemL1
FROM MONHOC CROSS JOIN DIEM
```

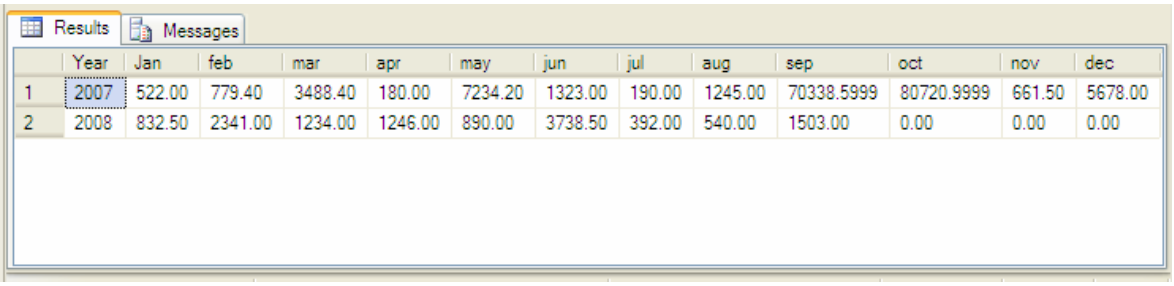
c) Truy vấn Cross tab

Trong một số trường hợp thống kê, ta cần phải xoay bảng kết quả, do đó có các cột được biểu diễn theo chiều ngang và các dòng được biểu diễn theo chiều dọc (được gọi là truy vấn cross tab).

Ví dụ 4.3. Ví dụ ta có một view tính tổng giá trị của một hóa đơn View_Order (OrderID, OrderDate, Month, Year, Total). Ta cần thống kê doanh thu theo từng tháng của các năm.

```
SELECT Year,
    SUM(CASE Month WHEN 1 THEN Total ELSE 0 END) AS Jan,
    SUM(CASE Month WHEN 2 THEN Total ELSE 0 END) AS feb,
    SUM(CASE Month WHEN 3 THEN Total ELSE 0 END) AS mar,
    SUM(CASE Month WHEN 4 THEN Total ELSE 0 END) AS apr,
    SUM(CASE Month WHEN 5 THEN Total ELSE 0 END) AS may,
    SUM(CASE Month WHEN 6 THEN Total ELSE 0 END) AS jun,
    SUM(CASE Month WHEN 7 THEN Total ELSE 0 END) AS jul,
    SUM(CASE Month WHEN 8 THEN Total ELSE 0 END) AS aug,
    SUM(CASE Month WHEN 9 THEN Total ELSE 0 END) AS sep,
    SUM(CASE Month WHEN 10 THEN Total ELSE 0 END) AS oct,
    SUM(CASE Month WHEN 11 THEN Total ELSE 0 END) AS nov,
    SUM(CASE Month WHEN 12 THEN Total ELSE 0 END) AS dec
FROM View_Order
GROUP BY Year
```

Kết quả:



Results | Messages

	Year	Jan	feb	mar	apr	may	jun	jul	aug	sep	oct	nov	dec
1	2007	522.00	779.40	3488.40	180.00	7234.20	1323.00	190.00	1245.00	70338.5999	80720.9999	661.50	5678.00
2	2008	832.50	2341.00	1234.00	1246.00	890.00	3738.50	392.00	540.00	1503.00	0.00	0.00	0.00

Sử dụng toán tử PIVOT và UNPIVOT

SQL Server 2005 đưa ra các toán tử đơn giản hơn cho việc tạo truy vấn cross tab, đó là toán tử PIVOT và UNPIVOT trong mệnh đề FROM của khối câu lệnh SELECT.

+ Toán tử PIVOT thực hiện xoay một biểu thức giá trị bảng (table valued expression) thành một bảng khác bằng việc đưa các giá trị duy nhất của một cột thành các cột và thực hiện các hàm thống kê trên các cột còn lại.

+ Toán tử UNPIVOT thực hiện quá trình ngược lại với quá trình thực hiện của toán tử PIVOT, xoay các cột của biểu thức bảng thành giá trị của một cột.

Cú pháp:

```
FROM { <table_source> } [ ,...n ]
<table_source> ::=
{
     <pivoted_table>
    | <unpivoted_table> [ ,...n ]
}

<pivoted_table> ::=
       table_source PIVOT <pivot_clause> table_alias

<pivot_clause> ::=
        ( aggregate_function( value_column )
        FOR pivot_column
        IN ( <column_list> )
    )

<unpivoted_table> ::=
        table_source UNPIVOT <unpivot_clause>
table_alias

<unpivot_clause> ::=
        ( value_column FOR pivot_column IN (
<column_list> ) )

<column_list> ::=
          column_name [ , ... ]
```

Trong đó:

+ *table_source* PIVOT <pivot_clause> : Chỉ định bảng *table_source* được xoay dựa trên cột *pivot_column*. *table_source* là một bảng hoặc biểu thức bảng. Output là một

bảng chứa tất cả các cột của *table_source* trừ cột *pivot_column* và *value_column*. Các cột của *table_source*, trừ *pivot_column* và *value_column*, được gọi là các cột phân nhóm của toán tử pivot.

+ *aggregate_function:* Là một hàm thống kê của hệ thống hoặc do người dùng định nghĩa. Hàm COUNT(*) không được phép sử dụng trong trường hợp này.

+ *value_column:* Là cột giá trị của toán tử PIVOT. Khi sử dụng với toán tử UNPIVOT, *value_column* không được trùng tên với các cột trong bảng input *table_source*.

+ FOR *pivot_column* : Chỉ định trục xoay của toán tử PIVOT. *pivot_column* là có kiểu chuyển đổi được sang **nvarchar()**. KHông được là các kiểu **image** hoặc **rowversion**.

  Khi UNPIVOT được sử dụng, *pivot_column* là tên của cột output được thu hẹp lại từ *table_source*. Tên cột này không được trùng với một tên nào trong *table_source*.

+ IN **(** *column_list* **)** : Trong mệnh đề PIVOT, danh sách các giá trị trong *pivot_column* sẽ trở thành tên các cột trong bảng output. Danh sách này không được trùng với bất kỳ tên cột nào tồn tại trong bảng input *table_source* mà đang được xoay.

  Trong mệnh đề UNPIVOT, danh sách các cột trong *table_source* sẽ được thu hẹp lại thành một cột *pivot_column*.

+ *table_alias:* Là tên bí danh của bảng output. *pivot_table_alias* phải được chỉ định.

+ UNPIVOT < unpivot_clause > : Chỉ định bảng input được thu hẹp bằng các cột trong *column_list* trở thành một cột gọi là *pivot_column*.

**** Hoạt động của toán tử PIVOT:***

Toán tử PIVOT thực hiện theo tiến trình sau:

+ Thực hiện GROUP BY dựa vào các cột phân nhóm trên bảng input_table và kết quả là ứng với mỗi nhóm cho một dòng out put trên bảng kết quả.

+ Sinh các giá trị ứng với các cột trong danh sách column list cho mỗi dòng output bằng việc thực thi như sau:

  - Nhóm các dòng được sinh từ việc GROUP BY ở bước trước dựa trên cột *pivot_column*.

    Đối với mỗi cột output trong *column_list*, chọn một nhóm con thỏa mãn điều kiện:

    ```
    pivot_column=CONVERT(<data type of pivot_column>, 'output_column')
    ```

  - *aggregate_function* định giá trị dựa tên cột *value_column* trong nhóm con này và kết quả được trả về của nó tương ứng là giá trị của cột *output_column*. Nếu nhóm con là rỗng thì SQL Server sinh giá trị NULL cho cột *output_column* đó. Nếu hàm thống kê là COUNT thì nó sinh giá trị 0.

Ví dụ 4.5. Ví dụ ta có một view tính tổng giá trị của một hóa đơn View_Order (OrderID, OrderDate, Month, Year, Total). Ta cần thống kê doanh thu theo từng tháng của các năm.

```
SELECT Year,[1]AS Jan,[2]AS feb, [3]AS mar,[4] AS apr,[5]
AS may,[6] AS jun,[7] AS jul,[8] AS aug,[9] AS sep,
[10]AS oct,[11] AS nov,[12] AS dec
FROM
    (SELECT Year, Month,Total
    FROM View_Order) p
PIVOT
    (Sum(Total) FOR Month IN
        ([1], [2], [3], [4], [5], [6], [7], [8], [9],
[10], [11], [12])
)AS pvt
```

Ví dụ 4.6. Sử dụng PIVOT

```
USE AdventureWorks
GO
SELECT VendorID, [164] AS Emp1, [198] AS Emp2, [223] AS
Emp3, [231] AS Emp4, [233] AS Emp5
FROM
```

```
(SELECT PurchaseOrderID, EmployeeID, VendorID
FROM Purchasing.PurchaseOrderHeader) p
PIVOT
(
COUNT (PurchaseOrderID)
FOR EmployeeID IN
( [164], [198], [223], [231], [233] )
) AS pvt
ORDER BY VendorID;
```

Ví dụ 4.7. Sử dụng UNPIVOT

```
CREATE TABLE pvt (VendorID int, Emp1 int, Emp2 int,
Emp3 int, Emp4 int, Emp5 int)
GO
INSERT INTO pvt VALUES (1,4,3,5,4,4)
INSERT INTO pvt VALUES (2,4,1,5,5,5)
INSERT INTO pvt VALUES (3,4,3,5,4,4)
INSERT INTO pvt VALUES (4,4,2,5,5,4)
INSERT INTO pvt VALUES (5,5,1,5,5,5)
GO
--Unpivot the table.
SELECT VendorID, Employee, Orders
FROM
    (SELECT VendorID, Emp1, Emp2, Emp3, Emp4, Emp5
    FROM pvt) p
UNPIVOT
    (Orders FOR Employee IN
        (Emp1, Emp2, Emp3, Emp4, Emp5)
)AS unpvt
```

d) UNION và UNION ALL

Toán tử UNION [ALL] dùng để hợp kết quả của hai hoặc nhiều câu truy vấn tương thích với nhau. Hai câu truy vấn tương thích là hai câu có cùng cấu trúc, tức là có cùng số cột và tập các cột tương ứng có cùng kiểu dữ liệu hoặc có các kiểu dữ liệu tương thích nhau. Cú pháp của câu lệnh:

```
select_statement UNION [ALL] select_statement
```

Tên của các cột trong phép toán UNION là tên các cột trong tập kết quả của khối câu lệnh SELECT thứ nhất trong UNION.

Theo mặc định phép toán UNION chỉ lấy đại diện cho tập các dòng trùng nhau. Nếu ta sử dụng từ khóa ALL, thì tất cả các dòng được cho vào bảng kết quả và các dòng trùng nhau sẽ không loại bỏ các dòng trùng nhau.

Ví dụ 4.8. Sử dụng UNION

```
SELECT * from LOP
UNION ALL
SELECT * from DMLOP
```

4.1.3. Lập trình cấu trúc trong SQL Server

a) Các toán tử

- *Toán tử gán:* Ký hiệu là dấu '=' được dùng để gán giá trị cho một biến hoặc một cột.

```
DECLARE @intValue int
SELECT @intValue = 1
PRINT @intValue
```

hoặc

```
DECLARE @intValue int
SET @intValue = 1
PRINT @intValue
```

- *Toán tử số học*: Đó là các phép toán cộng (+), trừ (-), nhân (∗), chia (/) và chia modul (%) .

```
12+4=16
12-4=8
12*4=48
12/4=3
15%2=1
```

- *Toán tử so sánh*: Đó là các phép toán so sánh giữa hai biểu thức và trả về giá TRUE hoặc FALSE. Đó là các phép so sánh: = (bằng), <> (khác), > (lớn hơn), >= (lớn hơn hoặc bằng), < (nhỏ hơn), <= (nhỏ hơn hoặc bằng).

- *Toán tử logic*: Kiểm tra điều kiện đúng của hai biểu thức, chúng thường được sử dụng cùng với các toán tử so sánh để trả về giá trị TRUE hoặc FALSE. Các toán tử logic được cho trong bảng 4.1 sau.

Bảng 4.1. Các toán tử logic

Toán tử	*Ý nghĩa*	*Ví dụ*
ALL	So sánh một giá trị vô hướng với một tập các giá trị của một cột được lấy từ một câu truy vấn con. ALL trả về giá trị TRUE nếu tất cả các giá trị trong cột trả về giá trị TRUE ngược lại trả về	5 > ALL (SELECT * FROM sales)

	giá trị FALSE.	
AND	Kết hợp và so sánh giữa hai biểu thức Boolean, nếu cả hai biểu thức đều TRUE thì nó trả về giá trị TRUE và ngược lại nó trả về giá trị FALSE.	5 > 7 AND 6 < 15
ANY	So sánh một giá trị vô hướng với một tập các giá trị của một cột được lấy từ một câu truy vấn con. Nó sẽ trả về giá trị TRUE nếu có bất cứ giá trị nào trong cột trả về giá trị TRUE. Nếu không có một giá trị nào trả về giá trị TRUE thì nó trả về giá trị FALSE. ANY tương tự như toán tử SOME.	5 > ANY (SELECT qty FROM sales)
BETWEEN	Kiểm tra giá trị có nằm giữa phạm vi được chỉ định hay không. Trả về giá trị TRUE nếu nó nằm trong khoảng giá trị đó và ngược lại trả giá trị FALSE.	5 BETWEEN (3 AND 10)
EXISTS	Kiểm tra xem có giá trị nào trả về khi thực hiện một câu truy vấn. Nếu có các giá trị trả về thì toán tử cho giá trị TRUE, ngược lại trả về giá trị FALSE.	EXISTS (SELECT * FROM test)
IN	Kiểm tra xem một giá trị có tồn tại trong một tập các giá trị hay không. Nếu giá trị mà thuộc tập giá trị đó thì toán tử trả về giá trị TRUE, ngược lại trả về giá trị FALSE.	5 IN (SELECT qty FROM sales)
LIKE	Dùng để so khớp các giá trị với một mẫu theo từ khóa LIKE. Nó sẽ trả về giá trị TRUE nếu khớp với mẫu ngược lại trả về giá trị FALSE. Ký tự % đại diện cho một dãy ký tự bất kỳ, _ đại diện cho một ký tự bất kỳ.	SELECT name WHERE name LIKE ‘S%’

NOT	Dùng để phủ định một biểu thức Boolean.	NOT 5 > 2
OR	Kết hợp và so sánh giữa hai biểu thức Boolean, nếu một trong hai biểu thức là TRUE thì nó trả về giá trị TRUE và ngược lại nó trả về giá trị FALSE.	5 > 2 OR 10 < 3
SOME	So sánh một giá trị vô hướng với một tập các giá trị của một cột được lấy từ một câu truy vấn con. Nó sẽ trả về giá trị TRUE nếu có bất cứ giá trị nào trong cột trả về giá trị TRUE. Nếu không có một giá trị nào trả về giá trị TRUE thì nó trả về giá trị FALSE. SOME tương tự như toán tử ANY.	5 > SOME (SELECT * FROM sales)

- *Toán tử ghép chuỗi (+)*: Dùng để ghép hai chuỗi với nhau thành một chuỗi. Toán tử ghép chuỗi được dùng với các kiểu dữ liệu char, varchar, nchar, nvarchar, text, và ntext.

```
SELECT 'This' + ' is a test.'
```

- *Toán tử bit*: Thực hiện thao tác với các bit-lavel với các kiểu dữ liệu Integer. Các toán tử đó được cho trong bảng 4.2.

Bảng 4.2. Các toán tử Bitwise

Toán tử	*Ý nghĩa*	*Ví dụ*
&	Thực hiện AND giữa các bit tương ứng giữa hai biểu diễn nhị phân của hai số integer.	7 & 51 = 3 (7=111, 51=110011, 3=11)
|	Thực hiện OR giữa các bit tương ứng giữa hai biểu diễn nhị phân của hai số integer.	7 | 51 = 55
^	Thực hiện XOR giữa các bit tương ứng giữa hai biểu diễn nhị phân của hai số integer. (hai bit giống nhau trả về bit 0, khác nhau trả về bit 1)	7 ^51 = 52

~	Thực hiện NOT của biểu thức biểu diễn nhị phân của mọt số nguyên	~7 = -8

b) Cấu trúc lặp

SQL Server cung cấp hai cấu trúc lặp đó là: cấu trúc WHILE và GOTO.

- Cấu trúc lặp WHILE: Câu lệnh WHILE sẽ kiểm tra điều kiện trước khi thực hiện lệnh. Một khối lệnh là một tập các câu lệnh được bao trong cặp từ khóa BEGIN …END. Cú pháp:

```
WHILE Boolean_expression
    {sql_statement| statement_block}
    [BREAK]
    {sql_statement| statement_block}
    [CONTINUE]
```

trong đó:

+ *Boolean_expression:* Là biểu thức điều kiện để kiểm tra điều kiện lặp. Vòng lặp sẽ được thực hiện khi biểu thức trả về giá trị True và kết thúc vòng lặp khi trả về giá trị False.
+ *sql_statement|statement_block*:Đó là câu lệnh SQL hoặc khối các câu lệnh SQL sẽ được lặp lại trong câu lệnh While. Khối các câu lệnh SQL được bao trong cặp từ khóa BEGIN … END
+ *BREAK:* Từ khóa dùng để chỉ định dừng việc thực thi vòng lặp hiện tại. Tất cả các câu lệnh sau từ khóa BREAK và trước từ khóa END sẽ bị bỏ qua.
+ *CONTINUE*: Từ khóa dùng để restart lại vòng lặp hiện tại tại ví trí bắt đầu. Tất cả các câu lệnh sau từ khóa CONTINUE và trước từ khóa END sẽ bị bỏ qua.

Ví dụ 4.5. Sử dụng cấu trúc lặp WHILE đơn giản.

```
Use pubs
go
CREATE TABLE WhileLoopTest
(
    LoopID        INT,
```

```
    LoopValue    VARCHAR(32)
)
GO
SET NOCOUNT ON
DECLARE  @intCounter    INT
DECLARE @vchLoopValue    VARCHAR(32)

SELECT @intCounter = 1
WHILE (@intCounter <= 100)
 BEGIN
    SELECT @vchLoopValue = 'Loop Interation #' +
           CONVERT (VARCHAR(4), @intCounter)
    INSERT INTO WhileLoopTest(LoopID, LoopValue)
           VALUES (@intCounter, @vchLoopValue)
    SELECT @intCounter = @intCounter + 1
 END
```

- Cấu trúc lặp GOTO: Tương tự như cấu trúc WHILE, GOTO có thể cho phép lặp một chuỗi câu lệnh cho đến khi điều kiện được thỏa mãn.

Chú ý: Câu lệnh GOTO không nhất thiết phải sử dụng trong các vòng lặp mà có thể sử dụng để thoát khỏi vòng lặp khác.

Để sử dụng câu lệnh GOTO, trước hết ta phải định nghĩa một nhãn. Nhãn là một câu lệnh chỉ định vị trí mà câu lệnh GOTO sẽ nhảy đến. Để tạo nhãn ta sử dụng cú pháp sau:

```
LABLE:
```

Để nhảy đến nhãn trong code ta sử dụng câu lệnh GOTO theo cú pháp sau:

```
GOTO  LABLE
```

Trong đó: LABLE là nhãn đã được định nghĩa ở trước đó trong code. Bằng việc sử dụng GOTO, ta có thể nhảy đến một vị trí bất kỳ trong code.

Ví dụ 4.6. Sử dụng cấu trúc lặp GOTO đơn giản.

```
Use pubs
Go
CREATE TABLE GotoLoopTest
(
    GotoID        INT,
    GotoValue     VARCHAR(32)
)
GO

SET NOCOUNT ON
```

```
    DECLARE    @intCounter    INT
    DECLARE    @vchLoopValue  VARCHAR(32)

    SELECT @intCounter = 0

    LOOPSTART:
    SELECT @intCounter = @intCounter + 1
    SELECT @vchLoopValue = 'Loop Iteration #' +
               CONVERT(VARCHAR(4), @intCounter)
    INSERT INTO GotoLoopTest(GotoID, GotoValue) VALUES
(@intCounter, @vchLoopValue)
    IF (@intCounter <= 1000)
    BEGIN
        GOTO LOOPSTART
    END
```

c) Cấu trúc rẽ nhánh

- *Cấu trúc IF...ELSE*: Cấu trúc IF…ELSE là một khối các câu lệnh dùng để rẽ nhánh dựa trên các tham số được cung cấp. Cú pháp của khối câu lệnh IF như sau:

```
IF expression
    BEGIN
    sql_statements
    END
[ELSE
    BEGIN
    sql_statements
    END]
```

Chú ý: Ta có thể sử dụng các cấu trúc IF lồng nhau.

Ví dụ 4.7. Sử dụng cấu trúc rẽ nhánh IF.

```
Use pubs
Go

CREATE PROCEDURE uspCheckNumber
    @intNumber    INT
AS

IF @intNumber < 1
   BEGIN
    PRINT 'Number is less that 1.'
    RETURN
   END

ELSE IF @intNumber = 1
  BEGIN
    PRINT 'One'
    RETURN
  END
```

```
ELSE IF @intNumber = 2
 BEGIN
    PRINT 'Two'
    RETURN
 END
ELSE IF @intNumber = 3
 BEGIN
    PRINT 'Three'
    RETURN
 END
ELSE IF @intNumber = 4
 BEGIN
    PRINT 'Four'
    RETURN
 END
ELSE IF @intNumber = 5
 BEGIN
    PRINT 'Five'
    RETURN
 END
ELSE IF @intNumber = 6
 BEGIN
    PRINT 'Six'
    RETURN
 END
ELSE IF @intNumber = 7
 BEGIN
    PRINT 'Seven'
    RETURN
 END
ELSE IF @intNumber = 8
 BEGIN
    PRINT 'Eight'
    RETURN
 END
ELSE IF @intNumber = 9
 BEGIN
    PRINT 'Nine'
    RETURN
 END
ELSE IF @intNumber = 10
 BEGIN
    PRINT 'Ten'
    RETURN
 END
ELSE
 BEGIN
```

```
    PRINT 'Number is greater than 10.'
    RETURN
END
```

- *Cấu trúc CASE:* Cấu trúc này được dùng để đánh giá một biểu thức và trả về một hoặc một số các kết quả dựa vào giá trị của biểu thức. Có 2 kiểu cấu trúc CASE khác nhau như sau:
    - *Simple CASE:* Với cấu trúc này, một biểu thức sẽ được dùng để so sánh với một tập các giá trị để xác định kết quả. Cú pháp như sau:

```
CASE case_expression
    WHEN expression THEN result
        [...n
    [ELSE else_result
END
```

    - Searched CASE: Đánh giá tập các biểu thức Boolean để xác định kết quả. Cú pháp của nó như sau:

```
CASE
    WHEN Boolean_expression THEN result
        [...n
    [ELSE else_result
END
```

Trong đó:

+ `case_expression`: Biểu thức dùng để SQL Server đánh giá giá trị trong câu lệnh Simple CASE.
+ `Expression`: Giá trị dùng để so sánh với biểu thức `case_expression` nếu đúng thì nó sẽ trả về kết quả.
+ `Result`: Kết quả sẽ được trả về nếu như giá trị biểu thức `case_expression` so với `Expression` là đúng.
+ `Boolean_expression`: SQL Server dùng biểu thức Boolean để rẽ nhánh, nếu biểu thực nhận giá trị True thì sẽ thực hiện kết quả `Result.`
+ `else_result`: Thực hiện các kết quả sau ELSE.

Ví dụ 4.8. Sử dụng cấu trúc rẽ nhánh CASE dùng trong cả hai trường hợp Simple Case và Searched Case.

```
Use pubs
Go
CREATE PROCEDURE uspCheckNumberCase
    @chrNumber    CHAR(2)
AS
IF  (CONVERT(INT,  @chrNumber)  <  1)  OR  (CONVERT(INT,
@chrNumber) > 10)
  BEGIN
     SELECT CASE
        WHEN CONVERT(INT, @chrNumber) < 1 THEN 'Number
is less than 1.'
        WHEN CONVERT(INT, @chrNumber) > 10 THEN 'Number
is greater than 10.'
     END
     RETURN
  END
SELECT CASE CONVERT(INT, @chrNumber)
    WHEN 1 THEN 'One'
    WHEN 2 THEN 'Two'
    WHEN 3 THEN 'Three'
    WHEN 4 THEN 'Four'
    WHEN 5 THEN 'Five'
    WHEN 6 THEN 'Six'
    WHEN 7 THEN 'Seven'
    WHEN 8 THEN 'Eight'
    WHEN 9 THEN 'Nine'
    WHEN 10 THEN 'Ten'
END
```

d) Cấu trúc WAITFOR

Cấu trúc WaitFor được dùng để ngăn việc thực thi một lô, thủ tục, hay một giao dịch cho đến một thời điểm nào đó hoặc sau một khoảng thời gian nào đó. Cú pháp của WAITFOR như sau:

```
WAITFOR { DELAY 'time' | TIME 'time' }
```

Trong đó:

+ `DELAY`: Chỉ định khoảng thời gian phải chờ. Tối đa là 24 giờ.
+ `TIME`: Chỉ định thời điểm thực thi một lô, thủ tục, hay một giao dịch.

Ví dụ 4.9. Sử dụng cấu trúc WAITFOR để chờ đến lúc 21^h30 thì thực hiện xóa bản ghi.

```
BEGIN
   WAITFOR TIME '21:30'
   DELETE FROM DMLOP WHERE MALOP='TH6A'
END
```

Ví dụ 4.10. Xây dựng thủ tục time_delay để chờ trong một khoảng thời gian nào đó và đưa ra thông báo khoảng thời gian đã chờ đó.

```
CREATE PROCEDURE time_delay @DELAYLENGTH char(9)
AS
DECLARE @RETURNINFO varchar(255)
BEGIN
    WAITFOR DELAY @DELAYLENGTH
    SELECT @RETURNINFO = 'A total time of ' +
        SUBSTRING(@DELAYLENGTH, 1, 2) +
        ' hours, ' +
        SUBSTRING(@DELAYLENGTH, 4, 2) +
        ' minutes, and ' +
        SUBSTRING(@DELAYLENGTH, 7, 2) +
        ' seconds ' +
        'has elapsed! Your time is up.';
    PRINT @RETURNINFO;
END;
GO
-- This next statement executes the time_delay procedure.
EXEC time_delay '00:05:00'
GO
```

e) Cấu trúc TRY…CATCH

Trong SQL Server 2005, cấu trúc TRY … CATCH được sử dụng để quản lý lỗi tương tự như các ngôn ngữ lập trình VB.NET, $C^{\#}$ và C^{++}. Cú pháp:

```
BEGIN TRY
      { sql_statement | statement_block }
END TRY
BEGIN CATCH
      { sql_statement | statement_block }
END CATCH[ ; ]
```

Hoạt động của cấu trúc TRY… CATCH:

+ Cấu trúc TRY…CATCH gồm hai phần: Khối TRY và khối CATCH. Khi một điều kiện lỗi được dò thấy ở một câu lệnh Transact-SQL thuộc khối TRY, điều khiển được chuyển sang khối

CATCH để xử lý. Sau khi khối CATCH điều khiển ngoại lệ, điều khiển được chuyển cho câu lệnh Transact-SQL ngay sau lệnh END CATCH.

+ Nếu không lỗi trong khối TRY, điều khiển được chuyển ngay lập tức cho câu lệnh sau END CATCH.

Ví dụ 4.11. Sử dụng cấu trúc TRY … CATCH để điều khiển lỗi.

```
BEGIN TRY
INSERT INTO [QLDiemSV].[dbo].[DMLOP]([MaLop],[TenLop], [Khoa])
     VALUES ('TH6A','Tin học 6A','6')
END TRY
BEGIN CATCH
Print ERROR_MESSAGE()
END CATCH
```

Ví dụ 4.11. Xây dựng thủ tục đưa ra thông tin lỗi.

```
USE QLDiemSV;
GO
BEGIN TRANSACTION;

BEGIN TRY
    -- Generate a constraint violation error.
    DELETE FROM LOP
        WHERE MaLop='TH5A';
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() as ErrorState,
        ERROR_PROCEDURE() as ErrorProcedure,
        ERROR_LINE() as ErrorLine,
        ERROR_MESSAGE() as ErrorMessage;

    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;
END CATCH;

IF @@TRANCOUNT > 0
    COMMIT TRANSACTION;
GO
```

f) Functions - Hàm

Hàm được dùng hoặc là định dạng và thao tác dữ liệu hoặc là trả về thông tin cho người sử dụng. Có hai loại hàm: hàm do hệ thống định nghĩa

hoặc hàm do người dùng định nghĩa. Hàm do hệ thống định nghĩa được tạo do Microsoft và được cài đặt khi SQL Server cài đặt. Hàm do người dùng định nghĩa được định nghĩa bởi người sử dụng bằng cách sử dụng câu lệnh CREATE FUNCTION . Đối với loại hàm này ta sẽ thảo luận chúng trong phần tiếp theo của chương.

Các hàm do hệ thống định nghĩa được chia thành các kiểu hàm sau: String functions, Date functions, Mathematical functions, aggregate Functions, System functions,.v.v...

- ***String functions:*** Là các hàm thao tác với dữ liệu kiểu ký tự. Sau đây là một số hàm thông dụng.

  + `CHARINDEX(string1, string2, start_position)`: Tìm vị trí bắt đầu của chuỗi ký tự chỉ định string1 trong chuỗi string2 và bắt đầu tìm ở vị trí start_position trong chuỗi string2.

    Ví dụ 4.9. Sử dụng hàm `CHARINDEX`

    ```
    SELECT CHARINDEX('test', 'This is a test', 1)
    ```

    Hàm sẽ trả về giá trị 11, vị trí bắt đầu của chuỗi 'test' trong chuỗi 'This is a test'.

  + `LEFT (string, number_of_characters)`: Trả về chuỗi gồm `number_of_characters` ký tự tính từ trái sang của chuỗi `string`.

    Ví dụ 4.10. Sử dụng hàm `LEFT`

    ```
    SELECT LEFT('This is a test', 4)
    ```

    Hàm sẽ trả về chuỗi 'This'

  + `LEN(string)`: Xác định độ dài của chuỗi ký tự *`string`*.

    Ví dụ 4.11. Sử dụng hàm `LEN`

    ```
    SELECT LEN('This is a test')
    ```

    Hàm sẽ trả về giá trị 14

  + `LOWER(string)`: Hàm trả về chuỗi ký tự thường.

Ví dụ 4.12. Sử dụng hàm `LOWER`

```
SELECT LOWER('This is a TEST')
```

Hàm sẽ trả về chuỗi 'this is a test'

+ `LTRIM(string)`: Cắt bỏ các ký tự trắng bên trái của chuỗi.

Ví dụ 4.13. Sử dụng hàm `LTRIM`

```
SELECT LTRIM('    This is a test    ')
```

Hàm sẽ trả về chuỗi 'This is a test   '

+ `RIGHT (string, number_of_characters)`: trả về chuỗi gồm `number_of_characters` ký tự tính từ phải sang của chuỗi `string`.

Ví dụ 4.14. Sử dụng hàm `RIGHT`

```
SELECT RIGHT('This is a test', 4)
```

Hàm sẽ trả về chuỗi 'test'

+ `RTRIM(string)`: Cắt bỏ các ký tự trắng bên phải của chuỗi.

Ví dụ 4.15. Sử dụng hàm `RTRIM`

```
SELECT RTRIM('    This is a test    ')
```

Hàm sẽ trả về chuỗi '   This is a test'

+ `SUBSTRING ( expression ,start , length )`: Hàm trả về chuỗi con gồm `length` ký tự của `expression` tính từ vị trí `start`.

```
SELECT x = SUBSTRING('abcdef', 2, 3)
```

+ `UPPER(string)`: Chuyển đổi các ký thường thành chữ hoa.

Ví dụ 4.16. Sử dụng hàm `UPPER`

```
SELECT UPPER('This is a TEST')
```

Hàm sẽ trả về chuỗi 'THIS IS A TEST'

Chú ý: Cần phải cẩn thận khi sử dụng các hàm, chẳng hạn khi ta sử dụng hàm UPPER trong vế trái của toán tử so sánh. Khi đó nó sẽ bắt SQL Server

phải thực hiện trên một bảng để tìm kiếm giá trị. Ta xét hai truy vấn trong ví dụ 4.17 sau:

```
select au_lname from authors where au_lname = 'Green'
select au_lname from authors where upper(au_lname) =
'Green'
```

Đối với truy vấn thứ hai, sử dụng hàm Upper mất thời gian lâu hơn so với truy vấn thứ nhất.

- ***Date Functions:*** Là các hàm làm việc với dữ liệu kiểu datetime. Một số hàm làm việc với các kiểu thông tin đặc biệt được gọi là `datepart`. Trước khi đi vào các hàm, ta xét các ký hiệu của `datepart` cho trong bảng 4.3.

Bảng 4.3. Các thành phần `datepart`

Ký hiệu `datepart`	`datepart`
yy	Year (năm)
yyyy	Year (năm)
q	Quarter (quý)
qq	Quarter (quý)
m	Month (tháng)
mm	Month (tháng)
dy	Dayofyear
y	Dayofyear
d	Day
dd	Day
wk	Week
ww	Week
dw	weekday
hh	hour
mi	Minute
n	minute
ss	Second

s	Second
ms	millisecond

Sau đây là một số hàm hay sử dụng:

+ `DATEADD (datepart, amount, date)`: Cộng thêm một số `amount` thời gian thành phần `datepart` của `date`.

Ví dụ 4.17. Sử dụng hàm `DATEADD`

```
SELECT DATEADD(year, 1, GETDATE())
```

Hàm sẽ trả về ngày hiện tại cộng thêm một năm.

+ `DATEDIFF (datepart, date1, date2)`: So sánh điểm khác nhau giữa hai ngày bằng việc sử dụng tham số `datepart`.

Ví dụ 4.18. Sử dụng hàm `DATEDIFF`

```
SELECT DATEDIFF(hour, '1/1/2008 12:00:00',
'1/1/2008 16:00:00')
```

Hàm sẽ trả về giá trị 4. Đây là điểm khác nhau giữa hai ngày, hai ngày chênh nhau 4 giờ.

```
SELECT DATEDIFF(hour, '1/1/2008 12:00:00',
'1/2/2008 16:00:00')
```

Hàm sẽ trả về giá trị 28. Đây là điểm khác nhau giữa hai ngày, hai ngày chênh nhau 28 giờ.

+ `DATEPART(datepart, date)`: Hàm trả về giá trị của thành phần `datepart` trong `date`.

Ví dụ 4.19. Sử dụng hàm `DATEPART`

```
SELECT DATEPART(month, '1/1/2008
16:00:00')
```

Hàm sẽ trả về giá trị tháng 1.

+ `DAY(date)`: Xác định số ngày của tháng trong dữ liệu ngày giờ date.

Ví dụ 4.20. Sử dụng hàm `DAY`

```
SELECT DAY ('7/22/1979 00:04:00')
```

Hàm sẽ trả về giá trị ngày là 22.

+ `GETDATE()`: Trả về giá trị ngày hiện tại của hệ thống.
+ `MONTH(date)`: Tương tự như hàm DAY, hàm MONTH trả về tháng của dữ liệu ngày giờ.
+ `YEAR(date)`: Trả về năm của dữ liệu ngày giờ.

- ***Mathematical Functions:*** Sau đây ta trình bày một số hàm toán học thông thường.
  + `ABS(number)`: Trả về giá trị tuyệt đối của số `number`.
  + `CEILING(number)`: Trả về số nguyên nhỏ nhất lớn hơn hoặc bằng `number`.
  + `FLOOR(number)`: Trả về số nguyên lớn nhất nhỏ hơn hoặc bằng `number`.
  + `ROUND(number,precision)`: Hàm làm tròn số `number` lấy `precision` chữ số sau dấu thập phân.
  + `SQUARE(number)`: Hàm trả về giá trị bình phương số `number`.
  + `SQRT(number)`: Hàm trả về giá trị căn bậc hai số `number`.
- ***Aggregate Functions*:** Các hàm tập hợp thực hiện tính toán trên một tập hợp các giá trị và trả về một giá trị đơn. Ngoại trừ hàm COUNT, hàm tập hợp bỏ qua các giá trị NULL.

  Các hàm tập hợp thường sử dụng với mệnh đề GROUP BY trong khối câu lệnh SELECT. Hàm tập hợp được phép dùng như là các biểu thức trong trường hợp:

  - Trong danh sách select của khối câu lệnh SELECT.

- Trong mệnh đề COMPUTE hoặc COMPUTE BY .
- Trong mệnh đề HAVING

Sau đây là một số hàm tập hợp hay được sử dụng:

+ `AVG ([ALL|DISTINCT]expression)`: Hàm trả về giá trị trung bình của tập các giá trị trong một nhóm.
  - `ALL`: Áp dụng cho các hàm tập hợp để chỉ định cho tất cả các giá trị. ALL là từ khóa mặc định.
  - `DISTINCT:` Chỉ định chỉ lấy một thể hiện duy nhất của một giá trị. Nghĩa là trong tập hợp có nhiều phần tử có cùng một giá trị thì chỉ lấy một giá trị đại diện cho nó.

Ví dụ 4.21. Sử dụng hàm `AVG`

```
USE pubs
SELECT AVG(advance), SUM(ytd_sales)
FROM titles
WHERE type = 'business'
```

+ `COUNT({[ALL|DISTINCT]expression]|*})`: Hàm trả về kiểu int số các phần tử của một nhóm.

Chú ý. Sử dụng hàm `COUNT`

- `COUNT(*)`: Trả về số các phần tử trong một nhóm bao gồm cả giá trị NULL và giá trị duplicates.
- `COUNT(ALL` ***`expression`***`)`: Thực hiện định giá trị cho *expression* tại mỗi dòng trong nhóm và trả về số các giá trị không NULL.
- `COUNT(DISTINCT` ***`expression`***`)`: Thực hiện định giá trị cho *expression* tại mỗi dòng trong nhóm và trả về số các giá trị duy nhất và không NULL.

Ví dụ 4.22. Sử dụng hàm `COUNT`

```
USE pubs
```

```
GO
SELECT COUNT(DISTINCT city)
FROM authors
GO
```

+ COUNT_BIG({[ALL|DISTINCT]expression}|*): Trả về số các phần tử trong một nhóm. Hàm COUNT_BIG làm việc như hàm COUNT. Điểm khác nhau giữa chúng là hàm COUNT trả về giá trị kiểu int còn hàm COUNT_BIG trả về giá trị kiểu bigint.

+ MAX ([ALL|DISTINCT]expression): Trả về giá trị lớn nhất trong biểu thức expression.

+ MIN ([ALL|DISTINCT]expression): Trả về giá trị lớn nhất trong biểu thức expression.

+ SUM([ALL|DISTINCT]expression): Trả về tổng của tất cả các giá trị của biểu thức hoặc tổng các giá trị DISTINCT của biểu thức expression. Hàm SUM chỉ áp dụng cho các cột kiểu số. Các giá trị NULL được bỏ qua.

Ví dụ 4.23. Sử dụng hàm SUM

```
USE pubs
GO
-- Aggregate functions
SELECT type, SUM(price), SUM(advance)
FROM titles
WHERE type LIKE '%cook'
GROUP BY type
ORDER BY type
GO
```

Ví dụ 4.24. Sử dụng hàm SUM để tính điểm trung bình trong CSDL QLDiemSV

```
SELECT DIEM.Masv,(Convert(real, Sum(
dbo.fncDiemCN(DiemL1,DiemL2)*MONHOC.SD
```

```
VHT))/convert(real,Sum(MONHOC.SDVHT)))
AS DTB
FROM DIEM INNER JOIN MONHOC ON
DIEM.MAMH=MONHOC.MaMH
GROUP BY DIEM.MaSV
```

- ***System Functions:*** Các hàm hệ thống là các hàm lấy thông tin hệ thống về các đối tượng và đã thiết lập trong SQL Server.

  + `CONVERT (data_type, expression)`: Chuyển đổi biểu thức `expression` thành kiểu dữ liệu `data_type`.

    Ví dụ 4.25. Sử dụng hàm `CONVERT`

    ```
    SELECT CONVERT(VARCHAR(5), 12345)
    ```

    Hàm sẽ trả về chuỗi '12345'.

  + `CAST (expression AS data_type)`: Chuyển đổi biểu thức `expression` thành kiểu dữ liệu `data_type`.

  + `CURRENT_USER`: Trả về người sử dụng hiện tại.

    Ví dụ 4.26. Sử dụng hàm `CURRENT_USER`

    ```
    SELECT CURRENT_USER
    ```

  + `DATALENGTH(expression)`: Trả về số byte được sử dụng trong biểu thức `expression`.

  + `HOST_NAME()`: trả về tên máy tính mà người sử dụng hiện tại đang login.

    Ví dụ 4.27. Sử dụng hàm `HOST_NAME()`

    ```
    SELECT HOST_NAME()
    ```

  + `SYSTEM_USER`: Hàm trả về tên của các User đang login hệ thống.

    Ví dụ 4.28. Sử dụng hàm `SYSTEM_USER`

    ```
    SELECT SYSTEM_USER
    ```

+ `USER_NAME()`: Hàm trả về username khi đưa số user ID.

Ví dụ 4.29. Sử dụng hàm `USER_NAME()`

```
SELECT name FROM sysobjects WHERE
USER_NAME(uid) = 'dbo'
```

4.2. Các store procedure – Các thủ tục

4.2.1. Khái niệm

Store Procedure là một tập các phát biểu T-SQL mà SQL Server biên dịch thành một kế hoạch thực thi đơn. Lần đầu tiên khi SQL Server thực thi store procedure thì nó biên dịch store procedure thành kế hoạch và lưu trong bộ nhớ đệm. Mỗi khi gọi thực hiện store procedure này thì nó sử dụng lại kế hoạch này mà không phải biên dịch lại lần nữa.

T- SQL store procedure tương tự như các ngôn ngữ lập trình khác, chúng chấp nhận các tham số nhập, trả về giá trị xuất thông qua tham số hoặc trả về thông điệp cho biết thủ tục thành công hay thất bại.

Các ứng dụng có thể giao tiếp với SQL Server thông qua hai cách:

+ Chương trình ứng dụng gửi các phát biểu T-SQL từ client đến server. Các phát biểu này được gửi qua mạng và được SQL Server biên dịch lại mỗi khi thực thi chúng.
+ Tạo store procedure, chúng được lưu và biên dịch thành một kế hoạch ở server. Như vậy với cách này, sử dụng store procedure sẽ giảm được lưu thông mạng, hiệu quả nhanh hơn so với cách gửi các phát biểu T-SQL.

4.2.2. Tạo store procedure

Cú pháp:

```
CREATE {PROC|PROCEDURE}[schema_name.]
procedure_name [ ; number ]
    [{@parameter [type_schema_name.] data_type }
        [VARYING][= default ][[ OUT|OUTPUT ]
         [,...n ]
```

```
[ WITH <procedure_option> [ ,...n ]
AS { [ BEGIN ] statements [ END ] }
[;]

<procedure_option> ::=
    [ ENCRYPTION ]
    [ RECOMPILE ]
```

Trong đó:

+ *procedure_name*: là tên của store procedure sẽ được tạo.
+ *parameter:* Là các tham số truyền vào store procedure, ta phải định nghĩa chúng trong phần khai báo của store procedure. Khai báo báo gồm tên của tham số (trước tên tham số sử dụng tiền tố @), kiểu dữ liệu của tham số và một số chỉ định đặc biệt phụ thuộc vào mục đích sử dụng của tham số đó.
+ ; number: Là số nguyên tùy chọn được sử dụng trong nhóm các thủ tục có cùng tên.
+ *data type:* Kiểu của tham số trong phần khai báo.
+ *[VARYING]*: Đây là tùy chọn được chỉ định khi cursor trả về như một tham số.
+ *[= default]* : Gán giá trị mặc định cho tham số. Nếu không gán giá trị mặc định thì tham số nhận giá trị NULL.
+ OUTPUT: Đây là từ khóa chỉ định tham số đó là tham số xuất. Tham số xuất không dùng được với kiểu dữ liệu Text và image.
+ [,...n]: Chỉ định rằng có thể khai báo nhiều tham số.
+ RECOMPILE:Chỉ định Database Engine không xây dựng kế hoạch cho thủ tục này và thủ tục sẽ được biên dịch tại thời điểm thực thi thủ tục.
+ ENCRYPTION:Chỉ định SQL Server sẽ mã hóa bản text lệnh CREATE PROCEDURE. Users không thể truy cập vào các bảng hệ thống hoặc file dữ liệu để truy xuất bản text đã mã hóa.

**** Thực thi store procedure trong SQL Server:*** Để thực thi một thủ tục trong SQL Server ta sử dụng cú pháp sau:

```
{ EXEC | EXECUTE }
    { module_name [ ;number ]}
        [ [ @parameter = ] { value
                           | @variable [ OUTPUT ]
                           | [ DEFAULT ]
                           }
        ]
      [ ,...n ]
      [ WITH RECOMPILE ]
```

Trong đó:

+ module_name: Là tên thủ tục cần thực hiện.
+ ;number: Chỉ định thủ tục trong nhóm thủ tục cùng tên.
+ @parameter: Tên tham số trong thủ tục.
+ @variable: Chỉ định biến chứa các tham số hoặc trả về tham số.
+ DEFAULT: Chỉ định lấy giá trị mặc định của biến.

Ví dụ 4.30. Xây dựng thủ tục XemDSSV.

```
Use QLDiemSV
Go
IF EXISTS(Select name from sysobjects
          where name ='p_DSSV' and type='p')
DROP PROCEDURE p_DSSV
GO
CREATE PROCEDURE p_DSSV
AS
SELECT MaSV, Hodem + ' '+TensV as Hoten, Ngaysinh, MaLop
      From HOSOSV
GO
```

- Thực thi thủ tục p_DSSV

```
Use QLDiemSV
Go
EXEC p_DSSV
```

**** Truyền tham số nhập vào trong store procedure.***

Ví dụ 4.32. Xây dựng thủ tục pp_DSSV để hiển thị danh sách sinh viên theo tham số mã lớp. Mã lớp được truyền vào khi thủ tục được thực hiện.

```
Use QLDiemSV
Go
IF EXISTS(Select name from sysobjects
where name ='p_DSSV' and type='p')
DROP PROCEDURE p_DSSV
GO
CREATE PROCEDURE p_DSSV
@parMaLop Varchar(10)='TH%'
AS
 SELECT MaSV, Hodem + ' '+TensV as Hoten, Ngaysinh
From HOSOSV Where MaLop like @parMaLop
GO
```

Gọi thực thi thủ tục trên với truyền giá trị cho tham số nhập như sau:

```
EXEC p_DSSV 'TH03A'
EXEC p_DSSV @parMaLop=DEFAULT
```

**** Sử dụng tham số xuất trong store procedure.***

Ví dụ 4.33. Xây dựng thủ tục pp_Siso để xuất giá trị sĩ số của một lớp theo tham số mã lớp. Mã lớp được truyền vào khi thủ tục được thực hiện.

```
Use QLDiemSV
Go
IF EXISTS(Select name from sysobjects where name
='pp_Siso' and type='p')
DROP PROCEDURE pp_Siso
GO
CREATE PROCEDURE pp_Siso
@parMaLop Char(10), @parSiso Int OUTPUT
AS
 SELECT @parSiso=count(*)
    From HOSOSV Where MaLop=@parMaLop
GO
DECLARE @siso int
```

```
exec pp_Siso 'TH03A',@parSiso=@siso OUTPUT
Print 'Si so lop TH03A là :'+ convert(varchar(3),@siso)
Go
```

Kết quả thực hiện chương trình:

```
Si so lop TH03A là :12
```

**** Sử dụng biến cục bộ:*** Các biến cục bộ được sử dụng trong bó lệnh, trong chương trình gọi (Scipt) hoặc trong thủ tục (xem ví dụ 4.5 và 4.6). Biến cục bộ thường được giữ các giá trị sẽ được kiểm tra trong phát biểu điều kiện và giữ giá trị sẽ được trả về bởi lệnh RETURN. Phạm vị của biến cục bộ trong store procedure là từ điểm biến đó được khai báo cho đến khi thoát store procedure. Ngay khi store procedure kết thúc thì biến đó không được tham chiếu nữa. Cú pháp khai báo biến cục bộ:

```
DECLARE <parameter> [AS] <data type>
```

Giống như khai báo các biến ở trên, trước tên biến phải có tiền tố @. Giá trị khởi tạo ban đầu của biến là NULL.

Để thiết lập giá trị của biến ta sử dụng cú pháp:

```
SET <parameter> = <expression>
SELECT <parameter> = <expression>
```

**** Câu lệnh PRINT:*** Dùng để hiển thị chuỗi thông báo tới người sử dụng. Chuỗi thông báo này nó thể dài tới 8000 ký tự. Cú pháp của lệnh PRINT như sau:

```
PRINT < messages>
```

**** Sử dụng SELECT đề trả về giá trị:*** Ta có thể trả về giá trị bằng việc sử dụng SELECT trong thủ tục hoặc trả về kết quả thiết lập từ truy vấn SELECT.

Ví dụ 4.34. Xây dựng thủ tục pp_Siso để xuất giá trị sĩ số của một lớp theo tham số mã lớp ra ngoài. Mã lớp được truyền vào khi thủ tục được thực hiện.

```
Use QLDiemSV
Go
```

```
    IF EXISTS(Select name from sysobjects where name
='pp_Siso' and type='p')
    DROP PROCEDURE pp_Siso
    GO
    CREATE PROCEDURE pp_Siso
    @parMaLop Char(10), @parSiso Int OUTPUT
    AS
     SELECT     @parSiso=count(*)     From     HOSOSV     Where
MaLop=@parMaLop
    GO
    DECLARE @siso int
    exec pp_Siso 'TH03A',@parSiso=@siso OUTPUT
    SELECT 'Si so lop TH03A là :'= @siso
    Go
```

**** Lệnh RETURN:*** Ta có thể sử dụng lệnh RETURN để thoát không điều kiện khỏi thủ tục. Khi lệnh RETURN được thực thi trong thủ tục, khi đó các câu lệnh sau RETURN trong thủ tục sẽ bị bỏ qua và thoát khỏi thủ tục để trở về dòng lệnh tiếp theo trong chương trình gọi.

Ngoài ra, ta có thể sử dụng lệnh RETURN để trả về giá trị cho chương trình gọi, giá trị trả về phải là một số nguyên, nó có thể là một hằng số hoặc một biến. Cú pháp như sau:

```
RETURN [ integer_expression ]
```

Ví dụ 4.35. Cho CSDL pubs. Xây dựng thủ tục `usp_4_31` kiểm tra một chủ đề có tồn tại trong bảng `titles` hay không? Nếu tồn tại một chủ đề thì hiển thị chủ đề đó. Nếu không tồn tại chủ đề đó thì thủ tục trả về giá trị 1 hoặc có nhiều hơn một chủ đề đó thì trả về giá trị 2.

```
Use pubs
Go
IF EXISTS(Select name from sysobjects
          where name ='usp_4_31' and type='p')
DROP PROCEDURE usp_4_31
GO
CREATE PROCEDURE usp_4_31
    @vchTitlePattern    VARCHAR(80) = '%'
AS
SELECT @vchTitlePattern = '%' + @vchTitlePattern + '%'
IF (SELECT COUNT(*) FROM titles
                   WHERE title LIKE @vchTitlePattern) < 1
BEGIN
    RETURN 1
END
IF (SELECT COUNT(*) FROM titles
                   WHERE title LIKE @vchTitlePattern) > 1
```

```
BEGIN
    RETURN 2
END
SELECT title, price FROM titles
      WHERE title LIKE @vchTitlePattern
RETURN 0
GO
DECLARE @intReturnValue    INT
EXEC @intReturnValue = usp_4_31 'Tin hoc'
IF (@intReturnValue = 1)
BEGIN
    PRINT 'There are no corresponding titles.'
END
IF (@intReturnValue = 2)
BEGIN
    PRINT 'There are multiple titles that match this
                    criteria.   Please   narrow   your
                    search.'
END
GO
```

4.2.3.Thay đổi, xóa, xem nội dung store procedure

a) Thay đổi store procedure

Cú pháp:

```
ALTER {PROC|PROCEDURE}[schema_name.]
procedure_name [ ; number ]
    [{@parameter [type_schema_name.] data_type }
        [VARYING][= default ][[ OUT|OUTPUT ]
         [,...n ]
[ WITH <procedure_option> [ ,...n ]
AS { [ BEGIN ] statements [ END ] }
[;]

<procedure_option> ::=
    [ ENCRYPTION ]
    [ RECOMPILE ]
```

Ví dụ 4.36. Ta sửa lại thủ tục p_DSSV trong ví dụ 4.32 như sau:

```
Use QLDiemSV
Go
ALTER PROCEDURE p_DSSV
@parMaLop Char(10)
AS
 SELECT MaSV, Hodem, TensV, Ngaysinh
```

```
        From HOSOSV Where MaLop=@parMaLop
GO
```

b) Xóa store procedure

Cú pháp:

```
DROP{ PROC|PROCEDURE}{[schema_name.] procedure }
```

Ví dụ 4.37. Xóa thủ tục p_DSSV:

```
Use QLDiemSV
Go
DROP PROCEDURE p_DSSV
Go
```

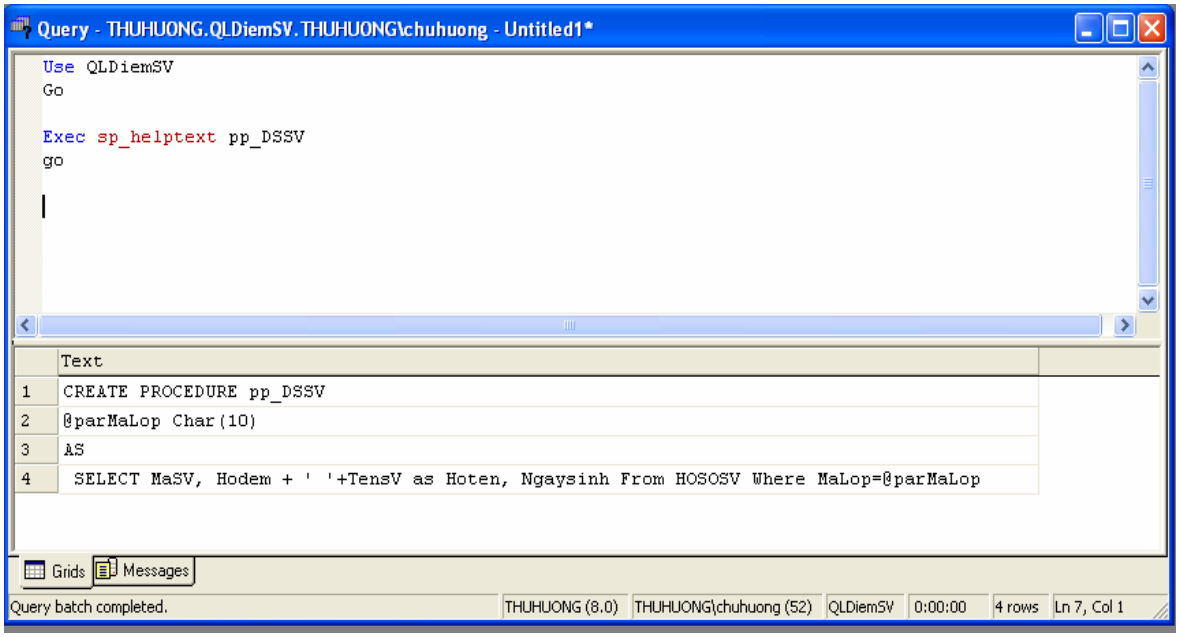
c) Xem nội dung store procedure

Để xem nội dung của thủ tục ta sử dụng thủ tục hệ thống sp_helptext.

Ví dụ 4.38. Xem nội dung thủ tục pp_DSSV:

```
Use QLDiemSV
Go
Exec sp_helptext pp_DSSV
Go
```

Kết quả việc thi hành các câu lệnh này cho trong hình 4.8.



Hình 4.8. Sử dụng thủ tục sp_helptext

4.3. Các store function – Các hàm

4.3.1. Các khái niệm

Tất cả các ngôn ngữ lập trình, bao gồm cả T-SQL, việc có các hàm tạo cho các ứng dụng trở lên mạnh mẽ. Ngoài ra, người lập trình có thể tự tạo một hàm riêng cho mình làm cho hệ thống dễ được mở rộng.

Một hàm - function - trong SQL Server được định nghĩa là một thủ tục đơn giản bao gồm một nhóm các câu lệnh SQL

4.3.2. Tạo các hàm

Một hàm do người dùng định nghĩa được tạo bằng cách sử dụng câu lệnh CREATE FUNCTION theo cú pháp sau:

```
CREATE FUNCTION [owner_name.]function_name
(
[ {@parameter_name scalar_data_type [= default]}
[,...n] ]
)
RETURNS scalar_data_type |
TABLE(column_definition | table_constraint
[,...n])

[WITH ENCRYPTION | SCHEMABINDING [,...n] ]
[AS]
[BEGIN function_body END] | RETURN [(]
select_statement [)]
```

Trong đó:

+ [owner_name.]: Chỉ định tên đối tượng sẽ sở hữu. Ta không phải bắt buộc chỉ định tên người sẽ tạo đối tượng sở hữu nó.
+ function_name: tên của hàm ta sẽ tạo.

+ `parameter_name`: Là các tham số Input cho hàm. Các tham số này xây dựng cũng tương tự như trong stored procedure.
+ `scalar_data_type`: Là kiểu dữ liệu vô hướng của tham số. Một hàm có thể nhận bất kỳ kiểu dữ liệu nào như là tham số trừ các kiểu timestamp, cursor, text, ntext, image.
+ `default`: Chỉ định giá trị mặc định cho tham số, tương tự như trong stored procedure.
+ `[,...n]`: Chỉ định một hàm có thể tạo nhiều tham số. Một hàm trong SQL Server có thể chứa tới 1024 tham số.
+ `RETURNS`: từ khóa này chỉ định kiểu dữ liệu hàm sẽ trả về. Kiểu dữ liệu của hàm có thể là một kiểu dữ liệu vô hướng hoặc một bảng.
+ `scalar_data_type`: Ta sẽ chỉ định kiểu dữ liệu nếu như hàm trả về một giá trị vô hướng. Ở đây ta phải chỉ định kiểu độ dài dữ liệu.
+ `TABLE`: Đây là kiểu dữ liệu cho phép hàm có thể trả về nhiều dòng dữ liệu.
+ `column_definition`: Định nghĩa các cột cho kiểu dữ liệu TABLE. Các cột này được định nghĩa tương tự như định nghĩa các cột trong bảng.
+ `table_constraint`: Định nghĩa các ràng buộc trong kiểu dữ liệu TABLE này.
+ `[,...n]`: Chỉ định có thể có nhiều cột và nhiều ràng buộc trong bảng.
+ `WITH ENCRYPTION`: Từ khóa chỉ định code của hàm sẽ được mã hóa trong bảng `syscomments`.
+ `SCHEMABINDING`: Từ khóa này chỉ định hàm được tạo để buộc vào tất cả các đối tượng mà nó tham chiếu.

+ `[,...n]`: Chỉ định có thể có nhiều từ khóa khác ngoài hai từ khóa trên.
+ `AS`: Từ khóa cho biết code của hàm bắt đầu.
+ `BEGIN`: Đi cùng với END để tạo thành bao khối bao các câu lệnh trong thân hàm.
+ `function_body`: thân của hàm.
+ `END`: Đi cùng với BEGIN để tạo thành bao khối bao các câu lệnh trong thân hàm.
+ `RETURN`: Từ khóa này sẽ gửi giá trị tới thủ tục gọi hàm.
+ `select_statement:` đi kèm với RETURN để gửi giá trị tới thủ tục gọi hàm.

4.3.3. Các ví dụ tạo các hàm

Ví dụ 4.38. Xây Dựng một hàm `fncGetThreeBusinessDays` trả về ngày làm việc thứ 3 tính từ ngày bắt đầu `@dtmDateStart`.

```
CREATE FUNCTION fncGetThreeBusinessDays
      (@dtmDateStart DATETIME)
    RETURNS DATETIME
AS
BEGIN
    IF DATEPART(dw, @dtmDateStart) = 4
    BEGIN
        RETURN(DATEADD(dw, 5, @dtmDateStart))
    END
    ELSE IF DATEPART(dw, @dtmDateStart) = 5
    BEGIN
        RETURN(DATEADD(dw, 5, @dtmDateStart))
    END
    ELSE IF DATEPART(dw, @dtmDateStart) = 6
    BEGIN
        RETURN(DATEADD(dw, 5, @dtmDateStart))
    END
```

```
    ELSE IF DATEPART(dw, @dtmDateStart) = 7
    BEGIN
        RETURN(DATEADD(dw, 4, @dtmDateStart))
    END

    RETURN(DATEADD(dw, 3, @dtmDateStart))
END
```

Thực hiện thử nghiệm hàm trên, ta xây dựng scipt sau:

```
DECLARE @dtmDate    DATETIME
SELECT      @dtmDate = '1/10/2008'
SELECT      DATENAME(dw, @dtmDate)
SELECT      DATENAME(dw, dbo.
                        fncGetThreeBusinessDays(@dtmDate))
```

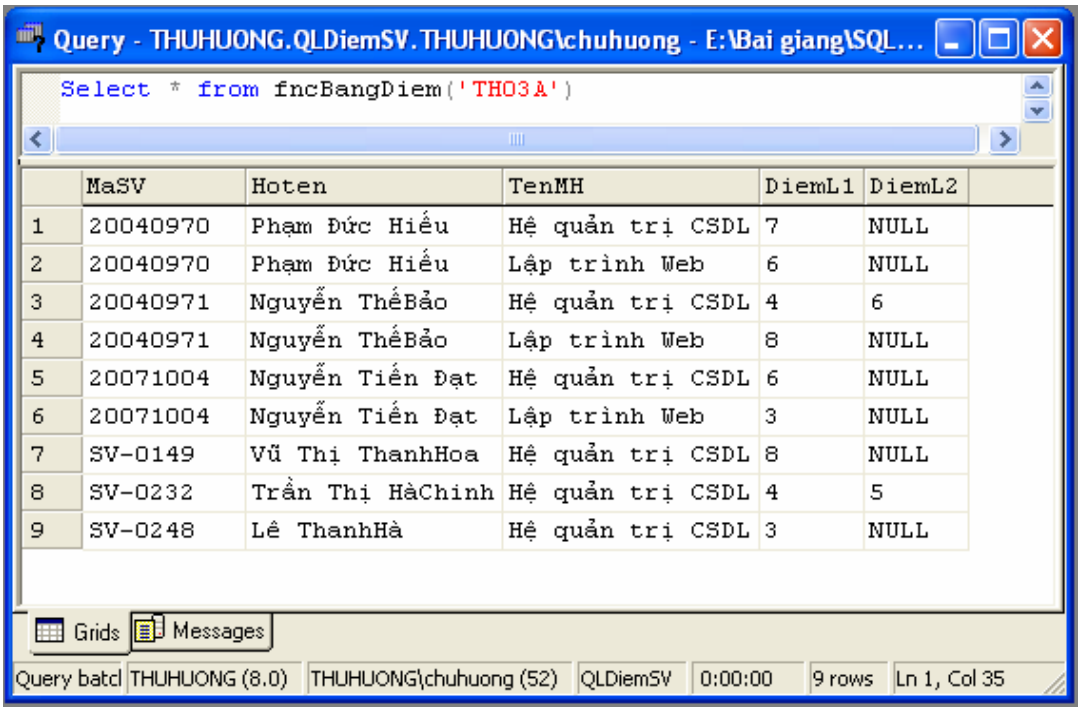
Ví dụ 4.39. Sử dụng hàm `fncGetThreeBusinessDays` trên để tính toán trên một cột trong một bảng.

```
CREATE TABLE OrderInfo
(
    OrderID         INT NOT NULL,
    ShippingMethod VARCHAR(16) NOT NULL,
    OrderDate       DATETIME NOT NULL DEFAULT GETDATE(),
    ExpectedDate    AS(
            dbo.fncGetThreeBusinessDays(OrderDate)
        )
)
GO

INSERT OrderInfo VALUES (1, 'UPS GROUND', GETDATE())
INSERT OrderInfo VALUES (2, 'FEDEX STANDARD',
                            DATEADD(dd, 2, GETDATE()))
INSERT OrderInfo VALUES (3, 'PRIORITY MAIL',
                            DATEADD(dd, 4, GETDATE()))
GO

SELECT OrderID,ShippingMethod,CONVERT(VARCHAR(12),
       OrderDate,1) + '(' + DATENAME(dw, OrderDate) + ')'
 AS 'OrderDate', CONVERT(VARCHAR(12),ExpectedDate,1)+
 '('+ DATENAME(dw, ExpectedDate) + ')' AS 'ExpectedDate'
FROM     OrderInfo
```

Ví dụ 3.40. Xây dựng hàm trả về các dòng dữ liệu gồm thông tin về điểm của các môn học theo Mã lớp.

```
CREATE FUNCTION fncBangDiem(@MaLop CHAR(10))
    RETURNS @TableName TABLE
    (   MaSV    CHAR(10),
        Hoten   nvarchar(100),
        TenMH    NVARCHAR(50),
        DiemL1   INT,
        DiemL2   INT    )
AS
BEGIN
INSERT INTO @TableName
    SELECT Di.MaSV, Ho.HoDem + ' '+ Ho.TenSV, Mo.TenMH,
                                    Di.DiemL1,Di.DiemL2
    FROM   DIEM Di
        JOIN HOSOSV Ho ON (Di.MaSV = Ho.MaSV)
        JOIN MONHOC Mo ON (Di.MaMH = Mo.MaMH)
    WHERE   Ho.MaLop = @MaLop
RETURN
END
GO
```

Thử nghiệm gọi hàm này trong đoạn script chương trình sau và kết quả cho trong hình 4.10: Select * from fncBangDiem('TH03A')



Hình 4.10. Gọi hàm `fncBangDiem`

Ví dụ 4.41. Xây dựng hàm đưa ra danh sách sinh viên của một lớp phụ thuộc tham số mã lớp đưa vào.

```
CREATE FUNCTION fncDSSV(@MaLop CHAR(10))
    RETURNS TABLE
   AS
RETURN
  SELECT MaSV, HoDem + ' '+ TenSV As Hoten, NgaySinh
    FROM HOSOSV
    WHERE  MaLop=@MaLop
GO
```

Gọi hàm này trong đoạn script chương trình sau:

```
Select * from fncDSSV ('TH03A')
```

4.3.4.Thay đổi, xóa, xem nội dung store function

a) Thay đổi các hàm

Để thay đổi các hàm ta dùng câu lệnh ALTER FUNCTION.

```
ALTER FUNCTION [owner_name.]function_name
(
[ {@parameter_name scalar_data_type [= default]}
[,...n] ]
)
RETURNS scalar_data_type |
TABLE(column_definition | table_constraint
[,...n])

[WITH ENCRYPTION | SCHEMABINDING [,...n] ]
[AS]
[BEGIN function_body END] | RETURN [(]
select_statement [)]
```

Ví dụ 4.42. Thay đổi hàm `fncDSSV`

```
ALTER FUNCTION fncDSSV(@MaLop CHAR(10))
   RETURNS TABLE
   AS
RETURN
```

```
    SELECT MaSV, HoDem + ' '+ TenSV As Hoten
      FROM HOSOSV
      WHERE  MaLop=@MaLop
GO
```

b) Xóa hàm

Để xóa hàm ta dùng câu lệnh DROP FUNCTION.

```
DROP FUNCTION { [ schema_name. ] function_name }
```

Ví dụ 4.43. Xóa hàm `fncDSSV`

```
DROP FUNCTION fncDSSV
```

4.4. Trigger

4.4.1. Khái niệm

Trigger là một kiểu stored procedure đặc biệt được kích nổ (thực thi) một cách tự động khi xảy ra một sự kiện trên Database server và không thể thực thi bằng tay. Trigger ược chia ra làm hai nhóm: DML và DDL trigger.

+ DML triggers (hay Standart triggers) thực thi khi một người sử dụng cố gắng sửa đổi dữ liệu thông qua sự kiện thao tác dữ liệu (data manipulation language - DML) INSERT, UPDATE, hoặc DELETE trên bảng hoặc view. DML triggers thường được sử dụng trong chính sách đảm bảo các quy tắc thương mại hoặc đảm bảo tính toàn vẹn dữ liệu.

+ DDL triggers được thực thi đáp ứng các sự kiện định nghĩa lược đồ dữ liệu (data definition language - DDL). Nhóm lệnh chính của các lệnh định nghĩa lược đồ dữ liệu là CREATE, ALTER, và DROP. Nhóm DDL trigger là nhóm trigger mới được bổ xung trong SQL Server 2005 Database Engine.

DML triggers có hai kiểu, chúng được xử lý khác nhau. Kiểu phổ biến với mọi người nhất đó là kiểu AFTER trigger và kiểu thứ hai đó là INSTEAD OF trigger.

- **AFTER Triggers:** Khi các câu lệnh thay đổi dữ liệu được thực hiện trên một bảng có định nghĩa trigger, một vài xử lý xuất hiện trước khi trigger thực sự nổ. Đầu tiên bộ truy vấn sẽ kiểm tra trên bảng có bất cứ các ràng buộc nào hay không. Nếu có, SQL Server sẽ tiến hành kiểm tra tính hợp lệ của dữ liệu trên mọi ràng buộc nào. Nếu dữ liệu đang thêm vào hoặc đang sửa đổi mà không thỏa các ràng buộc thì bộ truy vấn sẽ dừng câu lệnh trên và khi đó trigger sẽ không được kích nổ. Ngược lại, khi kiểm tra thành công thì các trigger sẽ được thực thi.

  Trước khi bất cứ câu lệnh nào chứa trigger thực sự được thực hiện, SQL Server tạo ra hai bảng đặc biệt lưu trong bộ nhớ có cùng cấu trúc với bảng mà trên đó trigger được tạo.

  + Table INSERTED chứa các giá trị đang được Add vào bảng.
  + Table DELETED chứa các giá trị đang bị xóa từ bảng.

  Nếu trigger được định nghĩa là INSERT trigger thì SQL Server sẽ chỉ tạo bảng INSERTED, còn nếu là DELETE trigger thì thì SQL Server sẽ chỉ tạo bảng DELETED. Cuối cùng là nếu trigger được định nghĩa là UPDATE trigger thì SQL Server sẽ tạo cả hai bảng vì INSERTED sẽ chứa ảnh của hàng sau khi thay đổi còn bảng DELETED chứa ảnh của hàng trước khi thay đổi.

- ***INSTEAD OF trigger:*** Là một đặc điểm thêm vào của SQL Server 2000. Như tên gọi đã ám chỉ, INSTEAD OF trigger được kích nổ thay cho hành động được sử dụng để kích nó.

  Nghĩa là, nếu một trigger được định nghĩa là INSTEAD OF INSERT trigger thì trigger này sẽ được nổ khi câu lệnh Insert được thực hiện trên bảng. Sau khi câu lệnh sửa đổi dữ liệu được gửi đi thì INSTEAD OF trigger được kích nổ và hiện lập tức ngay lập tức. Các ràng buộc không được kiểm tra trước khi trigger được kích nổ, mặc dù các bảng INSERTED, DELETED vẫn được tạo. Sau khi các bảng này được tạo thì quá trình xử lý trigger tương tự như quá trình xử lý của stored procedure. INSTEAD OF trigger có thể được tạo trên bảng hoặc trên view.

4.4.2. Tạo trigger

a) Dùng T-SQL để tạo trigger

Ta dùng T-SQL để tạo trigger theo cú pháp sau:

- Tạo DML Trigger:

```
CREATE TRIGGER trigger_name
ON {table | view }
[WITH ENCRYPTION]
{
{{FOR | AFTER | INSTEAD OF} { [DELETE] [,]
[INSERT] [,] [UPDATE] }
    [NOT FOR REPLICATION]
AS
    [ { IF UPDATE ( column )
            [ { AND | OR } UPDATE ( column ) ]
                [ ...n ]
        | IF ( COLUMNS_UPDATED ( ) {
bitwise_operator } updated_bitmask )
                { comparison_operator }
column_bitmask [ ...n ]
        } ]
        sql_statement [ ...n ]
}
```

Trong đó:

+ ***trigger_name:*** tham số chỉ định tên của trigger sẽ tạo.
+ ***ON***: Chỉ định lấy tên của bảng hoặc view mà ta sẽ xây dựng trigger trên đó.
+ ***table | view***: *Tên của bảng hoặc của view.*
+ ***WITH ENCRYPTION***: Dùng để chỉ định code của trigger mã hóa lưu trữ trên bảng ***syscomments*** mà người khác không thể xem code của trigger đó.
+ ***FOR:*** Định nghĩa hành động mà trigger được kích nổ và kiểu của trigger ta đang tạo. AFTER trigger sẽ là trigger mặc định nếu chỉ có chỉ định FOR.

+ *AFTER*: Chỉ định AFTER trigger. Tùy chọn này chỉ định riêng không lẫn với từ khóa INSTEAD OF. AFTER triggers không được định nghĩa trên view.
+ *INSTEAD OF*: Từ khóa chỉ định đây là INSTEAD OF trigger. Tùy chọn này chỉ định để không lẫn với từ khóa AFTER.
+ *DELETE*: Chỉ định rằng trigger ta đang tạo sẽ được kích nổ đáp ứng với hành động DELETE của bảng hoặc view.
+ *INSERT*: Chỉ định rằng trigger ta đang tạo sẽ được kích nổ đáp ứng với hành động INSERT của bảng hoặc view.
+ *UPDATE*: Chỉ định rằng trigger ta đang tạo sẽ được kích nổ đáp ứng với hành động UPDATE của bảng hoặc view.
+ *NOT FOR REPLICATION*: Chỉ định rằng bất cứ bản sao nào của các hành động chạy ngầm dưới bảng này đều không được kích nổ trigger này.
+ *AS*: Chỉ định phần code của trigger bắt đầu từ đây.
+ *IF UPDATE (column)*: Được dùng trong các trigger INSERT, UPDATE. Cấu trúc này được dùng để kiểm tra các sửa đổi trên cột chỉ định và sau đó là các hành động trên nó.
+ *{AND | OR} UPDATE (column)*: Chỉ định rằng ta có thể sử dụng chuỗi các cấu trúc UPDATE với nhau để kiểm tra một vài cột tại cùng một thời điểm.
+ ...n: Chỉ định ta có thể lặp lại các cấu trúc trên nếu cần thiết.
+ IF(COLUMNS_UPDATED():Chỉ dùng trong các trigger INSERT, UPDATE. Hàm trả về một bit chỉ định cột bị chỉnh sửa trong quá trình INSERT, UPDATE trên bảng cơ sở.
+ *bitwise_operator:* Được dùng để so sánh với bit trả về của hàm COLUMNS_UPDATED()

+ *updated_bitmask:* Được sử dụng để kiểm cột nào thực sự được Update trong câu lệnh Insert hoặc Update.
+ *sql_statement:* Các câu lệnh T-SQL
+ *... n:* Chỉ định lặp lại các câu lệnh T-SQL.

- Tạo DDL Trigger:

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH ENCRYPTION ]
{ FOR |AFTER } {event_type } [,...n ]
AS { sql_statement  [ ; ] [ ...n ]}
```

Trong đó:

+ DATABASE: Chỉ định phạm vi của DDL trigger là database hiện thời. Trigger sẽ kích nổ khi sự kiện event_type xảy ra trên database hiện thời.
+ ALL SERVER: Chỉ định phạm vi của DDL trigger là server hiện thời. Trigger sẽ kích nổ khi sự kiện event_type xảy ra trên server hiện thời.
+ event_type: Là tên của sự kiện mà là nguyên nhân kích nổ DDL trigger. Các sự kiện này có thể là: CREATE_FUNCTION, CREATE_INDEX, GRANT_DATABASE, CREATE_TABLE, ALTER_VIEW, ALTER_TABLE, DROP_TABLE, DROP_VIEW, .v.v...

Ví dụ 4.44. Tạo một AFTER INSERT Trigger. Trong ví dụ này ta định nghĩa AFTER INSERT Trigger trên bảng TriggerTableChild. Trigger này có nhiệm vụ kiểm tra xem có sự tương ứng với các dòng của bảng TriggerTableParent hay không. Nếu không có sự tương ứng nó sẽ thực hiện roll back (cuốn lại) lại giao dịch đó và dòng thông báo lỗi hiện lên. Nếu có sự tương ứng thì giao dịch được thực hiện một cách bình thường.

```
CREATE TABLE TriggerTableParent
(
    TriggerID    INT,
```

```
    TriggerText     VARCHAR(32)
)
GO

INSERT INTO TriggerTableParent VALUES (1, 'Trigger Text 1')
INSERT INTO TriggerTableParent VALUES (2, 'Trigger Text 2')
INSERT INTO TriggerTableParent VALUES (3, 'Trigger Text 3')
INSERT INTO TriggerTableParent VALUES (4, 'Trigger Text 4')
INSERT INTO TriggerTableParent VALUES (5, 'Trigger Text 5')
GO

CREATE TABLE TriggerTableChild
(
    TriggerID     INT,
    TriggerSubText     VARCHAR(32)
)
GO

CREATE TRIGGER trTriggerTableChildInsert
ON TriggerTableChild
FOR INSERT
AS
IF (SELECT COUNT(*)FROM TriggerTableParent TTP
    INNER JOIN INSERTED I ON (TTP.TriggerID= I.TriggerID))= 0
BEGIN
    ROLLBACK TRANSACTION
    RAISERROR ('No corresponding record was found in the
           TriggerTableParent table for this insert.', 11, 1)
END
ELSE
    Print 'This vas inserted'
GO
SET NOCOUNT ON
INSERT INTO TriggerTableChild VALUES(1, 'Sub Trigger Text 1')
INSERT INTO TriggerTableChild VALUES(2, 'Sub Trigger Text 2')
INSERT INTO TriggerTableChild VALUES(3, 'Sub Trigger Text 3')
INSERT INTO TriggerTableChild VALUES(6, 'Sub Trigger Text 6')
GO
```

Ví dụ 4.45. Dùng bảng INSERTED và DELETED

```
CREATE TRIGGER trTriggerTableParentUpdate1
ON TriggerTableParent1
AFTER UPDATE
AS
SET NOCOUNT ON

PRINT     'Contents of the INSERTED Table:'
SELECT     *
FROM     INSERTED
PRINT     'Contents of the DELETED Table:'
SELECT     *
FROM     DELETED
PRINT     'Contents of the TriggerTableParent Table:'
SELECT     TTP.* FROM     TriggerTableParent1 TTP
    INNER JOIN INSERTED I ON
    (TTP.TriggerID = I.TriggerID)

ROLLBACK TRANSACTION
GO
UPDATE TriggerTableParent1
SET TriggerText = 'Changed Trigger Text 1'
WHERE TriggerID = 1
```

Khi sử dụng lệnh UPDATE, kết quả cho thấy bảng INSERTED chứa các giá trị mới; bảng DELETED chứa các giá trị cũ và bảng TriggerTableParent1 chứa các giá trị mới.

Ví dụ 4.46. Sử dụng cấu trúc IF UPDATED trong UPDATE Trigger

```
CREATE TRIGGER trTriggerTableChildUpdate
ON TriggerTableChild1
AFTER UPDATE
AS
IF UPDATE(TriggerID)
BEGIN
    IF (SELECT COUNT(*)FROM     TriggerTableParent1 TTP
            INNER JOIN INSERTED I ON
            (TTP.TriggerID = I.TriggerID)) = 0
```

```
        BEGIN
            RAISERROR ('No parent record exists for this
                    modification. Transaction cancelled.', 11,
                1)
            ROLLBACK TRANSACTION
            RETURN
        END
    END
    GO

    UPDATE      TriggerTableChild1
    SET     TriggerID = 7
    WHERE       TriggerID = 1
    GO
```

Ví dụ 4.47. Tạo INSTEAD OF Trigger. Bởi vì INSTEAD OF Trigger được kích nổ trước khi bất cứ dữ liệu nào được sửa đổi trong CSDL, do đó vai trò của hai bảng INSERTED và DELETED bị giảm nhẹ hơn. Ta xét ví dụ sau tương tự như ví dụ 4.45.

```
CREATE TABLE TriggerTableParent2
(
    TriggerID     INT,
    TriggerText     VARCHAR(32)
)
GO
INSERT INTO TriggerTableParent2 VALUES(1,'Trigger Text 1')
INSERT INTO TriggerTableParent2 VALUES(2,'Trigger Text 2')
INSERT INTO TriggerTableParent2 VALUES(3,'Trigger Text 3')
INSERT INTO TriggerTableParent2 VALUES(4,'Trigger Text 4')
INSERT INTO TriggerTableParent2 VALUES(5,'Trigger Text 5')
GO
CREATE TABLE TriggerTableChild2
(
    TriggerID     INT,
    TriggerSubText     VARCHAR(32)
)
GO
CREATE TRIGGER trTriggerTableParent2InsteadOfUpdate
ON TriggerTableParent2
```

```
INSTEAD OF UPDATE
AS
SET NOCOUNT ON

PRINT     'Contents of the INSERTED Table:'
SELECT     *
FROM     INSERTED
PRINT     'Contents of the DELETED Table:'
SELECT     *
FROM     DELETED
PRINT     'Contents of the TriggerTableParent Table:'
SELECT     TTP.*
FROM     TriggerTableParent2 TTP
    INNER JOIN INSERTED I ON
    (TTP.TriggerID = I.TriggerID)

ROLLBACK TRANSACTION
GO

UPDATE     TriggerTableParent2
SET     TriggerText = 'Changed Trigger Text 1'
WHERE     TriggerID = 1
GO
```

Khi thực hiện lệnh Update, ta thấy bảng cơ sở chưa được chèn dữ liệu do trigger đã được thực thi trước khi có sự sửa đổi dữ liệu, đây chính là đặc điểm của INSTEAD OF Trigger. Cụ thể khi sử dụng lệnh UPDATE, kết quả cho thấy bảng INSERTED chứa các giá trị mới; bảng DELETED chứa các giá trị cũ và bảng TriggerTableParent2 chứa các giá trị cũ.

Ví dụ 4.48. View của các bảng và INSTEAD OF Trigger.

Một trong những đặc điểm nổi bật chính của INSTEAD OF Trigger là cho phép người sử dụng thực hiện các câu lệnh thay đổi dữ liệu trên view của nhiều bảng. Trong ví dụ này, ta xây dựng một INSTEAD OF Trigger thực hiện chức năng này.

```
SET NOCOUNT ON
GO
CREATE TABLE ViewTable1
```

```
(
    KeyColumn    INT,
    Table1Column    VARCHAR(32)
)
GO
INSERT INTO ViewTable1 VALUES (1, 'ViewTable1 Value 1')
INSERT INTO ViewTable1 VALUES (2, 'ViewTable1 Value 2')
INSERT INTO ViewTable1 VALUES (3, 'ViewTable1 Value 3')
INSERT INTO ViewTable1 VALUES (4, 'ViewTable1 Value 4')
INSERT INTO ViewTable1 VALUES (5, 'ViewTable1 Value 5')
INSERT INTO ViewTable1 VALUES (6, 'ViewTable1 Value 6')
INSERT INTO ViewTable1 VALUES (7, 'ViewTable1 Value 7')
INSERT INTO ViewTable1 VALUES (8, 'ViewTable1 Value 8')
INSERT INTO ViewTable1 VALUES (9, 'ViewTable1 Value 9')
INSERT INTO ViewTable1 VALUES (10, 'ViewTable1 Value 10')
GO

CREATE TABLE ViewTable2
(
    KeyColumn    INT,
    Table2Column    VARCHAR(32)
)
GO

INSERT INTO ViewTable2 VALUES (1, 'ViewTable2 Value 1')
INSERT INTO ViewTable2 VALUES (2, 'ViewTable2 Value 2')
INSERT INTO ViewTable2 VALUES (3, 'ViewTable2 Value 3')
INSERT INTO ViewTable2 VALUES (4, 'ViewTable2 Value 4')
INSERT INTO ViewTable2 VALUES (5, 'ViewTable2 Value 5')
INSERT INTO ViewTable2 VALUES (6, 'ViewTable2 Value 6')
INSERT INTO ViewTable2 VALUES (7, 'ViewTable2 Value 7')
INSERT INTO ViewTable2 VALUES (8, 'ViewTable2 Value 8')
INSERT INTO ViewTable2 VALUES (9, 'ViewTable2 Value 9')
INSERT INTO ViewTable2 VALUES (10, 'ViewTable2 Value 10')
GO

CREATE VIEW TestView1
AS
SELECT VT1.KeyColumn, VT1.Table1Column, VT2.Table2Column
```

```
FROM ViewTable1 VT1 INNER JOIN ViewTable2 VT2 ON
                              (VT1.KeyColumn = VT2.KeyColumn)
GO
INSERT INTO TestView1 VALUES (11, 'ViewTable1 Value 11',
'ViewTable2 Value 11')
GO
CREATE TRIGGER trTestView1InsteadOfInsert
ON TestView1
INSTEAD OF INSERT
AS
DECLARE    @intKeyColumn         INT
DECLARE @vchTable1Column    VARCHAR(32)
DECLARE @vchTable2Column    VARCHAR(32)
DECLARE @intError        INT

SET NOCOUNT ON
SELECT @intKeyColumn=KeyColumn, @vchTable1Column =
Table1Column,    @vchTable2Column = Table2Column
FROM    INSERTED
BEGIN TRANSACTION
 INSERT INTO ViewTable1
VALUES(@intKeyColumn,@vchTable1Column)
    SELECT @intError = @@ROWCOUNT
    INSERT INTO ViewTable2 VALUES(@intKeyColumn,
@vchTable2Column)
    SELECT @intError = @intError + @@ROWCOUNT
    IF ((@intError < 2) OR (@intError % 2) <> 0)
    BEGIN
        RAISERROR('An error occurred during the multitable
insert.', 1, 11)
        ROLLBACK TRANSACTION
        RETURN
    END
COMMIT TRANSACTION
GO
INSERT INTO TestView1 VALUES (11, 'ViewTable1 Value
11','ViewTable2 Value 11')
GO
```

Trong đoạn script trên, đầu tiên ta tạo 2 bảng và chèn 10 dòng vào hai bảng đó. Hai bảng đó có chung một cột là KeyColumn và ta tạo view thể hiện dữ liệu của hai bảng. Lệnh INSERT sẽ thực hiện chèn dữ liệu vào hai bảng. Lệnh INSERT đầu tiên sẽ bị lỗi vì có nhiều bảng trong mệnh đề FROM của câu lệnh SELECT tạo view đó. Lệnh tiếp theo sẽ được thực hiện bởi trigger đã tạo.

**** Tiến trình xử lý giao dịch - TRANSACTION***: DBMS phải đảm bảo một giao dịch được xử lý như một đơn vị cơ sở. Điều này có nghĩa là khi DBMS đang thực hiện tiến trình xử lý câu lệnh đầu tiên trong một giao dịch thì thì tiến trình này phải được tiếp tục cho đến khi tất cả các câu lệnh trong giao dịch được thực hiện thành công, giao dịch này không bị bẻ gãy. Tất cả các câu lệnh trong một giao dịch phải được xử lý như một đơn vị công việc.

Khi một tiến trình bị bẻ gãy ở giữa một giao dịch, như kết quả của việc re boot, hệ thống bị vỡ, thì DBMS phải đưa database về trạng thái tồn tại trước khi giao dịch được bắt đầu. Một câu lệnh SQL đặc biệt để làm điều này đó là ROLL BACK. Nếu tất cả câu lệnh trong đã được thực hiện thành công thì sự thay đổi Database được thực hiện bởi câu lệnh COMMIT. COMMIT và ROLL BACK là các câu lệnh SQL thao chuẩn ANSI/ISO, giống như các câu lệnh SELECT, INSERT, .v.v...

Ví dụ 4.49. DDL Trigger.

```
CREATE TRIGGER safety
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
   PRINT 'You must disable Trigger "safety" to
drop or alter tables!'
   ROLLBACK
```

4.4.3. Các thao tác quản lý trigger

**** Xem mã trigger:*** Dùng thủ tục sp_helptext.

Ví dụ 4.49. Xem code của trigger `trTestView1InsteadOfInsert`

```
Exec sp_helptext trTestView1InsteadOfInsert
go
```

**** Xem những trigger nào đang tồn tại trên một bảng hoặc một view:*** Dùng thủ tục sp_helptrigger.

Ví dụ 4.50. Xem các trigger trên bảng HOSOSV

```
Use QLDiemSV
Go
Exec sp_helptrigger HOSOSV
go
```

**** Thay đổi nội dung trigger:*** Để thay đổi trigger ta dùng câu lệnh ALTER TRIGGER theo cú pháp sau:

- Sửa DML Trigger:

```
ALTER TRIGGER trigger_name
ON {table | view }
[WITH ENCRYPTION]
{
{{FOR | AFTER | INSTEAD OF} { [DELETE] [,]
[INSERT] [,] [UPDATE] }
    [NOT FOR REPLICATION]
AS
    [ { IF UPDATE ( column )
            [ { AND | OR } UPDATE ( column ) ]
                [ ...n ]
        | IF ( COLUMNS_UPDATED ( ) {
bitwise_operator } updated_bitmask )
                { comparison_operator }
column_bitmask [ ...n ]
        } ]
        sql_statement [ ...n ]
}
```

- Sửa DDL Trigger:

```
ALTER TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH ENCRYPTION ]
{ FOR |AFTER } {event_type } [,...n ]
AS { sql_statement  [ ; ] [ ...n ]}
```

Ví dụ 4.51. Thay đổi nội dung trigger trTestView1InsteadOfInsert

```
ALTER TRIGGER trTestView1InsteadOfInsert
ON TestView1
INSTEAD OF INSERT
AS
DECLARE    @intKeyColumn         INT
DECLARE @vchTable1Column    VARCHAR(32)
DECLARE @vchTable2Column    VARCHAR(32)
DECLARE @intError        INT

SET NOCOUNT ON

SELECT @intKeyColumn=KeyColumn, @vchTable1Column =
Table1Column,    @vchTable2Column = Table2Column
FROM    INSERTED

BEGIN TRANSACTION
 INSERT INTO ViewTable1
VALUES(@intKeyColumn,@vchTable1Column)
    SELECT @intError = @@ROWCOUNT
    INSERT INTO ViewTable2 VALUES(@intKeyColumn,
@vchTable2Column)
    SELECT @intError = @intError + @@ROWCOUNT
    IF ((@intError < 2) OR (@intError % 2) <> 0)
    BEGIN
        ROLLBACK TRANSACTION
        RETURN
    END
COMMIT TRANSACTION
GO
```

**** Xóa một trigger***: Dùng câu lệnh DROP TRIGGER.

- Xóa DML Trigger:

```
DROP TRIGGER schema_name.trigger_name [ ,...n ]
```

- Xóa DDL Trigger:

```
DROP TRIGGER trigger_name [ ,...n ]
    ON { DATABASE | ALL SERVER }
```

Ví dụ 4.52. Xóa trigger `trTestView1InsteadOfInsert`

```
DROP TRIGGER trTestView1InsteadOfInsert
go
```

**** Vô hiệu hóa hoặc làm cho có hiệu lực một trigger ta dùng câu lệnh:***

```
DISABLE | ENABLE TRIGGER{[ schema . ]
trigger_name [ ,...n ] | ALL }
ON { object_name | DATABASE | ALL SERVER } [ ; ]
```

Ví dụ 4.53. Vô hiệu hóa trigger `trTriggerTableParent2InsteadOfUpdate` Trên bảng `TriggerTableParent2.`

```
ALTER TABLE TriggerTableParent2
DISABLE TRIGGER trTriggerTableParent2InsteadOfUpdate
GO.
```

Ví dụ 4.54. Vô hiệu hóa trigger `[trTriggerTableChildInsert]` trên bảng `TriggerTableChild.`

```
DISABLE TRIGGER [trTriggerTableChildInsert] ON
[dbo].[TriggerTableChild]
```

Ví dụ 4.55. Làm cho trigger `trTriggerTableParent2InsteadOfUpdate` trên bảng `TriggerTableParent2` có hiệu lực.

```
ALTER TABLE TriggerTableParent2
ENABLE TRIGGER trTriggerTableParent2InsteadOfUpdate
GO.
```

Chương 5. SQL SERVER VÀ LẬP TRÌNH ỨNG DỤNG

5.1. Mô hình kết nối ứng dụng đến SQL server

5.1.1. Mô hình ADO

ADO là phương thức truy xuất dữ liệu cũ của Microsoft, và nó đã trở thành chuẩn cho các môi trường Office, Web, và Visual Basic trước .NET. Sau đây là các sản phẩm phát triển của ADO:

+ Microsoft Office 200x
+ Visual Studio 6.0 (bao gồm tất cả các ngôn ngữ trong đó)
+ SQL Server 7.0, 2000

Data Access Objects (DAO), là chuẩn trước ADO, được sử dụng trong Office và Visual Basic, nhưng nó không được thiết kế để sử dụng trong các môi trường Web như Visual InterDev và các dữ liệu servers khác. Do đó ADO cho đến bây giờ đang là chuẩn để phát triển các ứng dụng.

Các mô hình đối tượng ADO (ADO Object Models)

Đây là các mô hình đối tượng, mang ý nghĩa số nhiều. Không như DAO chỉ bao gồm một mô hình đối tượng (one object model), ADO được phân chia thành các mô hình đối tượng (object models).

Các mô hình đối tượng của ADO làm việc với nhau đưa ra các objects và các collections cần thiết để ta làm việc với dữ liệu. Trong ADO có cặp mô hình đối tượng:

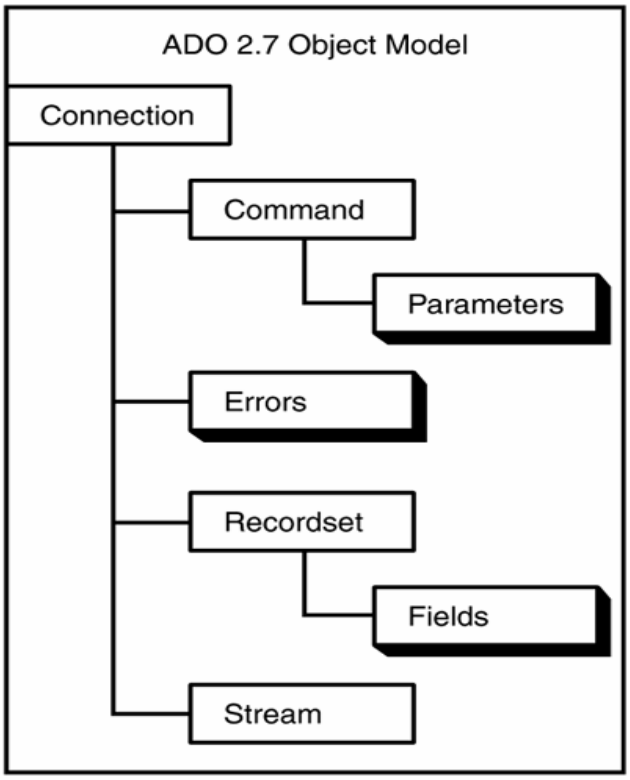
+ ActiveX Data Objects 2.7 (ADODB) cho phép ta tạo và làm việc với recordsets cũng như thực hiện quản lý lỗi.
+ ADO Extensions 2.7 for DDL and Security (ADOX) là ngôn ngữ định nghĩa dữ liệu, cho phép ta làm việc và sửa đổi lược đồ CSDL (database schema) và bảo mật các đối tượng

Tuy nhiên, các đối tượng này sẽ được sử dụng kết hợp với nhau. Chẳng hạn, khi muốn sửa đổi cấu trúc một bảng thì ta cần thư viện ADOX, nhưng

để nó hoạt động thì ta cần ADODB. Các ứng dụng thường làm việc với các bản ghi dữ liệu.

ActiveX Data Objects 2.7 (ADODB) Object Model

Mô hình đối tượng *ActiveX Data Objects* - *ADO* (Xem hình 5.1) bao gồm:



Hình 5.1. **Mô hình ADO**

+ Đối tượng *Connection*. Tương tự như đối tượng Database trong DAO, đó là nơi mà tất cả các thao tác làm việc của ta với ADO được bắt đầu. Tất cả các objects và collections đều được đề cập sau đối tượng *Connection*.

+ Đối tượng Errors collection/Error. Giống như *DAO errors collection* và *error object*, nó cho phép phát triển để quản lý lỗi.

+ Đối tượng *Command*. Cho phép chạy truy vấn trên một database và trả lại các bản ghi trên đối tượng Recordset, thao tác với cấu trúc CSDL, và thực hiện một thao tác dữ liệu. Một tập hợp các tham số sẽ được sử dụng với đối tượng Command.

+ *Recordset object*. Tương tự như *DAO Recordset object*, ta có thể mở các đối tượng Recordset để chỉ đọc hoặc tạo bảng động. Mỗi đối tượng Recordset có một tập các trường (*Field)*.

+ *Stream object*. Đối tượng này cho phép ta đọc cây đặc biệt – cấu trúc phân cấp (special tree-structured hierarchies), như e-mail messages hoặc các file hệ thống.

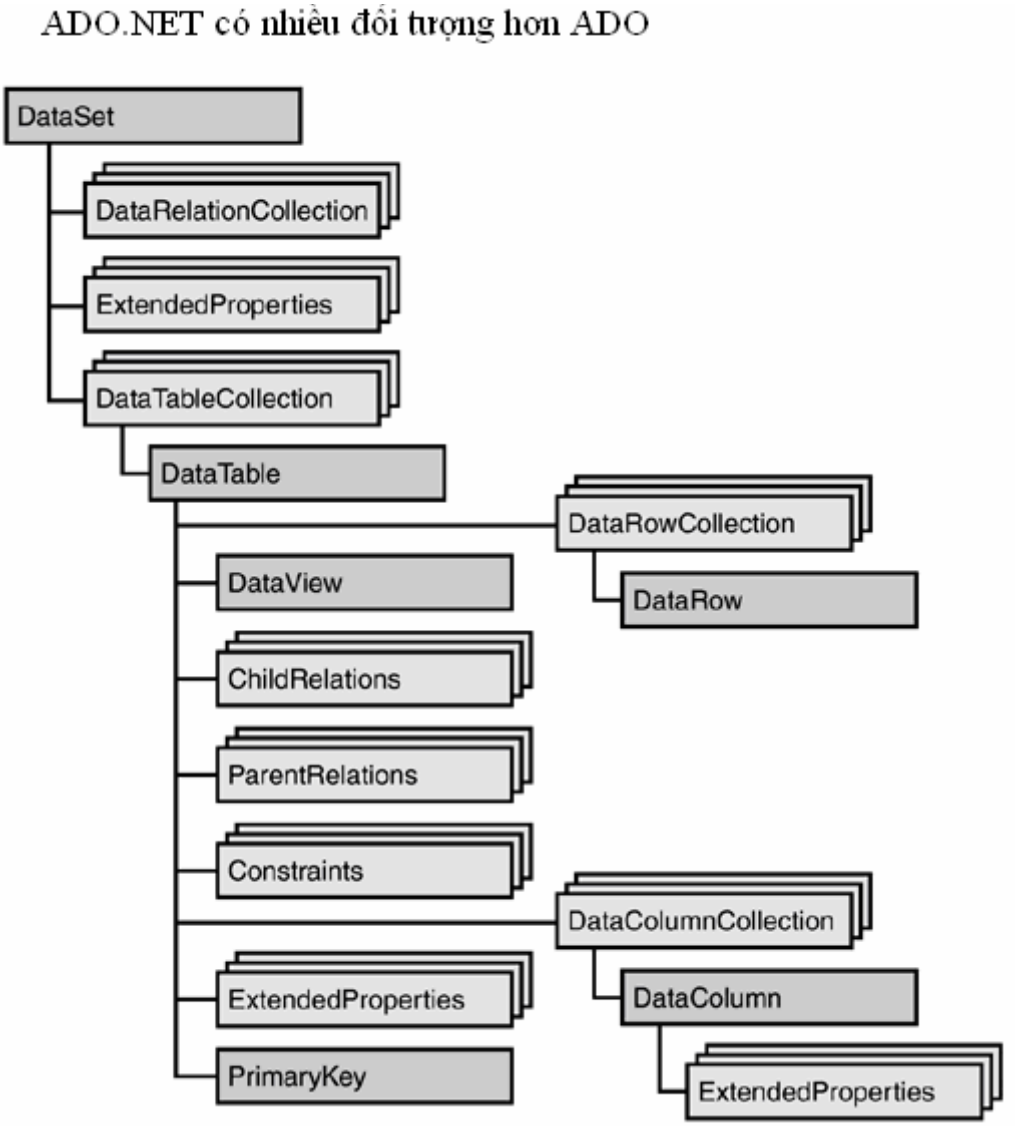
5.1.2. Mô hình ADO.NET

ADO.NET được thiết kế để cung cấp kiến trúc rời rạc *(disconnected architecture)*. Có nghĩa là các ứng dụng *connect* tới database truy nạp dữ liệu và lưu trữ trong memory. Sau đó *disconnect* với database và sử dụng bộ nhớ sao chép (memory copy) của dữ liệu đó. Nếu như database cần phải được update với các thay đổi trên memory copy, một kết nối mới (connection) được hình thành và database được update. Bộ nhớ chính lưu trữ dữ liệu là *DataSet*, nơi mà chứa các bộ nhớ lưu trữ dữ liệu khác, như là: Các đối tượng DataTable; ta có thể lọc và sắp xếp dữ liệu trên DataSet bằng việc sử dụng các đối tượng DataView.

Việc sử dụng kiến trúc *disconnected architecture* cung cấp rất nhiều lợi ích, trong đó, quan trọng nhất là cho phép ứng dụng trở lên *scale up*. Nghĩa là database của ta sẽ thực thi tốt như hỗ trợ hàng trăm users tuy nó chỉ hỗ trợ vài chục users. Điều đó có thể bởi vì ứng dụng connects tới database chỉ đủ để truy nạp hoặc update dữ liệu, bằng cách đó giải phóng các *connections* và database sẵn có cho các thể hiện khác của ứng dụng hoặc các ứng dụng khác sử dụng cùng database.

Các đối tượng trong ADO.NET

Đối tượng chính được sử dụng trong ADO.NET là đối tượng `DataSet`. Ta có thể thấy đối tượng `DataSet` và các thuộc tính, các phương thức, và các đối tượng của nó trong hình 5.2 dưới đây.



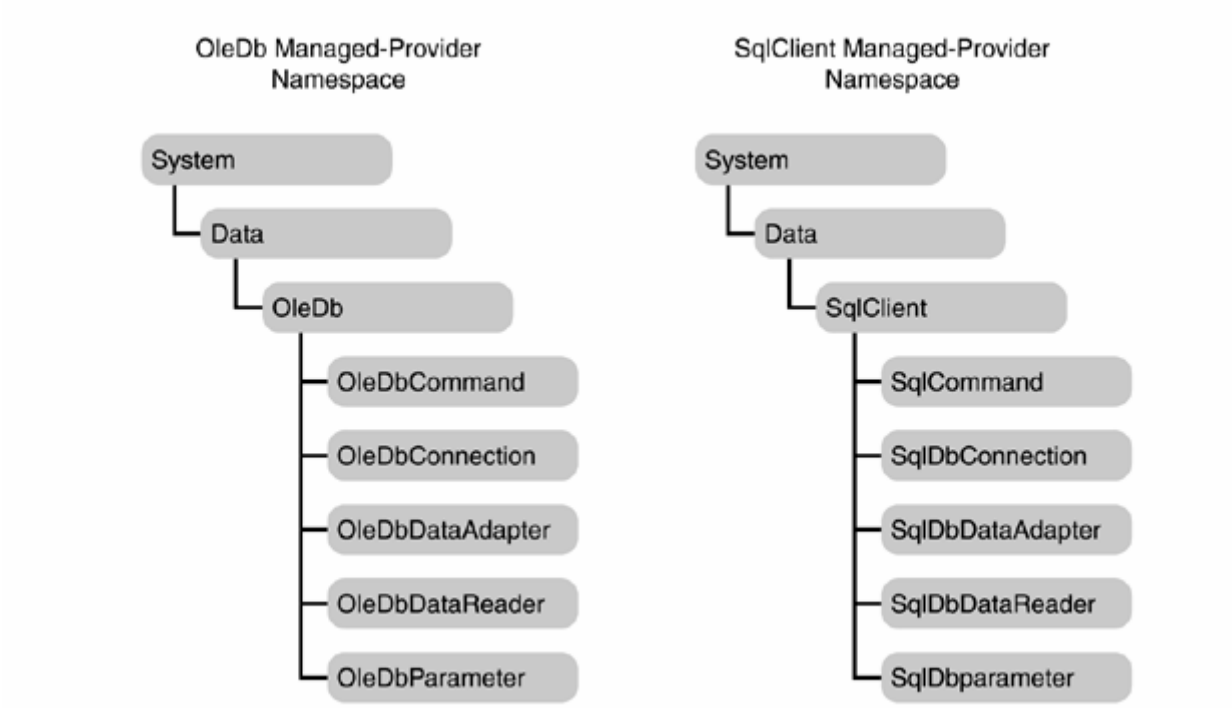
***Hình 5.2.* Mô hình ADO.NET**

Xét bảng Bảng 5.1 để tìm hiểu các mô tả ngắn ngọn về các đối tượng hay dùng.

Bảng5.1. ADO.NET Data Objects hay được dùng để thao tác dữ liệu	
Object	***Mục đích***

`DataSet`	Đối tượng này được sử dụng cùng với các điều khiển dữ liệu khác, lưu trữ các kết quả được trả về bởi các đối tượng *commands* và *data adapters*. Không như recordset của ADO và DAO, *Data set* thực sự mang lại một view phân cấp của dữ liệu. Bằng việc sử dụng các thuộc tính và các collections trong đối tượng `DataSet`, ta có thể nhận được toàn bộ các quan hệ (relations), các tables riêng biệt, rows, và columns.
`DataTable`	Một trong đối tượng của data set, là đối tượng `DataTable` cho phép ta có thể thao tác dữ liệu trong một table riêng biệt. Data table tương tự như đối tượng recordset trong ADO.
`DataView`	Bằng việc sử dụng đối tượng này, ta có thể lọc và sắp xếp dữ liệu, duy trì các views khác nhau của dữ liệu. Mỗi data table có một default view, nó là nơi khởi động data view mà có thể được sửa dổi và lưu trữ trong data view khác.
`DataRow`	Đối tượng này cho phép ta thao tác các rows của dữ liệu trong data tables. Nó có thể được xét như một cache của dữ liệu mà ta có thể thực hiện các thao tác adding, deleting, và modifying records. Ta có thể quay trở lại recordset, nơi mà ta sẽ chạy các câu lệnh SQL để update dữ liệu về server.
`DataColumn`	Như tên đề nghị, ta có thể nhận các thông tin tại các cột bằng việc sử dụng đối tượng `DataColumn`. Ta có thể lấy thông tin lược đồ như là dữ liệu bằng việc sử dụng đối tượng này. Ví dụ: nếu ta muốn tạo một list box của tên các trường, ta có thể lặp thông qua tập DataColumn của các dòng dữ liệu và truy lục tất cả các tên trường.
`PrimaryKey`	Đối tượng này cho phép ta chỉ định primary key cho table dữ liệu. Cách khác, khi ta sử dụng phương thức Find của data table.

.NET cũng cung cấp các classes, được gọi *data providers*, sử dụng tên miền (namespaces) để làm việc với các đối tượng ADO.NET và cung cấp các truy xuất với dữ liệu. Lõi của các lớp ADO.NET tồn tại trong *namespace System.Data*. Namespace này chứa một vài namespaces con. Phần quan trọng nhất của chúng là các tên miền *System.Data.SqlClient* và System.Data.OleDb. Chúng cung cấp các classes sử dụng cho việc truy cập *SQL Server databases* và *OLE (Object Linking and Embedding) DB- compliant databases* (phục hồi dữ liệu). Các lớp của OleDb và SqlClient cho trong hình 5.3. và Bảng 5.2. mô tả vắn tắt các đối tượng thường được sử dụng để thao tác với dữ liệu.



***Hình 5.3.* OleDb và SqlClient classes**

Bảng 5.2. .NET Data Provider Classes thường được sử dụng để thao tác dữ liệu

Object	***Mục đích***
`Command`	Tương tự như đối tượng ADO `Command`, cho phép ta thực hiện các stored procedures trong code. Khác với phiên bản ADO, ta có thể tạo đối tượng `DataReader` bằng việc sử dụng phương thức `ExecuteReader`.

`Connection`	Đây là đối tượng mở kết nối tới server và database mà ta muốn làm việc. Khác với đối tượng ADO `Connection`, cách thức kết nối phụ thuộc vào đối tượng mà ta muốn làm việc, như là đối tượng `DataReader` hay `DataSet`.
`DataAdapter`	Đối tượng `DataAdapter` cho phép ta tạo các câu lệnh SQL và điền dữ liệu vào *datasets*. Nó cũng cho phép tạo các *action queries* cần thiết, như là `Insert`, `Update`, và `Delete`.
`DataReader`	Đối tượng này tạo một *read-only*, *forward-only stream* của dữ liệu mà cho phép chúng đặt trên các điều khiển, như `ListBox` và `ComboBox`.
`Parameter`	Đây là đối tượng cho phép chỉ định các tham số mà các đối tượng `DataAdapter` có thể chỉ định và sử dụng

Ngoài ra, hai tên miền con khác cũng tồn tại trong tên miền System.Data, đó là: *System.Data.OracleClient* và *System.Data.Odbc.* Trong đó*, namespace System.Data.OracleClient* được sử dụng dành cho các CSDL Oracle. Các lớp *SqlClient* cung cấp các kết quả tốt nhất khi làm việc với các SQL Server databases; *OracleClient* cung cấp thực thi tối ưu khi truy cập vào Oracle databases. Namespace System.Data.Odbc cung cấp truy cập tới ODBC (Open Database Connectivity) data sources cũ mà không được support bởi công nghệ OleDb.

Như vậy, các tên miền *System.Data.SqlClient, System.Data.OleDb, System.Data.OracleClient*, và *System.Data.Odbc* được biết như là *data providers* trong ADO.NET.

Trong môn học, ta xây dựng ứng dụng truy xuất SQL Server databases nên sử dụng namespace SqlClient. Tuy nhiên trong ADO.NET, các data providers khác nhau làm việc theo một cách thức tương tự nhau. Vì vậy các kỹ xảo ta sử dụng ở đây có thể được dễ dàng chuyển giao cho các lớp OleDb classes. Với ADO.NET, ta sử dụng *data provider* này là phù hợp nhất cho

data source của ta. Ta không cần thiết phải học tất cả các interface mới, bởi vì tất cả các *data providers* làm việc theo một cách thức tương tự nhau.

5.1.3. Điểm khác nhau giữa ADO và ADO.NET

+ ADO làm việc với kết nối dữ liệu (connected data). Nghĩa là khi truy cập dữ liệu, chẳng hạn như viewing hoặc updating dữ liệu, thì nó là thời gian thực (real-time), với một kết nối được sử dụng trong suốt thời gian đó.

+ ADO.NET sử dụng dữ liệu theo kiểu disconnected. Khi truy cập dữ liệu, ADO.NET tạo bản copy của dữ liệu bằng việc sử dụng XML. ADO.NET chỉ chiếm giữ với việc mở một kết nối trong khoảng thời gian đủ để đẩy dữ liệu vào hoặc để thực hiện bất cứ yêu cầu updates nào. Với điều này, làm cho ADO.NET rất hữu hiệu cho các ứng dụng Web. Nó cũng tốt cho các ứng dụng desktop.

+ ADO có một đối tượng chính được dùng để tham chiếu dữ liệu, được gọi là *Recordset object.* Về cơ bản đối tượng này đưa ra một bảng view đơn của dữ liệu, tuy nhiên là ta có thể kết nối các bảng để tạo một tập các bản ghi mới. Với ADO.NET, ta có nhiều đối tượng khác nhau cho phép truy xuất dữ liệu theo các cách khác nhau. Đối tượng *DataSet* thực sự cho phép ta lưu trữ mô hình quan hệ của database.

+ ADO chỉ cho phép tạo client-side cursors, ngược lại ADO.NET cho phép chọn hoặc là sử dụng client-side hoặc server-side cursors. Trong ADO.NET, classes thực sự điều khiển sự làm việc của các cursors. Nó cho phép phát triển để lựa chọn cái tốt nhất.

Như vậy, tùy theo ứng dụng cụ thể ta có thể sử dụng các mô hình kết nối ứng dụng với SQL Server cụ thể. Trong phạm vi môn học ta sẽ sử dụng mô hình ADO.NET với các lớp *SqlClient* và sử dụng ngôn ngữ lập trình Visual Basic.NET.

5.2. Các lớp SqlClient trong mô hình ADO.NET

Danh sách sau chứa các classes ADO.NET mà chúng sẽ được sử dụng trong quá trình xây dựng ứng dụng sử dụng SQL Server Databases:

- ✓ SqlConnection
- ✓ SqlDataAdapter
- ✓ SqlCommand
- ✓ SqlParameter
- ✓ SqlDataReader

Chú ý: Chúng là các *SqlClient* cụ thể, trừ namespace OleDb có rất nhiều close tương đương. Mỗi khi ta muốn sử dụng lớp classes này, ta phải add một tham chiếu tới tên miền System.Data. Để đơn giản, ta có thể sử dụng từ khóa *Imports*, và lúc đó ta không cần phải điền đầy đủ các thành phần tên miền của SqlClient trong code, như đoạn code sau:

Imports System.Data.SqlClient

Nếu ta muốn sử dụng lõi của classes ADO.NET, như là *DataSet* và *DataView* không cần gõ đầy đủ tên miền, ta cần phải Import the tên miền System.Data, như đoạn code tiếp theo:

Imports System.Data

Ta nên quen thuộc với việc import các tên miền khác nhau trong project của ta. Tuy nhiên, để được toàn diện ta sẽ phủ chúng khi thực hành các bài tập. Bây giờ, ta xem xét các classes chính tồn tại trong namespace System.Data.SqlClient.

5.2.1. Class SqlConnection

Class SqlConnection có thể coi là "trái tim" của các classes mà ta sẽ thảo luận trong phần này, bởi vì nó cung cấp một connection tới SQL Server database.

Khi xây dựng một đối tượng SqlConnection, ta có thể chọn để chỉ định một chuỗi kết nối (connection strings) như một tham số. Chuỗi connection chứa tất cả các thông tin cần thiết để mở một connection tới your database.

Nếu ta không chỉ định chuỗi kết nối trong khi xây dựng, ta có thể thiết lập nó bằng việc sử dụng thuộc tính *SqlConnection.ConnectionString*.

Ta xem xét các chuỗi *connection string*s làm việc như thế nào?

**** Các tham số của chuỗi kết nối (Connection String Parameters)***

Phương thức để chuỗi connection string được xây dựng sẽ phụ thộc vào *data provider* gì ta đang sử dụng. Khi truy xuất SQL Server, ta thường cung cấp Server và Database, được chỉ dẫn theo bảng sau:

Tham số	***Mô tả***
Server	Tên của SQL Server mà ta muốn truy xuất. Nó thường là tên của máy tính đang chạy SQL Server. Ta có thể sử dụng *(local)* hoặc *localhost* nếu SQL Server trên cùng một máy đang chạy ứng dụng. Nếu bạn sử dụng các thể hiện tên của SQL Server, thì tham số nên chứa tên của máy tính theo sau là đấu gạch chéo ngược và tiếp đó là thể hiện tên của SQL Server.
Database	Là tên của database mà ta muốn connect tới.

Ta cũng cần một vài thông tin về authentication và thực hiện theo hai cách: bằng cách cung cấp *username* và *password* trong chuỗi kết nối (connection strings) hoặc bằng cách kết nối tới *SQL Server* sử dụng *NT account* mà ứng dụng đang chạy trong đó.

Nếu ta muốn connect tới server bằng việc chỉ định *username* và *password*, ta cần tính đến các parameters đó trong chuỗi kết nối connection, được cho trong bảng sau:

Tham số	***Mô tả***
User ID	Username sử dụng để connect tới database. account với user ID này phải tồn tại trong SQL Server và được phép truy cập đến database chỉ định.

Password	Là password của user đã chỉ định.

Tuy nhiên, SQL Server có thể được thiết lập sử dụng Windows NT account của user người đang chạy chương trình để mở connection. Trong trường hợp này, ta không cần chỉ định *username* và *password*. Ta chỉ cần phải chỉ định nó khi ta đang sử dụng *integrated security (security kết hợp* - Phương thức này được gọi là security kết hợp bởi vì SQL Server đang được kết hợp với Windows NT's security system và cung cấp connection an toàn nhất bởi vì các tham số User ID và Password parameters phải được chỉ định trong code). Ta sử dụng tham số Integrated Security này, và thiết lập là True khi ta muốn connect SQL Server đang sử dụng user's NT account hiện tại. Tất nhiên, với các làm việc này, user của ứng dụng phải được cho phép sử dụng SQL Server database. Giả dụ như là đang sử dụng SQL Server Enterprise Manager.

Để hiểu chức năng của các tham số này trong chuỗi kết nối ta khởi tạo một đối tượng connection, xét đoạn code sau. Nó sử dụng lớp *SqlConnection* để khởi tạo một đối connection mà được sử dụng chỉ định user ID và password trong chuỗi kết nối:

```
Dim objConnection As New SqlConnection _
    ("server=THUHUONG;database=QLDiemSV;" &_
    "user id=sa;password=pass2008")
```

Chuỗi kết nối này sẽ connects tới SQL Server database. Tham số Server chỉ định là database nằm trên máy THUHUONG. Tham số Database chỉ định database mà ta muốn truy cập, trong trường hợp này là database QLDiemSV. Cuối cùng, các tham số User ID và Password chỉ định User ID và password của user định nghĩa trên database. Ta thấy rằng, mỗi một tham số được gán một giá trị qua dấu **=**, và mỗi cặp *parameter* - *value* được phân cách nhau nhau bởi dấu chấm phảy. Hoặc

```
Dim objConnection As SqlConnection = New _
SqlConnection("Server=localhost;Database=pubs;" & _
"User ID=sa;Password=vbdotnet;")
```

Chuỗi kết nối này sẽ connects tới SQL Server database. Tham số Server chỉ định là database nằm trên máy local. Tham số Database chỉ định database mà ta muốn truy cập là database pubs. Cuối cùng, các tham số User ID và Password chỉ định User ID và password của user định nghĩa trên database.

**** Mở và đóng Connection***

Sau khi khởi tạo đối tượng connection với một chuỗi connection như đã trình bày ở phần trước, ta có thể gọi các phương thức của đối tượng SqlConnection như là Open và Close, thậm chí open và close một connection tới database được chỉ định trong chuỗi kết nối connection. Ví dụ, cho đoạn code sau:

```
' Open the database connection
objConnection.Open()
'  ... Use the connection

' Close the connection...
objConnection.Close()
```

Chú ý: Ta có thể sử dụng trình Wizard để thực hiện kết nối với CSDL theo sự chỉ dẫn của nó.

5.2.2. Class SqlCommand

Lớp SqlCommand đại diện cho các câu lệnh SQL để thực hiện truy vấn dữ liệu lưu trữ. Các câu lệnh là truy vấn select, insert, update, hoặc delete,.v.v… Có thể là các chuỗi SQL hoặc gọi một *stored procedure*. Và truy vấn được thực hiện có thể chứa các tham số hoặc không chứa các tham số.

Kiến trúc của lớp SqlCommand có một vài biến số, nhưng phương thức đơn giản nhất để khởi tạo một đối tượng SqlCommand với không tham số (parameters). Tiếp theo, sau khi đối tượng đã được khởi tạo, ta có thể thiết lập các thuộc tính ta cần để thực hiện các thao tác sắp tới. Đoạn code sau cho phép khởi tạo một đối tượng SqlCommand:

```
Dim objCommand As SqlCommand = New SqlCommand()
```

Khi sử dụng các *data adapters* (bộ điều hợp dữ liệu) và *datasets*, không cần gọi nhiều các đối tượng của chúng. Chúng có thể chủ yếu được sử dụng cho việc thực thi một câu lệnh select, delete, insert, hoặc update, mà đó là những gì ta đề cập trong chương này. Ta cũng có thể sử dụng các đối tượng command với *data reader*. Một *data reader* là một lựa chọn khác với DataSet mà nó sử dụng một vài tài nguyên hệ thống nhưng kém mềm dẻo.

**** Thuộc tính Connection***

Một vài thuộc tính phải được thiết lập trên đối tượng SqlCommand trước khi ta có thể thực hiện một câu truy vấn. Thuộc tính đầu tiên trong các thuộc tính này đó là thuộc tính Connection. Thuộc tính này thiết lập tới đối tượng SqlConnection như trong đoạn code sau:

```
objCommand.Connection = objConnection
```

Chú ý: Để câu lệnh được thực hiện thành công, thì connection phải được mở ở thời điểm thực hiện.

**** Thuộc tính CommandText***

Thuộc tính tiếp theo phải được thiết lập đó là thuộc tính CommandText. Thuộc tính này chỉ định chuỗi SQL hoặc *stored procedure* sẽ được thực hiện. Hầu hết các databases quy định đặt các các giá trị của chuỗi trong cặp dấu *" các giá trị của chuỗi "*, ví dụ:

```
Dim objConnection As New SqlConnection _
                ("server=THUHUONG;database=QLDiemSV;" &_
    "user id=sa;password=pass2008")
Dim objCommand As SqlCommand = New SqlCommand()
' Open the connection, execute the command
objConnection.Open()
' Set the SqlCommand object properties...
objCommand.Connection = objConnection
objCommand.CommandText = "INSERT INTO HOSOSV " & _
            "(MaSV, HoDem, TenSV, NgaySinh, MaLop) " & _
"VALUES('SV001','Nguyen Thi','Hoa','12/8/1988','TH05')"
```

Câu lệnh INSERT chỉ đơn giản nghĩa là "chèn một hàng mới vào bảng HOSOSV. Trong đó cột MaSV nhận giá trị 'SV001', HoDem nhận giá trị 'Nguyen Thi', TenSV nhận giá trị 'Hoa', NgaySinh nhận giá trị '12/8/1988' và cột MaLop nhận giá trị ' TH05'."

Với cách cơ bản này các câu lệnh INSERT làm việc trong SQL. Ta có, theo sau INSERT INTO là tên của bảng. Tiếp là một danh sách tên các cột trong cặp dấu ngoặc đơn. Tiếp theo nữa là từ khóa VALUES là danh sách các giá trị được chèn vào cột theo đúng thứ tự.

Trong ví dụ trên, với các giả thiết là ta biết các giá trị để chèn trong khi thực hiện chương trình, thường là không có thực trên hầu hết các ứng dụng. Cũng may, ta cũng có thể tạo các câu lệnh với các tham số và thiết lập giá trị cho các tham số đó một cách độc lập. Ta sẽ tìm hiểu cách sử dụng các tham số đó như thế nào trong phần Parameters Collection.

**** Parameters Collection***

Placeholders, ký hiệu @, là tiền tố các biến trong các câu lệnh SQL; Chúng được điền vào trước các tham số. Như vậy, nếu ta muốn cập nhật bảng HOSOSV được mô tả trong ví dụ trước nhưng không biết các giá trị tại thời điểm thiết kế, ta chỉ cần viết:

```
Dim objConnection As New SqlConnection _
                ("server=THUHUONG;database=QLDiemSV;" &_
    "user id=sa;password=pass2008")

Dim objCommand As SqlCommand = New SqlCommand()
' Open the connection, execute the command
objConnection.Open()
' Set the SqlCommand object properties...
objCommand.Connection = objConnection
objCommand.CommandText = "INSERT INTO HOSOSV " & _
            "(MaSV, HoDem, TenSV, NgaySinh, MaLop) " & _
            "VALUES(@MaSV,@Hodem,@TenSV,@NgaySinh,@MaLop)"
```

Ở đây, thay vì cung cấp các giá trị cụ thể, ta cung cấp các placeholders. Placeholders, như đã đề cập, luôn xuất hiện với ký hiệu @. Chúng không cần được đặt trước tên các trường trong các bảng của database mà chúng thể hiện, nhưng thường đơn giản hơn nếu ta sử dụng tên tham số trùng với tên trường dữ liệu, vì nó thường tốt hơn cho chính các văn bản code của ta.

Tiếp theo, ta cần tạo các tham số mà sẽ được sử dụng để truyền giá trị cho các placeholders khi câu lệnh SQL được thực hiện. Ta cần phải create và add các tham số vào *Parameters collection* của đối tượng *SqlCommand*.

Thuật ngữ *parameters* chỉ các tham số cần thiết cho câu lệnh SQL hoặc *stored procedure*, không chỉ các tham số được quy định chuyển để chuyển sang các phương thức của Visual Basic 2005.

Ta có thể truy cập vào tập Parameters collection của đối tượng SqlCommand bằng việc chỉ định thuộc tính Parameters. Sau khi truy cập vào tập Parameters collection, ta có thể sử dụng các thuộc tính và các phương thức của nó để tạo một hoặc nhiều hơn các tham số trong collection. Cách tốt nhất để add một tham số tới câu lệnh được mô tả trong ví dụ sau:

```
Dim objConnection As New SqlConnection _
                ("server=THUHUONG;database=QLDiemSV;" &_
    "user id=sa;password=pass2008")

Dim objCommand As SqlCommand = New SqlCommand()
' Open the connection, execute the command
objConnection.Open()
' Set the SqlCommand object properties...
objCommand.Connection = objConnection
objCommand.CommandText = "INSERT INTO HOSOSV " & _
            "(MaSV, HoDem, TenSV, NgaySinh, MaLop) " & _
            "VALUES(@MaSV,@Hodem,@TenSV,@NgaySinh,@MaLop)"
' Add parameters for the placeholders in the SQL in the
objCommand.Parameters.AddWithValue("@MaSV", txtMaSV.Text)
objCommand.Parameters.AddWithValue("@Hodem", txtHodem.Text)
objCommand.Parameters.AddWithValue("@TenSV", txtTenSV.Text)
objCommand.Parameters.AddWithValue("@NgaySinh",_
                    txtNgaySinh.Text).DbType = DbType.Date
objCommand.Parameters.AddWithValue("@MaLop", cboMaLop.Text)
```

Phương thức *AddWithValue* thừa nhận tên của tham số và đối tượng ta muốn add. Trong trường hợp này, ta sử dụng thuộc tính Text của đối tượng Text box trên cùng một Form.

**** Phương thức ExecuteNonQuery***

Cuối cùng, ta có thể thực hiện các câu lệnh. Để làm điều này, connection cần phải được mở. Ta có thể gọi phương thức ExecuteNonQuery của đối tượng SqlCommand. Phương thức này thực hiện câu lệnh SQL và là nguyên

nhân dữ liệu được chèn vào database. Để hoàn thành đoạn code, ta phải mở connection, thực hiện query, và đóng connection trở lại:

```
Dim objConnection As New SqlConnection _
                ("server=THUHUONG;database=QLDiemSV;" &_
    "user id=sa;password=pass2008")

Dim objCommand As SqlCommand = New SqlCommand()
' Set the SqlCommand object properties...
objCommand.Connection = objConnection
objCommand.CommandText = "INSERT INTO HOSOSV " & _
            "(MaSV, HoDem, TenSV, NgaySinh, MaLop) " & _
            "VALUES(@MaSV,@Hodem,@TenSV,@NgaySinh,@MaLop)"
' Add parameters for the placeholders in the SQL in the

objCommand.Parameters.AddWithValue("@MaSV", txtMaSV.Text)
objCommand.Parameters.AddWithValue("@Hodem", txtHodem.Text)
objCommand.Parameters.AddWithValue("@TenSV", txtTenSV.Text)
objCommand.Parameters.AddWithValue("@NgaySinh",_
                    txtNgaySinh.Text).DbType = DbType.Date
objCommand.Parameters.AddWithValue("@MaLop", cboMaLop.Text)
' Open the connection, execute the command
objConnection.Open()
' Execute the SqlCommand object to insert the new data...
Try
    objCommand.ExecuteNonQuery()
Catch SqlExceptionErr As SqlException
    MessageBox.Show(SqlExceptionErr.Message)
End Try
' Close the connection...
objConnection.Close()
```

**** Phương thức ExecuteReader***

Trái với phương thức ExecuteNonQuery, phương thức ExecuteReader của đối tượng SqlCommand sẽ thực hiện truy vấn trả về kết quả trên đối tượng DataReader. Và dữ liệu trên DataReader thường được đổ sang các điều khiển ListBox, ComboBox trên form ứng dụng.

```
    ' Declare local variables and objects...
    Dim objCommand As SqlCommand = New SqlCommand()
    Dim objDataReader As SqlDataReader

    ' Open the connection, execute the command
    objConnection.Open()
    ' Set the SqlCommand object properties...
    objCommand.Connection = objConnection
```

```
objCommand.CommandText = "Select MaLop, TenLop From Lop"

' Execute the SqlCommand object to insert the new data...
objDataReader = objCommand.ExecuteReader
cboMaLop.Items.Clear()

Do While (objDataReader.Read())
     cboMaLop.Items.Add(objDataReader.Item(0))
Loop
' Close the connection...
objConnection.Close()
```

5.2.3. Class SqlDataAdapter

Lớp *SqlDataAdapte*r tương tự như lớp *OleDbDataAdapter*. Điểm khác nhau chính là OleDbDataAdapter có thể truy cập vào bất cứ *data source* được supports OLE DB, trong khi *SqlDataAdapter* chỉ supports cho SQL Server databases. Ta có thể sử dụng trong cùng một cách như nhau; ta có thể cấu hình SqlDataAdapter sử dụng wizards, giống như là cấu hình cho *OleDbDataAdapter* (được cung để ta đang truy xuất vào SQL Server data source), ta sẽ trình bày trong ví dụ minh họa cụ thể ở phần sau. Trong phần này, ta tìm hiểu cách cấu hình và sử dụng SqlDataAdapter trong code, những nguyên tắc này cũng có thể áp dụng cho OleDbDataAdapter.

Data adapters hoạt động như một cầu nối giữa *data source* và các đối tượng lưu dữ liệu trong bộ nhớ, như là *DataSet*. Để truy xuất data source, chúng sử dụng các đối tượng *command*, ta đã xem xét ở trên. Các đối tượng *command* này liên quan đến các connections, vì vậy *data adapter* này dựa vào các đối tượng *command* và connection để truy xuất và điều khiển data source.

Thuộc tính SelectCommand của lớp SqlDataAdapter được sử dụng nắm giữ một SqlCommand để gọi dữ liệu từ *data source*. Sau đó *data adapter* đẩy kết quả của truy vấn vào DataSet hoặc DataTable.

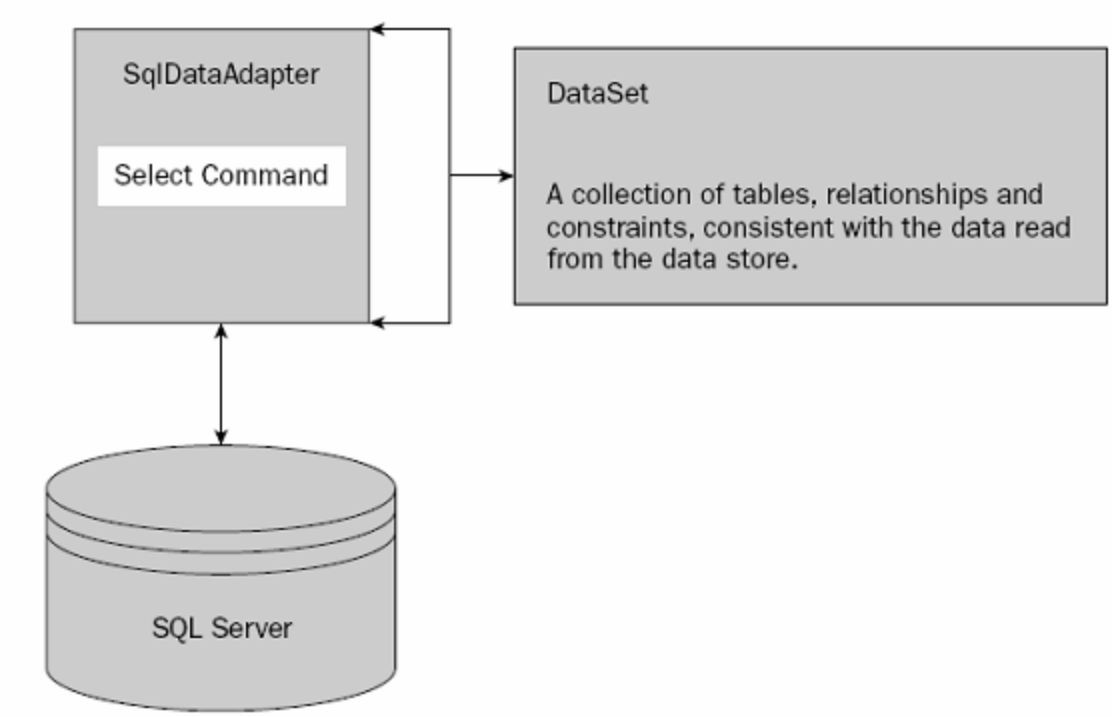
SqlDataAdapter cũng có các thuộc tính UpdateCommand, DeleteCommand, and InsertCommand. Chúng cũng là các đối tượng SqlCommand, được sử dụng để ghi các thay đổi với DataSet hoặc DataTable quay trở lại data source. Vấn đề này có thể thấy phức tạp, nhưng các tools thực tế rất dễ sử dụng. Ta đã học các câu lệnh SQL trong chương trước để viết

SelectCommand, và các tools được gọi *command builders* mà ta có thể tạo một cách tự động các câu lệnh khác dựa trên chúng.

Xét thuộc tính SelectCommand và xét xem ta có thể tạo các câu lệnh cho việc updating, deleting, và inserting records như thế nào.

**** Thuộc tính SelectCommand***

Thuộc tính SelectCommand của lớp SqlDataAdapter được dùng để đẩy dữ liệu vào từ SQL Server database DataSet, như hình vẽ Hình 5.4.



Hình 5.4. Thuộc tính Select Command

Khi ta muốn đọc dữ liệu từ kho dữ liệu, đầu tiên ta phải thiết lập thuộc tính SelectCommand của lớp SqlDataAdapter. Thuộc tính này là đối tượng SqlCommand và nó được dùng để chỉ định dữ liệu được select và select dữ liệu như thế nào. Vì vậy, thuộc tính SelectCommand cũng có các thuộc tính của nó, và ta cần thiết lập chúng như là thiết lập các thuộc tính trên các câu lệnh thông thường. Ta đã xem xét các thuộc tính sau của đối tượng SqlCommand:

- ✓ Connection: Thiết lập đối tượng SqlConnection để truy xuất vào data store.

- ✓ CommandText: Thiết lập các câu lệnh SQL hoặc tên stored procedure được sử dụng để selectdữ liệu.

Trong các ví dụ trước của các đối tượng SqlCommand, ta đã sử dụng trực tiếp các câu lệnh SQL. Nếu ta muốn sử dụng các stored procedures, ta cần phải sử dụng thêm thuộc tính CommandType, thuộc tính này thiết lập một giá trị xác định xem thuộc tính CommandText được biên dịch như thế nào.

Trong chương này, ta tập trung vào các câu lệnh SQL, nhưng các stored procedures thường cũng rất hữu dụng, đặc biệt là nếu chúng đã tồn tại trên database. Nếu ta muốn sử dụng chúng, thì thiết lập thuộc tính CommandText với tên của stored procedure (chú ý là phải được đóng trong cặp dấu ngoặc kép "tên của stored procedure hoặc chuỗi SQL " bởi trình trình biên dịch biên dịch chúng như là một string), và thiết lập thuộc tính CommandType thành CommandType.StoredProcedure.

- ***Thiết lập SelectCommand với một chuỗi SQL***

Xét xem làm thế nào, ta có thể thiết lập các thuộc tính trên trong code. Đoạn code sau cho ta thấy các cách thiết lập điển hình khi thực hiện chuỗi SQL string:

```
' Declare a SqlDataAdapter object...
Dim objDataAdapter As New SqlDataAdapter()
' Assign a new SqlCommand to the SelectCommand property
objDataAdapter.SelectCommand = New SqlCommand()
' Set the SelectCommand properties...
objDataAdapter.SelectCommand.Connection = objConnection
objDataAdapter.SelectCommand.CommandText = _
"SELECT MaSV, HoDem, TenSV, FROM HOSOSV ORDER BY TenSV,
Hodem"
```

Đầu tiên trong đoạn code trên khai báo đối tượng SqlDataAdapter. Đối tượng này có thuộc tính SelectCommand được thiết lập bằng SqlCommand; Ta chỉ cần thiết lập các thuộc tính của command. Ta thiết lập các thuộc tính bằng việc thiết lập thuộc tính Connection, để đối tượng connection được hợp lệ thì nó phải được tạo trước đoạn code trên. Tiếp theo, ta thiết lập các thuộc tính của thuộc tính CommandText bằng các câu lệnh SQL SELECT.

- ***Thiết lập SelectCommand với một Stored Procedure***

Trong đoạn code tiếp theo, ta tìm hiểu cách thiết lập các thuộc tính trên khi sử dụng để thực hiện một stored procedure. Một stored procedure là một nhóm các câu lệnh SQL và đã được lưu trữ trong database dưới một tên duy nhất và được thực hiện như một unit. stored procedure trong ví dụ này (usp_select_DSSV) sử dụng cùng câu lệnh SQL ta đã sử dụng trong phần trước:

```
' Declare a SqlDataAdapter object...
Dim objDataAdapter As New SqlDataAdapter()
' Assign a new SqlCommand to the SelectCommand property
objDataAdapter.SelectCommand = New SqlCommand()
' Set the SelectCommand properties...
objDataAdapter.SelectCommand.Connection = objConnection
objDataAdapter.SelectCommand.CommandText = "usp_select_DSSV"
objDataAdapter.SelectCommand.CommandType =
CommandType.StoredProcedure
```

Thuộc tính CommandText bây giờ được chỉ định bằng tên của stored procedure mà ta muốn thực hiện thay vì chuỗi SQL đã được chỉ định trong ví dụ trước. Cũng phải chú ý đến thuộc tính CommandType. Trong ví dụ trước ta không cần phải chuyển đổi thuộc tính này, bởi vì nó được mặc định là CommandType.Text, mà ta cần thiết lập để thực hiện các câu lệnh SQL. Trong ví dụ này, ta phải thiết lập giá trị CommandType.StoredProcedure, để chỉ dẫn rằngthuộc tính CommandText chứa tên của stored procedure để thực hiện.

Sử dụng Command Builders để tạo các Commands khác

SelectCommand là tất cả những gì ta cần thiết để chuyển dữ liệu từ database vào DataSet. Sau khi ta cho phép các users thay đổi đối với DataSet, ta muốn ghi những thay đổi đó trở lại database. Ta có thể làm điều này bằng việc điều chỉnh các đối tượng câu lệnh SQL cho các công việc inserting, deleting, and updating. Một lựa chọn khác, ta có thể sử dụng các stored procedures. Cả hai giải pháp phụ thuộc vào sự am hiểu về SQL. Rất may là, có một cách đơn giản hơn, ta có thể sử dụng *command builders* để tạo các câu lệnh này. Nó chỉ có một dòng trong các dòng sau:

```
' Declare a SqlDataAdapter object...
```

```
Dim objDataAdapter As New SqlDataAdapter()
' Assign a new SqlCommand to the SelectCommand property
objDataAdapter.SelectCommand = New SqlCommand()
' Set the SelectCommand properties...
objDataAdapter.SelectCommand.Connection = objConnection
objDataAdapter.SelectCommand.CommandText = "usp_select_DSSV"
objDataAdapter.SelectCommand.CommandType =
CommandType.StoredProcedure
' automatically create update/delete/insert commands
Dim objCommandBuilder As SqlCommandBuilder = New _
SqlCommandBuilder(objDataAdapter)
```

Như vậy, ta có thể sử dụng SqlDataAdapter để ghi những thay đổi dữ liệu trở lại database. Ta tìm hiểu kỹ hơn về vấn đề này trong phần ví dụ minh họa. Để hiểu, ta xét các phương thức mà lấy dữ liệu từ database đẩy vào DataSet:Phương thức Fill .

**** Phương thức Fill***

Ta dùng phương thức Fill để dữ liệu đẩy dữ liệu mà đối tượng SqlDataAdapter sử dụng SelectCommand của nó lấy từ kho lưu trữ dữ liệu vào đối tượng DataSet. Tuy nhiên, trước khi làm điều đó, ta phải khởi tạo đối tượng DataSet. Để sử dụng đối tượng DataSet trong project của ta, ta phải add một tham chiếu tới System.Xml.

```
' Declare a SqlDataAdapter object...
Dim objDataAdapter As New SqlDataAdapter()
' Assign a new SqlCommand to the SelectCommand property
objDataAdapter.SelectCommand = New SqlCommand()
 ' Set the SelectCommand properties...
objDataAdapter.SelectCommand.Connection = objConnection
objDataAdapter.SelectCommand.CommandText = "usp_select_DSSV"
objDataAdapter.SelectCommand.CommandType =
CommandType.StoredProcedure
Dim objDataSet as DataSet = New DataSet()
```

Ta có hai đối tượng DataSet và SqlDataAdapter, có thể điền dữ liệu vào DataSet. Phương thức Fill có nhiều phiên bản, nhưng ta sẽ chỉ xét thảo luận một phiên bản hay được sử dụng nhiều nhất. Cú pháp của phương thức Fill được cho như sau:

```
SqlDataAdapter.Fill(DataSet, string)
```

Đối số *DataSet* chỉ định một đối tượng DataSet hợp lệ mà dữ liệu được lưu trữ ở trong đó. Đối số *string* gán với tên mà ta muốn bảng có trong DataSet. Chú ý rằng, một DataSet có thể chứa nhiều bảng. Ta có thể sử dụng bất cứ tên nào ta muốn, nhưng thường tốt nhất là sử dụng tên của các bảng nơi mà dữ liệu trong database đến. Nó giúp cho văn bản code của ta và làm cho code dễ bảo trì.

Đoạn code sau gọi phương thức Fill. Chuỗi "authors" được chỉ định là đối số string. Nó là tên ta muốn sử dụng khi đang thao tác trong các phiên bản nhớ của bảng này; nó cũng là tên của bảng trong data source.

```
' Declare a SqlDataAdapter object...
Dim objDataAdapter As New SqlDataAdapter()

'Create an instance of a new select command object
objDataAdapter.SelectCommand = New SqlCommand()

 ' Set the SelectCommand properties...
objDataAdapter.SelectCommand.Connection = objConnection
objDataAdapter.SelectCommand.CommandText = "usp_select_DSSV"
objDataAdapter.SelectCommand.CommandType =
CommandType.StoredProcedure
Dim objDataSet as DataSet = New DataSet()

 ' Fill the DataSet object with data...
objDataAdapter.Fill(objDataSet, "DSSV")
```

Phương thức Fill sử dụng thuộc tính SelectCommand.Connection để connect tới database. Nếu connection đã được mở, data adapter sẽ sử dụng nó để thực hiện SelectCommand. Và nếu như connection đã đóng thì data adapter sẽ mở nó, thực hiện SelectCommand, và sau đó đóng trở trở lại. Bây giờ ta đã có dữ liệu trong bộ nhớ, và có thể bắt đầu thao tác nó không phụ thuộc vào data source.

Chú ý: Lớp DataSet không có Sql tại vị trí bắt dầu tên lớp của nó. Bởi vì DataSet không nằm trong namespace System.Data.SqlClient, nó nằm trong lớp cha namespace System.Data. Các lớp trong tên miền này có liên quan căn bản với thao các dữ liệu trong bộ nhớ.

Một lần nữa, ta đã có dữ liệu được load vào DataSet, tính chất quan trọng của data source không còn nữa (trừ khi ta muốn ghi trở lại). Ta hãy xét hai trong các lớp trong namespace: DataSet và DataView.

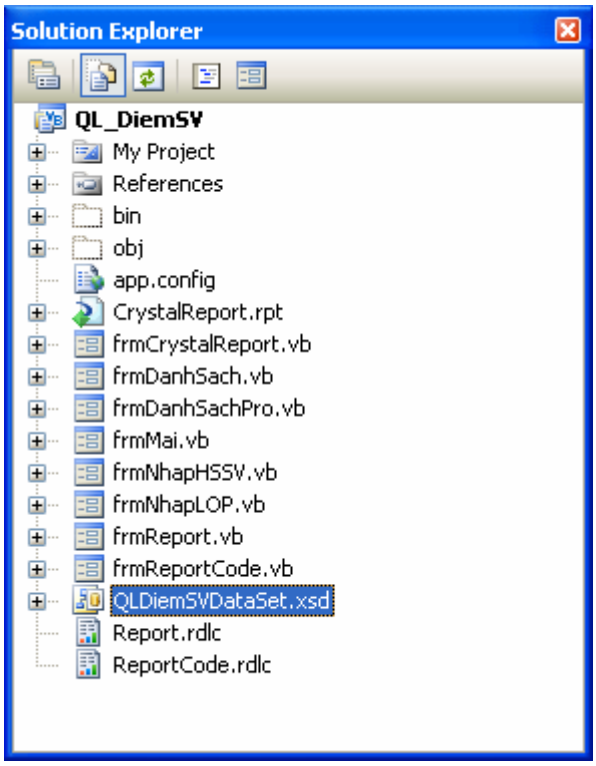
5.2.4. Class DataSet

Lớp DataSet được dùng để lưu trữ dữ liệu được lấy từ một kho dữ liệu và lưu trữ dữ liệu trong bộ nhớ trên client. Đối tượng DataSet chứa một tập các tables, relationships, và các ràng buộc constraints thích hợp với dữ liệu đọc từ kho dữ liệu. Nó hoạt động như một bộ máy database gọn nhẹ, cho phép ta lưu trữ tables, edit data, và chạy các truy vấn dựa vào việc sử dụng đối tượng DataView.

Dữ liệu trong DataSet không được connect từ kho dữ liệu, và ta có thể làm việc với dữ liệu độc lập với kho dữ liệu. Ta có thể thao tác dữ liệu trong đối tượng DataSet bằng việc adding, updating, và deleting the records. Ta có thể áp các thay đổi này quay trở lại kho dữ liệu gốc sau khi sử dụng data adapter.

Dữ liệu trong đối tượng DataSet được duy trì trong Extensible Markup Language (XML), nghĩa là ta có thể save DataSet như là một file hoặc đơn giản chuyển nó trên network. XML được bảo vệ từ ta, những người phát triển, và chúng ta không bao giờ nên edit trực tiếp XML. Tất cả các thao tác editing XML được thực hiện thông qua các thuộc tính và các phương thức của lớp DataSet. Nhiều nhà phát triển thích sử dụng XML và lựa chọn để thao tác XML biểu diễn trực tiếp DataSet, nhưng điều này không cần thiết.

Giống như bất cứ bản document XML nào, DataSet có một *schema* (một file cấu trúc của dữ liệu trên một hoặc nhiều XML files). Khi ta sinh ra một kiểu dataset dùng wizard, XML Schema Definition (XSD) file được add vào Solution Explorer, như hình 5.6.



Hình 5.6. File XML Schema Definition

File này là một XML schema cho dữ liệu QLDiemSVDataSet chứa nó. Từ đây, Visual Studio .NET tạo ra một lớp kế thừa từ DataSet và sử dụng schema cụ thể này. Một lược đồ DataSet chứa các thông tin về tables, relationships, and constraints được lưu trữ trong DataSet. Thêm nữa, nó được bảo vệ từ phía ta, và ta không cần biết XML làm việc với DataSet.

5.2.5. DataView

Lớp DataView được sử dụng tiêu biểu cho việc sorting, filtering, searching, editing, và navigating dữ liệu từ DataSet. Một DataView là *bindable*, nghĩa là nó có thể bị buộc vào các điều khiển giống như cách DataSet bị buộc vào các điều khiển. Hơn nữa, ta sẽ tìm hiểu nhiều hơn về cách buộc dữ liệu trong code trong phần tiếp theo của chương.

Một DataSet có thể chứa một số các đối tượng *DataTable*; khi ta sử dụng phương thức Fill của lớp SqlDataAdapter để add dữ liệu vào DataSet, như vậy ta đã thực sự đang tạo một đối tượng DataTable bên trong DataSet này. DataView cung cấp view thông dụng của DataTable; ta có thể sắp xếp và lọc dữ liệu, như ta thực hiện trên truy vấn SQL.

Ta có thể tạo một DataView từ dữ liệu được chứa trong DataTable mà đang chỉ chứa dữ liệu mà ta muốn hiển thị. Ví dụ, nếu dữ liệu trong DataTable chứa tất cả các Sinh viên được sắp xếp bởi *Tên* và *Họ đệm*, ta có thể tạo một DataView chứa tất cả *Sinh viên* được sắp xếp bởi Tên và sau đó là Họ đệm. Hoặc, nếu muốn, ta có thể tạo một DataView chỉ chứa một vài sinh viên chứa tên nào đó.

Mặc dù ta có thể view dữ liệu trong DataView trong nhiều cách khác nhau từ DataTable cơ bản, nó vẫn là cùng dữ liệu. Các thay đổi với DataView ảnh hưởng tới DataTable cơ bản một cách tự động và các thay đổi với DataTable cơ bản tự động ảnh hưởng bất kỳ đối tượng DataView mà đang hiển thị DataTable đó.

Kiến trúc cho lớp DataView khởi tạo một thể hiện mới của lớp DataView và lấy DataTable là một đối số. Đoạn code sau khai báo một đối tượng DataView và khởi tạo nó đang sử dụng bảng HosoSV từ DataSet có tên là objDataSet. Chú ý rằng code đó truy cập vào tập hợp Tables của đối tượng DataSet, bằng việc chỉ định thuộctính Tables và tên bảng:

```
' Set the DataView object to the DataSet object...
Dim objDataView = New DataView(objDataSet.Tables("HosoSV"))
```

**** Thuộc tính Sort***

Khi *DataView* đã được khởi tạo và hiển thị dữ liệu, ta có thể thay đổi view của dữ liệu đó. Ví dụ, giả sử ta muốn sắp xếp dữ liệu khác với thứ tự trong DataSet. Để sắp xếp dữ liệu trong DataView, ta thiết lập thuộc tính Sort và chỉ định cột hoặc các cột mà ta muốn sắp xếp. Đoạn code sau sẽ sắp xếp dữ liệu trong DataView theo Tên và sau đó theo Họ đệm của danh sách Sinh viên:

```
objDataView.Sort = "TenSV, Hodem"
```

Chú ý: Thuộc tính Sort giống như cú pháp của mệnh đề ORDER BY trong câu lệnh SQL. Như trong SQL mệnh đề ORDER BY, thao tác sắp xếp trên DataView thường sắp xếp theo thứ tự mặc định là tăng dần. Nếu ta muốn thay đổi theo thứ tự sắp xếp giảm, ta cần phải có chỉ định từ khóa DESC:

```
objDataView.Sort = "TenSV, Hodem DESC"
```

**** Thuộc tính RowFilter***

Khi đã khởi tạo DataView, ta có thể lọc các dòng dữ liệu mà chúng sẽ chứa. Nó tương tự như chỉ định trong mệnh đề WHERE của câu lệnh SQL SELECT; chỉ những hàng thỏa mãn điều kiện sẽ được giữ lại trên view. Dữ liệu cơ bản không bị ảnh hưởng. Thuộc tính RowFilter chỉ định một điều kiện lọc mà có thể áp dụng được trên DataView. Cú pháp này cũng tương tự như mệnh đề SQL WHERE. Nó chứa ít nhất tên một cột, theo sau là một toán tử và giá trị. Nếu giá trị là string thì nó phải được bao trong cặp dấu 'nháy đơn', ví dụ ta có đoạn code sau:

```
' Set the DataView object to the DataSet object...
objDataView = New DataView(objDataSet.Tables("HosoSV"))
objDataView.RowFilter = "TenSV = 'Hoa'"
```

Nếu bạn muốn lấy lại các ròng trừ các dòng có Tên là ='Hoa'

```
' Set the DataView object to the DataSet object...
objDataView = New DataView(objDataSet.Tables("authors"))
objDataView.RowFilter = "TenSV <> 'Hoa'"
```

Ta cũng có thể chỉ định điều kiện lọc phức tạp hơn như trong SQL. Ví dụ, sử dụng toán tử AND:

```
objDataView.RowFilter = "TenSV <> 'Hoa' AND Hodem LIKE 'N*'"
```

**** Phương thức Find***

Nếu ta muốn tìm kiếm cho một hàng dữ liệu trong DataView, ta gọi phương thức Find. Phương thức Find tìm kiếm dữ liệu trong cột khóa sắp xếp của DataView. Do đó, trước khi gọi phương thức Find, ta trước hết hãy sắp xếp DataView trên cột chứa dữ liệu mà ta muốn tìm kiếm. Cột trong DataView đã được sắp xếp trở thành cột khóa sắp xếp trong đối tượng DataView object.

Ví dụ, giả sử ta muốn tìm kiếm Sinh viên có TenSV là 'Anh'. Ta cần sắp xếp dữ liệu trong DataView theo TenSV và thiết lập cột này thành cột khóa sắp xếp trong DataView, và sau đó gọi phương thức Find, như đoạn code sau:

```
Dim intPosition as Integer
objDataView.Sort = "TenSV"
intPosition = objDataView.Find("Ann")
```

Phương thức Find tìm kiếm và trả về vị trí của bản ghi trong DataView. Trong trường hợp ngược lại, DataView trả về giá trị null nếu không tìm thấy. Nếu phương thức Find tìm thấy một bản khớp thì nó sẽ dừng tìm kiếm và trả về vị trí của bản ghi đầu tiên tìm thấy. Nếu ta muốn có nhiều hơn một bản ghi trong data store, ta phải lọc dữ liệu trong DataView, dữ liệu của ta sẽ được hiển thị.

Phương thức Find không phân biệt dạng chữ, nghĩa là khi tìm Sinh viên có tên là 'Hoa', ta có thể nhập là 'hoa'.

Phương thức Find tìm chính xác từng cụm từ, nghĩa là ta phải nhập toàn bộ cụm từ đó. Ví dụ, giả sử ta muốn tìm kiếm Sinh viên có Họ đệm là 'Đinh Thanh' thì ta phải nhập đầy đủ cụm đó.

```
objDataView.Sort = "Hodem"
intPosition = objDataView.Find("Đinh Thanh")
```

Ta thấy DataView có thể được sắp xếp trên nhiều hơn một cột tại cùng một thời điểm. Nếu ta muốn sắp xếp nhiều hơn một cột, ta cần cung cấp một mảng các giá trị để phương thức Find thay cho chỉ một giá trị đơn. Ví dụ, ta muốn tìm kiếm 'Đinh Thanh Hoa' xuất hiện trong DataView:

```
Dim intPosition As Integer
Dim arrValues(1) As Object
objDataView.Sort = "Hodem, TenSV"
' Find the author named "Đinh Thanh Hoa".
arrValues(0)= "Đinh Thanh"
arrValues(1) = "Hoa"
intPosition = objDataView.Find(arrValues)
```

5.3. Ví dụ minh họa

Mục đích của phần này đưa ra cho sinh viên các kỹ thuật lập trình quản trị CSDL cơ bản như là cập nhật dữ liệu, hiển thị dữ liệu, tìm kiếm dữ liệu, sắp xếp dữ liệu, xây dựng các báo cáo thống kê, .v.v... Tác giả trình bày chi tiết từng bước thiết kế chúng và thông qua đó sinh viên có thể tổng hợp các kỹ năng đó để xây dựng cho mình một ứng dụng quản lý CSDL hoàn chỉnh.

Khi lập trình, ta thường dùng kết hợp giữa wizards và coding để lập trình một cách nhanh chóng và dễ dàng. Các components có thể tạo bằng thao tác

kéo thả có thể được sinh trong code đúng như các đối tượng được tạo trong code. Trong ví dụ minh họa, đề hiểu được bản chất ta tập chung trên code.

5.3.1. CSDL trong ví dụ minh họa

Trong phần này, ta thiết kế một CSDL QL_DiemSV rất đơn giản nhằm mục đích minh họa cho các kỹ năng lập trình CSDL, gồm 4 quan hệ sau:

- Bảng Hồ sơ sinh viên (HOSOSV)

Design Table 'HOSOSV' in 'QLDiemSV' on 'THUHUONG'

Column Name	Data Type	Length	Allow Nulls
MaSV	char	10	
HoDem	nvarchar	50	
TenSV	nvarchar	50	
NgaySinh	datetime	8	✓
MaLop	char	10	

- Bảng danh mục Lớp (LOP)

Design Table 'LOP' in 'QLDiemSV' on 'THUHUONG'

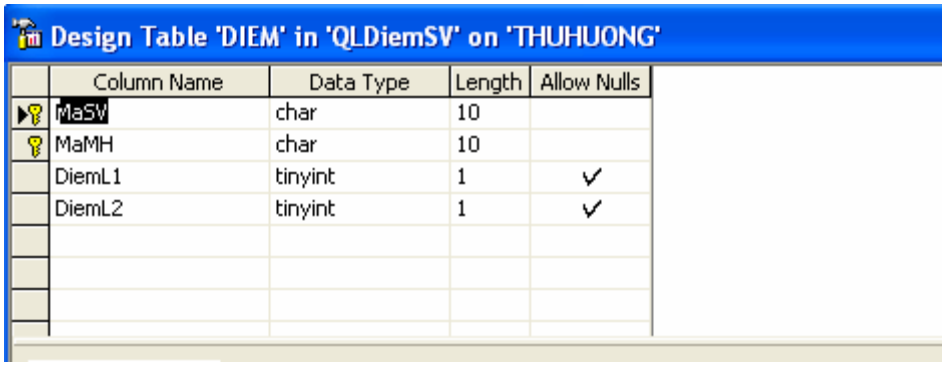
Column Name	Data Type	Length	Allow Nulls
MaLop	char	10	
TenLop	nvarchar	50	
Khoa	char	4	✓

- Bảng danh mục môn học (MONHOC)

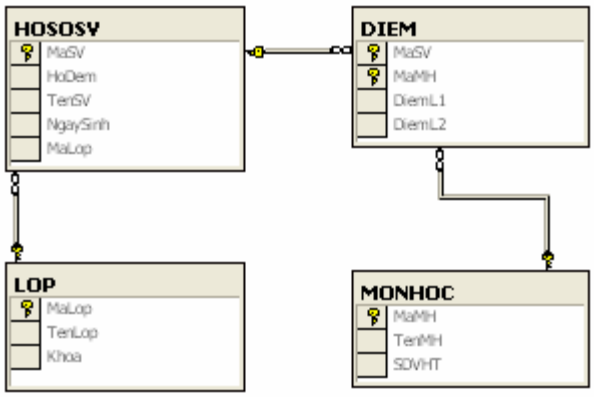
Design Table 'MONHOC' in 'QLDiemSV' on 'THUHUONG'

Column Name	Data Type	Length	Allow Nulls
MaMH	char	10	
TenMH	nvarchar	50	
SDVHT	int	4	

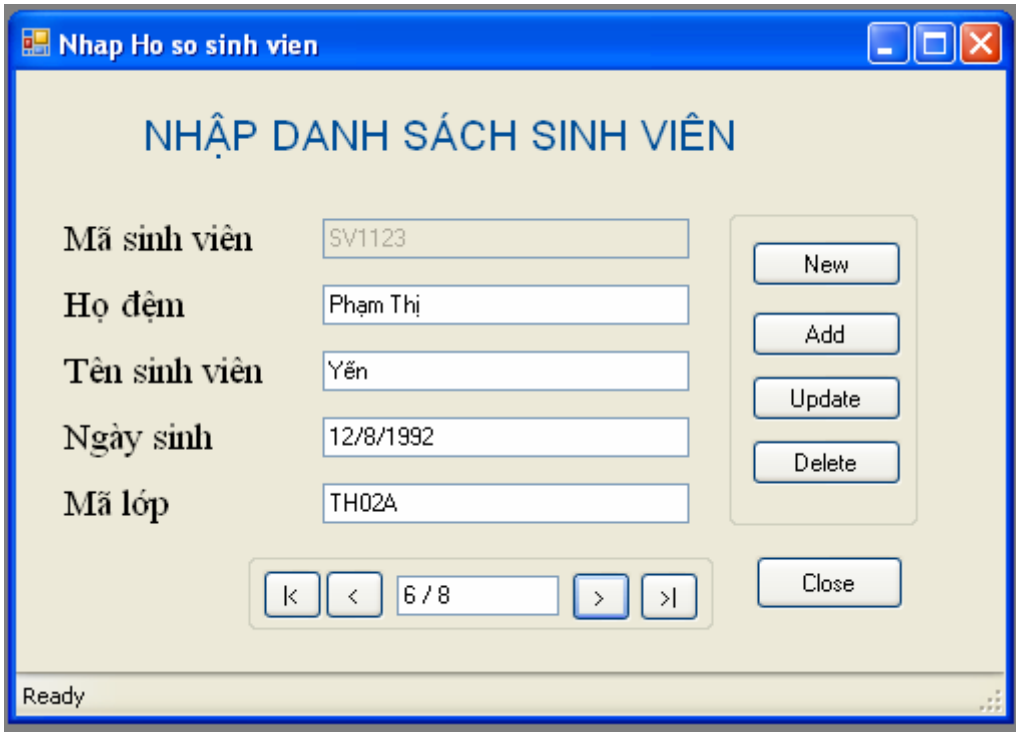
- Bảng điểm (DIEM)



- Quan hệ giữa các bảng:



5.3.2. Xây dựng Form nhập DSSinhVien



Ta thực hiện xây dựng from trên thông qua các đối tượng sau.

1. Tạo một Project Windows Application tên là QL_DiemSV.

2. Tạo Form Nhập hồ sơ sinh viên (frmNhapHSSV) như hình có các đối tượng sau:

Object	***Property***	***Setting***
Form	Name Text	frmNhapHSSV Nhap Ho so sinh vien
Label	Name Text	Label1 NHẬP DANH SÁCH SINH VIÊN
Label	Name Text	lblMaSV Mã sinh viên
Label	Name Text	lblHodem Họ đệm
Label	Name Text	lblTenSV Tên sinh viên
Label	Name Text	lblMaLop Mã lớp
Textbox	Name	txtMaSV
Textbox	Name	txtHodem
Textbox	Name	txtTenSV
Textbox	Name	txtMaLop
Combo	Name	cboMaLop
Button	Name Text	btnNew New
Button	Name Text	btnAdd Add
Button	Name Text	btnUpdate Update
Button	Name Text	btnDelete Delete

Textbox	Name	txtRecordPosition
Button	Name Text	btnMoveFirst |<
Button	Name Text	btnMovePrevious <
Button	Name Text	btnMoveNext >
Button	Name Text	btnMoveLast >|
StatusStrip	Name Text	StatusStrip1 StatusStrip1
ToolStripStatusLabel	Name Text	ToolStripStatusLabel1 ToolStripStatusLabel1
Button	Name Text	btnClose Close

Sau khi thiết kế các đối tượng trên ta được một form như hình,

3. Import các tên miền cần thiết. Mở code editor và chèn đoạn code sau.

```
' Import Data and SqlClient namespaces...
Imports System.Data

Imports System.Data.SqlClient
```

4. Tiếp theo khai báo các đối tượng global trong phạm vi của form này.

```
' Import Data and SqlClient namespaces...
Imports System.Data
Imports System.Data.SqlClient

Public Class frmNhapHSSV
#Region "Khai bao cac ket noi"
    Dim objConnection As New SqlConnection _
       ("server=THUHUONG;database=QLDiemSV;user id=sa;password=pass2008")
    Dim objDataAdapter As New SqlDataAdapter( _
        "SELECT MaSV, HoDem, TenSV, NgaySinh, MaLop " & _
        "FROM HOSOSV ", objConnection)
    Dim objDataSet As DataSet
    Dim objDataView As DataView
    Dim objCurrencyManager As CurrencyManager

#End Region
```

Chú ý: Khi khai báo kết nối cần phải đảm bảo đúng tên server, user id và password.

5. Thủ tục đầu tiên ta sẽ tạo đó là thủ tục FillDataSetAndView. Thủ tục có thể coi là thủ tục khởi tạo các đối tượng.

```
Private Sub FillDataSetAndView()
    ' Initialize a new instance of the DataSet object...
    objDataSet = New DataSet()

    ' Fill the DataSet object with data...
    objDataAdapter.Fill(objDataSet, "Hoso")

    ' Set the DataView object to the DataSet object...
    objDataView = New DataView(objDataSet.Tables("Hoso"))

    ' Set our CurrencyManager object to the DataView object...
    objCurrencyManager = CType(Me.BindingContext(objDataView),
CurrencyManager)
  End Sub
```

6. Thủ tục tiếp theo sẽ thực hiện kết nối các điều khiển trên form với đối tượng Dataview.

```
Private Sub BindFields()


    ' Clear any previous bindings...
    txtMaSV.DataBindings.Clear()
    txtHodem.DataBindings.Clear()
    txtTenSV.DataBindings.Clear()
    txtNgaySinh.DataBindings.Clear()
    txtMaLop.DataBindings.Clear()

    ' Add new bindings to the DataView object...
    txtMaSV.DataBindings.Add("Text", objDataView, "MaSV")
    txtHodem.DataBindings.Add("Text", objDataView, "HoDem")
    txtTenSV.DataBindings.Add("Text", objDataView, "TenSV")
    txtNgaySinh.DataBindings.Add("Text", objDataView, "Ngaysinh")
    txtMaLop.DataBindings.Add("Text", objDataView, "MaLop")

    ' Display a ready status...
    ToolStripStatusLabel1.Text = "Ready"
  End Sub
```

7. Thủ tục tiếp theo là sẽ hiển thị vị trí của bản ghi hiện thời trên form.

```
Private Sub ShowPosition()
    ' Display the current position and the number of records
    txtRecordPosition.Text = objCurrencyManager.Position + 1 & _
    " / " & objCurrencyManager.Count()

End Sub
```

8. Ta đã xây dựng các thủ tục. Ta sẽ gọi các thủ tục đó. Double-click Form Designer, chọn sự kiện Load form sau đó add đoạn code sau.

```
 Private Sub frmNhapHSSV_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load

  ' Fill the DataSet and bind the fields...
```

```
        Me.txtMaSV.Enabled = False
        FillDataSetAndView()
        BindFields()

       ' Show the current record position...
        ShowPosition()

    End Sub
```

Bây giờ ta sẽ chạy thử form: Debug\Start Debugging (F5)

9. Tiếp theo ta sẽ viết code cho các navigation buttons. Để thực hiện duyệt các bản ghi trên form.

```
    Private Sub btnMoveFirst_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnMoveFirst.Click

        ' Set the record position to the first record...
        objCurrencyManager.Position = 0

        ' Show the current record position...
        ShowPosition()
    End Sub

    Private Sub btnMovePrevious_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnMovePrevious.Click
        ' Move to the previous record...
        objCurrencyManager.Position -= 1
        ' Show the current record position...
        ShowPosition()
    End Sub

    Private Sub btnMoveNext_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnMoveNext.Click
        ' Move to the next record...
        objCurrencyManager.Position += 1

        ' Show the current record position...
        ShowPosition()
    End Sub

    Private Sub btnLast_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnLast.Click
        ' Set the record position to the last record...
        objCurrencyManager.Position = objCurrencyManager.Count - 1

        ' Show the current record position...
        ShowPosition()

        End Sub
```

10. Thêm các bản ghi.

- Trước hết ta đưa đoạn code sau vào thủ tục của nút New.

```
    Private Sub btnNew_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnNew.Click

        txtMaSV.Text = ""
        txtHodem.Text = ""
```

```
        txtTenSV.Text = ""
        txtNgaySinh.Text = ""
        txtMaLop.Text = ""

        Me.txtMaSV.Enabled = True
        Me.cboMaLop.Visible = True
        Me.txtMaLop.Visible = False

        ' Declare local variables and objects...
        Dim objCommand As SqlCommand = New SqlCommand()
        Dim objDataReader As SqlDataReader

        ' Open the connection, execute the command
        objConnection.Open()
        ' Set the SqlCommand object properties...
        objCommand.Connection = objConnection
        objCommand.CommandText = "Select MaLop, TenLop From Lop"

        ' Execute the SqlCommand object to insert the new data...
        objDataReader = objCommand.ExecuteReader
        cboMaLop.Items.Clear()

        Do While (objDataReader.Read())
            cboMaLop.Items.Add(objDataReader.Item(0))
        Loop

        ' Close the connection...
        objConnection.Close()
    End Sub
```

Đối với thủ tục này, ta sẽ cho các đối tượng Text box về rỗng để nhập dữ liệu. Riêng Text box txtMaLop sẽ được ẩn đi và thay vào đó là Combo box cboMaLop. Do mối quan hệ giữa hai quan hệ HOSOSV và LOP, cboMaLop sẽ chứa MaLop các lớp có trong danh mục Lop.

Trong phần này, ta có sử dụng đối tượng DataReader của lớp SqlDataReader. Khai báo đối tượng:

```
' Declare local variables and objects...
Dim objCommand As SqlCommand = New SqlCommand()
Dim objDataReader As SqlDataReader
```

Thực hiện đối tượng Command và lưu kết quả để đọc vào đối tượng objDataReader. Và truy xuất dữ liệu đó.

```
' Execute the SqlCommand object to insert the new data...
objDataReader = objCommand.ExecuteReader
cboMaLop.Items.Clear()

Do While (objDataReader.Read())
    cboMaLop.Items.Add(objDataReader.Item(0))
Loop
```

- Chèn dữ liệu vào CSDL. Đưa đoạn code sau vào sự kiện click của nút lệnh Add.

```
Private Sub btnAdd_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnAdd.Click
        ' Declare local variables and objects...
        Dim intPosition As Integer
        Dim objCommand As SqlCommand = New SqlCommand()

        ' Save the current record position...
        intPosition = objCurrencyManager.Position
        ' Open the connection, execute the command
        objConnection.Open()
        ' Set the SqlCommand object properties...
        objCommand.Connection = objConnection
        objCommand.CommandText = "INSERT INTO HOSOSV " & _
            "(MaSV, HoDem, TenSV, NgaySinh, MaLop) " & _
            "VALUES(@MaSV,@Hodem,@TenSV,@NgaySinh,@MaLop)"
        ' Add parameters for the placeholders in the SQL in the
        ' CommandText property...
        objCommand.Parameters.AddWithValue("@MaSV", txtMaSV.Text)
        objCommand.Parameters.AddWithValue("@Hodem", txtHodem.Text)
        objCommand.Parameters.AddWithValue("@TenSV", txtTenSV.Text)
        objCommand.Parameters.AddWithValue("@NgaySinh",_
                                    txtNgaySinh.Text).DbType = DbType.Date
        objCommand.Parameters.AddWithValue("@MaLop", cboMaLop.Text)

        ' Execute the SqlCommand object to insert the new data...
        Try
            objCommand.ExecuteNonQuery()
        Catch SqlExceptionErr As SqlException
            MessageBox.Show(SqlExceptionErr.Message)
        End Try

        ' Close the connection...
        objConnection.Close()

        ' Fill the dataset and bind the fields...
        FillDataSetAndView()
        BindFields()
        ' Set the record position to the one that you saved...
        objCurrencyManager.Position = intPosition
        ' Show the current record position...
        ShowPosition()
        ' Display a message that the record was added...
        ToolStripStatusLabel1.Text = "Record Added"
        Me.txtMaSV.Enabled = False
        Me.cboMaLop.Visible = False
        Me.txtMaLop.Visible = True

    End Sub
```

11. Thực hiện Update các bản ghi.

```
Private Sub btnUpdate_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnUpdate.Click
        ' Declare local variables and objects...
        Dim intPosition As Integer
        Dim objCommand As SqlCommand = New SqlCommand()

        ' Save the current record position...
```

```
        intPosition = objCurrencyManager.Position

        ' Set the SqlCommand object properties...
        objCommand.Connection = objConnection
        objCommand.CommandText = "UPDATE HOSOSV " & _
                "SET  HoDem = @HoDem, TenSV=@TenSV, NgaySinh=@NgaySinh,
MaLop=@MaLop Where MaSV = @MaSV"
        objCommand.CommandType = CommandType.Text
        ' Add parameters for the placeholders in the SQL in the
        ' CommandText property...
        objCommand.Parameters.AddWithValue("@Hodem", txtHodem.Text)
        objCommand.Parameters.AddWithValue("@TenSV", txtTenSV.Text)
        objCommand.Parameters.AddWithValue("@NgaySinh",
txtNgaySinh.Text).DbType = DbType.Date
        objCommand.Parameters.AddWithValue("@MaLop", txtMaLop.Text)
        objCommand.Parameters.AddWithValue_
("@MaSV", BindingContext(objDataView).Current("MaSV"))

        ' Open the connection...
        objConnection.Open()
        ' Execute the SqlCommand object to update the data...
        objCommand.ExecuteNonQuery()
        ' Close the connection...
        objConnection.Close()
        ' Fill the DataSet and bind the fields...
        FillDataSetAndView()
        BindFields()
        ' Set the record position to the one that you saved...
        objCurrencyManager.Position = intPosition
        ' Show the current record position...
        ShowPosition()

        ' Display a message that the record was updated...
        ToolStripStatusLabel1.Text = "Record Updated"

    End Sub
```

12. Thực hiện Delete các bản ghi.

```
Private Sub btnDelete_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnDelete.Click
        ' Declare local variables and objects...
        Dim intPosition As Integer
        Dim objCommand As SqlCommand = New SqlCommand()

        ' Save the current record position - 1 for the one to be
        ' deleted...
        intPosition = Me.BindingContext(objDataView).Position - 1

        ' If the position is less than 0 set it to 0...
        If intPosition < 0 Then
            intPosition = 0
        End If

        ' Set the Command object properties...
        objCommand.Connection = objConnection
        objCommand.CommandText = "DELETE FROM HOSOSV " & _
                "WHERE MaSV = @MaSV;"
        ' Parameter for the MaSv field...
        objCommand.Parameters.AddWithValue _
                        ("@MaSV",
BindingContext(objDataView).Current("MaSV"))
```

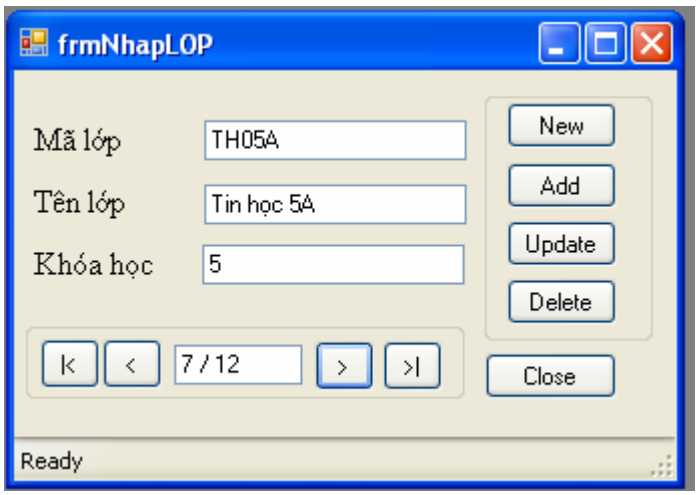
```
        ' Open the database connection...
        objConnection.Open()
        ' Execute the SqlCommand object to update the data...
        objCommand.ExecuteNonQuery()

        ' Close the connection...
        objConnection.Close()

        ' Fill the DataSet and bind the fields...
        FillDataSetAndView()
        BindFields()

        ' Set the record position to the one that you saved...
        Me.BindingContext(objDataView).Position = intPosition

        ' Show the current record position...
        ShowPosition()

        ' Display a message that the record was deleted...
        ToolStripStatusLabel1.Text = "Record Deleted"

    End Sub
```

13. Viết code cho nút Close

```
Private Sub btnClose_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnClose.Click
        Me.Close()
End Sub
```

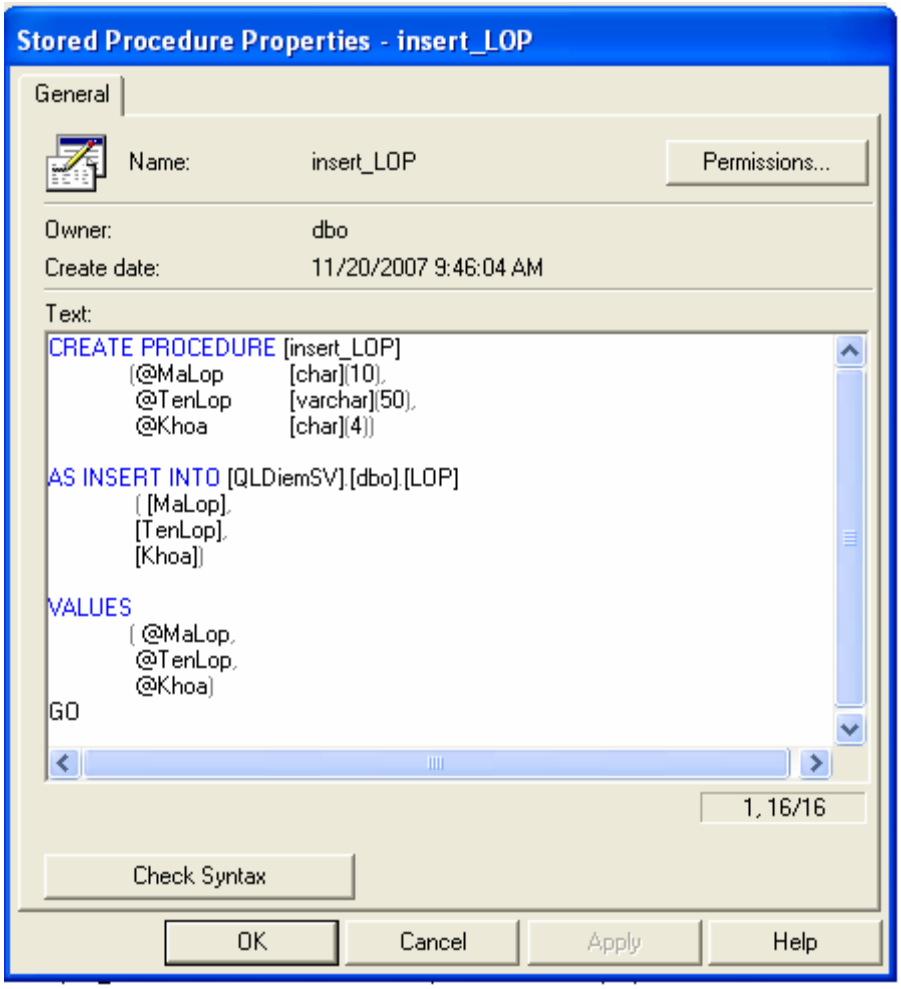
14. Ta có thể biên dịch và chạy ứng dụng.

5.3.3. Xây dựng Form nhập DSLop



Ta thực hiện tương tự như Form trên. Nhưng nút Add ta sẽ minh họa cho việc sử dụng Stored Procedure.

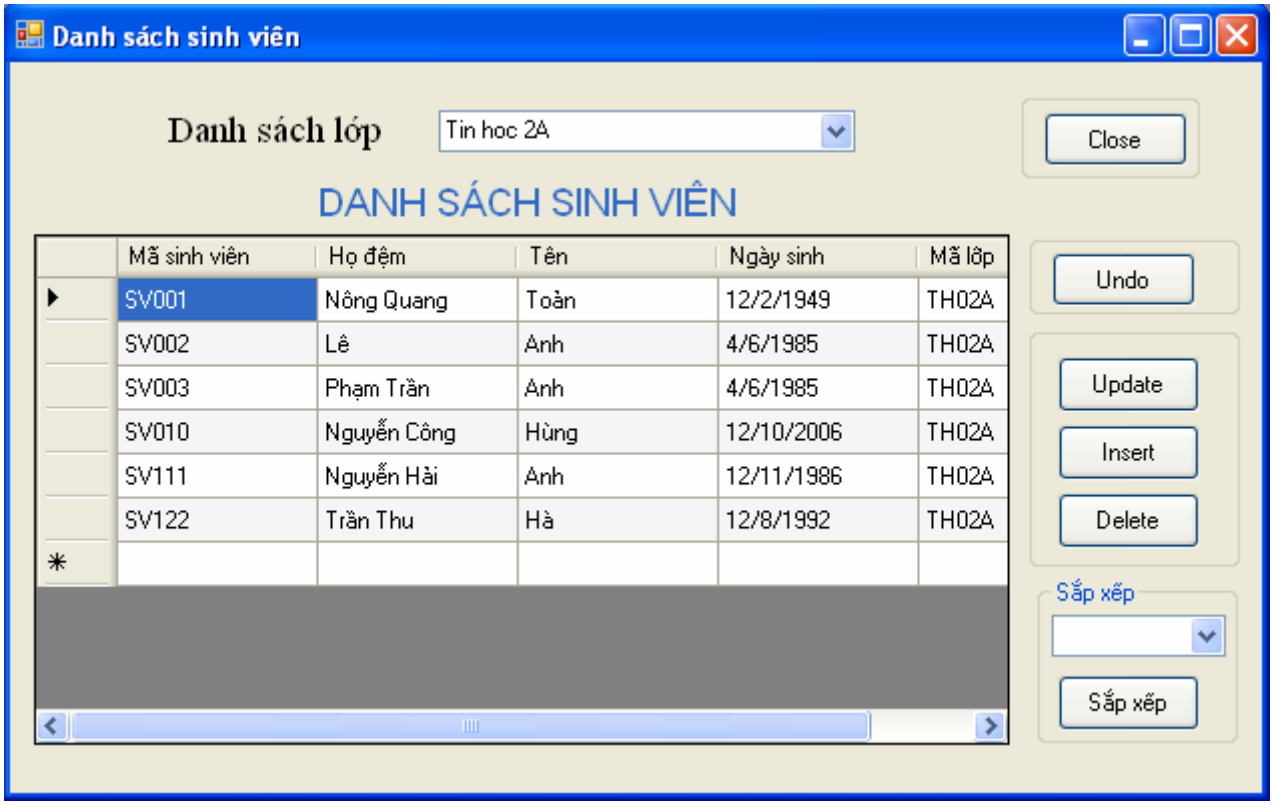
- Trong SQL server ta xây dựng StoredProcedure như sau



- Viết code cho nút Add này như sau:

```
Private Sub btnAdd_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnAdd.Click

        ' Declare local variables and objects...
        Dim intPosition As Integer
        Dim objCommand As SqlCommand = New SqlCommand()

        ' Save the current record position...
        intPosition = objCurrencyManager.Position

        ' Open the connection, execute the command
        objConnection.Open()
        ' Set the SqlCommand object properties...
        objCommand.Connection = objConnection
        objCommand.CommandText = "insert_Lop"
        objCommand.CommandType = CommandType.StoredProcedure

        ' Add parameters for the placeholders in the SQL in the
        ' CommandText property...
        objCommand.Parameters.AddWithValue("@MaLop", txtMaLop.Text)
        objCommand.Parameters.AddWithValue("@TenLop", txtTenLop.Text)
        objCommand.Parameters.AddWithValue("@Khoa", txtKhoa.Text)

          ' Execute the SqlCommand object to insert the new data...
```

```
        Try
            objCommand.ExecuteNonQuery()
        Catch SqlExceptionErr As SqlException
            MessageBox.Show(SqlExceptionErr.Message)
        End Try

        ' Close the connection...
        objConnection.Close()

        ' Fill the dataset and bind the fields...
        FillDataSetAndView()
        BindFields()

        ' Set the record position to the one that you saved...
        objCurrencyManager.Position = intPosition

        ' Show the current record position...
        ShowPosition()

        ' Display a message that the record was added...
        ToolStripStatusLabel1.Text = "Record Added"

    End Sub
```



5.3.4. Xây dựng Form hiển thị danh sách sinh viên.

Mục đích của Form thực hiện hiển thị danh sách sinh viên theo từng lớp. Ngoài ra ra cho phép thực hiện các thao tác:

- Cập nhật dữ liệu trên Grid.
- Sửa đổi dữ liệu trên Grid.
- Xóa dữ liệu trên Grid.
- Sắp xếp dữ liệu.

1. Xây dựng Form như hình trên gồm các control sau:

Object	***Property***	***Setting***
Form	Name Text	frmDanhSach Danh sách sinh viên
Label	Name Text	lblDSL Danh sách lớp
Label	Name Text	lblDSSV DANH SÁCH SINH VIÊN
DataGridView	Name	dgDanhsach
Button	Name Text	btnClose Close
Button	Name Text	btnInsert Insert
Button	Name Text	btnUpdate Update
Button	Name Text	btnDelete Delete
Button	Name Text	btnUndo Undo
Button	Name Text	btnSort Sắp xếp
ComboBox	Name	cboMaLop
ComboBox	Name Items	cboSX Tên; Họ tên; Mã sinh viên

2. Import các tên miền cần thiết. Mở code editor và chèn đoạn code sau.

```
' Import Data and SqlClient namespaces...
Imports System.Data
Imports System.Data.SqlClient
```

3. Tiếp theo khai báo các đối tượng global trong phạm vi của form này.

```
#Region "Khai bao cac ket noi"
    Dim objConnection As New SqlConnection _
            ("server=THUHUONG;database=QLDiemSV;user
id=sa;password=12102006")
    Dim objDataAdapter As New SqlDataAdapter()
    Dim objDataSet As DataSet
    Dim objDataView As DataView
#End Region
```

Chú ý: Khi khai báo kết nối cần phải đảm bảo đúng tên server, user id và password.

4. Thủ tục đầu tiên ta sẽ tạo đó là thủ tục FillDataSetAndView. Thủ tục có thể coi là thủ tục khởi tạo các đối tượng.

```
    Private Sub FillDataSetAndView()

        ' Initialize a new instance of the DataSet object...
        objDataSet = New DataSet()
        objDataAdapter.SelectCommand = New SqlCommand()
        objDataAdapter.SelectCommand.Connection = objConnection
        objDataAdapter.SelectCommand.CommandText = _
        "SELECT MaSV, HoDem, TenSV, NgaySinh, MaLop FROM HOSOSV WHERE
MaLop= '" & cboMaLop.SelectedItem(0) & "'"

        objDataAdapter.SelectCommand.CommandType = CommandType.Text

        ' Open the database connection...
        objConnection.Open()

        ' Fill the DataSet object with data...
        objDataAdapter.Fill(objDataSet, "HosoSV")

        ' Close the database connection...
        objConnection.Close()

        ' Set the DataView object to the DataSet object...
        objDataView = New DataView(objDataSet.Tables("HosoSV"))

    End Sub
```

5. Thủ tục tiếp theo là thủ tục thực hiện trang trí DataGridView

```
Private Sub Grid()
' * trang tri Datagrid
' Declare and set the currency header alignment property...
Dim objAlignRightCellStyle As New DataGridViewCellStyle
objAlignRightCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight

' Declare and set the alternating rows style...
Dim objAlternatingCellStyle As New DataGridViewCellStyle()
objAlternatingCellStyle.BackColor = Color.WhiteSmoke
dgDanhSach.AlternatingRowsDefaultCellStyle = objAlternatingCellStyle

' Change column names and styles using the column index
dgDanhSach.Columns(0).HeaderText = "Mã sinh viên"
dgDanhSach.Columns(1).HeaderText = "Họ đệm"
dgDanhSach.Columns(2).HeaderText = "Tên"
dgDanhSach.Columns(3).HeaderText = "Ngày sinh"
dgDanhSach.Columns(4).HeaderText = "Mã lớp"

' Clean up
objAlternatingCellStyle = Nothing
objAlignRightCellStyle = Nothing
End Sub
```

6. Viết code cho thủ tục Load form.

```
Private Sub frmDanhSach_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load

 ' Dien du lieu vào cboMaLop
 Dim objDA_Lop As New SqlDataAdapter()
 Dim objDS_Lop As New DataSet()

 objDA_Lop.SelectCommand = New SqlCommand()

 objDA_Lop.SelectCommand.Connection = objConnection
 objDA_Lop.SelectCommand.CommandText = "SELECT MaLop, TenLop FROM Lop"
 objDA_Lop.SelectCommand.CommandType = CommandType.Text
 objConnection.Open()
 ' Fill the DataSet object with data...
 objDA_Lop.Fill(objDS_Lop, "DMLop")
 ' Close the database connection...
 objConnection.Close()

 Me.cboMaLop.DataSource = objDS_Lop.Tables("DMLop")
 Me.cboMaLop.DisplayMember = "TenLop"
 Me.cboMaLop.ValueMember = "MaLop"

 'Khởi tạo dữ liệu
  FillDataSetAndView()
  Me.dgDanhSach.AutoGenerateColumns = True

  Me.dgDanhSach.DataSource = objDataView
  Grid()
End Sub
```

7. Viết code cho nút lệnh Close để thực hiện đóng form.

```
Private Sub btnClose_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnClose.Click
        me.Close
End Sub
```

8. Viết code cho nút lệnh Update để thực hiện ghi lại các thay đổi dữ liệu vềcơ sở dữ liệu.

```
   Private Sub btnUpdate_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnUpdate.Click
        ' automatically create update/delete/insert commands
        Dim objCommandBuilder As SqlCommandBuilder = New
SqlCommandBuilder(objDataAdapter)
        objDataAdapter.UpdateCommand = objCommandBuilder.GetUpdateCommand
        ' Open the connection, execute the command
        objConnection.Open()
        objDataAdapter.Update(objDataSet, "HosoSV")
        objDataSet.Tables("HosoSV").AcceptChanges()
        objConnection.Close()
   End Sub
```

9.Viết code cho nút lệnh Insert để thực hiện chèn dữ liệu thực sự vào cơ sở dữ liệu.

```
Private Sub btnInsert_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnInsert.Click
        ' automatically create update/delete/insert commands
        Dim objCommandBuilder As SqlCommandBuilder = New
SqlCommandBuilder(objDataAdapter)
        objDataAdapter.InsertCommand = objCommandBuilder.GetInsertCommand
        ' Open the connection, execute the command
        objConnection.Open()
        objDataAdapter.Update(objDataSet, "HosoSV")
        objDataSet.Tables("HosoSV").AcceptChanges()
        objConnection.Close()
End Sub
```

10.Viết code cho nút lệnh Delete để thực hiện xóa thực sự dữ liệu ở CSDL, sau khi thực hiện xóa tạm trên Grid.

```
  Private Sub btnDelete_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnDelete.Click
        ' automatically create update/delete/insert commands
        Dim objCommandBuilder As SqlCommandBuilder = New
SqlCommandBuilder(objDataAdapter)
        objDataAdapter.DeleteCommand = objCommandBuilder.GetDeleteCommand
        ' Open the connection, execute the command
        objConnection.Open()
```

```
        objDataAdapter.Update(objDataSet, "HosoSV")
        objDataSet.Tables("HosoSV").AcceptChanges()
        objConnection.Close()
        Me.Refresh()
End Sub
```

11.Viết code cho nút lệnh Undo để thực hiện Undo lại các thao tác tạm trên Grid mà chưa thi hành thực sự đối với CSDL.

```
  Private Sub btnUndo_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnUndo.Click
        Me.Refresh()
        FillDataSetAndView()

        Me.dgDanhSach.AutoGenerateColumns = True
        Me.dgDanhSach.DataSource = objDataView
        Grid()
End Sub
```

12.Viết code cho nút lệnh Sắp xếp để thực hiện để thực hiện sắp xếp các bản ghi trên Grid theo 3 tiêu chí.

- Sắp xếo theo mã sinh viên
- Sắp xếp theo tên sinh viên
- Sắp xếp theo Họ và tên sinh viên.

```
Private Sub btnSort_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnSort.Click
        ' Sort property of the DataView object...
        FillDataSetAndView()
        Select Case cboSX.SelectedIndex
            Case 0 'Ten
                objDataView.Sort = "TenSV"
            Case 1 'Hoten
                objDataView.Sort = "TenSV, Hodem"
            Case 2 'Ma SV
                objDataView.Sort = "MaSV"
        End Select

        Me.dgDanhSach.AutoGenerateColumns = True
        Me.dgDanhSach.DataSource = objDataView
        Grid()
End Sub
```

13.Viết code cho sự kiện chọn dữ liệu trêm cboMaLop. Đoạn code này nhằm mục đích đồng bộ hóa dữ liệu trên Form.

```
  Private Sub cboMaLop_SelectedIndexChanged(ByVal sender As Object, ByVal
e As System.EventArgs) Handles cboMaLop.SelectedIndexChanged
```

```
        FillDataSetAndView()
        ' Me.dgDanhSach.AutoGenerateColumns = True
        Me.dgDanhSach.DataSource = objDataSet
        Me.dgDanhSach.DataMember = "HosoSV"
        Grid()
    End Sub
```

14. Ta có thể thực hiện cho chạy và thử nghiệm form trên.

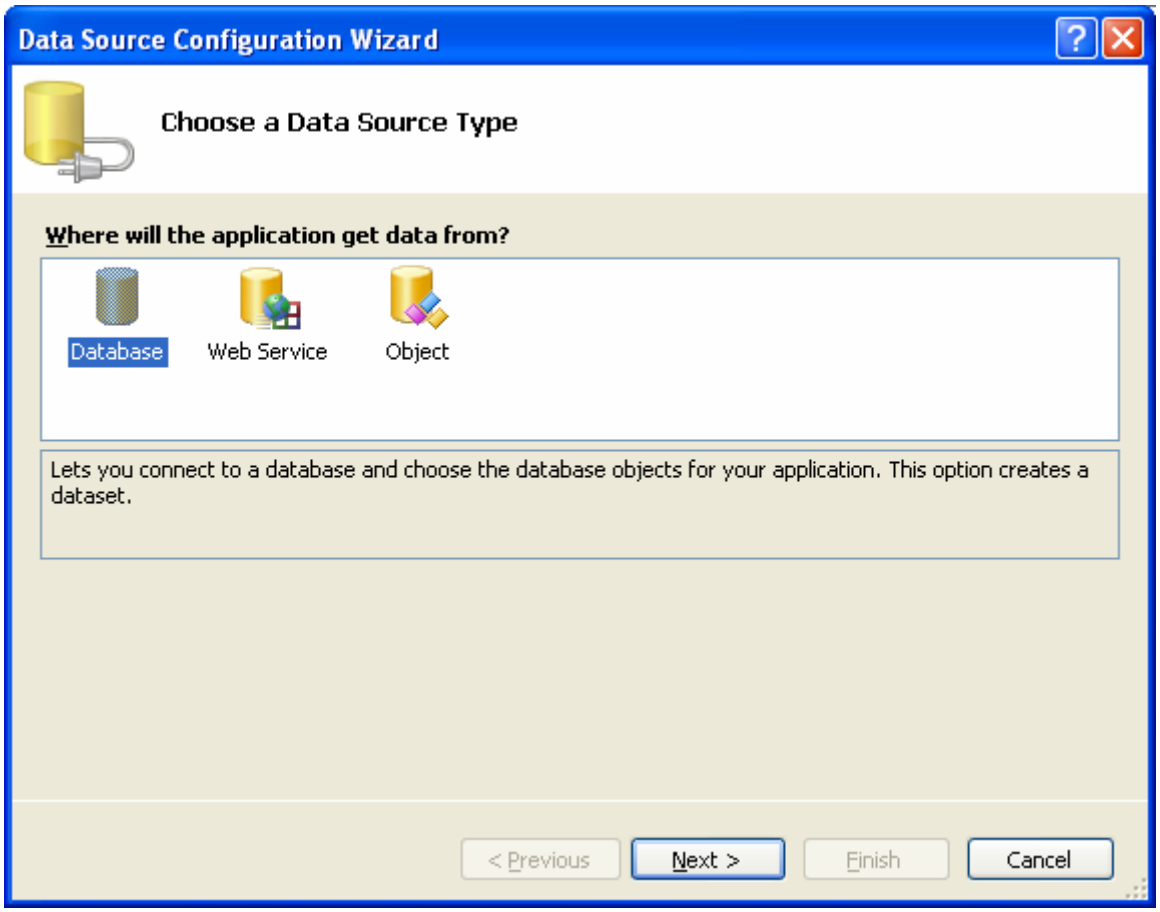
5.3.5. Xây dựng báo cáo dùng Report.

5.3.5.1. Tạo file Report.rdlc

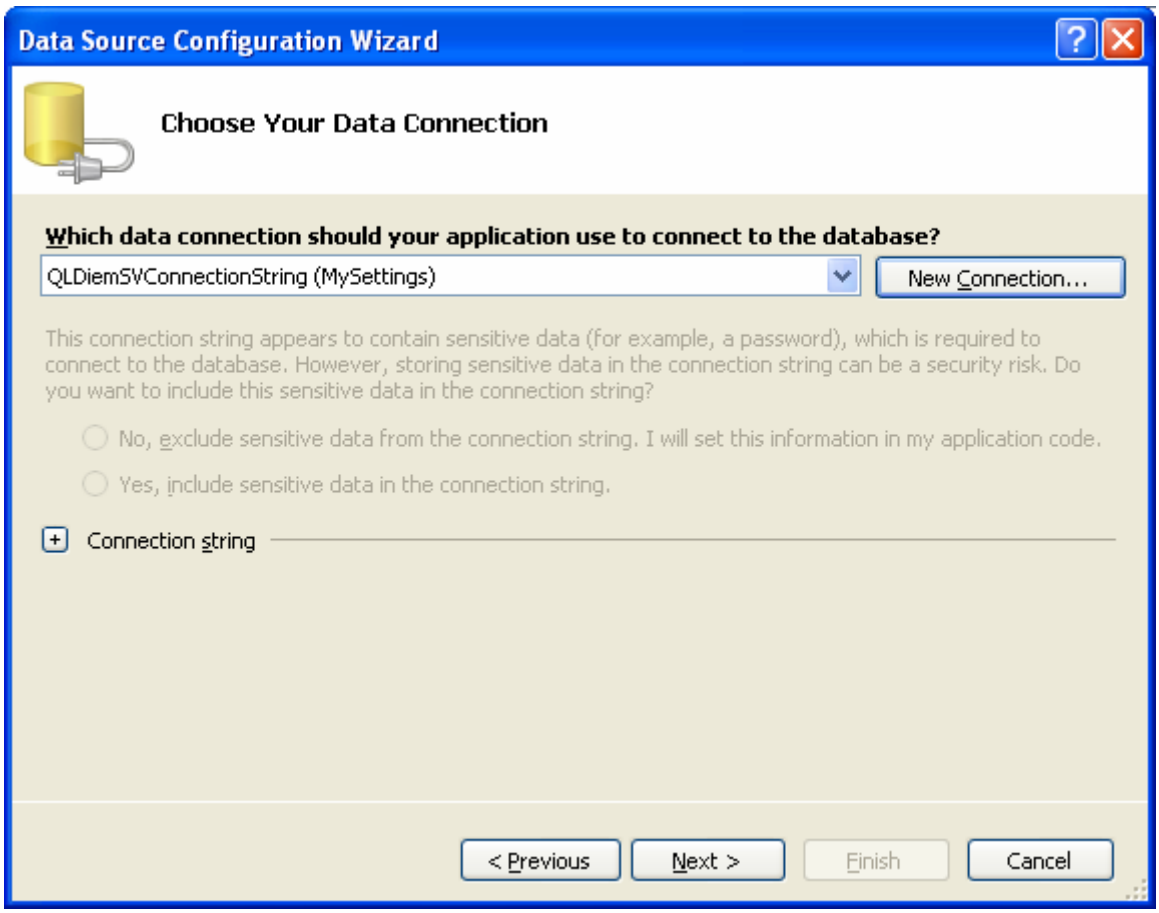
1. Right - Click trên Project *QL_DiemSV/Add/New Item/* chọn *template Report*. Xuất hiện cửa sổ thiết kế Report.

2. Từ menu *Data* chọn *Add New Data source* để tạo nguồn dữ liệu mới sẽ thể hiện trên report hoặc chọn Show data sources nếu đã tồn tại trước đó. Ở đây ta chọn *Add New Data source*. Các cửa sổ Wizard sẽ chỉ dẫn ta thực hiện.

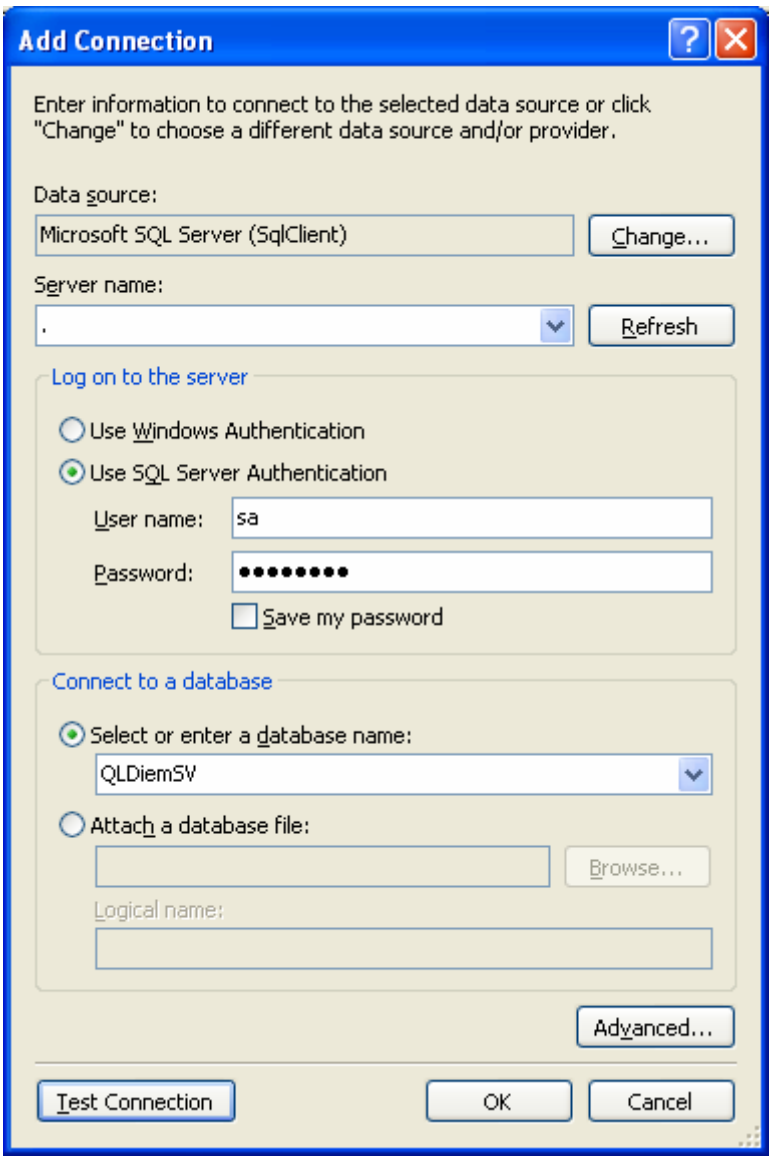
Cửa sổ 1. Chọn Database/ Nút Next.



Cửa sổ 2: Click nút New Connection

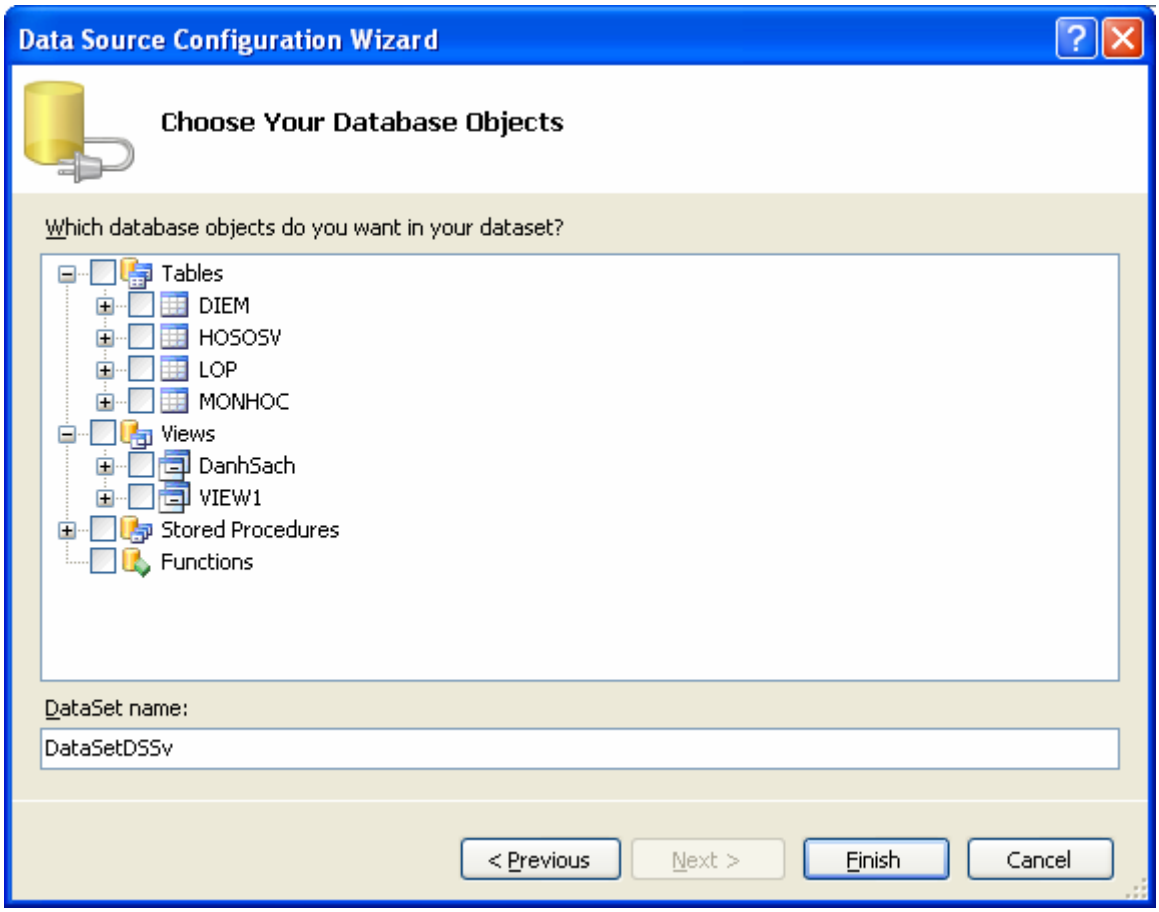

Data Source Configuration Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
QLDiemSVConnectionString (MySettings)
New Connection...
This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?
No, exclude sensitive data from the connection string. I will set this information in my application code.
Yes, include sensitive data in the connection string.
Connection string
< Previous
Next >
Finish
Cancel

Cửa sổ 3: Cửa sổ Add new Connection. Ta thực hiện điền các tham số như hình.



Sau khi chọn OK, nó sẽ quay trở lại cửa sổ 2.

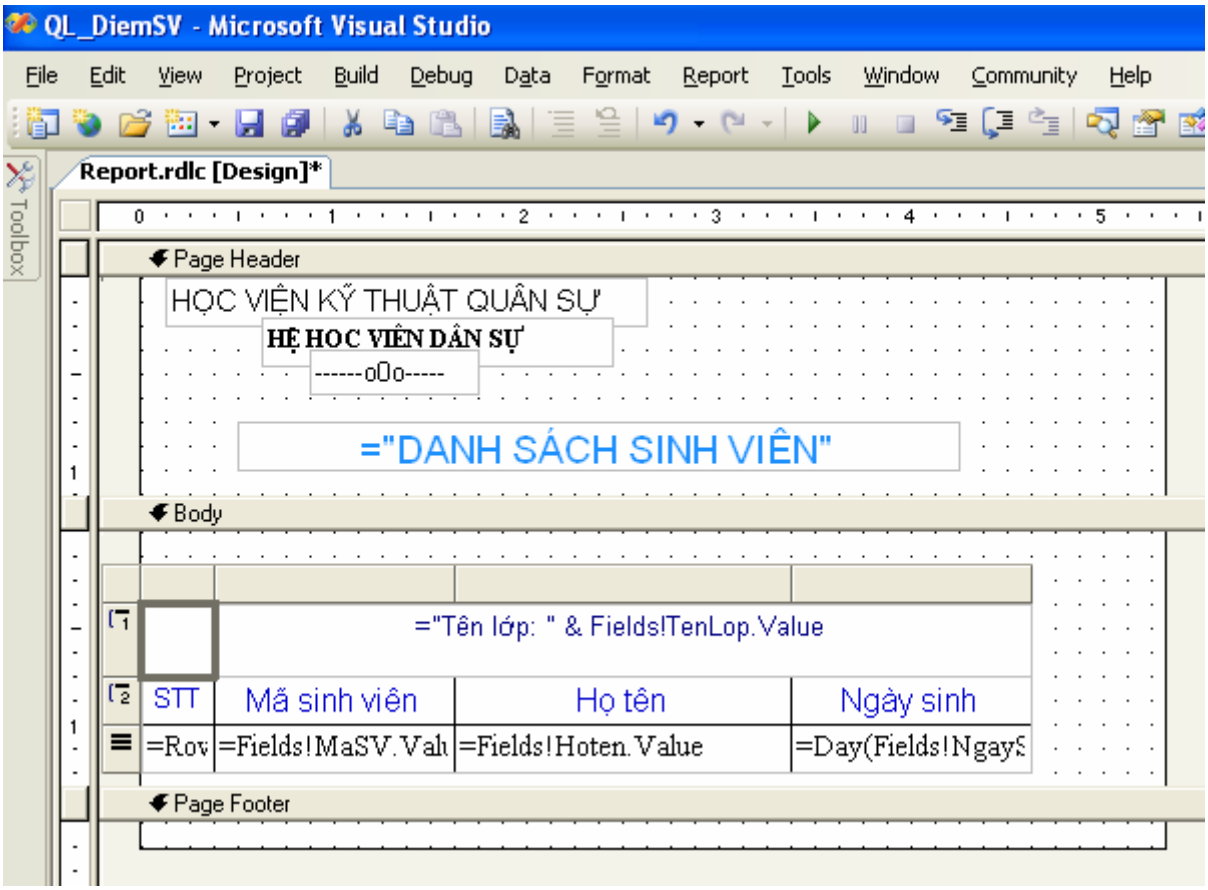
Cửa sổ 4. Chọn dữ liệu sẽ thể hiện trên Report và đặt tên cho Dataset.



Chọn view Danh sach → Click nút Finish.

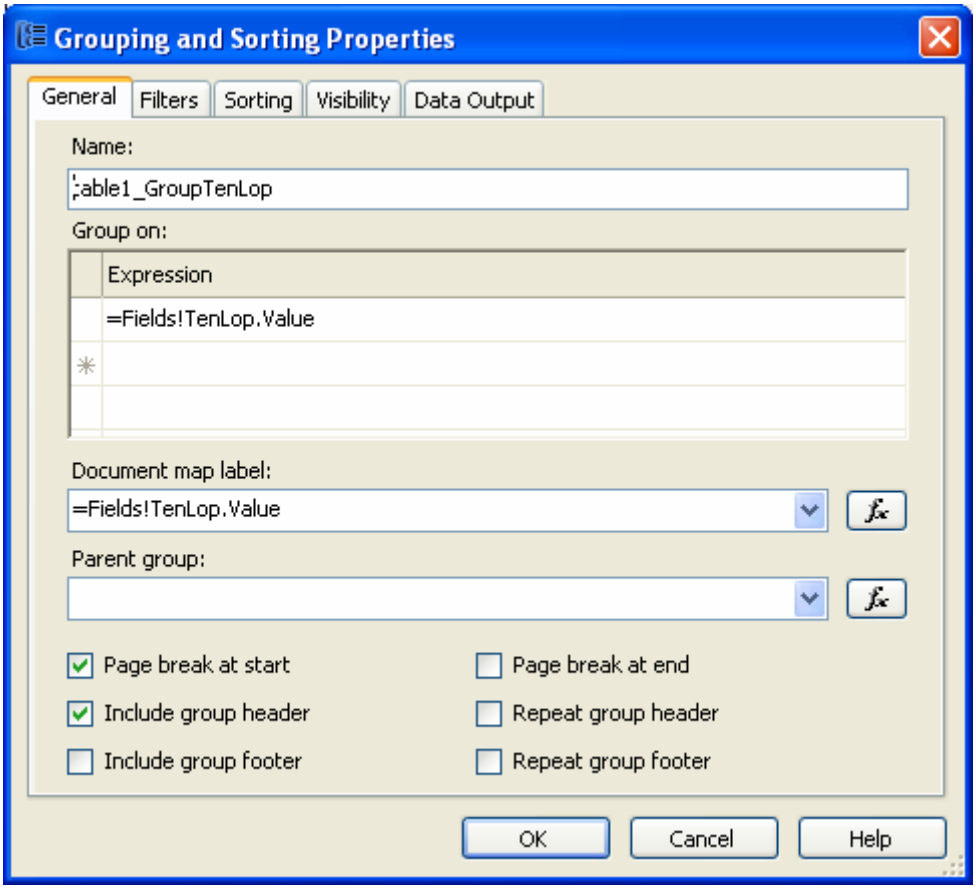
3. Sau khi có nguồn dữ liệu, trong cửa sổ thiết kế Report ta sẽ thiết kế dữ liệu sẽ hiển thị trên Report. Như hình, gồm các đối tượng sau:

Object	*Property*	*Setting*
Textbox	Name Value	textbox1 HỌC VIỆN KỸ THUẬT QUÂN SỰ
Textbox	Name Value	textbox2 **HỆ HỌC VIÊN DÂN SỰ**
Textbox	Name Value	Textbox3 ------o0o-----
Textbox	Name Value	Textbox4 ="DANH SÁCH SINH VIÊN"
Table	Name Datasetname	Table1 DatasetDSSv
Textbox	Name Value	Textbox5 ="Tên lớp: " & Fields!TenLop.Value
Textbox	Name Value	Textbox6 **STT**
Textbox	Name Value	Textbox7 Mã sinh viên
Textbox	Name Value	Textbox8 Họ tên
Textbox	Name Value	Textbox9 Ngày sinh
Textbox	Name Value	Textbox10 `=RowNumber("table1_Group2")`
Textbox	Name Value	Textbox11 `=Fields!MaSV.Value`
Textbox	Name Value	Textbox12 `=Fields!Hoten.Value`
Textbox	Name Value	Textbox13 `=Fields!Ngaysinh.Value`


QL_DiemSV - Microsoft Visual Studio
File Edit View Project Build Debug Data Format Report Tools Window Community Help
Report.rdlc [Design]*
Toolbox
Page Header
HỌC VIỆN KỸ THUẬT QUÂN SỰ
HỆ HỌC VIÊN DÂN SỰ
------oOo-----
="DANH SÁCH SINH VIÊN"
Body
="Tên lớp: " & Fields!TenLop.Value
STT
Mã sinh viên
Họ tên
Ngày sinh
=Rov
=Fields!MaSV.Valı
=Fields!Hoten.Value
=Day(Fields!Ngay
Page Footer

**** Chèn group vào bảng.***

- Trong cửa sổ thiết kế Right-click vào Table tại dòng ta muốn chèn Group và chọn Insert Group.



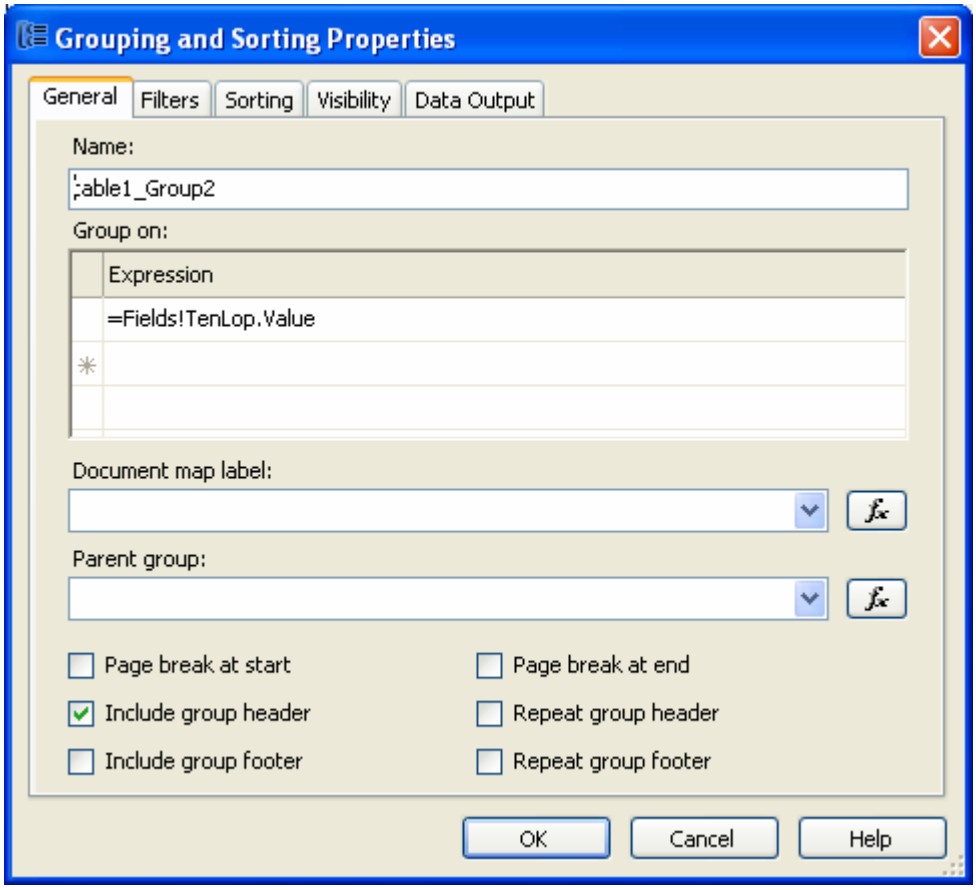
- Trên tab ***General*** ta điền các tham số:
  - + *Name:* Tên của group
  - + *Group on:* Chọn biểu thức mà dựa vào đó dữ liệu gom nhóm.
  - + Document map label: Đánh hoặc chọn biểu thức được sử dụng làm nhãn ánh xạ.
  - + *Parent group:* Nếu group này là nhóm phân cấp ngược (phân cấp đệ quy) ta sẽ đánh hoặc chọn biểu thức làm nhóm cha (recursive group parent).
  - + *Option chọn Page break at start* hoặc *Page break at end*: để thay thế cho một page break tại một vị trí bắt đầu hoặc kết thúc một thể hiện của nhóm.

+ *Include group header* hoặc *Include group footer:* Cho hiển thi hay không hiển thị Header và Footer của nhóm trên Table.

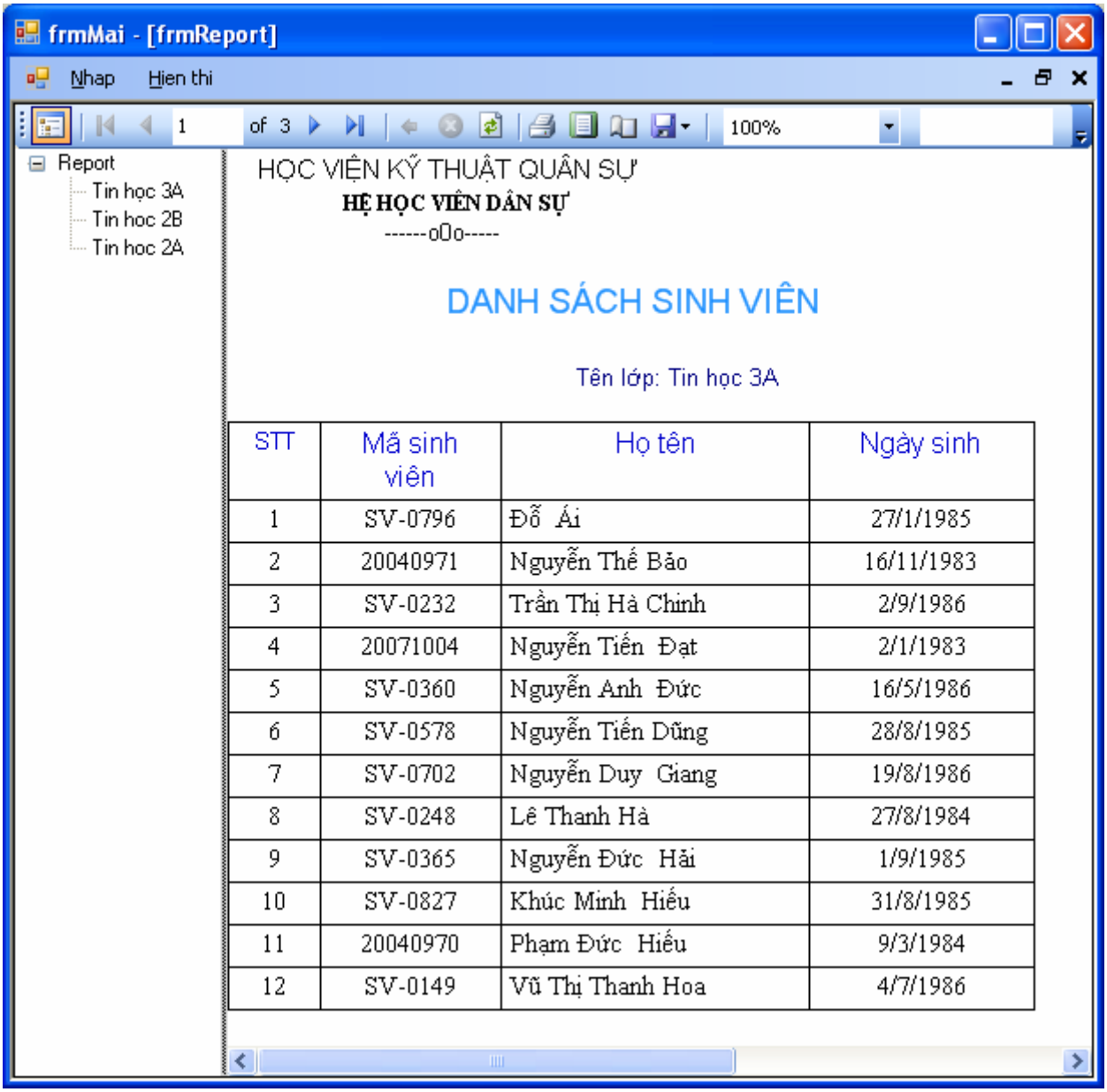
+ *Repeat group header hoặc Repeat group footer:* để lặp lại group header hoặc footer trên mỗi trang mà trong đó bảng xuất hiện.

- Trên tab ***Fillters:*** Chọn hoặc đánh biểu thức dùng để lọc dữ liệu trên nhóm.
- Trên tab ***Sorting***: Chọn hoặc đánh biểu thức dùng để sắp xếp dữ liệu trên nhóm.
- Trên tab ***Visibility***: chọn Visible.
- Trên tab Data Output: Chọn Yes.

Trong ví dụ ta chèn hai Group: table1_GroupTenLop và table1_Group2



5.3.5.2. Hiển thị dữ liệu.



STT	Mã sinh viên	Họ tên	Ngày sinh
1	SV-0796	Đỗ Ái	27/1/1985
2	20040971	Nguyễn Thế Bảo	16/11/1983
3	SV-0232	Trần Thị Hà Chinh	2/9/1986
4	20071004	Nguyễn Tiến Đạt	2/1/1983
5	SV-0360	Nguyễn Anh Đức	16/5/1986
6	SV-0578	Nguyễn Tiến Dũng	28/8/1985
7	SV-0702	Nguyễn Duy Giang	19/8/1986
8	SV-0248	Lê Thanh Hà	27/8/1984
9	SV-0365	Nguyễn Đức Hải	1/9/1985
10	SV-0827	Khúc Minh Hiếu	31/8/1985
11	20040970	Phạm Đức Hiếu	9/3/1984
12	SV-0149	Vũ Thị Thanh Hoa	4/7/1986

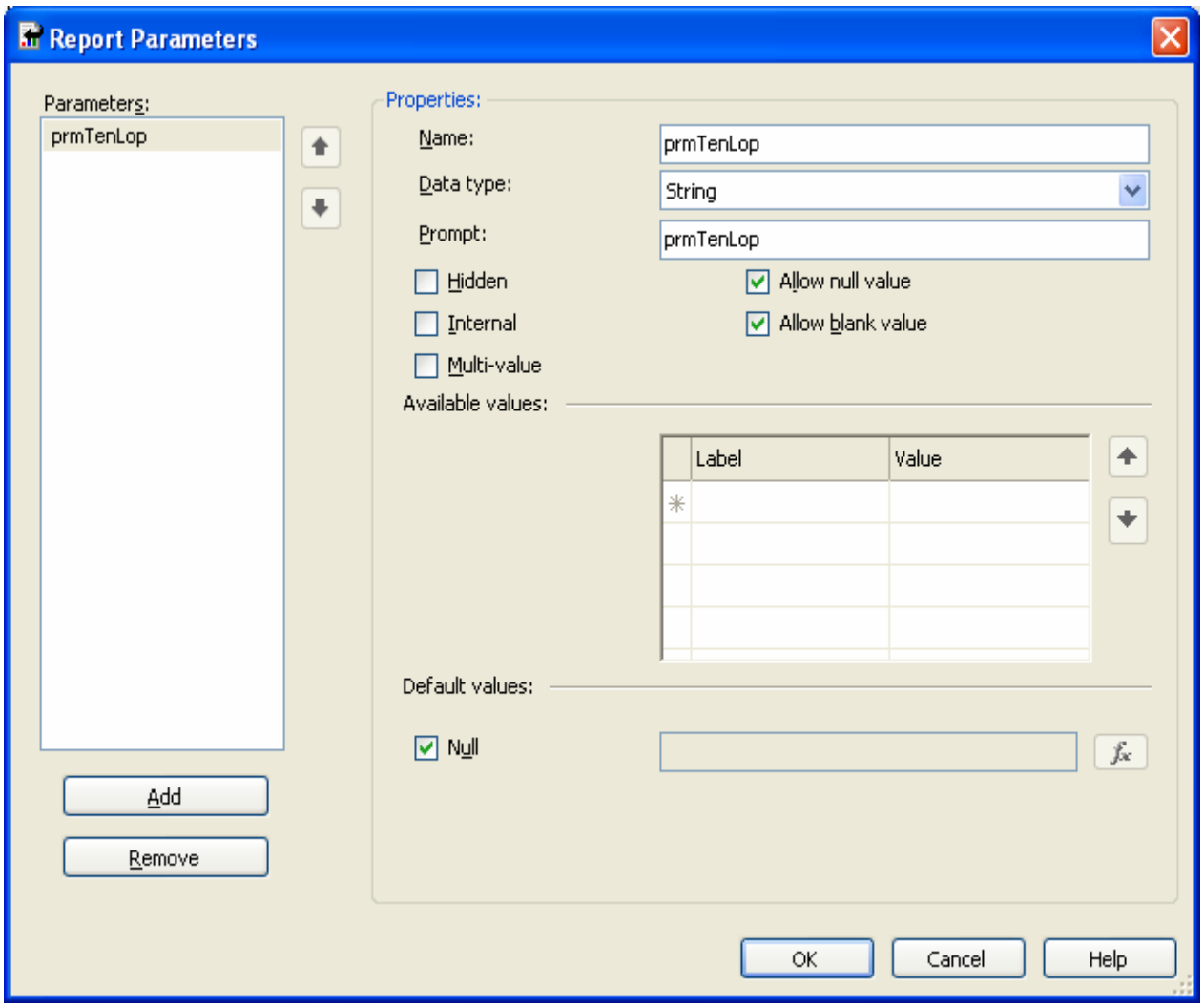
Để kết Report vào form, ta xây dựng form hiển thị. Đặt một điều khiển Report Viewer. Trên Report Viewer Tasks chọn các mục sau:

- Choose report: Chọn report ta vừa thiết kế (report.rdlc)
- Dock in parent container. Chọn.

5.2.5.3. Sử dụng tham số trong Report.

a) Tạo tham số

Trong cửa sổ thiết kế Report, để thiết lập tham số ta thực hiện như sau: Từ menu Report\Report Paramaters để Add các tham số cho Report.



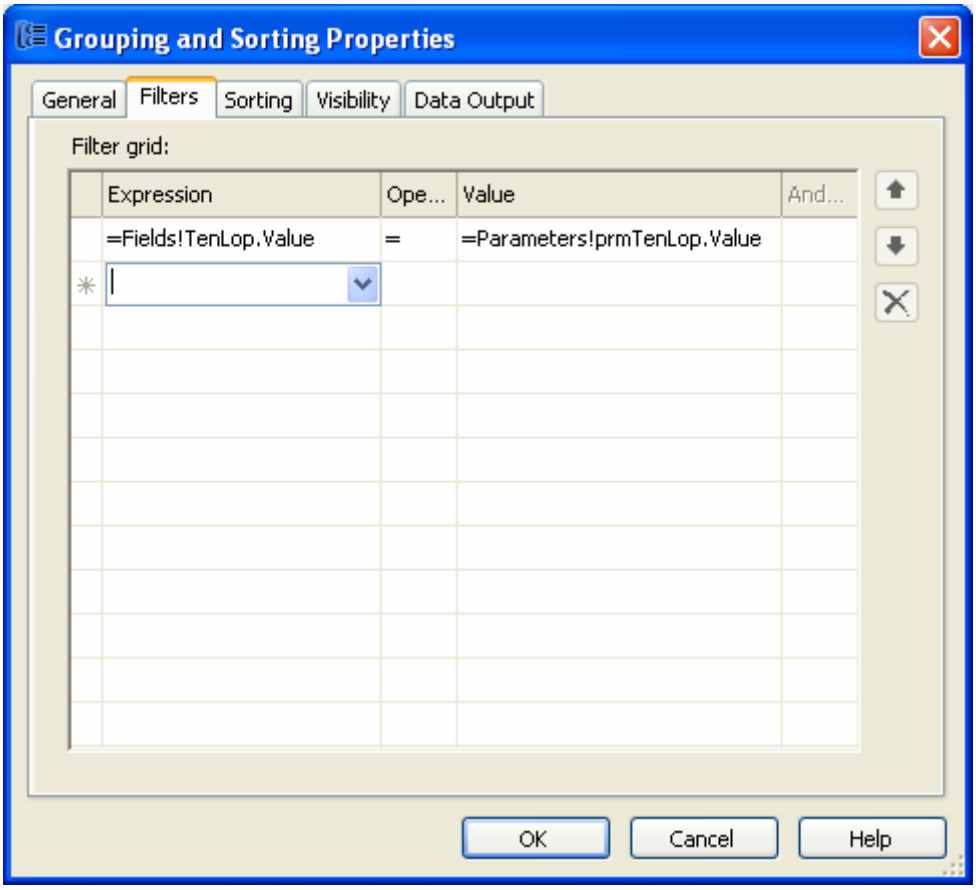
- Chọn nút Add để tạo thêm một tham số mới gồm các thông tin:
  - Name: Tên tham số
  - Data type: Kiểu dữ liệu của tham số
  - Prompt: Điền đoạn text sẽ xuất hiện sau *parameter text box* khi người sử dụng thực hiện chạy report.
  - Allow null value: Chọn khi cho phép tham số nhận giá trị null.
  - Allow blank value: Chọn khi cho phép thma số nhận giá trị blank.
  - Available values: Đưa ra một danh sách các giá trị sẵn có mà người sử dụng có thể lựa chọn.

- Lable: Chứa nhãn sẽ được sử dụng để hiển thị cho người sử dụng.
- Value: Là giá trị sẽ được sử dụng để chuyển qua Report sever cho tham số.

▪ Default values: Giá trị mặc định cho tham số.

- Chọn nút Remove để xóa tham số đã chọn.

Sau khi đã tạo tham số, ta có thể sử dụng tham số trong khi thiết kế Report. Trong ví dụ ta này ta xây dựng một report như trên, ngoài ra add thêm tham số prmTenLop và lọc dữ liệu theo tham số.



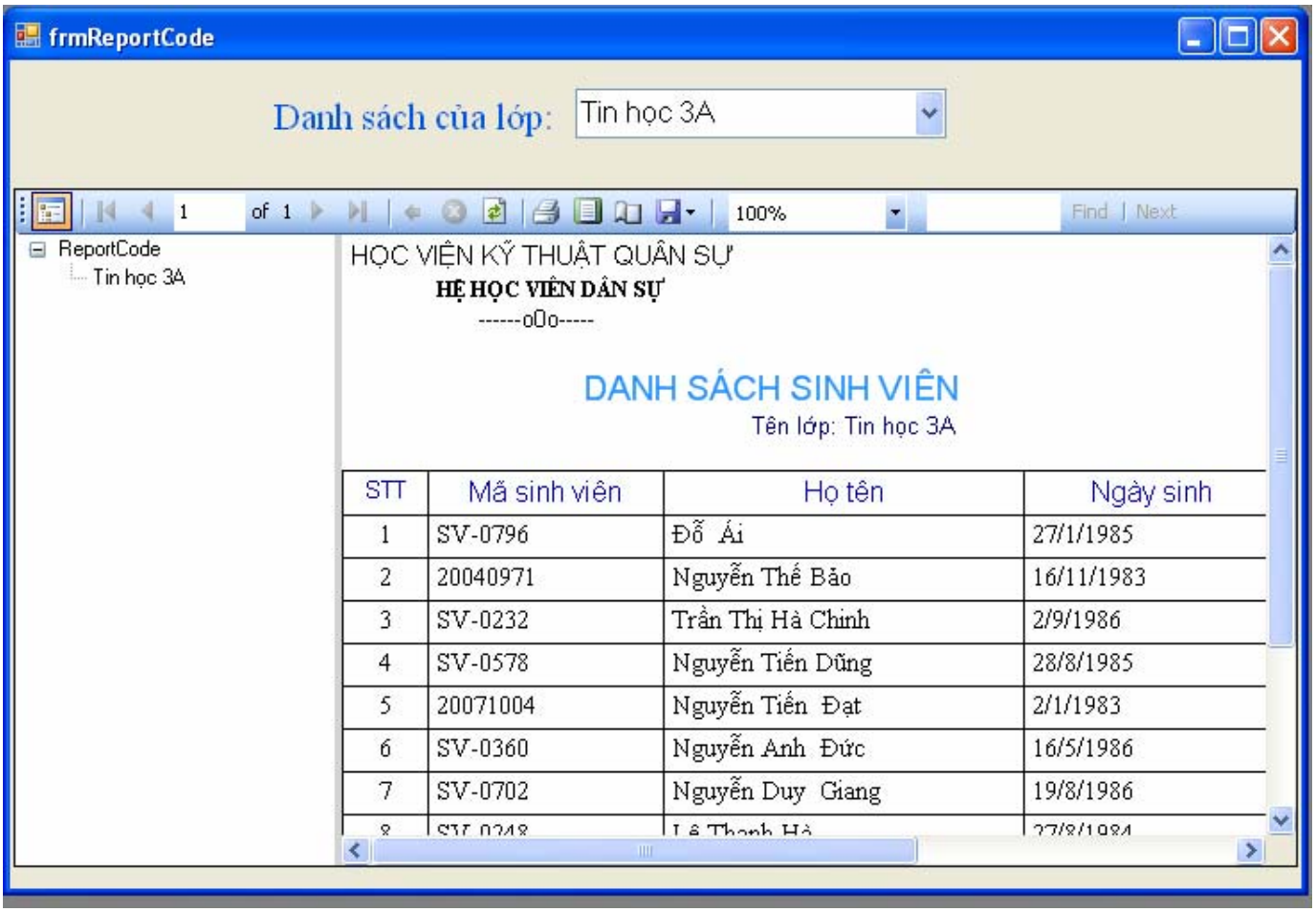
b)Truyền giá trị cho tham số:

Để truyền giá trị cho tham số ta sử dụng phương thức SetParameters của lớp Report trong tên miền Microsoft.Reporting.WinForms. Cụ thể ta sử dụng như sau:

```
' Gán giá trị cho tham số
```

```
Dim p As New ReportParameter("prmTenLop",
Me.cboMaLop.SelectedItem(1).ToString)
' Truyền giá trị vào report
Me.rvwDSReport.LocalReport.SetParameters(New
ReportParameter() {p})
```

Giả sử ta có form hiển thị Report như sau:



Có các đối tượng sau:

Object	***Property***	***Setting***
Form	Name Text	frmReportCode frmReportCode
Label	Name Text	lblDSL Danh sách của lớp
ComboBox	Name	cboMaLop
Report Viewer	Name Choose report	rvwDSL Chọn Report đã tạo như trên

1. Import các tên miền cần thiết.

```
' Import Data and SqlClient namespaces...
Imports System.Data
Imports System.Data.SqlClient
Imports Microsoft.Reporting.WinForms
```

2. Khai báo các kết nối

```
    Dim objConnection As New SqlConnection _
            ("server=THUHUONG;database=QLDiemSV;user
id=sa;password=12102006")
```

3. Code so sự kiện load form.

```
    Private Sub frmReportCode_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load

    ' Dien dư lieu vào cboLop
    Dim objDA_Lop As New SqlDataAdapter()
    Dim objDS_Lop As New DataSet()
    objDA_Lop.SelectCommand = New SqlCommand()
    objDA_Lop.SelectCommand.Connection = objConnection
    objDA_Lop.SelectCommand.CommandText = "SELECT MaLop, TenLop FROM Lop"
    objDA_Lop.SelectCommand.CommandType = CommandType.Text
    objConnection.Open()
    ' Fill the DataSet object with data...
    objDA_Lop.Fill(objDS_Lop, "DMLop")
    ' Close the database connection...
    objConnection.Close()

    Me.cboMaLop.DataSource = objDS_Lop.Tables("DMLop")
    Me.cboMaLop.DisplayMember = "TenLop"
    Me.cboMaLop.ValueMember = "MaLop"

     ' Set tham so cho Report
     Dim p As New ReportParameter("prmTenLop",
     Me.cboMaLop.SelectedItem(1).ToString)
      Me.rvwDSReport.LocalReport.SetParameters(New ReportParameter() {p})

     'This line of code loads data into the 'QLDiemSVDataSet.DanhSach'
     Me.objTableAdapterDSSV.Fill(Me.objDataSetDSSV.DanhSach)
        Me.rvwDSReport.RefreshReport()

    End Sub
```

4. Đồng bộ hóa dữ liệu trên form.

```
    Private Sub cboMaLop_SelectedIndexChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles cboMaLop.SelectedIndexChanged
     Dim p As New ReportParameter("prmTenLop",
     Me.cboMaLop.SelectedItem(1).ToString)
     Me.rvwDSReport.LocalReport.SetParameters(New ReportParameter() {p})
     Me.rvwDSReport.RefreshReport()
    End Sub
```

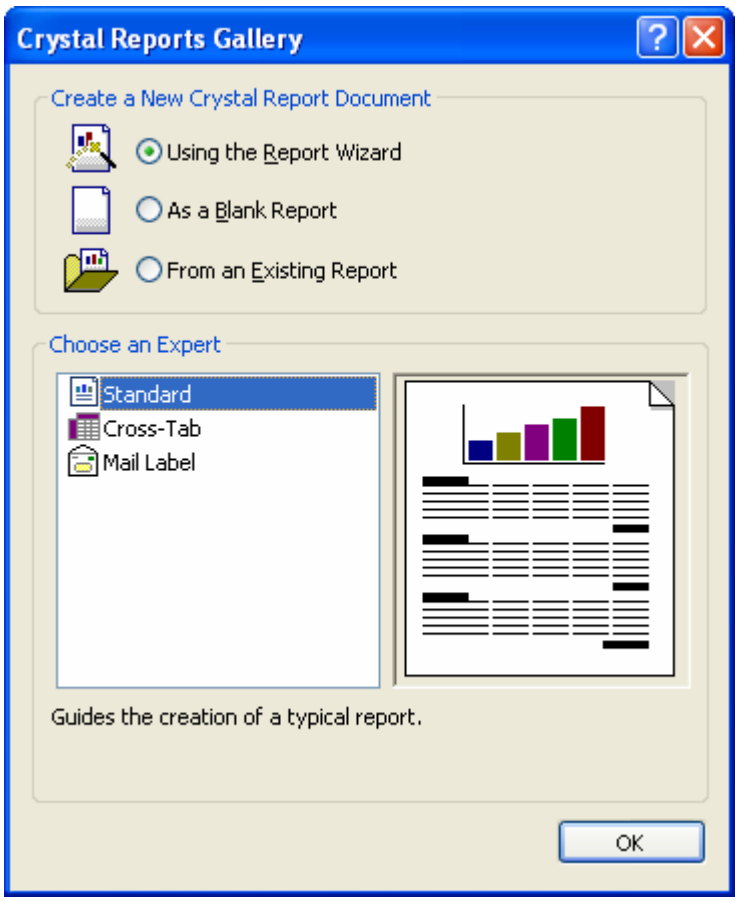
5.3.6. Xây dựng report dùng Crystal Report.

Trong phần này hướng dẫn xây dựng Report bằng Crystal report một cách đơn giản nhất, sử dụng Wizard. Trong khi thiết kế ta nên dùng wizard để thiết kế sơ bộ, sau đó chuyển vào cửa sổ thiết kế để thiết kế chi tiết.

Các bước thiết kế report.

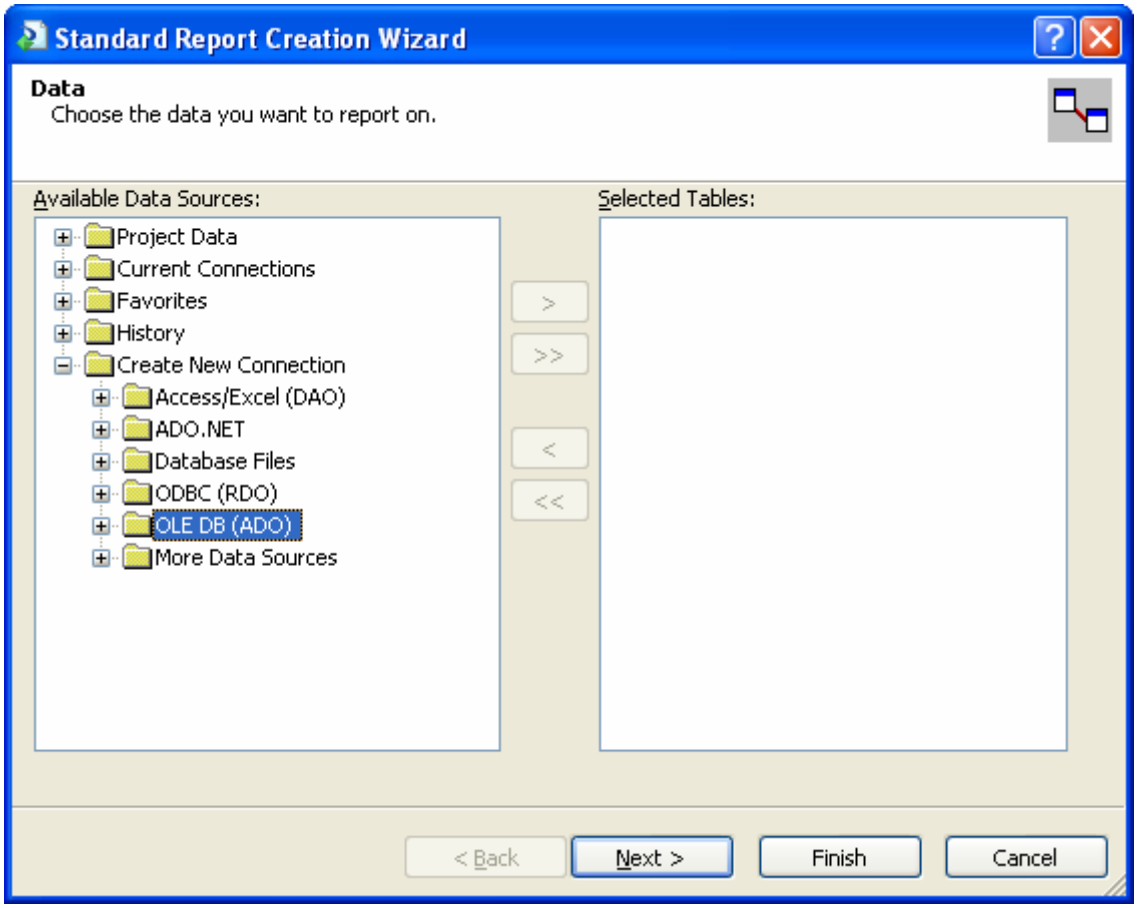
1. Right – Click trên Project QL_DiemSV/Add/New Item/ chọn template Crystal Report. Điền tên file của Crystal Report. Xuất hiện cửa sổ sau:

Cửa sổ 1.



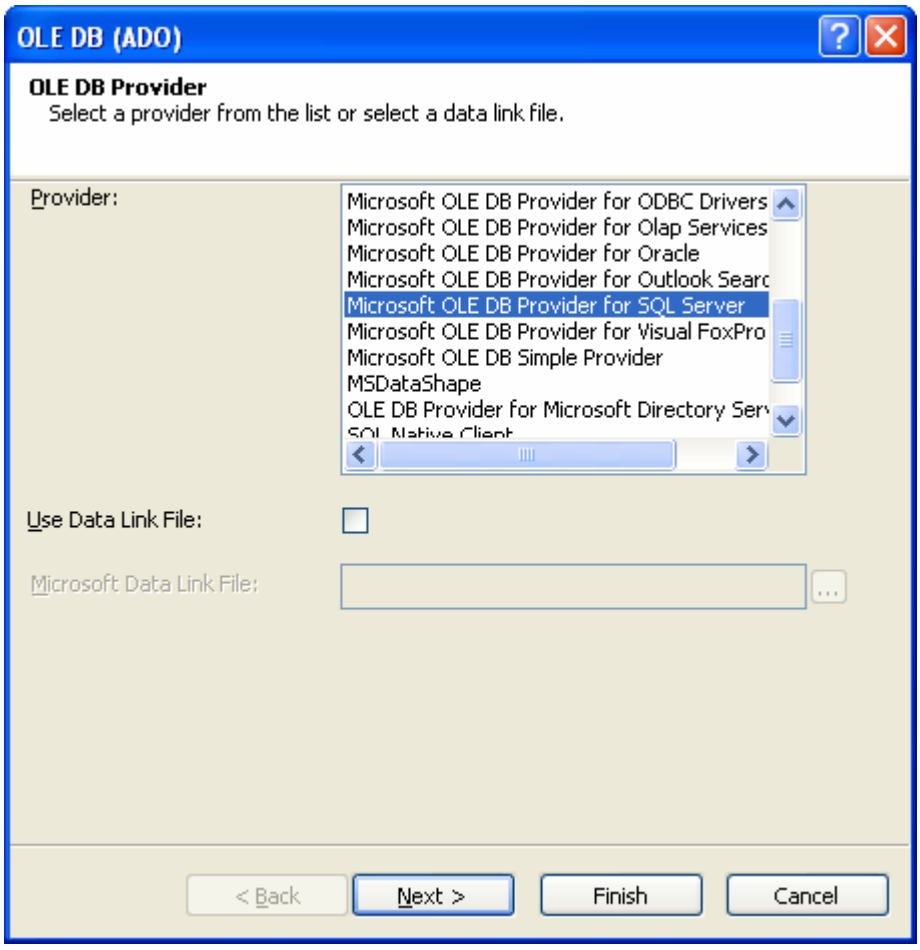
2. Chọn Using the Repor Wizart và Standard → OK.

Cửa sổ 2.



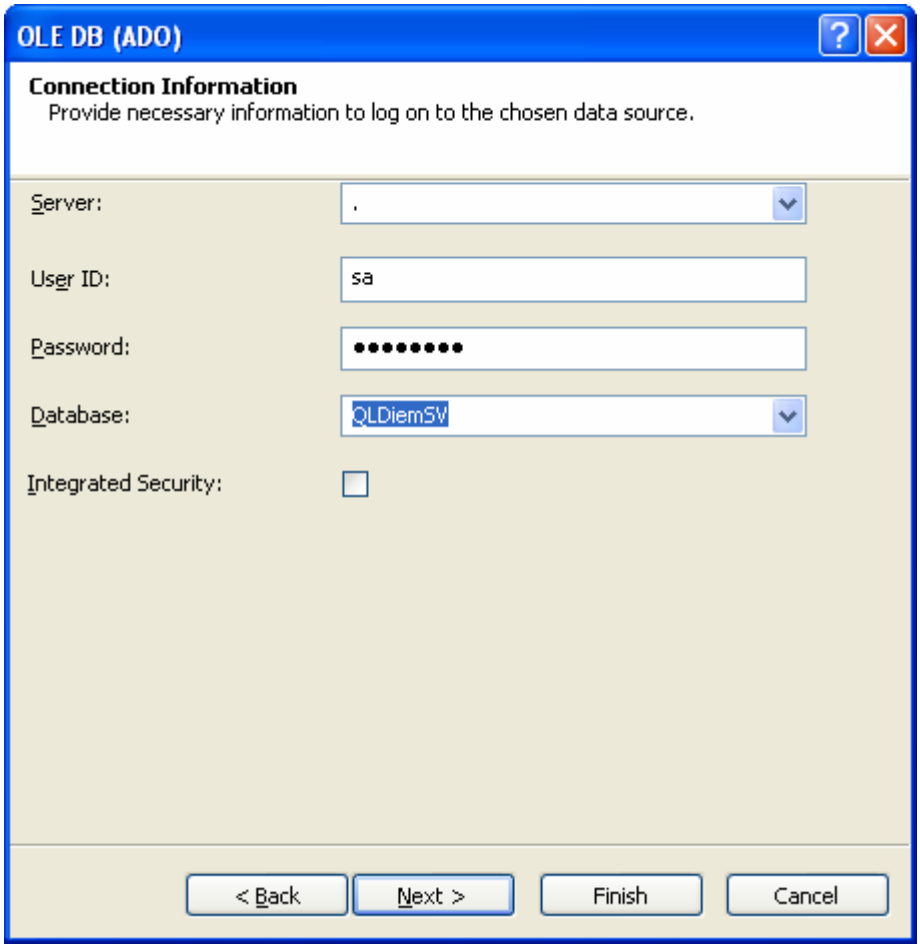
Chọn OLE DB (ADO) → Xuất hiện cửa sổ 3.

Cửa sổ 3:



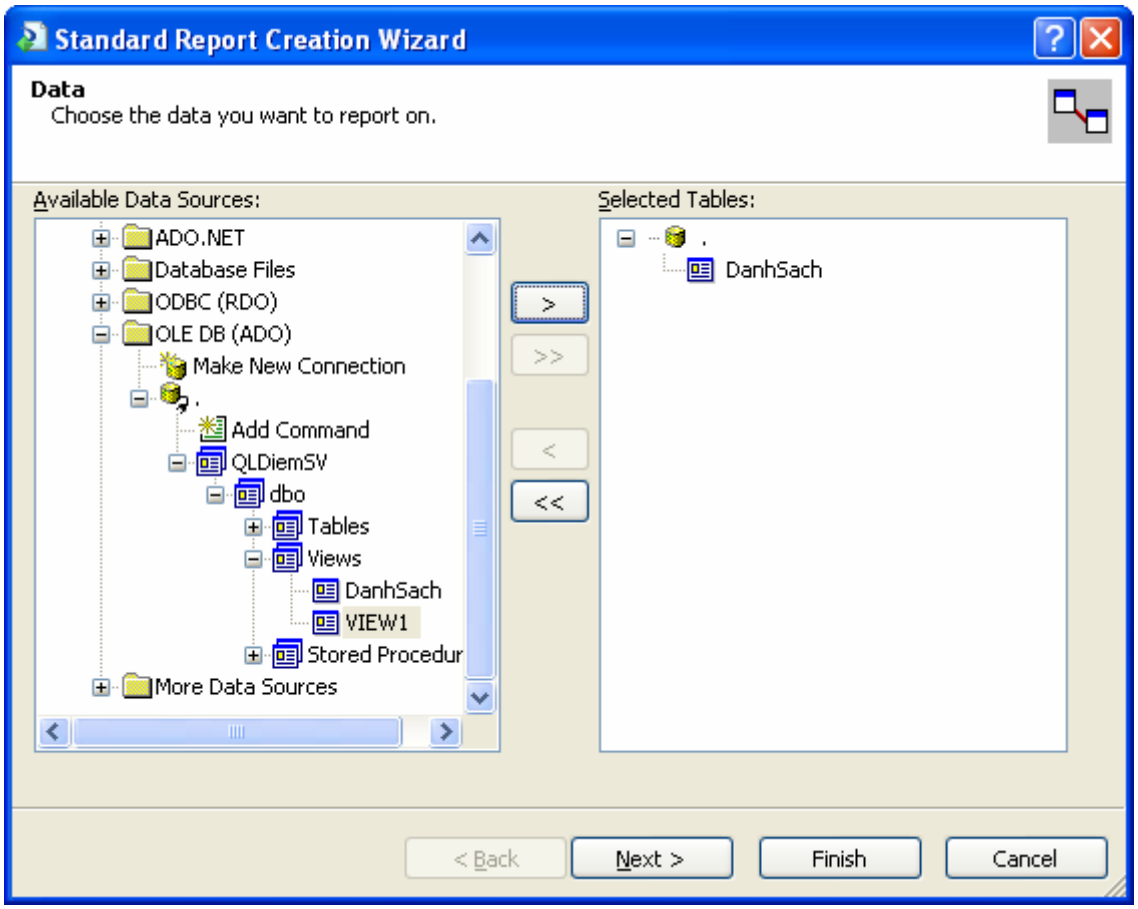
Chọn Microsoft OLE DB Provider for SQL Server. → Next.

Cửa sổ 4:

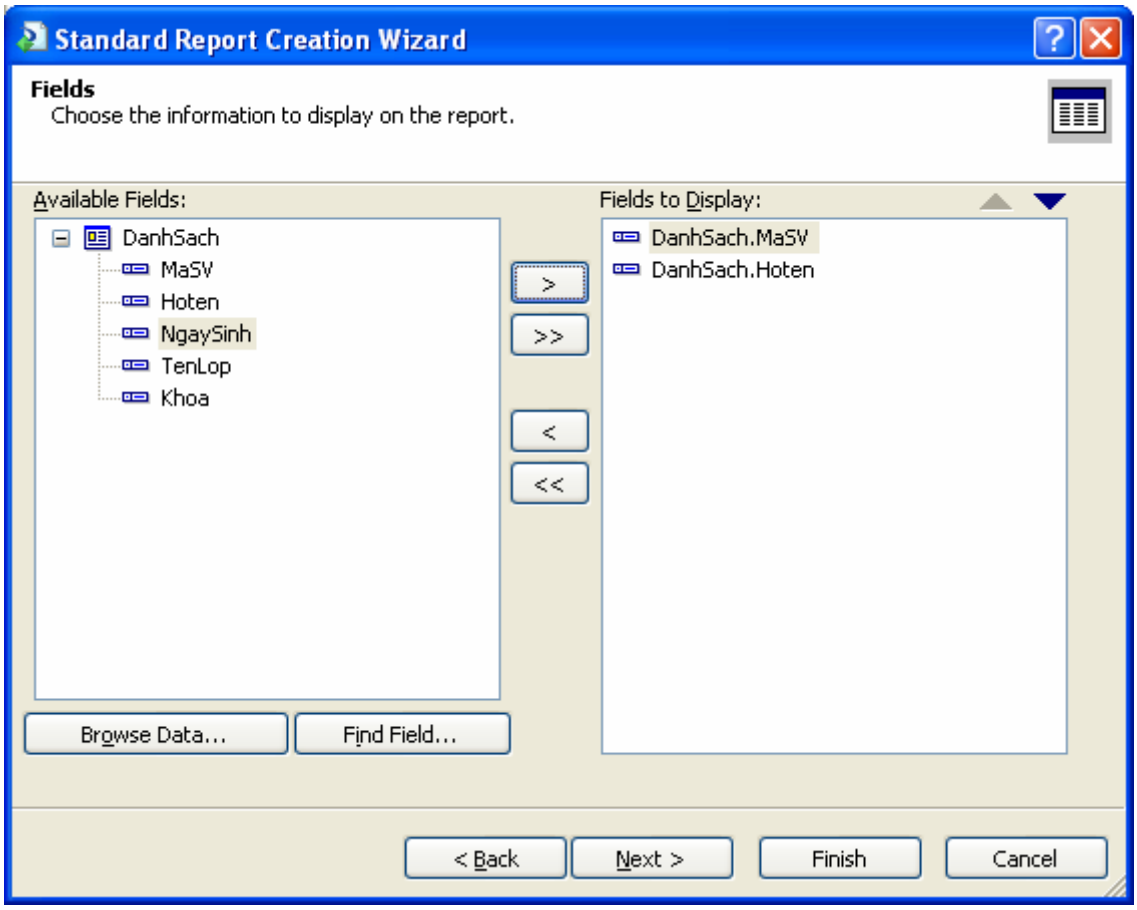


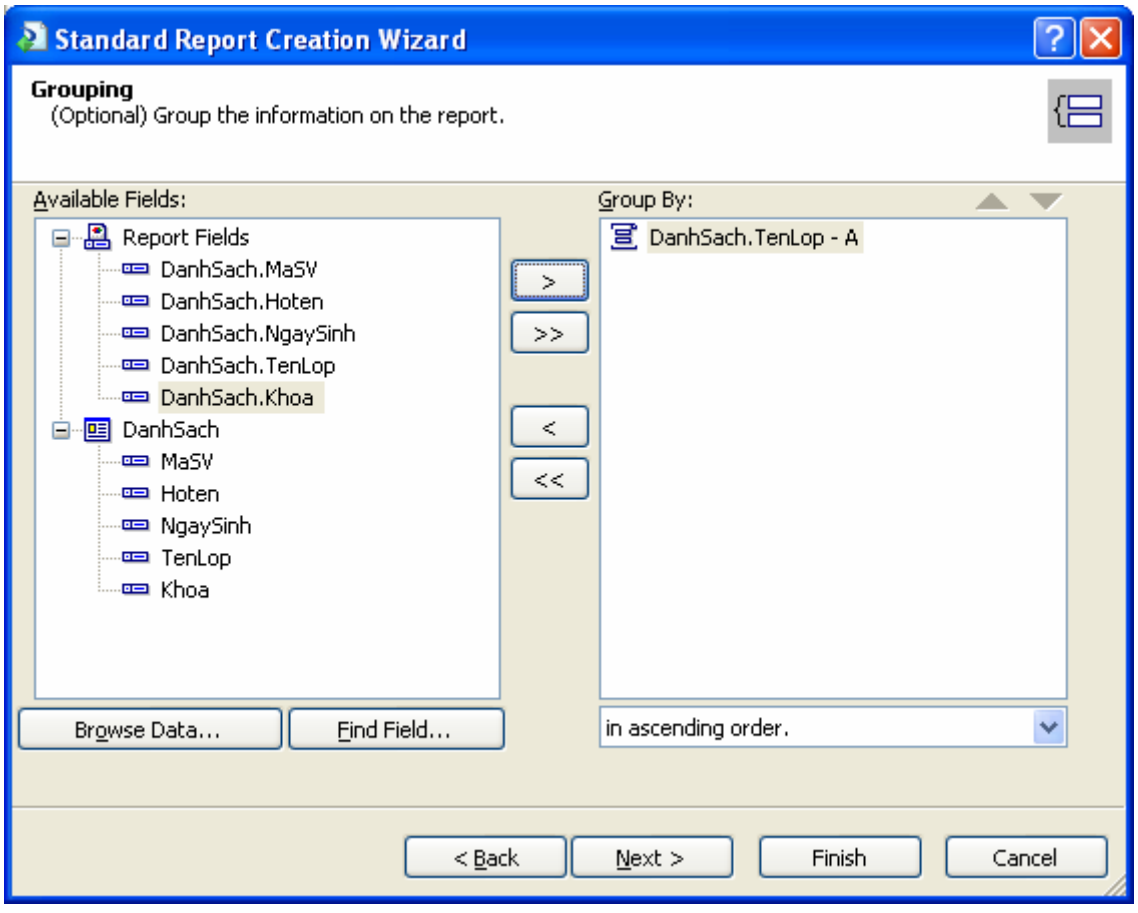
Chọn server, User, password và Database. →Finish. Để quay lại cửa sổ 2. Chọn dữ liệu sẽ được hiển thị trên report.

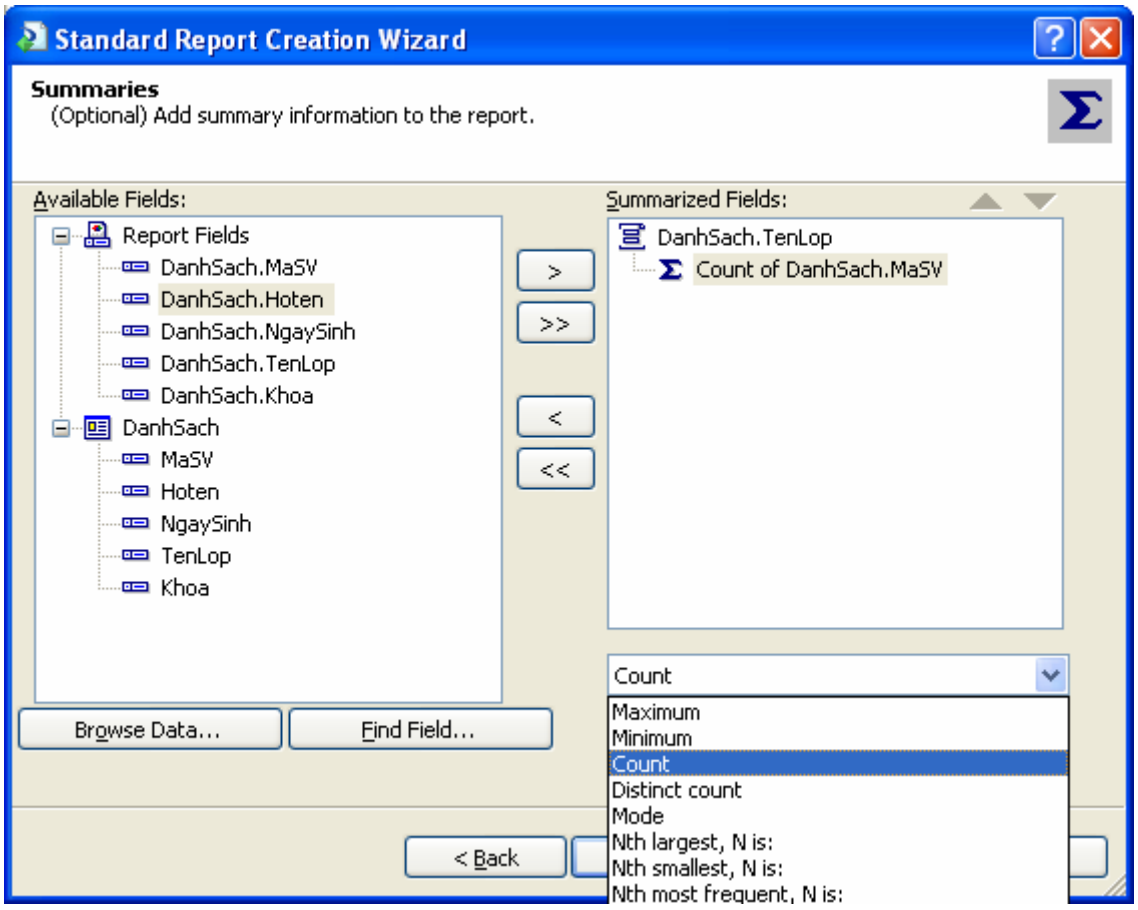
Cửa sổ 2: Chọn nguồn dữ liệu cho Report.

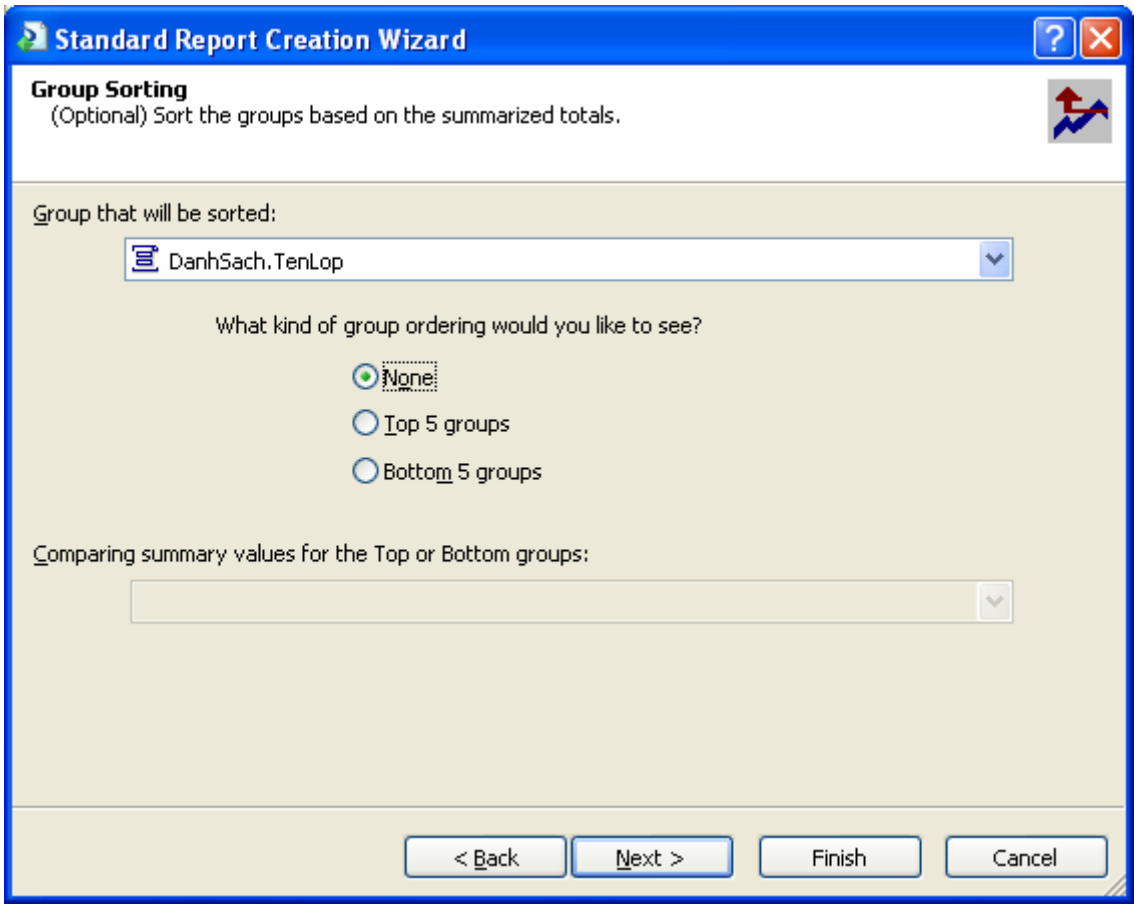


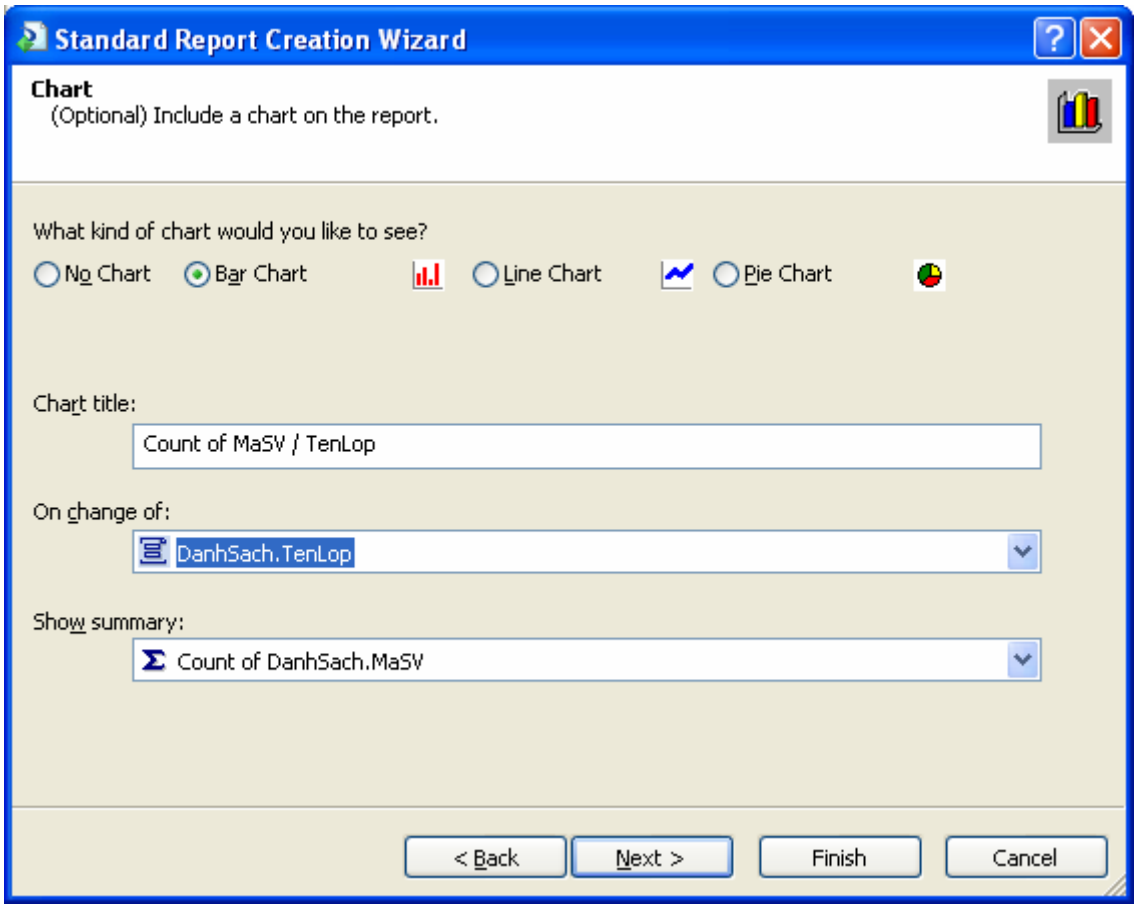
Chọn Next để tiếp tục theo sự chỉ dẫn của wizard.

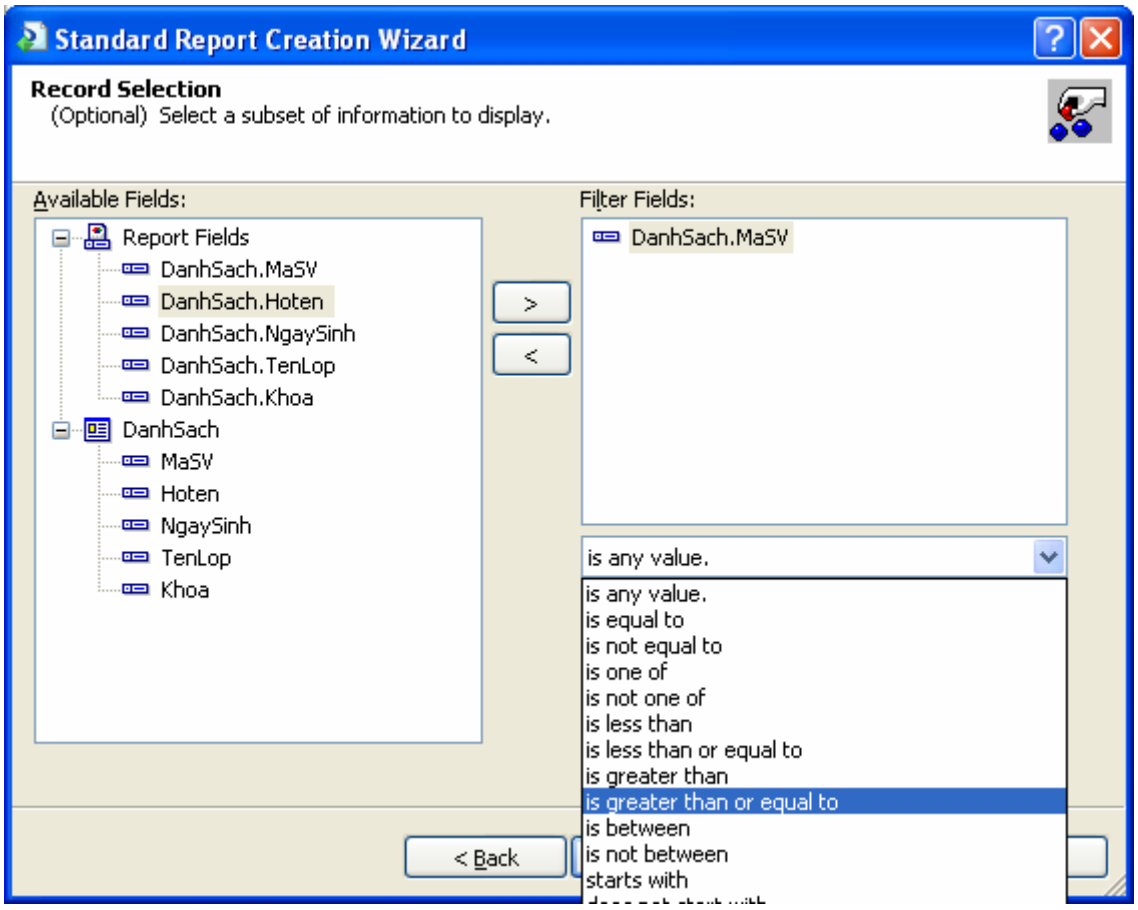

Standard Report Creation Wizard
Fields
Choose the information to display on the report.
Available Fields:
DanhSach
MaSV
Hoten
NgaySinh
TenLop
Khoa
Fields to Display:
DanhSach.MaSV
DanhSach.Hoten
>
>>
<
<<
Browse Data...
Find Field...
< Back
Next >
Finish
Cancel

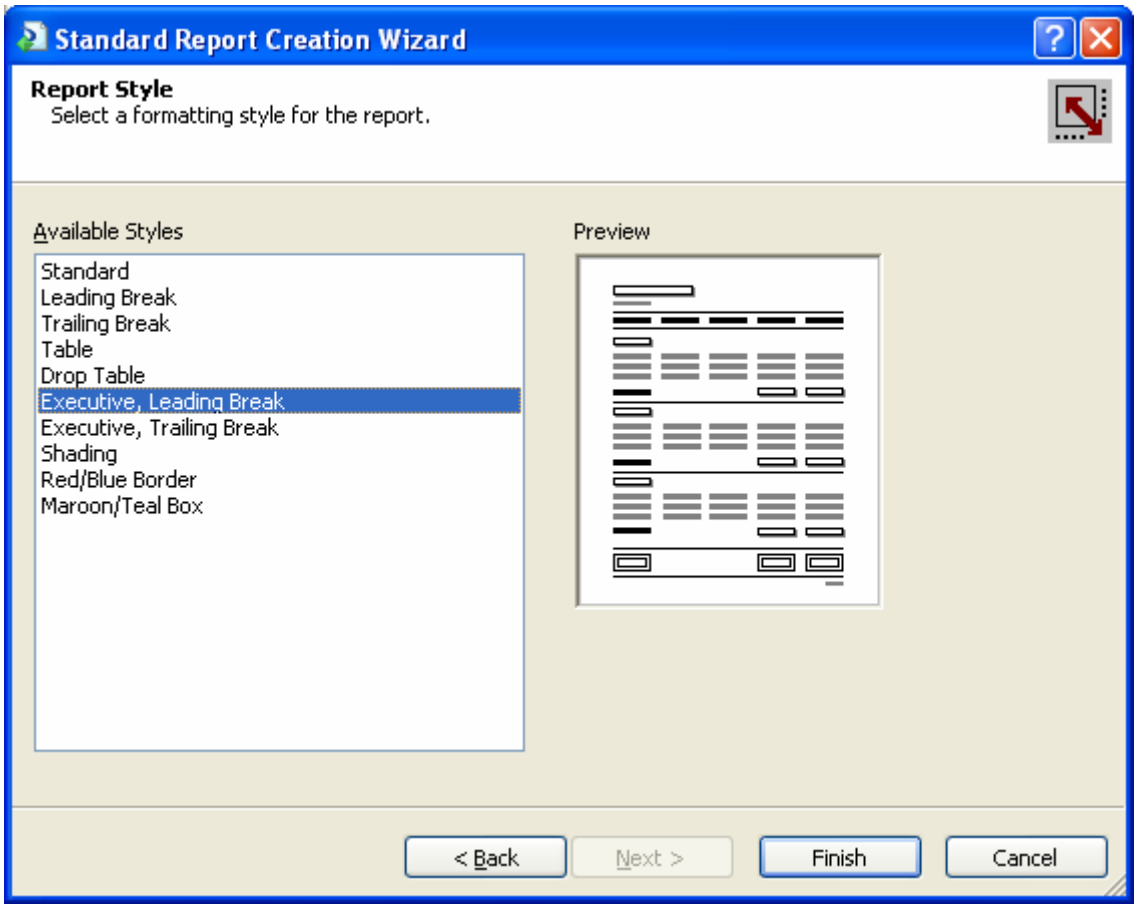

Standard Report Creation Wizard
Grouping
(Optional) Group the information on the report.
Available Fields:
Report Fields
DanhSach.MaSV
DanhSach.Hoten
DanhSach.NgaySinh
DanhSach.TenLop
DanhSach.Khoa
DanhSach
MaSV
Hoten
NgaySinh
TenLop
Khoa
Group By:
DanhSach.TenLop - A
>
>>
<
<<
Browse Data...
Find Field...
in ascending order.
< Back
Next >
Finish
Cancel

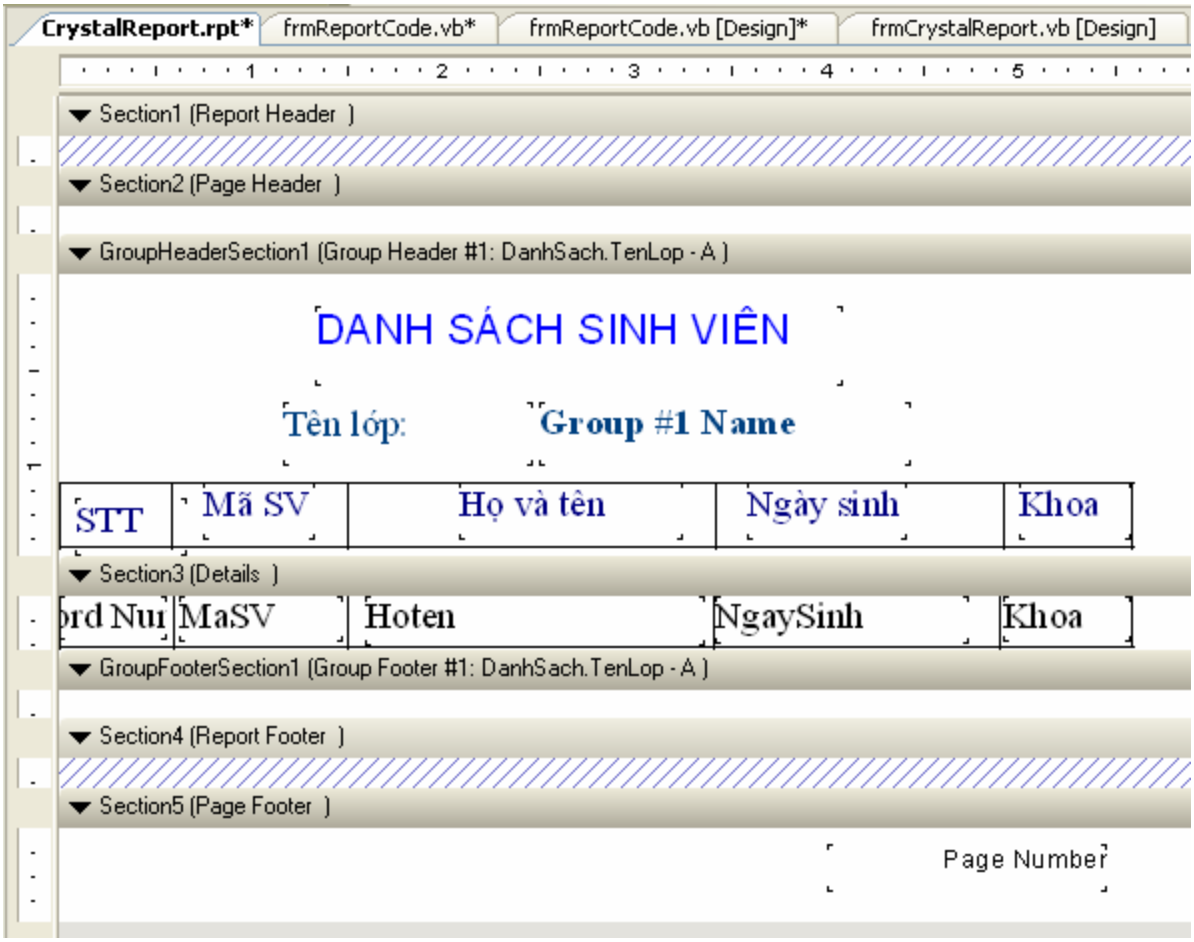

Standard Report Creation Wizard
Summaries
(Optional) Add summary information to the report.
Available Fields:
Report Fields
DanhSach.MaSV
DanhSach.Hoten
DanhSach.NgaySinh
DanhSach.TenLop
DanhSach.Khoa
DanhSach
MaSV
Hoten
NgaySinh
TenLop
Khoa
>
>>
<
<<
Summarized Fields:
DanhSach.TenLop
Count of DanhSach.MaSV
Count
Maximum
Minimum
Count
Distinct count
Mode
Nth largest, N is:
Nth smallest, N is:
Nth most frequent, N is:
Browse Data...
Find Field...
< Back


Standard Report Creation Wizard
Group Sorting
(Optional) Sort the groups based on the summarized totals.
Group that will be sorted:
DanhSach.TenLop
What kind of group ordering would you like to see?
None
Top 5 groups
Bottom 5 groups
Comparing summary values for the Top or Bottom groups:
< Back
Next >
Finish
Cancel


Standard Report Creation Wizard
Chart
(Optional) Include a chart on the report.
What kind of chart would you like to see?
No Chart
Bar Chart
Line Chart
Pie Chart
Chart title:
Count of MaSV / TenLop
On change of:
DanhSach.TenLop
Show summary:
Count of DanhSach.MaSV
< Back
Next >
Finish
Cancel


Standard Report Creation Wizard
Record Selection
(Optional) Select a subset of information to display.
Available Fields:
Report Fields
DanhSach.MaSV
DanhSach.Hoten
DanhSach.NgaySinh
DanhSach.TenLop
DanhSach.Khoa
DanhSach
MaSV
Hoten
NgaySinh
TenLop
Khoa
Filter Fields:
DanhSach.MaSV
is any value.
is any value.
is equal to
is not equal to
is one of
is not one of
is less than
is less than or equal to
is greater than
is greater than or equal to
is between
is not between
starts with
< Back


Standard Report Creation Wizard
Report Style
Select a formatting style for the report.
Available Styles
Standard
Leading Break
Trailing Break
Table
Drop Table
Executive, Leading Break
Executive, Trailing Break
Shading
Red/Blue Border
Maroon/Teal Box
Preview
< Back
Next >
Finish
Cancel


CrystalReport.rpt*
frmReportCode.vb*
frmReportCode.vb [Design]*
frmCrystalReport.vb [Design]
Section1 (Report Header)
Section2 (Page Header)
GroupHeaderSection1 (Group Header #1: DanhSach.TenLop - A)
DANH SÁCH SINH VIÊN
Tên lớp:
Group #1 Name
STT
Mã SV
Họ và tên
Ngày sinh
Khoa
Section3 (Details)
MaSV
Hoten
NgaySinh
Khoa
GroupFooterSection1 (Group Footer #1: DanhSach.TenLop - A)
Section4 (Report Footer)
Section5 (Page Footer)
Page Number

Sau khi đã xây dựng được file *.rpt. Ta xây dựng form để hiển thị report trên. Thiết kế form như hình gồm một đối tượng Crystal Report Viewer. Trên Crystal Report Viewer tasks thực hiện các mục:

- Choose a crystal report: Chọn Report vừa tạo trên.
- Dock in parent container: Chọn.