MỤC LỤC

BÀI 1:
GIỚI THIỆU MICROSOFT SQL SERVER 2000

1.1 Tổng quan về SQL Server 2000.

Microsoft SQL Server 2000 là một hệ quản trị cơ sở dữ liệu quan hệ (Relational database management system - RDBMS), nó cung cấp các dịch vụ quản lý và lưu trữ dữ liệu cho các tổ chức thương mại lớn, cùng với việc truy xuất dữ liệu hỗ trợ đối với các người dùng thông qua internet. Nó cũng hỗ trợ khả năng truy xuất một cách dễ dàng đối với các tổ chức nhỏ hơn và các người dùng riêng biệt. SQL Server 2000 chấp nhận và thực thi các yêu cầu của các khách hàng (Client) đối với việc hiệu chỉnh và xoá dữ liệu, cũng như các lệnh tạo các đối tượng như là các cơ sở dữ liệu (database) và các bảng (Table). SQL Server cho phép người dùng truy xuất và sắp xếp dữ liệu theo cách quan hệ, việc lưu trữ dữ liệu một cách hiệu quả theo dạng dòng và cột. Các lệnh của người dùng được gửi như là các câu lệnh truy vấn giao tác (Transact-SQL). Transact-SQL (T-SQL) là một ngôn ngữ dùng bởi SQL Server 2000 để truy vấn một cơ sở dữ liệu hoặc hiệu chỉnh nội dung của nó.

Một client từ một chương trình ứng dụng hoặc một người dùng gửi các câu lệnh T-SQL (các truy vấn hoặc hiệu chỉnh) thông qua mạng đến SQL Server để xử lý. Các chương trình ứng dụng mà gửi các câu lệnh có thể được viết bằng các ngôn ngữ như Visual Basic, Visual C++, hoặc Java. Chương này sẽ giúp bạn làm quen với Microsoft SQL Server cũng như thảo luận về các thành phần và thiết kết của nó.

Microsoft SQL Server phát triển một cách nhảy vọt trong nhiều năm. Phiên bản mới nhất, SQL Server 2000 hỗ trợ nổi bật hơn về tính bảo mật, lưu trữ và tính sẵn dùng của dữ liệu. SQL Server 2000 được tích hợp chặt chẽ với hệ điều hành Windows 2000, việc nâng cao bảo mật và các công cụ dựa trên giao diện đồ họa đã cung cấp trong SQL Server 2000 là cho các tác vụ quản trị trở nên dễ dàng hơn.

Các câu lệnh T-SQL dùng để hiệu chỉnh hoặc truy xuất dữ liệu có thể được chia thành ba nhóm khác nhau:

- **Ngôn ngữ định nghĩa dữ liệu (Data Definition Language - DDL)** được dùng để định nghĩa các đối tương của cơ sở dữ liệu như các kiểu dữ liệu, bảng và các truy vấn.
- **Ngôn ngữ thao tác dữ liệu (Data Manipulation Language -DML)** được dùng để chọn, chèn, cập nhật và xóa dữ liệu trong các đối tượng của CSDL.
- **Ngôn ngữ điều khiển dữ liệu (Data Control Language - DCL)** được dùng để gán hoặc hủy bỏ các quyền từ các người dùng của CSDL.

SQL Server 2000 chạy trên hệ thống mạng Windows NT 4 hoặc Windows 2000 Server bằng cách dùng nhiều bộ xử lý và có thể được cài đặt như một hệ thống CSDL cá nhân (personal desktop) trên các máy trạm Windows NT 4, Windows 2000 Professional, Windows 98, và Windows Me. Bạn có thể dùng cùng một CD để cài đặt phiên bản Server SQL Server 2000 hoặc Personal SQL Server 2000. Ngoài ra bạn có thể cài đặt nhiều thể hiện (instance) của SQL Server 2000 trên cùng một máy tính, mỗi thể hiện có một người chủ (owner) và tập các người dùng (user) của nó.

Cơ sở dữ liệu (CSDL) - Databases

Một CSDL hay là một database tương tự một tập tin dữ liệu, nơi để lưu trữ dữ liệu. Giống như một tập tin dữ liệu, một CSDL không trình bày thông tin một cách trực tiếp đến người dùng; muốn xem dữ liệu, người dùng phải chạy một ứng dụng truy xuất dữ liệu từ CSDL và khi đó dữ liệu được trình bày trong một dạng dễ hiểu và thuận tiện.

Khi dữ liệu được lưu trong các tập tin dữ liệu thì phải có ứng dụng (được lập trình) để làm việc với cấu trúc đặc biệt của mỗi tập tin dữ liệu. Ngược lại, một CSDL chứa dựng một catalog mà các ứng dụng có thể dùng để xác định dữ liệu được tổ chức như thế nào. Các ứng dụng CSDL cùng loại có thể dùng catalog để trình bày đến người dùng các dữ liệu từ các CSDL khác nhau một cách linh động mà không bị phụ thuộc vào một dạng mẫu đặc biệt.

Các hệ thống CSDL mạnh tức là các tập tin dữ liệu được tổ chức theo một mức độ cao. Một CSDL được thiết kế tốt, thì không có sự trùng lắp thông tin khi người dùng hoặc ứng dụng cập nhật tại cùng thời điểm. Các mãnh thông tin liên quan được kết nhóm với nhau trong một cấu trúc đơn hoặc mẫu tin và các mối quan hệ có thể được định nghĩa giữa các cấu trúc này và các mẫu tin này.

CSDL quan hệ (Relational Database)

Mặc dù có nhiều cách khác nhau để tổ chức dữ liệu trong một CSDL, các CSDL quan hệ là CSDL hiệu quả nhất. Các hệ thống CSDL quan hệ là một ứng dụng thuộc về lý thuyết tập hợp toán học đối với vấn đề tổ chức dữ liệu một cách hiệu quả. Trong một CSDL quan hệ, dữ liệu được thu thập vào các bảng (table).

Một bảng trình bày lớp các đối tượng quan trọng đối với một tổ chức. Mỗi bảng được xây dựng thành các cột và các dòng (được gọi là các thuộc tính và các bộ trong lý thuyết quan hệ). Mỗi cột trình bày vài thuộc tính của đối tượng được thể hiện trong bảng. Ví dụ, bảng **NHANVIEN** sẽ có các cột điển hình đối với các thuộc tính như Họ, Tên, mã nhân viên, phòng ban, bậc lương, công việc … Mỗi dòng trình bày một thể hiện của đối tượng thể hiện trong bảng. Ví dụ, một dòng trong bảng **NHANVIEN** trình bày thông tin về nhân viên có mã nhân viên là 12345.

Khi tổ chức dữ liệu vào các bảng, bạn có nhiều cách khác nhau để định nghĩa các bảng. Lý thuyết CSDL quan hệ định nghĩa một phương thực gọi là dạng chuẩn (normalization) mà nó đảm bảo rằng tập các bảng mà bạn định nghĩa sẽ được tổ chức dữ liệu một cách hiệu quả.

SQL

Để làm việc với dữ liệu trong một CSDL, bạn phải dùng một tập các lệnh và câu lệnh (một ngôn ngữ) được hỗ trợ bởi DBMS. Bạn có thể dùng vài ngôn ngữ khác nhau với CSDL quan hệ; thông thường nhất là SQL. Ngôn từ của SQL được hỗ trợ bởi SQL Server được gọi Transact-SQL, và Transact-SQL là ngôn ngữ chính được dùng bởi hệ quản trị CSDL SQL Server.

XML

Extensible Markup Language (XML) là một ngôn ngữ lập trình siêu văn bản, dùng để mô tả nội dung của của tập dữ liệu và cách dữ liệu sẽ được kết xuất ra thiết bị hoặc hiển thị trên trang Web.

SQL Server 2000 bắt đầu hỗ trợ cho XML. Các đặc tính mới này bao gồm:

- Khả năng truy xuất đến SQL Server thông qua URL.

- Hỗ trợ cho lượt đồ dữ liệu XML (XLM-Data schemas) và khả năng để chỉnh định các truy vấn XPath đối với các lược đồ này.
- Khả năng truy xuất và ghi dữ liệu XML:
- Truy xuất dữ liệu XML bằng cách dùng câu lệnh SELECT và mệnh đề FOR
- Ghi dữ liệu XML bằng OpenXML rowset provider.
- Microsoft SQL Server 2000 OLEDB provider (SQLOLEDB) cho phép tài liệu XML được gán như văn bản lệnh và trả về các tập kết quả như một luồng.

Các ấn bản của SQL Server 2000

Microsoft® SQL Server™ 2000 sẵn có các ấn bản sau:

SQL Server 2000 Enterprise Edition

Dùng như một sản phẩm database server. Hỗ trợ tất cả các đặc tính có sẵn trong SQL Server 2000, và phân chia các mức hiệu năng để hỗ trợ các trang Web lớn và các tiến trình giao tác trực tuyến của doanh nghiệp (online transaction processing OLTP) và các hệ thống kho dữ liệu.

SQL Server 2000 Standard Edition

Dùng như một database server cho các nhóm và bộ phận nhỏ.

SQL Server 2000 Personal Edition

Dành cho các người dùng lưu động - những người có thời gian không kết nối với network nhưng vẫn chạy ứng dụng mà đòi hỏi sự lưu trữ dữ liệu SQL Server. Cũng sử dụng khi đang chạy một ứng dụng stand-alone mà yêu cầu việc lưu trữ dữ liệu SQL Server trên máy tính client.

SQL Server 2000 Developer Edition

Danh cho các nhà lập trình phát triển ứng dụng mà dùng SQL Server 2000 như là nơi lưu trữ dữ liệu. Mặc dù Developer Edition hỗ trợ tất cả các đặc tính của Enterprise Edition cho phép người phát triển viết và kiểm tra các ứng dụng có thể sử dụng các đặc tính, Developer Edition chỉ cấp quyền sử dụng như là hệ thống phát triển hoặc kiểm tra ứng dụng, không phải là một sản phẩm server.

SQL Server 2000 Windows CE Edition

Microsoft® SQL Server 2000™ Windows® CE Edition (SQL Server CE) được dùng khi lưu trữ dữ liệu trên thiết bị Windows CE. Có khả năng đồng bộ hóa dữ liệu với bất kỳ ấn bản của SQL Server 2000 để sự đồng bộ hóa dữ liệu Windows CE với CSDL chính.

SQL Server 2000 Enterprise Evaluation Edition

Ấn bản với đầy đủ đặc tính, có thể tãi về từ trang Web. Với mục tiêu chỉ cho sử dụng ước lượng các đặc tính của SQL Server; phiên bản này sẽ ngưng chạy sau 120 ngày tãi về.

Ngoài các ấn bản này của SQL Server 2000, **SQL Server 2000 Desktop Engine** là một thành phần mà cho phép người phát triển ứng dụng phân tán một bản sao của SQL Server 2000 relational database engine với ứng dụng của họ. Trong khi các chức năng của database engine trong SQL Server 2000 Desktop Engine là tương tự như database engine trong các ấn bản SQL Server, kích cở của các Desktop Engine databases không vượt quá 2 GB.

1.2 Các thành phần của SQL Server 2000

Các thành phần của Server

- **SQL Server service**

  Thực thi SQL Server database engine. Có một SQL Server service cho mỗi thể hiện (instance) của SQL Server đang chạy trên máy tính.

- **SQL Server Agent service**

  Thực thi các tác nhân mà chạy các tác vụ quản trị SQL Server theo thời lịch. Chỉ có một SQL Server Agent service cho mỗi instance của SQL Server đang chạy trên máy tính. SQL Server Agent cho phép định nghĩa và lập lịch các tác vụ mà chạy dựa trên thời lịch hoặc tuần hoàn.

- **Microsoft Search service** (chỉ ở Windows NT và Windows 2000)

  Thực thi bộ máy tìm kiếm full-text (full-text search engine). Chỉ có một dịch vụ bất chấp số các instance SQL Server trên máy tính.

- **MSDTC service** (Chỉ ở Windows NT và Windows 2000)

  Quản trị các giao tác phân tán. Chỉ có một service, bất chấp số các instance SQL Server trên máy tính.

- **MSSQLServerOlAPService service** (chỉ ở Windows NT và Windows 2000)

  Thực thi SQL Server 2000 Analysis Services. Chỉ có một service, bất chấp số các instance SQL Server trên máy tính.

Các công cụ đồ họa (Graphical Tools)

- **SQL Server Enterprise Manager:** công cụ quản trị CSDL và server chính, nó cung cấp một giao diện Microsoft Management Console (MMC).
- **SQL Profiler:** tạo cơ hội các người quản trị một công cụ tinh vi để theo dõi và phân tích giao thông mạng đến và đi từ một server đang chạy SQL Server 2000.
- **SQL Query Analyzer:** dùng để tạo và quản trị các đối tượng CSDL và kiểm tra các câu lệnh Transact-SQL, các batch, script một cách tương tác.
- **SQL Server Service Manager:** được dùng để start, stop, và pause các dịch vụ của SQL Server.
- **Client Network Utility**: dùng để quản trị các client Net-Libraries và định nghĩa các bí danh server bao gồm các tham số kết nối server tuy chọn nếu cần.
- **Server Network Utility:** dùng để quản trị các server Net-Libraries.
- **SQL Server Books online:** là một tài liệu trực tuyến hỗ trợ với Microsoft® SQL Server™ 2000. Bạn có thể tìm thông tin trong SQL Server Books Online bằng cách:
  - Điều hướng thông qua bảng nội dung.
  - Gõ một từ khóa trong index.
  - Gõ một từ hoặc một cụm từ và thực hiện việc tìm kiếm.

1.3 Kiến trúc của CSDL quan hệ (relational Database Architecture)

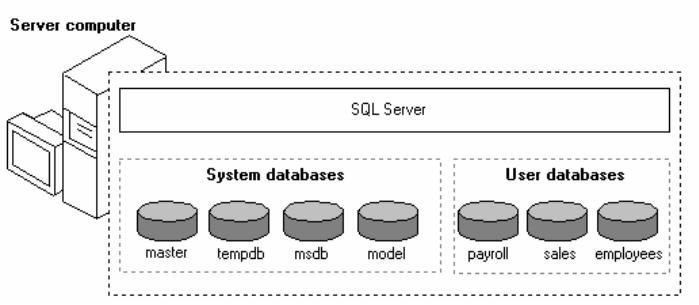
Trong SQL Server 2000, dữ liệu được lưu trong các CSDL. Về vật lý, một CSDL bao gồm hai hoặc nhiều hơn hai tập tin trên một hoặc nhiều đĩa, sự thể hiện vật lý này chỉ thấy được bởi nhà quản trị và nó trong suốt đối với người dùng. Sự chọn lựa vật lý của CSDL là trách nhiệm chính của người quản trị CSDL. Về mặc logic, một CSDL được xây

dựng thành các thành phần (component) được hiển thị đối với người dùng, chẳng hạn các bảng (Table), các khung nhìn (view) và các thủ tục (stored procudure). Sự chọn lựa logic của CSDL (chẳng hạn như thiết kế các bảng và các chỉ mục) là trách nhiệm chính của người thiết kế CSDL.

Các CSDL chứa các dữ liệu của người dùng được gọi là CSDL của người dùng (user database). Ngoài ra, các hoạt động của SQL Server tùy thuộc vào bốn CSDL hệ thống (system Database): **master, model, Tempdb,** và **msdb**. Các CSDL này tồn tại trong mỗi thể hiện của SQL Server.

CSDL hệ thống và CSDL người dùng

- CSDL **Master:** được dùng để điều khiển các thao tác và thông tin mức hệ thống của SQL Server. Lưu trữ thông tin hệ thống trong 16 bảng hệ thống, gọi là system catalog. System catalog lưu trữ các thông tin tài khoản người dùng, bao gồm các mục như là bảo mật, ID, mật khẩu, các CSDL lưu trên server, các biến môi trường, các thông điệp lỗi hệ thống, và các thủ tục hệ thống.
- CSDL **Model:** được dùng như là một CSDL mẫu cho tất cả các CSDL mới được tạo ra trong hệ thống. CSDL Model có thể được tùy biến để tạo một cấu trúc CSDL mặc định mới cho CSDL mới. Ví dụ, nếu một bảng cần phải tồn tại trong CSDL mới sau này thì các bảng đó nên được tạo trong CSDL model. Các CSDL mới vừa được tạo sẽ có kích cở ít nhất bằng kích cở CSDL model.
- CSDL **Tempdb**: được dùng như là vùng lưu chứa tạm thời đới với các bảng và các thủ tục tạm. SQL Server 2000 hỗ trợ hai loại bảng tạm: bảng tạm toàn cục (global temporary table) và bảng tạm cụ bộ (local temporary table). Tên của bảng tạm toàn cục được bắt đầu ##, có thể được truy xuất đối với tất cả các client, trong khi tên bảng tạm cụ bộ bắt đầu #, chỉ có thể truy xuất đối với những client mà tạo chúng. Làm việc trong Tempdb rất nhanh vì các hoạt động không được ghi nhận lại. Tuy nhiên, khi client kết thúc kết nối đến Server thì toàn bộ các bảng và thủ tục trong TempDB sẽ bị xóa.
- CSDL **Msdb:** SQL Server Agent dùng CSDL msdb cho các tác vụ khác nhau, như lập biểu, cảnh báo và ghi nhận các thao tác. Dữ liệu được lưu trong các bảng hệ thống trong CSDL msdb. Các bảng hệ thống được dùng bởi SQL Server Agent là *sysalerts, syscategories, sysdownloadlist, sysjobhistory, sysjobs, sysjobschedules, sysjobservers, sysjobsteps, sysnotifications, sysoperators, systargetservergroupmembers, systargetservergroups, systargetservers và systaskids*. Ngoài ra, CSDL msdb có 7 bảng dùng cho thao tác dự phòng và phục hồi dữ liệu: *backupfile, backupmediafamily, backupmediaset, backupset, restorefile, restorefilegroup, và restorehistory*. SQL Server có thể tự động tăng hoặc giảm kích cở của CSDL khi các dòng được thêm vào hoặc xóa đi.
- CSDL **Pubs, Northwind:** là hai CSDL ví dụ được dùng trong các tài liệu của SQL Server
- Các CSDL khác là các CSDL do người dùng tạo ra.



Hình 1: **Các CSDL hệ thống và CSDL của người dùng**

Khái niệm về cơ sở dữ liệu quan hệ: Databases, Tables, Columns, Views, Datatypes, và Database Schemas

Dữ liệu trong một RDBMS như SQL Server 2000 được sắp xếp trong một số đối tượng, dễ thấy nhất là bảng (table). Dữ liệu được lưu trong các Table theo dòng và cột. Dữ liệu liên quan đến các mục thực tế như các nhân viên, các sản phẩm, việc gửi hàng, người tham gia …. Được lưu trong các table riêng biệt. Ví dụ, hệ thống quản lý nhân viên của một công ty có thể có một bảng gọi là nhân viên, bảng này dùng để lưu chi tiết tất cả của các nhân viên trong công ty. Chi tiết của nhân viên bao gồm Họ nhân viên, tên nhân viên, địa chỉ, thành phố, số chứng minh nhân dân… Thỉnh thoảng cần ẩn các thông tin riêng tư từ người dùng bằng cách dùng một truy vấn. Các truy vấn là các câu lệnh SQL lưu trong CSDL và có thể được tham chiếu đến các câu lệnh SQL theo cách như các table. SQL Server cũng sử dụng các kiểu dữ liệu, mỗi cột có thể có các kiểu khác nhau. Tất cả các kiểu khác nhau này của các đối tượng, giống như bảng, truy vấn và kiểu dữ liệu được lưu trong một CSDL. Trong môi trường SQL Server, một lược đồ CSDL (database schema) là tập hợp các đối tượng CSDL có liên quan đến việc sử dụng một tên duy nhất và nó thuộc về một người dùng đơn lẻ.

Quản lý dữ liệu chính là lưu trữ với sự trợ giúp của một số đối tượng được cung cấp cung cấp bởi SQL Server 2000. Bảng 2 liệt kê các đối tượng chính trong một database của SQL Server 2000.

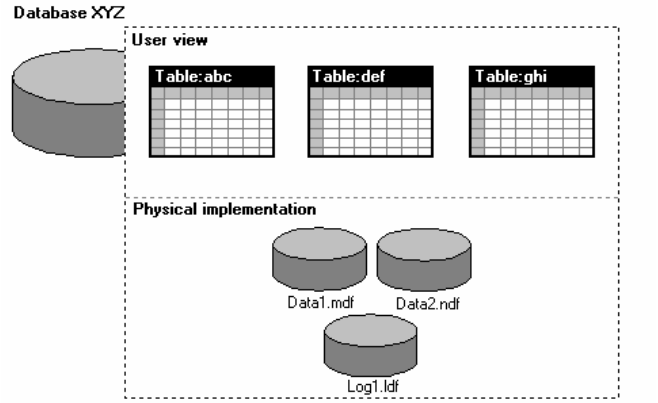
Bảng 1:

Đối tượng	Mô tả
Bảng - Table	Một table là một tập hợp của dữ liệu lưu trữ trong nhiều dòng nhiều cột.
Mặc định – Default	Một default là một cài đặt hệ thống, nghĩa là không được chỉ định bởi người dùng.
Cột - Column	Cột là các đối tượng dùng lưu trữ các phân tử riêng biệt bên trong một dòng dữ liệu.
Dòng – Row	Dòng là một mục đơn của dữ liệu bên trong một bảng, nó bao gồm tập các cột.
Kiểu dữ liệu - Datatype	Kiểu dữ liệu là các đối tượng dùng để xác định loại của dữ liệu có thể được lưu trong cột. Các loại dữ liệu khác nhau là tinyInt, smallInt, int, real, float, smalldatetime, datetime, smallmoney, money, char và sql_variant.
Chỉ mục – Index	Chỉ mục là một đối tượng của CSDL, dùng để tăng tốc các truy vấn bằng cách dò tìm dữ liệu theo giá trị khóa thay cho việc quét toàn bộ bảng.

	SQL Server 2000 hỗ trợ các chỉ mục clustered và non-clustered. Chỉ mục Clustered (Clustered indexe) là các chỉ mục mà sắp xếp các dòng theo cách dùng B-Tree.
Bẩy lỗi – Trigger	Trigger được ưa chuộng bởi vài nhà lập trình. Chúng chính là các thủ tục mà có thể chạy khi dữ liệu trong bảng được hiệu chỉnh như là cập nhập, xóa, chèn. Các trigger thường được dùng để đảm bảo tính thống nhất giữa các bảng liên kết. Các trigger có thể được tạo trên bảng hoặc truy vấn.
Quy tắt - Rule	Rule là đối tượng CSDL, nó được gắn kết với một cột hoặc một định nghĩa người nhằm hạn chế các giá trị. Ví dụ, một rule có thể đảm bảo số chứng minh nhân dân chỉ gồm các chữ số.
Thủ tục - Stored Procedures	Thủ tục là một tập các câu lệnh T-SQL được dịch trước (pre-compile) và được thực thi như một đối tượng. Do chúng đã được dịch trước nên các thủ tục sẽ thực thi nhanh.
Khung nhìn - View	View là một cách nhìn khác của table. Dùng để hiển thị các dữ liệu trong một hoặc nhiều table. Để sử dụng được view, người dùng phải có quyền trên view và tất cả các đối tượng phụ thuộc.

Cấu trúc vật lý của một CSDL

Mỗi CSDL bao gồm ít nhất một tập tin dữ liệu (data file) và một tập tin log (transaction log file). Các tập tin này không được chia xẻ với các CSDL khác. Để tối ưu hiệu năng và để hỗ trợ khả năng chịu lỗi (fault tolerance), tập tin dữ liệu và log được trãi dài trên nhiều đĩa và thông thường dùng một Raid (Redundant array of independent disks)



Hình 2: **Khung nhìn người dùng và triển khai vật lý của một CSDL**

Nhân bản - Replication

Nhân bản cho phép nhiều thể hiện của SQL Server ở vị trí từ xa có cùng dữ liệu. Vị trí từ xa có thể bao gồm các người dùng di động hoặc các site kết nối thông quan internet, dial-up hay intranet. Sự phân chia vật lý của dữ liệu cải thiện hiệu năng của tổ chức khi dữ liệu cần được xử lý tại các nơi khác để trình bày ở site khác để tham chiếu.

Bảo mật trong SQL Server.

SQL Server 2000 dùng hai mức bảo mật khi kiểm tra sự hợp lệ của một người dùng. Đó là chứng thực (authentication) và authorization

- **Authentication**

  Để kết nối đến một thể hiện của SQL Server 2000 thì người dùng phải chỉ định một định danh đăng nhập hợp lệ (ID). Một ID đăng nhập là một tài khoản định danh điều khiển sự truy xuất đến một thể hiện của SQL Server 2000. SQL Server kiểm tra ID đăng nhập có được phép kết nối đến thể hiện của SQL Server hay không. Sự kiểm tra của một ID đăng nhập được gọi là chứng thực (authentication). SQL Server 2000 dùng hai loại chứng thực: Windows authentication và SQL Server authentication. Hai chế độ chứng thực (Authentication Mode): Windows Authentication mode, mixed mode.

- **Authorization**

  Sau khi một ID - tài khoản đăng nhập được chứng thực, SQL Server 2000 xác định ID có được ủy nhiệm để thi hành các thao tác trong CSDL hay không. Một ID đăng nhập tự nó không đưa ra các quyền (permission) truy xuất đến các đối tượng trong một CSDL. Một ID phải có một sự ủy nhiệm hoặc có quyền hợp lệ. Điều này ngăn chặn một đăng nhập tự động truy xuất đến các CSDL trong một thể hiện của SQL Server 2000.

Chúng ta sẽ tìm hiểu bảo mật trong SQL Server kỹ hơn ở trong bài bảo mật.

1.4 Giới thiệu về Transact-Sql

Transact_SQL là ngôn ngữ dùng chủ yếu trong SQL Server.

- ✓ Có đầy đủ tính chất của một ngôn ngữ lập trình.
- ✓ SQL là một chuẩn do IBM đề ra và tất cả ngôn ngữ lập trình bổ sung thêm một số tính năng riêng của ngôn ngữ lập trình đó.
- ✓ Đối với SQL Server thì các ứng dụng muốn truyền thông với SQL Server đều phải các câu lệnh T-SQL đến Server.
- ✓ Câu lệnh T-SQL là tập hợp các đoạn mã mà thực thi một vài hành động lên các đối tượng hoặc dữ liệu của một CSDL. SQL Server cung cấp 3 loại câu lệnh T-SQL: DDL, DCL, và DML

Ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL): Dùng để định nghĩa và quản lý các đối tượng và đặc tính của đối tượng một CSDL. Lệnh DDL hỗ trợ việc định nghĩa, khai báo và chỉnh sửa các đối tượng của CSDL như Databases, Tables, Views. Mỗi lớp các đối tượng thì DDL có các lệnh như CREATE, ALTER, VÀ DROP (Ví dụ CREATE TABLE, ALTER TABLE, vaØ DROP TABLE).

Ngôn ngữ điều khiển dữ liệu (Data Control Language – DCL): Dùng để hiệu chỉnh các quyền (Permission) trên các đối tượng của CSDL. Các permission được điều khiển bởi lệnh các lệnh GRANT (gán quyền), REVOVE (huỹ bỏ quyền), DENY (từ chối quyền)

Ngôn ngữ thao tác dữ liệu (Data Manipulation Language – DML): Các lệnh này được dùng để truy vấn (SELECT), chèn (INSERT), cập nhật (UPDATE), xóa (DELETE) các dữ liệu của các đối tượng được tạo bởi DDL.

BÀI 2:
CÀI ĐẶT SQL SERVER 2000

2.1 Lập kế hoạch cài đặt SQL SERVER 2000

Các yêu cầu về phần cứng

Phần cứng	Yêu cầu tối thiểu
Máy tính	Intel hoặc tương thích. Pentium 166 MHz hoặc cao hơn.
Bộ nhớ (RAM)	Enterprise Edition: 64 MB; 128 MB hoặc hơn Standard Edition: 64 MB. Personal Edition: 64 MB đối với Windows 2000; 32 MB đối với các hệ điều hành khác. Developer Edition: 64 MB. Desktop Engine: 64 MB đối với Windows 2000; 32 MB đối với các hệ điều hành khác.
Không gian đĩa cứng	Các thành phần CSDL SQL Server: 95 đến 270 MB; 250 MB điển hình. Analysis Services: 50 MB; 130 MB điển hình. English Query: 80 MB. Desktop Engine only: 44 MB.
Màn hình	VGA hoặc độ phân giải cao hơn. 800 x 600 hoặc độ phân giải cao hơn SQL Server.
Thiết bị chuột	Microsoft Mouse hoặc tương thích.
CD	

2.2 Quyết định những chọn lựa cấu hình cài đặt SQL SERVER 2000

2.2.1 Xác định tài khoản người dùng (User account) thích hợp cho SQL Server Sevice và SQL Server Agent Service

Mỗi dịch vụ của SQL Server 200 chạy trong một ngữ cảnh bảo mật của một user account. Bạn có thể chọn hoặc **local system account** hoặc **domain user account**. Thông thường các Sevice đều chạy chung một user account.

Local system account là một account hệ thống của Win NT hoặc Win 2000 với đầy đủ các quyền quản trị trên máy tính cục bộ. Tuy nhiên, Account này không có

quyền truy cập mạng. Tuy nhiên, trong môi trường Client/Server, bạn nên tạo và dùng một domain user account có tính chuyên biệt dành cho các dịch vụ. Sự lựa chọn một domain user account cho phép những dịch vụ của SQL Server truyền thông với những cài đặt SQL Server khác, tài nguyên mạng (như là chia sẻ tập tin) trên những máy tính khác trong môi trường domain.

Domain user account mà bạn chọn phải có đầy đủ các quyền truy xuất trên máy tính cục bộ, nhưng không cần phải là thành viên của nhóm Administrator cục bộ hoặc dommain administrator. Những quyền được chỉ định này phải bao gồm quyền log on như là một dịch vụ, quyền truy xuất và thay đổi thư mục SQL Server, quyền truy xuất và thay đổi tập tin CSDL, đọc và ghi các khóa bất kỳ nào đó trong Registry của Windows. Bạn không cần lo lắng việc gán các quyền này, chương trình cài đặt SQL Server 2000 sẽ gán những quyền này một cách tự động cho domain user account mà bạn chỉ định.

2.2.2 Chọn một chế độ xác thực (Authentication Mode)

SQL server 2000 hỗ trợ 2 chế độ xác thực: Windows authentication Mode và Mixed mode. Mặc định là Authentication Mode.

Chế độ **Windows Authentication Mode** chỉ cho phép các user của hệ điều hành mới có thể kết nối với SQL Server. Chế độ **Mixed mode** cho phép tất cả các user của hệ điều hành hoặc SQL server đều có thể kết nối đến SQL Server.

Nếu bạn đang trong giai đoạn tìm hiểu và thử nghiệm SQL Server thì bạn nên chọn Mixed Mode. Nếu bạn triển khai ứng dụng thì bạn nên sử dụng Windows authentication mode, với chế độ này tính bảo mật được tăng cao nhờ tích hợp thêm các khả năng bảo mật của hệ điều hành.

2.2.3 Xác định Collation

Collation là tập hợp những qui tắt quản lý, nó sẽ ảnh hưởng đến cách lưu trữ dữ liệu, thứ tự sắp xếp… Ví dụ khi bạn cài đặt hệ điều hành, bạn chọn lựa ngôn ngữ, bàn phím được sử dụng. Mỗi ngôn ngữ sẽ có bộ ký tự khác nhau do đó có những code page khác nhau và dựa trên cơ sở này hệ điều hành sẽ cài đặt các numbers, currencies, Times, Dates khác nhau. Khi bạn cài đặt SQL Server, chương trình cài đặt sẽ xác định một collation mặc định cho SQL Server dựa trên các cài đặt của hệ điều hành.

2.2.4 Chọn Network Libraries

SQL Server sử dụng Network libraries để gửi các packet giữa SQL Server khách và chủ. Server và Client phải có ít nhất một Network Libraries chung.

Các Network Libraries: NEBEUI, TCP/IP, IPX/SPX, Apple Talk ADSP, Banyan Vines, VIA giginet San.

2.2.5 Xác định client Licensing Mode

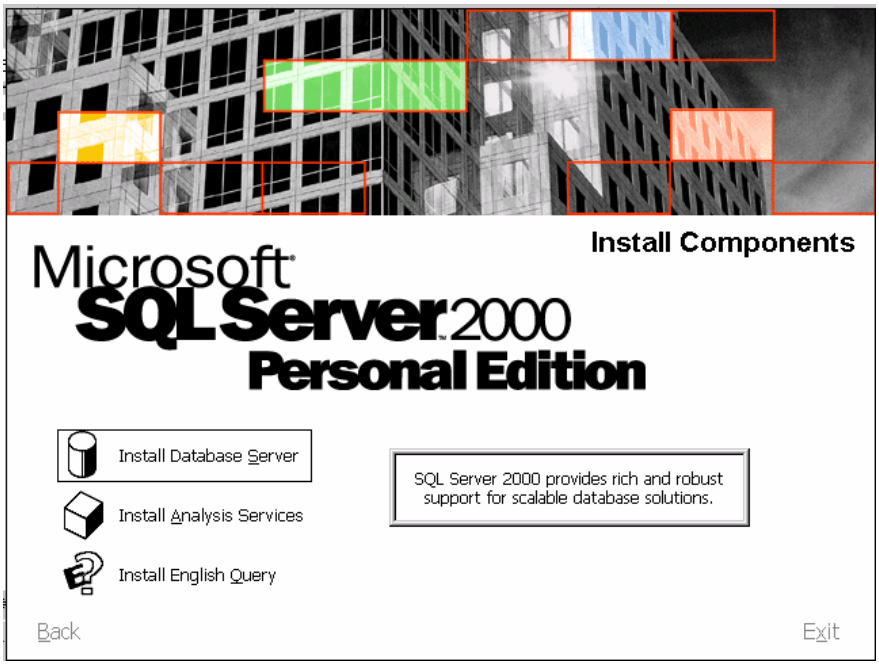
SQL Server cung cấp 2 kiểu client licensing: Per processor và Per seat.

2.3 Các bước cài đặt SQL Server 2000.

Bạn đưa điã CD chứa chương trình cài đặt SQL Server hoặc chạy tập tin Setup từ một vị trí nào đó trên mạng.

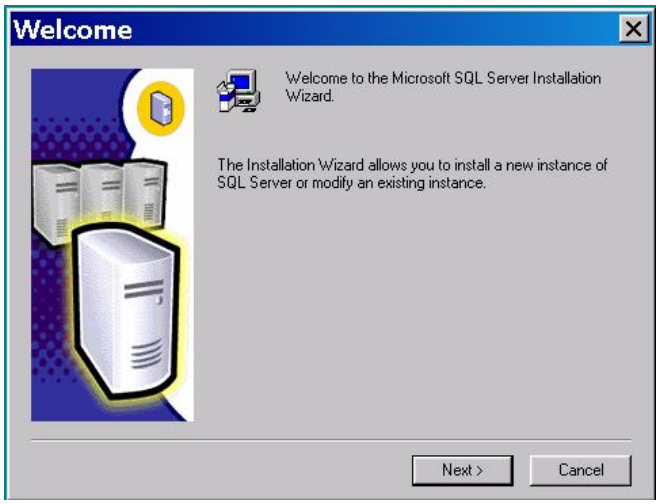
Hình 3: Màn hình đầu tiên khi đưa đĩa SQL Server Persional Edition vào

Chọn SQL Server 2000 Components



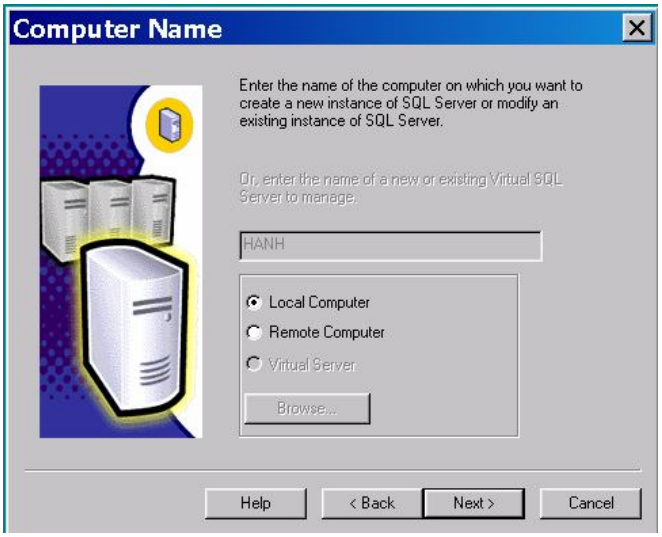
Hình 4: Chọn thành phần cần cài đặt

Chọn Install Database Server



Hình 5: Màn hình Welcome của MicroSoft SQL Server

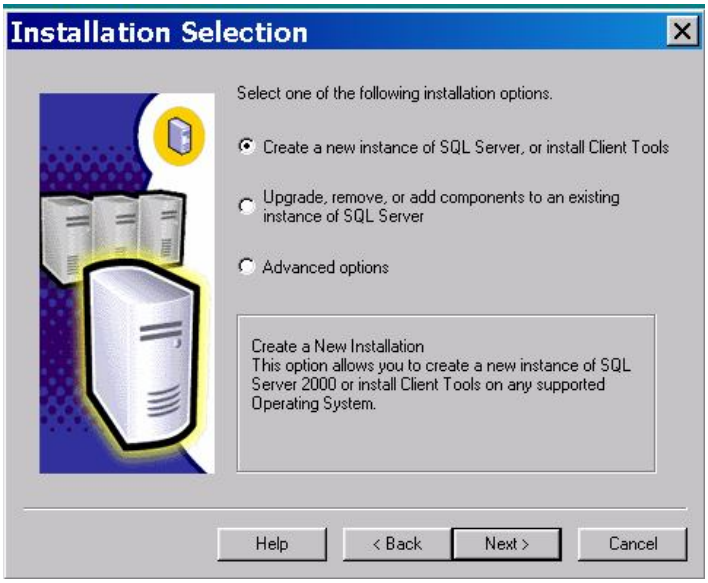
Chọn Next



Hình 6: Chọn tên máy tính nơi mà sẽ cài 1 instance mới hoặc hiệu chỉnh instance có sẵn

Local Computer : Cài trên máy Local
Remote Computer : Cài đặt từ xa
Virtual Server : Cài trên Server ảo

Chọn Local Computer, chọn Next



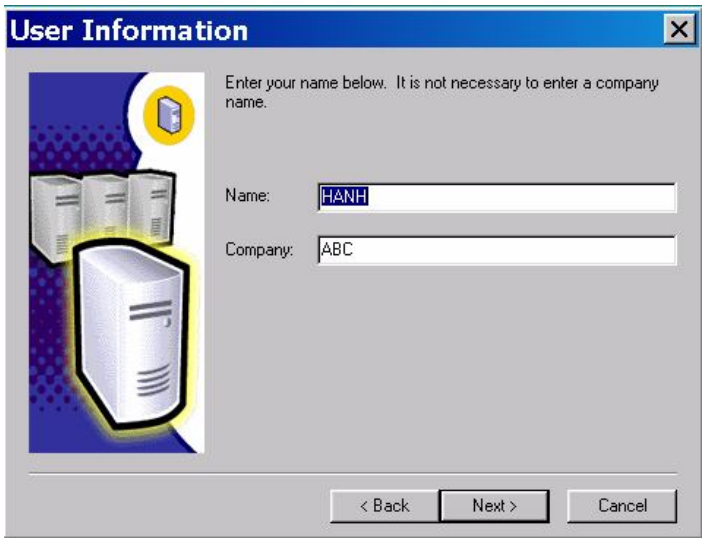
Hình 7: Chọn hình thức cài đặt

Create a new instance of SQL Server, or install client tools: Cài một instance mới trên server hoặc cài các client tools

Upgrade, remove, or add components to an existing instance of SQL Server: Nâng cấp, hủy bỏ, hoặc thêm các component vào instance có sẵn.

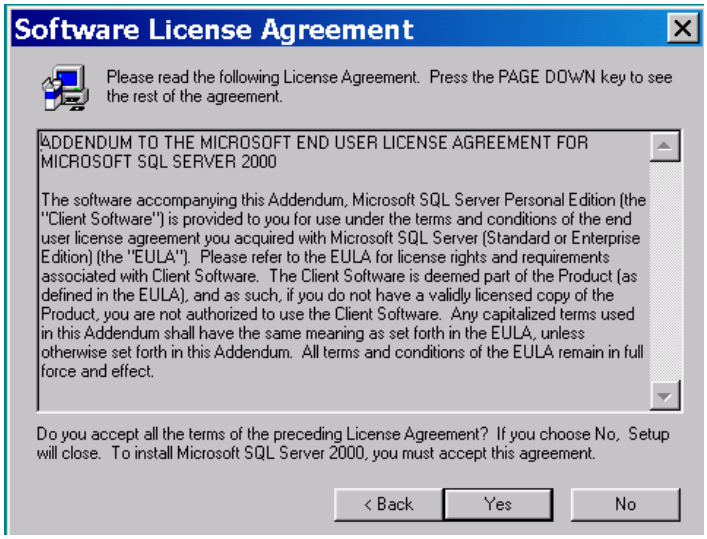
Advanced option: Một số chức năng nâng cao hoặc tạo tập tin cài đặt không tham dự.

Chọn Create a new instance of SQL Server, or install client tools**,** chọn Next



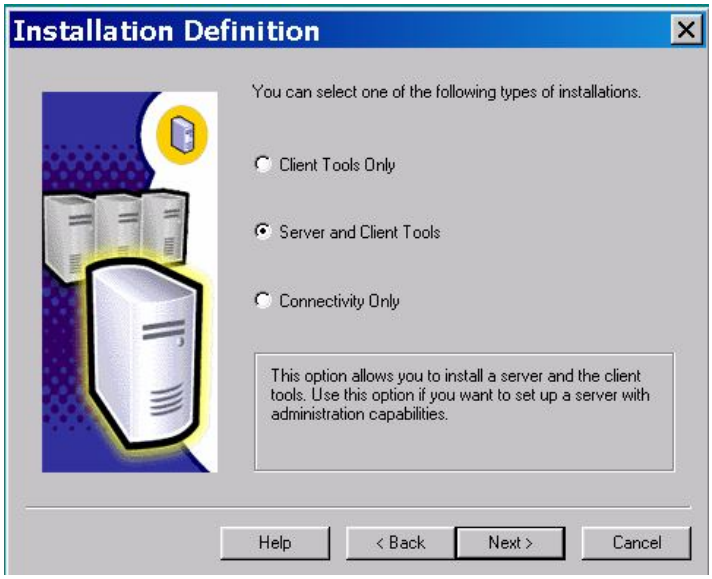
Hình 8: Thông tin của người dùng

Chọn Next



Hình 9: Thông báo Lincense

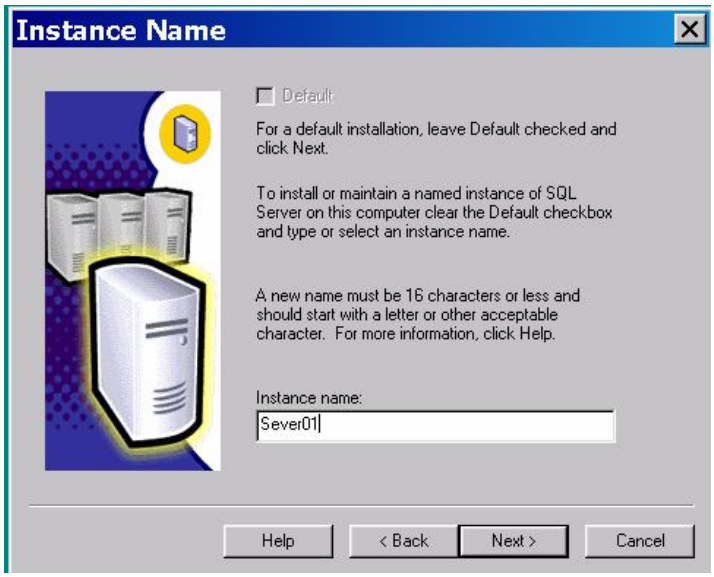
Chọn next



Hình 10: Xác định công cụ cài đặt

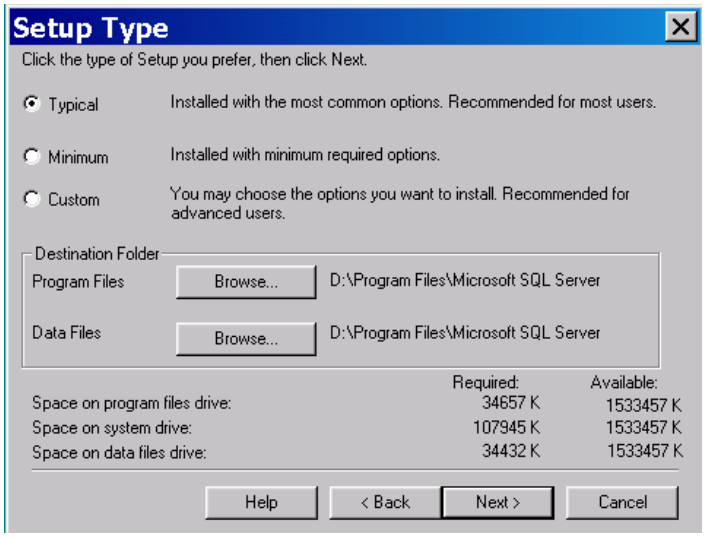
Client Tools Only	: Chỉ cài đặt các tools của client
Server and Client Tools	: Cài đặt các tools của Server và Client
Connectivity Only	: Cài đặt các thành phần truy xuất dữ liệu và các thư viện network

Chọn Server and ClientTools



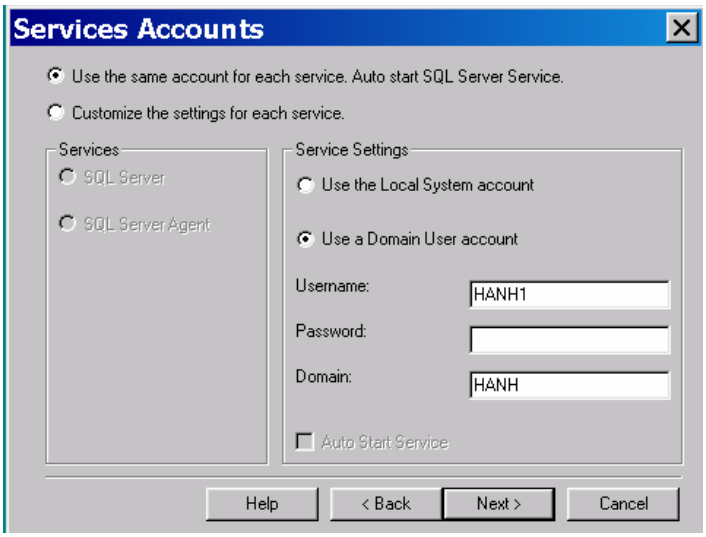
Hình 11: Qui định tên của instance

Gỏ vào Server01, chọn Next



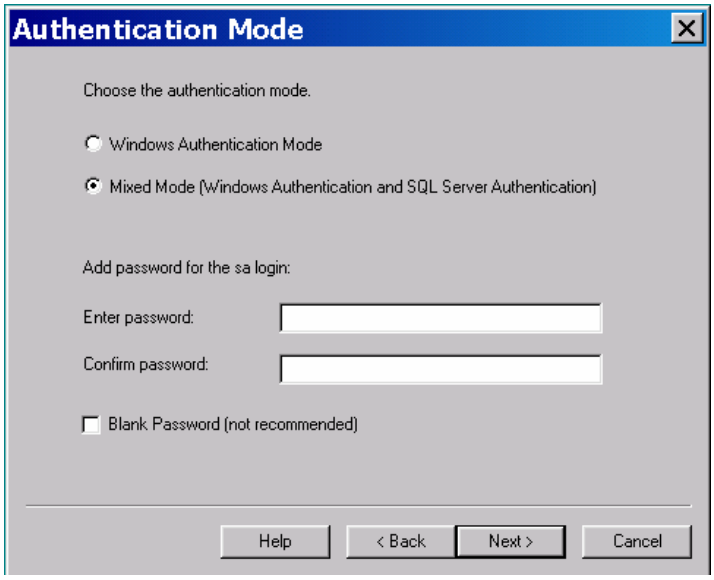
Hình 12: Chọn loại cài đặt

Chọn Typical, chọn Next



Hình 13: xác định Serverice Account để chạy các Service

Chọn Next

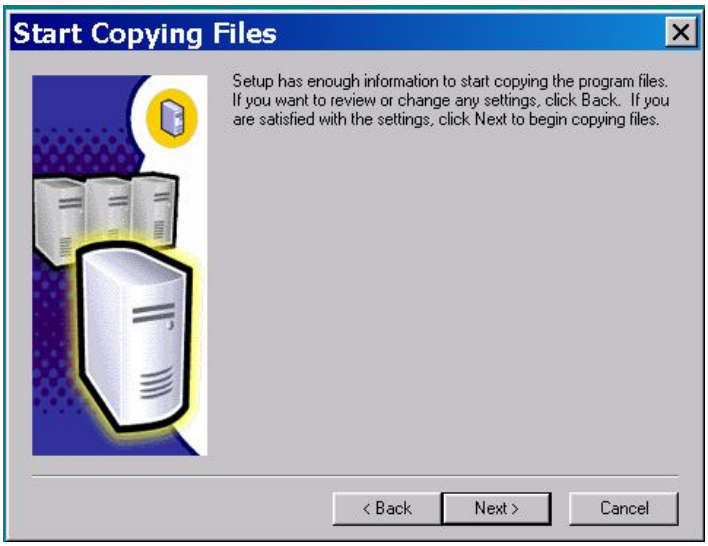


Hình 14: Xác định chế độ chứng thực

Windows Authentication Mode: chế độ xác thực của Windows

Mixed Mode: Chế độ xác thực vừa của Windows và vừa của SQL Server

Chọn Mixed Mode



Hình 15: Thông báo là SQL Server sẽ bắt đầu được cài đặt

Chọn Next, tiến trình cài đặt bắt đầu

Hình 16: Kết thúc cài đặt

2.4 Tạo tập tin cài đặt không tham dự (unattended) và cài đặt từ xa (Remote) SQL Server 2000

Có 3 cách để tạo ra tập tin khởi đầu cài đặt dành cho việc cài SQL Server 2000 không tham dự.

Cách 1: Chương trình Setup SQL Server 2000 cung cấp lựa chọn Advanced Option để ghi nhận lại tập tin không tham dự .ISS. Nếu lựa chọn này được chọn thì quá trình cài đặt sẽ được ghi nhận lại thành tập tin .ISS trong thư mục \Winnt, và SQL Server không được cài đặt thực sự trong tiến trình này.

Cách 2: tập tin .ISS được cung cấp trong đĩa SQL Server 2000, trong thư mục gốc. Bạn có thể dùng trực tiếp từ các tập tin này hoặc hiệu chỉnh bằng các trình soạn thảo văn bản. Tập tin đó là SQLINS.ISS, SQLCLI.ISS; SQLCST.ISS

Cách thứ 3: Tập tin .ISS được tự động tạo ra ngay sau khi mỗi lần bạn cài đặt SQL Server, nó nằm trong \WinNt. Tuy nhiên, nếu dùng tập tin này thì bạn phải chỉnh sửa nó nằng một trình soạn thảo văn bản để thêm phần [SdFinish-0]

2.5 Kết quả của việc cài đặt SQL SERVER 2000

Trình cài đặt SQl server tạo ra một loạt các thư mục để lưu các tập tin thi hành và cấu hình khác nhau. Theo ngầm định, SQL Server được cài đặt trên ổ đĩa C: trong thư mục MSSQL. Dưới thư mục MSSQL là vài thư mục khác:

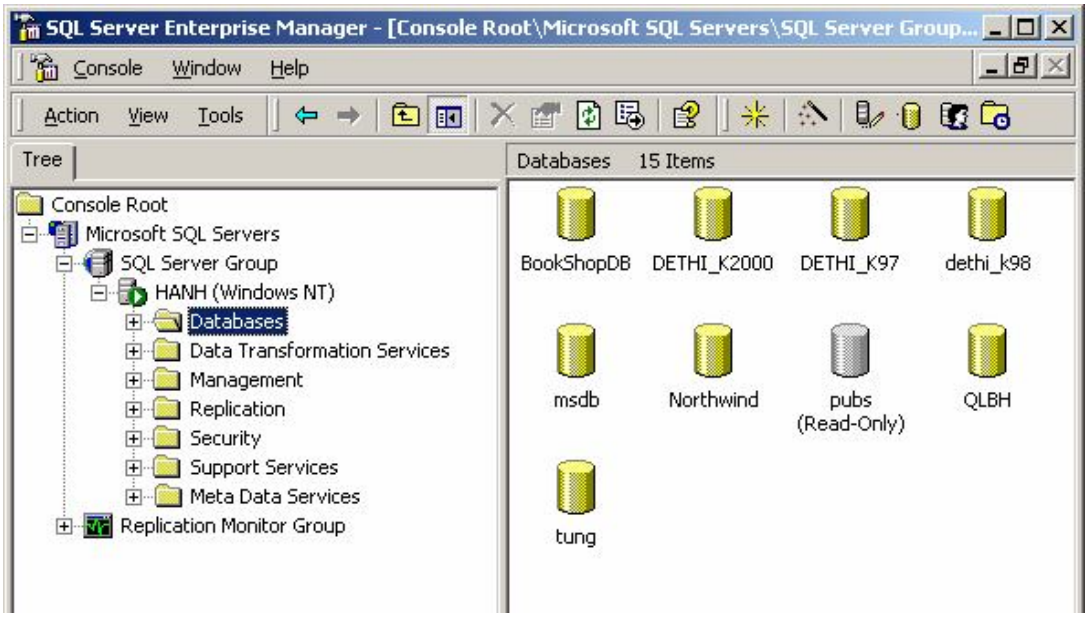
Backup	Thư mục vẫn còn trống ngay sau khi cài đặt. Thư mục này dùng để lưu trữ các tập tin dự phòng.
BIN	Các thư viện mạng phía client
BINN	Các tập thi hành và các tập tin kết hợp. Kể cả các tập tin thi hành chính của SQL Server, và mọi công cụ điều hành được nạp.
CHARSETS	Các bộ ký tự sắp xếp cho các kiểu cài đặt khác nhau.
DATA	Vị trí ngầm định cho các CSDL.
INSTALL	SQL Server Books Online và các chỉ mục.
LOG	Các sổ theo dõi lỗi SQL Server. Các tập tin văn bản tương đương với sổ theo dõi sự kiện Windows NT, nhưng chi tiết hơn.
REPLDATA	Vị trí ngầm định cho dữ liệu tạm thời được dùng trong khi sao lặp.
SNMP	Các MIB (Management Information Bases = cơ sở thông tin quản trị) cho SQL Server.
SQLOLE	các mẫu về các dùng OLE automation của Visual Basic để quản lý SQL Server.
SYSBOLS	Gỡ rối các ký hiệu do các lập trình viên sử dụng.

BÀI 3:
CÁC CÔNG CỤ CỦA SQL SERVER

3.1 SQL ENTERPRISE MANAGER

Enterprise Manager còn gọi tắt là EM, là một công cụ chính dành cho nhà quản trị server và CSDL. Enterprise Manager cho phép bạn dừng và khởi động một Server, cũng như cho phép bạn thực hiện các tác vụ sau:

- ✓ Đăng ký một server.
- ✓ Cấu hình các server cục bộ hoặc từ xa.
- ✓ Cấu hình và quản lý một cài đặt với nhiều server (multiple-server).
- ✓ Cài đặt các bảo mật đăng nhập (login security), thêm các người dùng (user), các nhà quản trị hệ thống (system administrator), và các điều hành viên.
- ✓ Gán một password nhà quản trị hệ thống.
- ✓ Tạo và lập biểu cho các công việc (job).
- ✓ Tạo các cảnh báo và cấu hình giao tiếp đến nhà quản trị hệ thống thông qua e-mail.
- ✓ Cài đặt và quản trị các CSDL, các bảng (table), các chỉ mục (index), các truy vấn (view), các thủ tục (stored procedure), các qui tắt (rule), các bẫy lỗi (trigger), các mặc định (default), các thiết bị dự phòng (backup device), và các vết lỗi (error log).
- ✓ Quản lý các service khác.



Hình 17: Cửa sổ Enterprise Manager của SQL Server

3.2 SQL SERVER SERVICES MANAGER

SQL Server Manager là một công cụ cho phép khởi động, tạm dừng hoặc dừng các dịch vụ (service) trong SQL Server. SQL Server hoạt động được thông qua các dịch mà nó tự cung cấp.

3.2.1 Các dịch vụ của SQL Server

Một dịch vụ (*service)* là một chương trình hoặc tiến trình thực thi một chức năng đặc biệt nào đó nhằm hỗ trợ cho các chương trình khác. SQL Server cung cấp các dịch vụ sau: *SQL Server Service, SQL Server Agent Service, MicroSoft Search Service và Microsoft Distributed Transaction Coordinator*. Mỗi dịch vụ có chức năng và nhiệm vụ riêng, hỗ trợ cho các hoạt động của SQL Server. Nếu bạn cài đặt một hoặc nhiều hơn thể hiện (instance) của SQL Server thì tên của service cho mỗi instance của SQL Server là MSSQL$*InstanceName*, $*InstanceName* là một tên của instance mà bạn chỉ định ở lần cài đặt. Ứng với mỗi SQL Server Agent service cho mỗi instance được gọi là SQLAGENT$*InstanceName*. Tuy nhiên, nhiều instances của SQL Server, sẽ chỉ có một Microsoft Distributed Transaction Coordinator và Microsoft Search.

SQL SERVER SERVICE

Khi bạn khởi động SQL Server có nghĩa là dịch vụ SQL Server service được khởi động ở Windows NT hoặc Windows 2000. Dịch vụ này quản lý các tập tin CSDL, xử lý các câu lệnh T-SQL, định vị tài nguyên giữa các kết nối của người dùng hiện hành, đảm bảo tính nhất quán dữ liệu, và nhiều hơn nữa.

SQL SERVER AGENT SERVICES

SQL Server Agent hỗ trợ việc lập biểu và thực thi các công việc (job), các cảnh báo (alert), thông báo, và kế hoạch duy trì CSDL. Không có service này, công việc quản trị của bạn sẽ trở nên khó khăn nhiều. SQL Server Agent cho phép bạn thực hiện tự động các thao tác nhằm duy trì CSDL.

Ví dụ: bạn có thể tạo một job để thực hiện tự động dự phòng (backup) dữ liệu mỗi đêm vào lúc 1 giờ sáng và một job khác thực hiện dự phòng transaction log mỗi 30 phút một lần.

Để kiểm tra hiệu năng hệ thống của bạn cảnh báo hiện trạng hiệu năng để báo cho bạn nếu server CPU đã hoạt động trên 90%. SQL Server Agent phải chạy để thực thi các tác vụ kiểu như vậy.

Dịch vụ này có thể được cấu hình khởi chạy một các tự động hoặc chạy một cách thủ công. Bạn nên cấu hình cho nó khởi động tự động để đảm bảo rằng các job, alerts, và notification sẽ có thể được thực thi.

MICROSOFT DISTRIBUTED TRANSACTION COORDINATOR.

Dùng quản lý các giao tác phân tán.

MICROSOFT SEARCH.

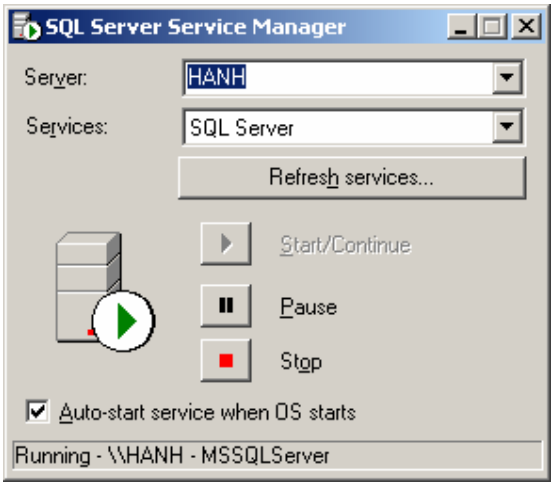
Dịch vụ Microsoft Search cho phép tạo một chỉ mục full-text và cơ chế tìm kiếm. Chuẩn SQL-92 định nghĩa các khả năng tìm kiếm với phép so sánh ký tự bằng, nhỏ hơn, lớn hơn một hằng số ký tự, giá trị ký tự có chứa mẫu chuỗi.

Dùng dịch vụ Microsoft Search cho phép Microsoft® SQL Server™ 2000 và SQL Server 7.0 được hỗ trợ nhiều tìm kiếm tinh vi trên cột chuỗi ký tự.

3.2.2 Khởi động, tạm ngưng, dừng các dịch vụ của SQL Server

Để khởi động hay dừng các dịch vụ SQL Server bằng cách dùng SQL Server Service Manager thì thực hiện các bước sau:

1. Click chọn **Start → Programs → Microsoft SQL Server**, và chọn **Service Manager** để hiển thị công cụ SQL Service Manager như hình



Hình 18: SQL Server Service Manager.

2. Tên server cục bộ xuất hiện trong mục **Server** và các dịch vụ của SQL Server xuất hiện trong mục **Service.** Trong danh sách xổ xuống, chọn tên của server và dịch vụ mà bạn muốn điều khiển.

   Bạn có thể khởi động (start) và dừng (stop) các service đang được chọn bằng cách click vào ứng tương ứng. Nếu Service ở trạng thái dừng thì click vào Start/Continue để chạy tiếp.

3. **Auto-start service when OS Server:** cấu hình tự động chạy dịch vụ khi hệ điều hành khởi động.

Lưu ý:

Nếu SQL Server và SQL Server Agent service không được cấu hình chạy tự động thì bạn phải khởi động nó một cách thủ công

3.3 OSQL

OSQL là một dấu nhắc tiện ích dùng để truy vấn một instance của SQL Server 2000 một cách tương tác bằng T-SQL, thủ tục hệ thống, tập tin lệnh. Nó cũng được dùng để xem xét các công việc hoặc bó lệnh, kể cả các lệnh của hệ điều hành đối với SQL Server 2000. Bạn có thể dùng osql bằng cách kết nối vào server và

thực thi các lệnh trong chế độ tương tác hoặc bằng cách kết nối vào và thực thi các lệnh như một phần của cú pháp lệnh osql.

Cú pháp lệnh của osql như sau:

Osql –S servername – U login_id – P password (1)

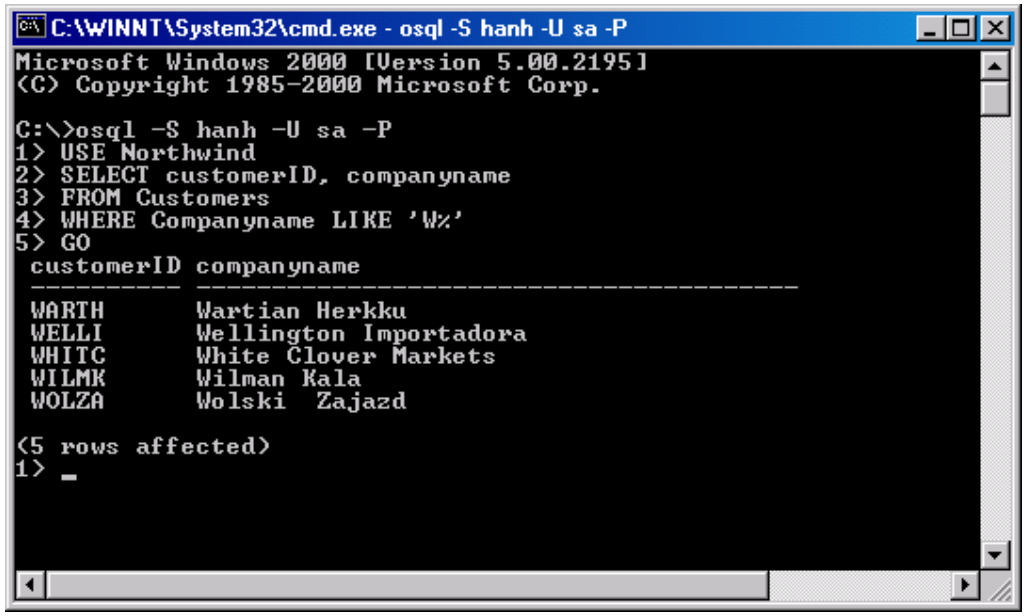
Osql –S servername – E (2)

Chúng ta dùng (1) khi bạn kết nối đến server bằng một tài khoản đăng nhập của SQL server, dùng (2) khi bạn kết nối bằng tài chế độ chứng thực là Windows. Khi dùng Osql để kết nối đến SQL Server 2000, có nhiều tham số mà bạn có thể dùng như là một phần của chuỗi kết nối. Nên nhớ rằng, Osql phân biệt chữ thường, chữ hoa.

- S servername	Chỉ định tên của Server của SQL Server mà bạn muốn kết nối.
- U login_id	Chỉ định tên của tài khoản đăng nhập
- P password	Chỉ định password của tài khoản đăng nhập nếu có
- E	Kết nối bằng tài khoản đăng nhập hiện hành của windows

Khi bạn dùng osql trong chế độ tương tác, nó hiện thị thứ tự các dòng. Chúng ta gõ các câu lệnh SQL, mỗi câu lệnh trên mỗi dòng. Osql không thực thi câu truy vấn của bạn cho đến khi bạn gõ từ khóa GO trên một dòng. Không làm việc bằng osql, gõ EXIT để tắt kết nối và đóng tiện ích osql.

Ví dụ:



Hình 19: Minh họa tiện ích osql

3.4 SQL QUERY ANALYZER

3.4.1 Giới thiệu.

SQL Query Analyzer là giao diện người dùng đồ họa (Graphical User Interface – GUI) dành cho các nhà lập trình. Query Analyzer cho phép thực hiện:

- Tạo các truy vấn (query), bó lệnh (script) và thực thi (execute) chúng để tác động đến CSDL của SQL Server.

- Tạo các đối tượng của CSDL một cách nhanh chóng từ những script được định nghĩa trước.
- Sao chép nhanh chóng các đối tượng của CSDL
- Tạo và thực thi các thủ tục (Stored procedures), hàm người dùng (user-defined function)
- Tìm lỗi (Debug) các thủ tục.
- Tìm lỗi (Debug) các vấn đề hiệu năng của của truy vấn *(Show Execution Plan, Show Server Trace, Show Client Statistics, Index Tuning Wizard).*
- Định vị các đối tượng trong các CSDL, xem và làm việc với các đối tượng.
- Chèn, cập nhật, xóa các mẫu tin trong table một cách nhanh chóng.

3.4.2 Khởi động Query Analyzer

SQL Query Analyzer có thể được khởi động từ SQL Server Enterprice Manager hoặc từ Start Menu hoặc từ của sổ Command bằng cách thực thi tiện ích ISQLW.

Khi bạn khởi động SQL Query Analyzer, thì hộp thoại Connect to SQL Server xuất hiện. Khi đó bạn phải xác định chế độ chứng thực được dùng để kết nối tới SQL Server.

Hình 20: Hộp thoại kết nối

Bạn có thể mở cùng lúc nhiều cửa sổ Query Analyzer để cho phép bạn làm việc trong các CSDL khác nhau hoặc thực thi các script khác nhau trong cùng một thời điểm. Mỗi cửa sổ tạo kết nối riêng biệt đến server. Những kết nối duy trì các cài đặt khác nhau và mỗi cửa sổ có CSDL hiện hành. Nếu bạn cố gắng thực thi một thao tác dành riêng trên một CSDL từ một query analyzer trong khi một query analyzer khác đang dùng CSDL, thao tác sẽ bị lỗi.

3.4.3 Thành phần chính của Query Analyzer

SQL Query Analyzer bao gồm các cửa sổ, hộp thoại, hướng dẫn (wizard) giúp chúng ta thiết kế các tác vụ (Task) cần thiết để tạo các CSDL, lưu trữ, khai thác dữ liệu trong các CSDL đó.

Cửa số Query Analyzer

- **Thanh tiêu đề** (Title bar): Hiển thị tên của Server, CSDL hiện hành, và tài khoản kết nối tới.
- Công cụ **Database** trên thanh công cụ, cho biết và cho phép bạn thay đổi CSDL được kết nối hiện hành.
- **Editor pane**: dùng để đưa vào các câu lệnh và thực thi các câu lệnh T-SQL

Màu của mã lệnh trong Query Analyzer:

Màu	Ý nghĩa
Đỏ	Chuỗi ký tự
Đỏ đậm	Thủ tục
Xanh lá	Bảng hệ thống
Xanh lá đậm	Chú giải
Đỏ tươi	Hàm hệ thống
Xanh	Từ khóa
Xám	Toán tử

Màu của mã lệnh bạn có thể biết là bạn nhập câu lệnh vào là đúng hay sai. Bạn có thể đổi màu qui định bằng các chọn trang fonts trong hộp thoại Tools → Option.

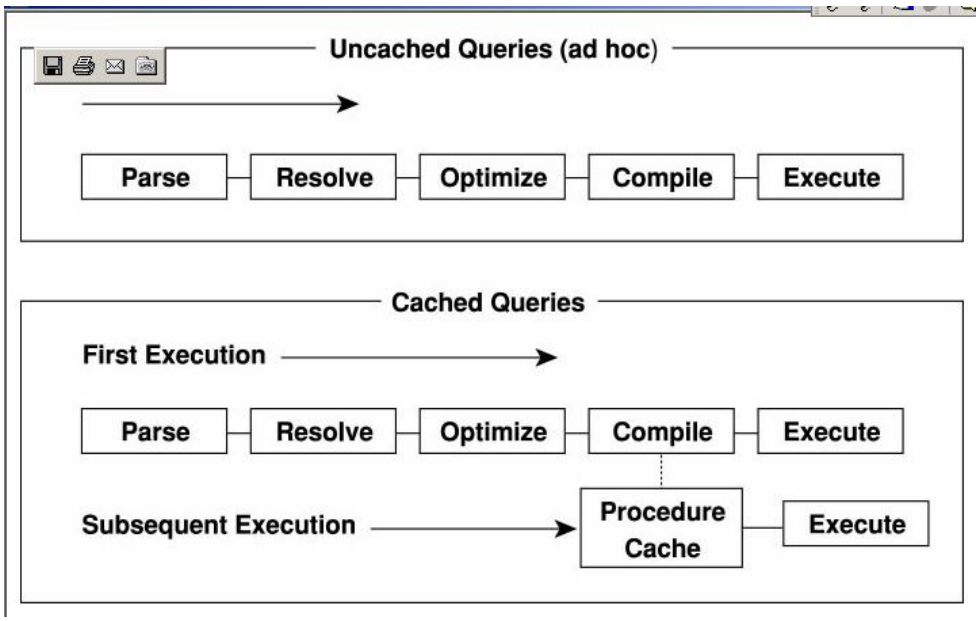
Results pane*:* hiển thị kết quả của truy vấn được thực thi. Một hoặc nhiều hơn một trang được hiển thị trong results pane.

Trang Messages: Hiển thị thông báo và các lỗi gửi trả từ server.

Trang Results: Hiển thị những kết quả như văn bản tự do.

Trang Results Grid: Hiển thị các kết quả trong bảng kẻ lưới. Dữ liệu trong lưới chỉ để xem, không thể hiệu chỉnh được. Một vài truy vấn yêu cầu server trả về nhiều hơn một tập kết quả thì sẽ có nhiều hơn một trang Results Grid sẽ được hiển thị. Nếu lưới kết quả trống sau khi thi hành một truy vấn thì truy vấn không được trả về một bảng kết quả

Trang Execution Plan: Hiển thị một biểu đồ của kế hoạch thực thi của truy vấn hiện hành. Bật tắt Execution Plan ta chọn từ thực đơn Query



Hình 21: Thành phần chính của Query Analyzer.

Lưu ý:

- Bạn có thể nhập vào chuỗi các câu lệnh mới hoặc mở một tập lệnh có sẵn. Khi bạn đã làm việc xong với tập lệnh bạn có thể lưu nó thành một tập tin để dàng dùng lại sau (tập tin có phần mở rộng là .SQL).
- Truy vấn có thể là một câu lệnh đơn hoặc nhiều câu lệnh. Những câu lệnh có thể không thể thực thi như là một phần của cùng một truy vấn với những câu lệnh khác. Trong trường hợp này được viết cách nhau bởi từ khóa GO.
- Những câu lệnh có thể gõ trên cùng một hàng hoặc trãi dài qua nhiều hàng. Do câu lệnh T-SQL thì quá dài để đặt trên một dòng, nên ta gõ chúng trên nhiều dòng, điều này sẽ làm cho chúng ta dễ đọc các câu lệnh.
- Nếu không có lệnh được chọn thì khi bạn thi hành truy vấn thì toàn bộ nội dung của query pane sẽ được thực thi

3.4.4 Một vài phím nóng dùng trong Query Analyzer

Hành động	Phím nóng
Thực thi	CTRL+E or F5
Tìm kiếm	CTRL+F
Chuyển văn bản đang chọn thành chữ hoa	CTRL+SHIFT+U
Chuyển văn bản đang chọn thành chữ thường	CTRL+SHIFT+L
Các kết quả dạng văn bản	CTRL+T
Những kết quả trong lưới	CTRL+D
Giúp đỡ về Query Analyzer	F1

Giúp đỡ một câu lệnh T-SQL được chọn	SHIFT+F1

BÀI 4:
LÀM VIỆC VỚI CỞ SỞ DỮ LIỆU SQL SERVER

4.1 Thiết kế một cơ sở dữ liệu.

Thiết kế CSDL là một bước vô cùng quan trọng trong tiến trình phát triển ứng dụng. Trong quá trình thiết kế, bạn phải quyết định những table nào cần để lưu trữ các dữ liệu của bạn. Quá trình thiết kế một CSDL logic được thực hiện độc lập với các hệ quản trị CSDL. Một số lưu ý khi thiết kế một CSDL logic:

- Các bảng và tên của chúng (còn gọi là các thực thể).
- Tên các cột (các thuộc tính) của mỗi bảng.
- Các đặc tính của cột như là giá trị duy nhất, cho phép null hay không, và kiểu của dữ liệu mà cột sẽ lưu chứa.
- Khóa chính (Primary key) cho mỗi bảng. Đó là một cột hoặc tập các cột mà chứa các giá trị được định nghĩa không trùng lặp ở các dòng trong bảng. Mỗi bảng chỉ có thể có một khóa chính. Mặc dù khóa chính là không yêu cầu bắt buộc phải có nhưng các bảng nên luôn luôn có.
- Các mối quan hệ (Relationship) giữa các bảng. Các dòng trong một bảng phụ thuộc một hoặc nhiều dòng khác trong bảng khác. Những phụ thuộc trong những bảng này được gọi là mối quan hệ. Để định nghĩa mối quan hệ, một cột hoặc tập các cột trong một bảng được gọi là khóa ngoại (foreign key) nếu nó tham chiếu đến khóa chính của bảng khác.

Ví dụ: mỗi dòng trong một bảng đơn hàng (DONHANG) phụ thuộc vào một dòng trong bảng khách hàng (KHACHHANG) bởi vì mỗi đơn hàng phải được đặt bởi một khách hàng. Điều này chính là nối quan hệ giữa bảng DONHANG và bảng KHACHHANG. Bảng đơn hàng phải có một cột lưu giữ các giá trị được tham chiếu đến một dòng riêng lẻ trong bảng KHACHHANG. Các dòng trong bảng HOADON phải được đảm bảo chỉ tham chiếu đến một khách hàng vì vậy mối quan hệ nên dựa trên cơ sở khóa chính của bảng KHACHHANG. Cột của bảng đơn hàng mà tham chiếu đến khóa chính của bảng khách hàng được gọi là khóa ngoại.

Các kiểu mối quan hệ: Ba kiểu của mối quan hệ có thể giữa các bảng:

- **One-to-One.** Mỗi dòng trong bảng chính có quan hệ đến chỉ một dòng trong bảng quan hệ. Một mối quan hệ one-to-one là được thực hiện bởi định nghĩa khoá ngoại là duy nhất (không trùng).
- **One-to-Many.** Mỗi dòng trong bảng chính được liên quan đến một hoặc nhiều dòng trong bảng quan hệ. Ví dụ một khách hàng có thể đặt nhiều đơn hàng, nhưng một đơn hàng không thể được đặt bởi nhiều khách hàng.
- **Many-to-Many.** Nhiều dòng trong một bảng liên quan đến nhiều dòng trong bảng khác.

Ví dụ: một tác giả có thể viết nhiều quyển sách và một quyển sách có thể được viết bởi nhiều hơn một tác giả. Mối quan hệ many-to-many giữa 2 bảng là thực hiện bằng cách tạo một bảng thứ ba và tạo một mối quan hệ one-to-many đến bảng chức năng này từ mỗi bảng ban đầu.

Một CSDL sau khi thiết kế được đánh giá thông qua các dạng chuẩn. CSDL đạt chuẩn cao thì CSDL đó lưu trữ đầy đủ thông tin, không bị trùng lấp, có tính nhất quán cao.

Các dạng chuẩn của CSDL quan hệ

- **Dạng chuẩn 1:** Tất cả các thuộc tính đều được định nghĩa ở dạng 1 giá trị đơn hoặc không ở dạng repeating group.

Ví dụ: Các quan hệ sau không đạt dạng chuẩn 1

OrderID	CustomID	OrderDate	Items	OrderTotal
1111	101	1/1/02	4 apples	1000
1111	103	1/2/02	5 bananas	900
Không đạt chuẩn 1 vì thuộc tính Items không là giá trị đơn				

OrderId	CustomId	Items	Quarter1	Items	Quarter2	Items	Quarter3	…
Không đạt chuẩn 1 vì thuộc tính Items và Quarter có dạng repeating group								

- **Dạng chuẩn 2:** Tất cả các thuộc tính không khóa phụ thuộc đầy đủ vào khóa.

ProductName	SupplerName	CompanyName	SupplerPhoneNumber
Giả sử ProductName không trùng, ProductName là khóa. Vi phạm chuẩn 2 vì SupplerPhoneNumber chỉ phụ thuộc vào SupplerName mà không phụ thuộc vào ProductName (khóa)			

- **Dạng chuẩn 3:** Tất cả thuộc tính không khóa phụ thuộc đầy đủ và không phụ thuộc bắt cầu vào khóa.

CompanyName	Address	City	Region	Postcode
Giả sử CompanyName là không trùng Vi phạm chuẩn 3 vì từ City và Region thì ta sẽ biết được Postcode, như vậy PostCode phụ thuộc vào City và Region. Trong đó CompanyName là một khóa dự tuyển (Candidate)				

Lưu ý:

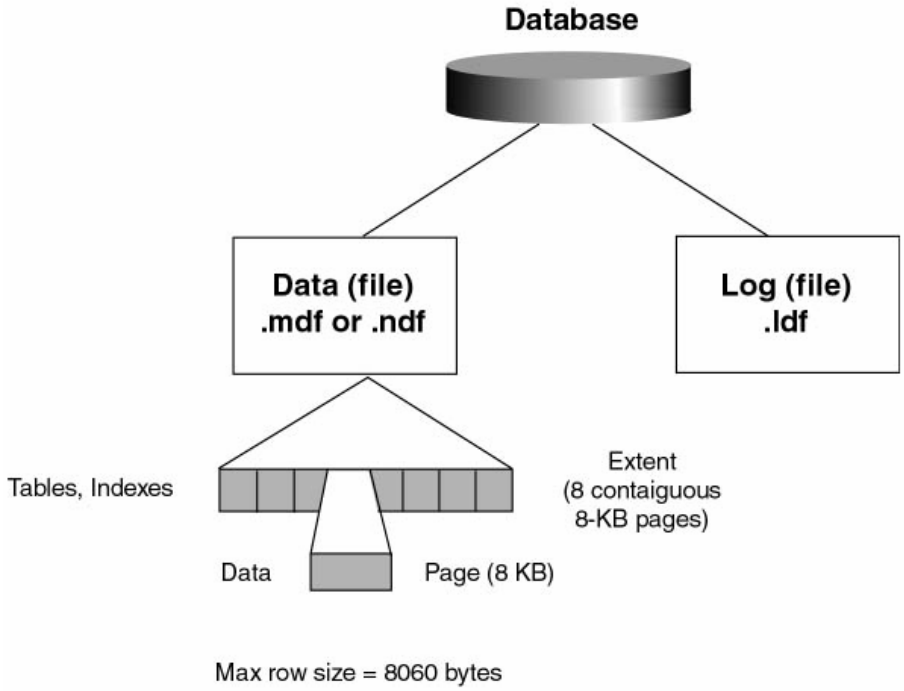
- ✓ Một cột (field) có giá trị lặp đi lặp lại một giá trị thì giá trị đó gọi là dữ liệu dư thừa.
- ✓ Các giá trị này người sử dụng có thể gõ sai chính tả khi đó nên suy nghĩ để quyết định có nên nảy sinh thêm 1 table hay không.
- ✓ Hoặc trong tương lai có thể nảy sinh ra các giá trị mới của field đó thì việc phát sinh Table là ưu việt.
- ✓ Khi xây dựng dữ liệu của khóa chính, có 2 trường phái: khóa có ngữ nghĩa và khóa không có ngữ nghĩa. Để giải quyết các lỗi phát sinh khi dùng giá trị khoá có ngữ nghĩa, các chuyên gia khuyên là ngữ nghĩa của khoá chỉ dùng để tham khảo và không nên dùng nó khi viết các ứng dụng.

4.2 Cơ sở dữ liệu của SQL SERVER 2000

Một CSDL của SQL Server là tập hợp các bảng (Table) dùng để chứa các dữ liệu và những đối tượng khác, chẳng hạn Views, indexes, store procedures, và triggers… Chúng được ấn định để hỗ trợ các hoạt động được thực hiện với dữ liệu. Dữ liệu được chứa trong một CSDL thường liên quan đến một chủ đề hay tiến trình đặc biệt. SQL Server có thể hỗ trợ nhiều CSDL. Mỗi một CSDL có thể liên quan hay không liên quan đến các CSDL khác. Ví dụ một SQL Server có thể có CSDL chứa dữ liệu về nhân sự, và một CSDL khác chứa các đơn hàng.

Khi bạn tạo một CSDL, bạn phải xây dựng cấu trúc lưu trữ dữ liệu. Các trúc này bao gồm ít nhất một tập tin dữ liệu (data file) và một tập tin vết giao tác (transaction log file). Bạn nên hiểu cách thức Microsoft SQL Server 2000 lưu trữ dữ liệu, cũng như chức năng của transaction log trước khi bạn làm việc với CSDL của SQL Server.

Cách dữ liệu được lưu trữ trong CSDL của SQL Server:



Hình 22: Cách lưu trữ dữ liệu

Các tập tin CSDL SQL Server

Một CSDL được lưu chứa trong các tập tin vật lý trên đĩa cứng, một CSDL trãi dài trên ít nhất là hai tập tin. Một vài **tập tin dữ liệu (data file)**, và một **tập tin vết (transaction log file)**. Những tập tin này được chỉ định khi CSDL được tạo hay hiệu chỉnh. SQL Server 2000 cho phép ba loại tập tin CSDL:

- **Primary data files:** một CSDL có một primary data file dùng để ghi nhận lại tất cả những tập tin khác trong CSDL, và lưu trữ dữ liệu. Theo ngầm định, tên của primary data file có phần mở rộng là **.MDF**.
- **Secondary data files:** một CSDL có thể không có hoặc có nhiều secondary data files, dùng để lưu các đối tượng của CSDL. Theo ngầm định, tên của secondary data file có phần mở rộng là **.NDF**.

- **Log files:** một CSDL có ít nhất một log file dùng chứa những thông tin cần thiết cho việc phục hồi tất cả những giao tác (transaction) trong CSDL. Theo ngầm định, log file có phần mở rộng là **.LDF**.

Mỗi tập tin CSDL có năm thuộc tính: *tên tập tin logic, một tên tập tin vật lý, một kích thước (size) ban đầu, một kích thước tối đa (maximum size), và gia số tăng kích thước (growth increment).* Các thuộc tính của mỗi tập tin, theo cùng với những thông tin khác được ghi chú trong bảng hệ thống **sysfiles**, một dòng ứng với mỗi tập tin đuợc dùng trong một CSDL.

Nhóm tập tin (Filegroups)

Các Filegroup cho phép bạn kết nhóm các tập tin nhằm mục đích quản trị và sắp xếp dữ liệu. Các Filegroup có thể cải thiện hiệu năng CSDL bằng cách cho phép một CSDL được tạo ngang qua nhiều đĩa hoặc hệ thống RAID. Bạn có thể tạo các bảng và các chỉ mục trên đĩa được chỉ rõ bằng cách dùng filegroup. Có 3 dạng filegroup.

- **Primary filegroup** bao gồm primary data file và tất cả những tập tin khác không đặt trong những filegroup khác. Các bảng hệ thống (System table) – Dùng định nghĩa các người dùng (user), các đối tượng (object), và các quyền (permission) đối với một CSDL – là được đặt trong primary filegroup của CSDL đó. SQL Server tự động tạo các bảng hệ thống của từng CSDL khi chúng ta tạo CSDL.
- **User-defined filegroups** là bất kỳ các filegroup xác định mặc định bởi người dùng trong suốt tiến trình tạo hoặc hiệu chỉnh CSDL. Một bảng hoặc một chỉ mục có thể được tạo và đặt trong một user-defined filegroup chỉ định.
- **Default filegroup:** Chứa tất cả đối tượng các trang (page) của các bảng và các chỉ mục mà không được chỉ định filegroup khi chúng được tạo ra. Theo mặc nhiên, Default filegroup là primary filegroup. Các thành viên của role db_owner database có thể chuyển đổi trạng thái default từ một filegroup này cho filegroup khác. Tại một thời điểm chỉ có thể có một default filegroup. Nếu default filegroup không được chỉ định thì primary filegroup là default filegroup một các tự động. Lệnh hiệu chỉnh CSDL (ALTER DATABASE) được dùng để thay đổi default filegroup.

```
ALTER DATABASE database_name MODIFY FILEGROUP filegroup_name DEFAULT
```

Cách cấp phát khoảng không để lưu trữ

Dữ liệu được lưu trữ trong các khối 8Kb liền nhau của không gian của đĩa được gọi là trang (page). Có nghĩa là một CSDL có thể lưu trữ 128 page mỗi megabyte (MB).

Các dòng không thể trãi dài trên các page và phải nằm gọn trong 1 page. Tổng số lớn nhất của dữ liệu trong một dòng đơn là 8060 bytes (8192, trừ overhead). Tổng số khoảng không lớn nhất mà có thể sử dụng bởi các dòng trên một trang là 8094 bytes.

Các bảng, các chỉ mục, … được lưu trữ trong các extent. Một extent có 8 page kề nhau, hoặc 64 KB. Vì vậy, một CSDL có 16 extent mỗi megabyte. Có thể có đến 8 đối tượng nhỏ có thể chia sẻ trong một extent (Mixed Extent). Khi một bảng tăng lên 8 page, nó dùng extent đồng dạng (Uniform extent).

SQL Server dùng 2 kiểu đồ thị định vị (allocation map) để ghi nhận lại định vị của các extent:

- **Global Allocation Map (GAM)**

Các trang GAM ghi nhận các extent đã được cấp phát. Mỗi một GAM kiểm soát 64000 extent, hoặc gần 4 GB dữ liệu. GAM có một bit ứng với mỗi extent trong vùng mà có kiểm soát. Nếu bit đó mang giá trị là 1 thì extent là trống; nếu bit đó mang giá trị 0 thì extent đã được cấp phát.

- Shared Global Allocation Map (SGAM)

  Các trang SGAM ghi nhận các extent được sử dụng như là các mixed extent và có ít nhất một trang chưa dùng. Mỗi SGAM kiểm soát 64000 extent, hay gần 4 GB dữ liệu. SGAM có một bit ứng với extent trong vùng mà nó kiểm soát. Nếu bit mang giá trị 1 thì extent được đang sử dụng như là một mixed extent và hiện có trang trống (free page); nếu mang giá trị 0, thì extent đã được dùng như là mixed extent hay nó là một mixed extent mà tất cả các trang đều được dùng.

Mỗi extent có một giá trị bit đặt trong GAM và SGAM dựa trên cơ sở có đang được dùng:

Sử dùng hiện thời của extent	**Bit GAM**	**Bit SGAM**
Trống, chưa được dùng	1	0
Uniform extent, hoặc mixed extent đầy	0	0
Mixed extent với các free page	0	1

Các extent được quản lý với một thuật toán đơn giản. Để định một extent đồng dạng, SQL Server tìm trên GAM một bit 1 và đặt nó thành bit 0. để tìm một mixed extent với những free page, SQL Server tìm trên SGAM một bit 1. Để định một mixed extent, SQL Server tìm trên GAM một bít 1, đặt nó thành 0, và đặt vào bit tương ứng trên SGAM giá trị 1. Một extent trống, SQL Server đảm bảo bít trên GAM mang giá trị 1 và trên SGAM mang giá trị 0

Transaction Log làm việc như thế nào?

Transaction log ghi nhận sự hiệu chỉnh dữ liệu – các câu lệnh INSERT, UPDATE, và DELETE –khi chúng được thi hành. Tiến trình ghi vết ghi nhận lại:

- Một sự thay đổi dữ liệu được gửi từ ứng dụng.
- Khi một sự thay đổi được thực hiện thì các trang dữ liệu ảnh hưởng được tải lên từ tập tin dữ liệu trong bộ nhớ (gọi là *data cache*), nếu các trang không sẵn sàng trong data cache từ truy vấn trước đó.
- Mỗi câu lệnh hiệu chỉnh dữ liệu thì luôn luôn được ghi trong log như nó được tạo. Thay đổi thì luôn luôn ghi nhận lại thành vết và được ghi vào tập tin log (log file) trước khi thay đổi đó được tác động trong CSDL. Kiểu của log này gọi là *write-ahead log*.
- Khi các trang dữ liệu hiện nằm trong data cache, và những trang log được ghi nhận lại trên đĩa trong một tập tin transaction log thì tiến trình checkpoint ghi tất cả các transaction đã hoàn tất (committed transaction) vào CSDL trên đĩa.

Một transaction đơn có thể có nhiều hiểu chỉnh dữ liệu. Mỗi transaction bắt đầu với một lệnh BEGIN TRANSACTION. Nếu ứng dụng hoàn tất tất cả sự hiệu chỉnh dữ liệu một

cách thành công thì transaction kết thúc với lệnh COMMIT TRANSACTION (như là một transaction được nối là transactin hoàn tất- *committed transaction*).

Trong suốt quá trình hoạt động, tiến trình checkpoint đều đặn thường xuyên kiểm tra các transaction đã hoàn tất mà sự hiệu chỉnh dữ liệu chưa được ghi vào tập tin dữ liệu. Tiến trình checkpoint ghi những hiệu chỉnh này vào tập tin dữ liệu và checkpoint các transaction cho biết rằng nó đã được viết vào tập tin dữ liệu chưa.

Nếu hệ thống bị hỏng hóc, tiến trình phục hồi (recovery process) tự động chạy khi SQL Server được khởi động lại. Tiến trình này sử dụng transaction log để quay ngược lại đến các transaction hoàn tất mà chưa từng được "checkpointed" và xoá bỏ đến (roll back) các transaction chưa hoàn tất.

Cơ chế tự động ghi nhận vết trong SQL Server là không là một lựa chọn (option) (có nghĩa là bạn không thể tắt nó đi), tất cả các hiệu chỉnh dữ liệu đều phải đi qua transaction log (Có 2 phương pháp tải một lượng dữ liệu lớn mà có thể được thực hiện mà **không dùng transaction log**, đó là chương trình **bulk copy** và lệnh **SELECT INTO**). Dữ liệu vrất quan trọng nên transaction log không bao giờ đầy (full) bởi điều này sẽ ngăn chặn hiệu chỉnh dữ liệu trong CSDL.

4.3 Tạo, hiệu chỉnh cở sở dữ liệu SQL SERVER

4.3.1 Giới thiệu

Để tạo một CSDL, trước hết bạn phải định nghĩa một tên cho CSDL, kích cở của nó, và các tập tin primary data file, secondary data file và file group dùng để lưu trữ nó. Bạn nên xem xét vài nhân tố sau trước khi bạn tạo CSDL:

- Quyền để tạo một CSDL mặc nhiên phải là thành viên của sysadmin và DBCreator fixed server role, mặc dù quyền này có thể gán cho bất kỳ user nào.
- User - người tạo ra CSDL trở thành chủ (owner) của CSDL.
- Có thể có tối đa 32767 CSDL có thể tạo trong một server.
- Tên của CSDL phải đặt theo qui tắt định danh.

Khi tạo CSDL bạn nên chỉ định dung lượng lớn nhất có thể có của một CSDL, điều này sẽ ngăn chặn sự gia tăng không kiểm soát kích thước của CSDL. SQL Server tạo CSDL thông qua 2 bước:

- SQL Server sử dụng một bản sao của CSDL Model để khởi tạo CSDL mới và biến đổi nó.
- Sau đó SQL Server nhồi đầy phần còn lại của CSDL bởi các trang trống.

Các phương pháp tạo, hiệu chỉnh một CSDL của SQL Server.

Cách 1: dùng SQL Enterprise Manager

Cách 2: dùng Create Database Wizard.

Cách 3: dùng câu lệnh CREATE DATABASE.

4.3.2 Tạo cơ sở dữ liệu

Tạo bằng Database Wizard (ơÛ tại cửa sổ Enterprise Manager)

1. Mở rộng **server group**, sau đó nới rộng **server** nơi mà sẽ tạo CSDL.

2. Chọn thực đơn **Tools → Wizards**.

3. Mở rộng **Database**.
4. Nhấp phải chuột tại **Create Database Wizard**.
5. Hoàn tất các bước trong Wizard.

Tạo bằng Enterprise Manager (ở tại cửa sổ Enterprise Manager)

1. Mở rộng **server group**, sau đó nới rộng **server** nơi mà sẽ tạo CSDL.
2. Nhấp nút phải chuột tại nút **Database,** chọn **New DataBase**.
3. Khai báo các thông tin cần thiết, sau đó chọn OK:

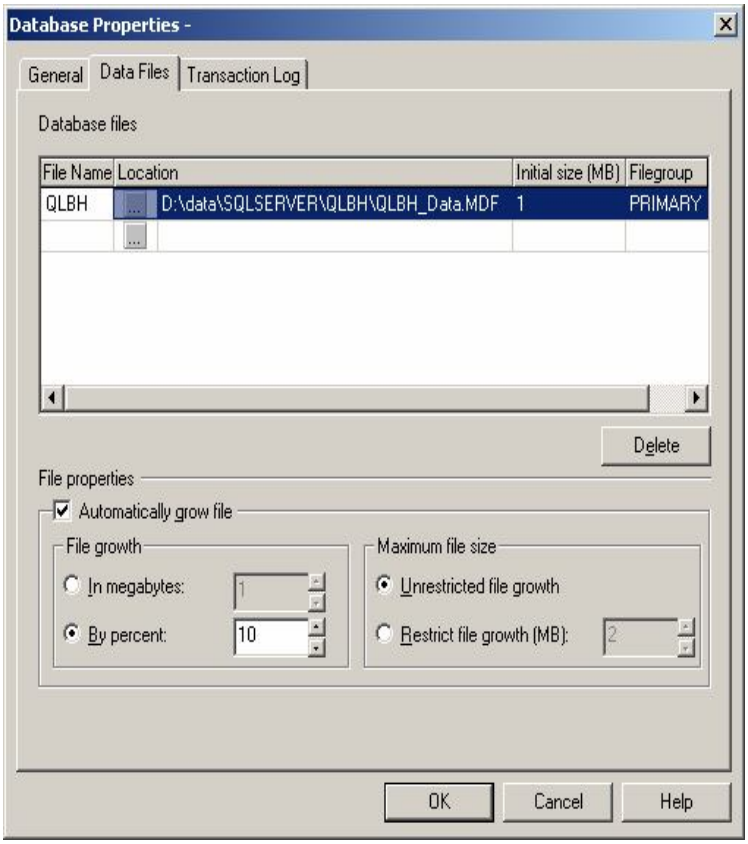
Trang General

+ Name: <Tên logic của CSDL>

Trang Data Files

+ Location : <Tên của tập tin Datafile, vị trí lưu tập tin trên đĩa>
+ Initial size: <Kích cở khởi tạo CSDL>
+ File Group : <Tên của File Group>
+ File properties: Khai báo một số thuộc tính khác như tỉ lệ gia tăng (File Growth), kích cở tối đa (maximun size)

Trang Transaction log: Tương tự như trang Data files nhưng khai báo cho tập tin log.



Hình 23: Hộp thoại xem thuộc tính của SQL Server

Tạo bằng câu lệnh Create Database (gõ lệnh trong cửa sổ Query Analyzer).

Cú pháp

```
CREATE DATABASE database_name
[ ON
   [ < filespec > [,...n ] ]
   [, < filegroup > [,...n ] ]
]
[ LOG ON { < filespec > [,...n ] } ]
[ COLLATE collation_name ]
[ FOR LOAD | FOR ATTACH ]


< filespec > ::=
[ PRIMARY ]
([ NAME = logical_file_name, ]
   FILENAME = 'os_file_name'
   [, SIZE = size ]
   [, MAXSIZE = { max_size | UNLIMITED } ]
   [, FILEGROWTH = growth_increment ]) [,...n ]
< filegroup > ::=
FILEGROUP filegroup_name < filespec > [,...n ]
```

Thực hiện:

(1) Gõ lệnh trong cửa sổ Query Analyzer

(2) Gọi thực thi câu lệnh

Ví dụ: Tạo CSDL có tên là SalesDB, tập tin dữ liệu tên là SalesDB_dat.mdf đặt trong C:\Data, kích cở khởi tạo là 10MB, kích thước tối đa là 50MB, tỉ lệ gia tăng là 5MB, và tập tin vết tên là SalesDB_log.ldf đặt trong C:\Data, kích thước khởi tạo là 5MB, kích thước tối đa là 25MB, tỉ lệ gia tăng là 10%.

```
CREATE DATABASE SalesDb
ON
(NAME = SalesDb_dat,
   FILENAME = 'c:\data\salesDB_dat.mdf',
   SIZE = 10,
   MAXSIZE = 50,
   FILEGROWTH = 5)
LOG ON
(NAME = 'SalesDb_log',
   FILENAME = 'c:\data\salesDB_log.ldf',
   SIZE = 5MB,
   MAXSIZE = 25MB,
   FILEGROWTH = 10%)
GO
```

Lưu ý:

- Thư mục **Data** phải hiện hữu trong **C:**

- Sao khi gõ câu lệnh, đánh dấu chọn khối lệnh và nhấn F5 để thực thi.

4.3.3 Thao tác trên cơ sở dữ liệu của SQL Server

Khi làm việc với CSDL, bạn có thể thực hiện trực tiếp trong cửa sổ Enterprise Manager hoặc dùng các câu lệnh T-SQL trong cở sổ Query Analyzer.

4.3.3.1 Kiểm tra sử tồn tại của cơ sở dữ liệu

Cách 1: Tại cửa sổ EM, kiểm tra sự tồn tại của CSDL trong nhánh DataBase.

Cách 2: Tại cửa sổ QA, thực hiện câu lệnh Sp_helpdb <Tên CSDL>

Ví dụ: Sp_helpDB SalesDB

4.3.3.2 Xem, thay đổi thuộc tính của cở sở dữ liệu.

Cách 1: Dùng Enterprise Manager

Mở nút **DataBase**, R_Click tại tên CSDL cần xem hoặc hiệu chỉnh

Chọn **Properties**

Thay đổi tuỳ ý

Cách 2: Dùng T_SQL

Cú pháp

```
ALTER DATABASE database
{ ADD FILE < filespec > [,...n ] [ TO FILEGROUP filegroup_name ]
| ADD LOG FILE < filespec > [,...n ]
| REMOVE FILE logical_file_name
| ADD FILEGROUP filegroup_name
| REMOVE FILEGROUP filegroup_name
| MODIFY FILE < filespec >
| MODIFY NAME = new_dbname
| MODIFY FILEGROUP filegroup_name {filegroup_property | NAME =
new_filegroup_name }
| SET < optionspec > [,...n ] [ WITH < termination > ]
| COLLATE < collation_name >
}
```

```
< filespec > ::=
(NAME = logical_file_name
   [, NEWNAME = new_logical_name ]
   [, FILENAME = 'os_file_name' ]
   [, SIZE = size ]
   [, MAXSIZE = { max_size | UNLIMITED } ]
   [, FILEGROWTH = growth_increment ])
```

Ví dụ 1: Chỉnh sửa kích cở của tập tin log file của SalesDb thành 10MB

```
ALTER DATABASE  SalesDb
MODIFY FILE (NAME='salesdb_log', size=10MB)
```

Ví dụ 2: Bổ sung thêm một tập tin dữ liệu SalesDB_data2

```
ALTER DATABASE SalesDB
ADD FILE (NAME=SalesDB_data2,
FILENAME='C:\data\SalesDb2.mdf',SIZE=10 MB,
MAXSIZE=20MB)
```

4.3.3.3 Xóa cơ sở dữ liệu.

Cách 1: Dùng Enterprise Manager

Nhấn nút phải chuột tại tên CSDL, chọn Delete

Cách 2: Dùng câu lệnh T-SQL

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

Ví dụ: `DROP DATABASE SalesDB`

4.3.3.4 Đổi tên cơ sở dữ liệu.

Dùng hàm sp_renamedb theo cú pháp sau

sp_renamedb [@dbname =] '*old_name*', [@newname =] '*new_name*'

4.3.3.5 Tạo một script cho CSDL và các đối tượng của CSDL.

Đôi khi, bạn cần sao chép cấu trúc của CSDL hoặc cấu trúc các đối tượng của CSDL, thì bạn sẽ thực hiện tạo script cho chúng. Khi có script bạn sẽ mở và thực thi đoạn script tại của sổ Query Analyzer để tái tạo lại các đối tượng.

Các bước thực hiện:

- Mở rộng một server group; mở rộng một server.
- Mở rộng nhánh DataBase, nhấp phải tại CSDL muốn tạo script, trỏ đến **All Tasks**, nhấp **General SQL SQL Script…**
- Khai báo các lựa chọn thích hợp.

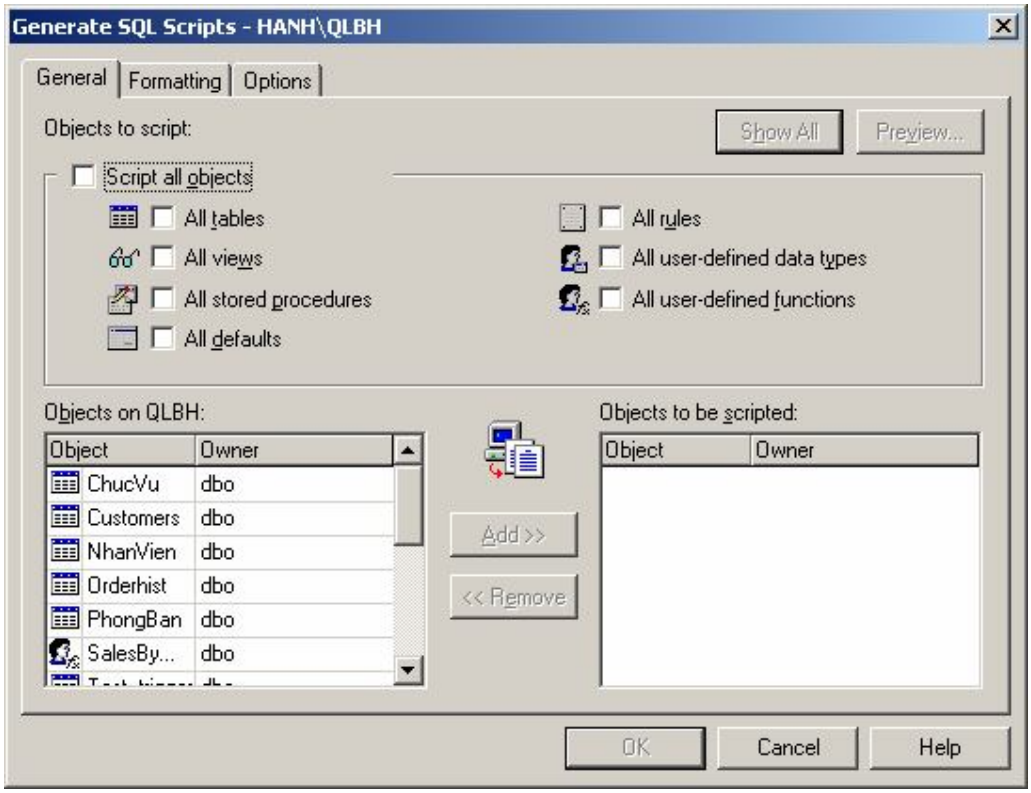
  + Trang General: chọn đối tượng cần tạo csript.

  + Trang Formating: chọn các tùy chọn định dạng script.

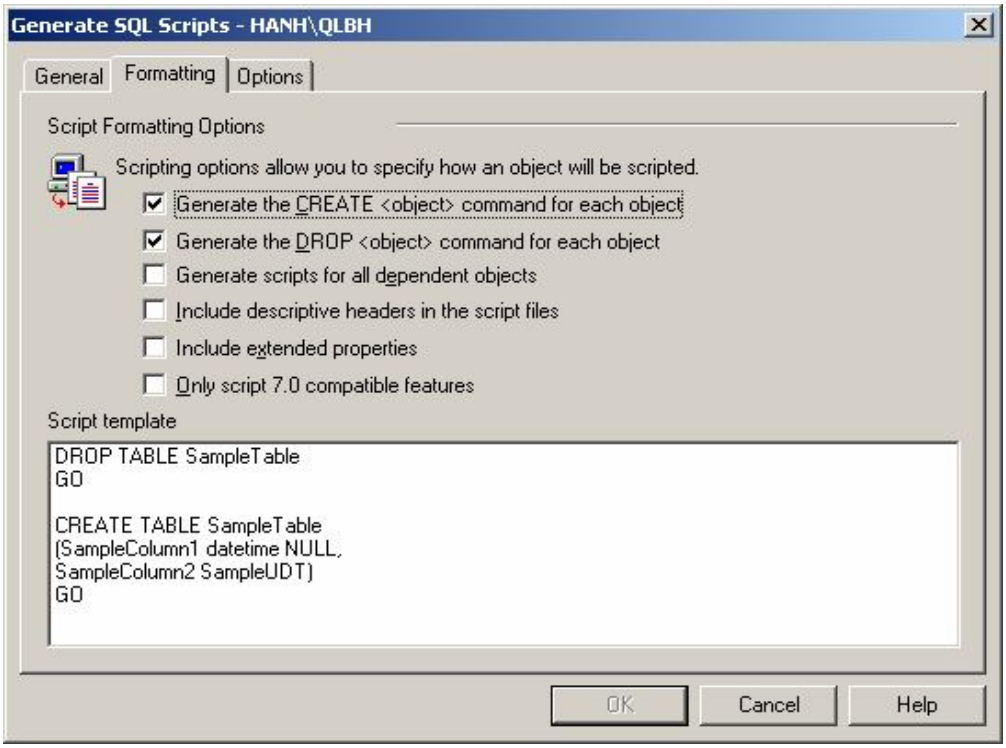
  - *Generate the CREATE <object> command for each object:* Tạo script theo cách sử dụng định nghĩa đang có của nó. Lựa chọn này được chọn theo mặc định.
  - *Generate the DROP <object> command for each object:* Bổ sung vào script cho mỗi đối tượng câu lệnh drop khi tạo script. Lựa chọn này được chọn theo mặc định.
  - *General scripts for all dependent objects:* Tự động tạo các script cho các đối tượng có liên quan với đối tượng đang tạo script.
  - *Include descriptive headers in the script files:* Thêm lời chú thích được bổ sung vào tập tin script cho mỗi đối tượng tạo script.

+ Trang options: hãy chọn các tùy chọn *security-related, table-related,* và *script file-related*

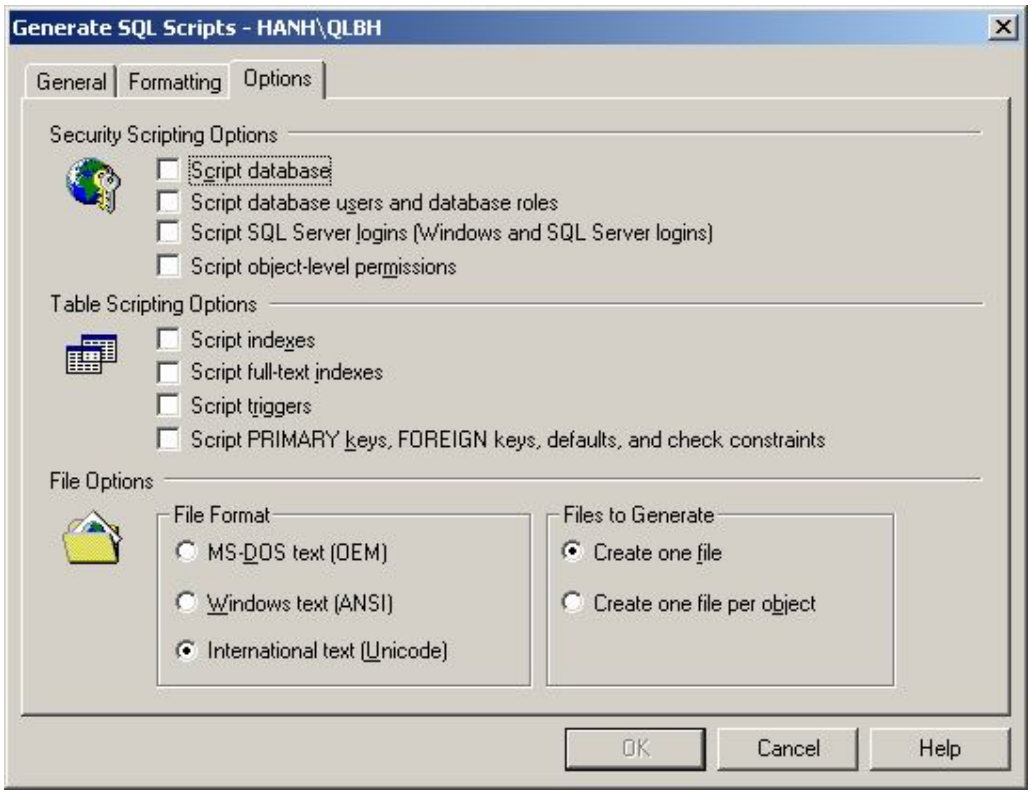
- Trên trang **General**, nhấp **PreView** để xem trước nội dung của script được tạo ra.



Hình 24: Trang General của hộp thoại phát sinh script các đối tượng của CSDL



Hình 25: Trang Formating của hộp thoại phát sinh script các đối tượng của CSDL



Hình 26: Trang Option của hộp thoại phát sinh script các đối tượng của CSDL

Sử dụng Script vừa tạo:

- Chuyển đến vị trí mới cần tài tạo lại CSDL/ các đối tượng CSDL.
- Vào cửa sổ Query Analazer, mở tập tin Script.
- Hiệu chỉnh các vị trí vật lý nếu cần.
- Cho thực thi đoạn Script
- Kiểm tra kết quả.

BÀI 5:
KIỂU DỮ LIỆU – LÀM VIỆC VỚI BẢNG

5.1 Kiểu dữ liệu (data type)

Kiểu dữ liệu là một đặc tính của một cột (Column). Nó định rõ loại dữ liệu và dạng dữ liệu được nhập vào cột.

Có 2 nhóm:

- **System-Supplied datatype:** Các kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server.
- **User-defined datatype:** Các kiểu dữ liệu của người dùng tự định nghĩa dựa trên các kiểu dữ liệu cơ bản.

5.1.1 System-Supplied Datatype.

System-Supplied Datatype **là** kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server. Các đối tượng lưu chứa dữ liệu đều có một kiểu dữ liệu để lưu, các đối tượng đó có thể là

- Các cột (Column) trong các bảng.
- Các tham số (parameters) trong strored procedures.
- Các biến (Variables) trong stored procedure, function, script, batch.
- Các hàm T_SQL trả về một hoặc nhiều giá trị thuộc một kiểu dữ liệu nào đó.

Ta có thể dùng các kiểu dữ liệu để tạo các ràng buộc toàn vẹn dữ liệu. Ví dụ cột TENNV thì không thể được định nghĩa với kiểu dữ liệu là Date, vì cột Date chỉ chấp nhận giá trị ngày.

Khi ta gán kiểu dữ liệu cho một đối tượng nào đó thì ta cần quan tâm đến các tính chất sau:

- Loại dữ liệu được chứa đựng bởi đối tượng.
- Chiều dài lưu trữ giá trị hoặc là kích cở của nó.
- Tính đúng của số (đối với các kiểu số).

Các kiểu dữ liệu cơ bản:

Loại	Kiểu dữ liệu cơ sở	Kích cở	Vùng giá trị	Mô tả
Binary	Binary	8 KB	“0”…”9”, “a”..”f”, “A”..”F”	Chứa các bit thông tin
	Varbinary	8 KB	“0”…”9”, “a”..”f”, “A”..”F”	
	Image	2^31 -1 bytes	2^31 –1 bytes	Dữ liệu hình ảnh
Character	Char	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Varchar	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Text	2147483647 bytes	2^31-1 ký tự (2147483647)	Ký tự hoặc chuỗi
Date and Time	Datetime	8 bytes	01/01/1753->31/12/9999	Chuỗi biểu diễn ngày giờ
	Smalldatetime	4 bytes	1/1/1900 -> 6/6/2079	Chuỗi biểu diễn ngày giờ
Decimal	Decimal	17 bytes	-10^38-1 -> 10^38-1	Số thực
	Numeric	17 bytes	-10^38-1 -> 10^38-1	Số thực
Foating point	Float	8 bytes	-1.79E+308 -> 1.79E+308	Số thực
	Real	4 bytes	-3.40E+38 ->3.40E+38	Số thực
Integer	Bigint	8 bytes	-2^63 -> 2^63	Số nguyên

	Int	4 bytes	-2^31 -> 2^31-1	Số nguyên
	Smallint	2 bytes	-2^15 -> 2^15-1	Số nguyên
	Tinyint	1 bytes	0..255	Số nguyên
Monetary	Money	8 bytes	-2^63 -> 2^63-1	Dữ liệu tiền tệ
	Smalmoney	4 bytes	-214748.3648 -> 214748.3648	Dữ liệu tiền tệ
Special	Bit	1 bytes	0 hoặc 1	Dữ liệu có một trong hai trạng thái 0 hoặc 1
	Cursor	Kiểu DL cho biến hoặc giá trị trả về của procedure, tham chiếu đến 1 mẫu tin		
	Timestamp	8 bytes	Chuỗi có dạng: 0x000000100000a90	Theo dõi mẫu tin nào bị thay đổi dữ liệu
	Uniqueidentifier	16 bytes	Số thập lục phân	
	SQL_variant	Là kiểu dữ liệu có thể chứa bất kỳ loại dữ tùy ý của SQL Server ngoại trừ **text**, **ntext, image,** and the **timestamp** data type		
	Table			
Unicode	Nchar		4000 ký tự	Ký tự hoặc chuỗi
	Nvarchar		4000 ký tự	Ký tự hoặc chuỗi
	Ntext		2^30-1 ký tự	Ký tự hoặc chuỗi

5.1.2 User-defined datatype.

Người sử dụng có thể dựa yêu cầu cần lưu trữ và các kiểu dữ liệu cơ bản để định nghĩa ra một kiểu dữ liệu của người dùng dùng để lưu trữ một dữ liệu đặc biệt nào đó. SQL Sever cho phép bạn cải tiến các kiểu dữ liệu để đảm bảo tính nhất quán khi làm việc trong môi trường dữ liệu đa dạng trong các bảng hay các CSDL khác nhau.

- User-defined data type không cho phép bạn định nghĩa các kiểu dữ liệu phức hoặc có cấu trúc.
- Mỗi một User-defined data type có thể được định nghĩa riêng cho một CSDL hoặc cho toàn bộ các CSDL. Nếu User-defined data type được định nghĩa trong CSDL Master thì nó được dùng chung cho toàn bộ các CSDL.
- Các User-defined data type mà bạn tạo trong CSDL model thì sẽ có trong tất cả các CSDL mới tạo một cách tự động.
- Mỗi user-defined data type được lưu thành một mẫu tin trong bảng **systypes.**
- Bạn có thể tạo và xóa user-defined data type bằng các thủ tục hệ thống. Tên của kiểu dữ liệu phải tuân thủ qui tắt định danh và phải là duy nhất trong mỗi CSDL. Định nghĩa mỗi user-defined data type trong giới hạn của các kiểu dữ liệu cơ bản. Phải chỉ định mặc định là chấp nhận giá trị NULL hay NOT NULL khi đối tượng không có giá trị.

Tạo một User-Defined Data Type

Dùng thủ tục hệ thống ***sp_addtype*** để tạo một user-defined data type.

sp_addtype *type*, *system_data_type* [,'NULL' | 'NOT NULL']

Ví dụ 1: Tạo kiểu dữ liệu tên là **isbn** với kiểu dữ liệu cơ bản là **smallint** và **không chấp nhận giá trị Null**

```
EXEC sp_addtype isbn, 'smallint', NOT NULL
```

Ví dụ 2: Tạo kiểu dữ liệu tên là **zipcode** với kiểu dữ liệu cơ bản là **char**, **độ dài tối đa là 10 và chấp nhận giá trị Null**

```
EXEC sp_addtype zipcode, 'char(10)', NULL
```

Ví dụ 3: Tạo kiểu dữ liệu tên là **longstring** với kiểu dữ liệu cơ bản là **varchar**, **độ dài tối đa là 63** và **chấp nhận giá trị Null**

```
EXEC sp_addtype longstring, 'varchar(63)', NULL
```

Xoá một User-Defined Data Type: dùng thủ tục hệ thống *sp_droptype để xóa một* user-defined data type từ bảng systypes. Một user-defined data type không thể xóa được nếu nó được tham chiếu bởi các bảng và những đối tượng khác.

Sp_droptype *type*

Ví dụ:

```
EXEC sp_droptype isbn
```

Xem các user-defined data types trong CSDL hiện hành: dùng thủ tục **sp_help** hoặc truy vấn trong **information_schema.domains**

Ví dụ:

```
Use SalesDB
Sp_help
 hoặc
SELECT domain_name, data_type,
character_maximum_length
FROM information_schema.domains
ORDER BY domain_name
```

5.2 Làm việc với bảng của SQL Server

Bảng là một đối tượng của CSDL và là nơi chứa đựng các dữ liệu về một thực thể nào đó ví dụ Khách hàng, đơn đặt hàng, tồn kho,… Một bảng là một tập hợp các cột (Column). Mỗi một cột đại diện cho một thuộc tính của dữ liệu trong bảng.

Khi bạn tạo một table, bạn phải chỉ định rõ tên của bảng, tên cột, kiểu dữ liệu của cột. Tên cột phải duy nhất trong một bảng, có thể dùng tên trùng nhau ở các bảng khác nhau trong cùng một CSDL. Phải chỉ rõ một kiểu dữ liệu cho mỗi cột và những lựa chọn khác nếu cần. Bạn có thể tạo:

- Tối đa 2 tỉ table cho mỗi CSDL.
- Tối đa 1024 cột trong mỗi bảng.
- 8060 bytes mỗi dòng (kiểu image và text dùng 16 bytes mỗi dòng)

Hai cách cơ bản làm việc trên bảng: **Enterprise Manager (tự nghiên cứu), câu lệnh T-SQL.**

5.2.1 Tạo một bảng mới

Cú pháp lệnh

```
CREATE TABLE
  [ database_name.[ owner ] . | owner. ] table_name
  ({ < column_definition >
```

```
    | column_name AS computed_column_expression
    | < table_constraint > ::= [ CONSTRAINT constraint_name ] }
      | [ { PRIMARY KEY | UNIQUE } [,...n ]
  )

[ ON { filegroup | DEFAULT } ]
[ TEXTIMAGE_ON { filegroup | DEFAULT } ]

< column_definition > ::= { column_name data_type }
  [ COLLATE < collation_name > ]
  [ [ DEFAULT constant_expression ]
    | [ IDENTITY [ (seed, increment) [ NOT FOR REPLICATION ] ] ]
  ]
  [ ROWGUIDCOL]
  [ < column_constraint > ] [ ...n ]

< column_constraint > ::= [ CONSTRAINT constraint_name ]
  { [ NULL | NOT NULL ]
    | [ { PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
      [ WITH FILLFACTOR = fillfactor ]
      [ON {filegroup | DEFAULT} ] ]
    ]
    | [ [ FOREIGN KEY ]
      REFERENCES ref_table [ (ref_column) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
      [ NOT FOR REPLICATION ]
    ]
    | CHECK [ NOT FOR REPLICATION ]
    (logical_expression)
  }
< table_constraint > ::= [ CONSTRAINT constraint_name ]
  { [ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ]
    { (column [ ASC | DESC ] [,...n ]) }
    [ WITH FILLFACTOR = fillfactor ]
    [ ON { filegroup | DEFAULT } ]
  ]
  | FOREIGN KEY
    [ (column [,...n ]) ]
    REFERENCES ref_table [ (ref_column [,...n ]) ]
    [ ON DELETE { CASCADE | NO ACTION } ]
    [ ON UPDATE { CASCADE | NO ACTION } ]
    [ NOT FOR REPLICATION ]
  | CHECK [ NOT FOR REPLICATION ]
    (search_conditions)
  }
```

Ví dụ 1:

```
USE SalesDb
GO
CREATE TABLE Employees
```

```
(
    EmployeeID SMALLINT IDENTITY(1,1) NOT NULL,
    FirstName NVARCHAR(30) NOT NULL,
    LastName NVARCHAR(30) NOT NULL,
    Address1 NVARCHAR(60) NOT NULL,
    City NVARCHAR(15) NOT NULL,
    State CHAR(2) NOT NULL,
    Phone VARCHAR(24) NOT NULL,
    DOB DATETIME NOT NULL,
    HireDate DATETIME NOT NULL,
    PositionID TINYINT NOT NULL
)
```

5.2.2 Hiệu chỉnh bảng

Thao tác cơ bản hiệu chỉnh bảng gồm thêm cột, xóa cột, thay đổi thuộc tính của cột. Để thực hiện ta sẽ dùng câu lệnh ALTER TABLE[1].

Thêm các cột

```
ALTER TABLE <TableName>
ADD <Column Definition>[,...n]
```

Ví dụ: Thêm cột Address vào bảng Employees

```
ALTER TABLE Employees
ADD  Address2 NVARCHAR(6) NOT NULL DEFAULT 'N/A'
```

Lưu ý: Nếu bảng đã có sẵn dữ liệu và cột thêm vào được định nghĩa là NOT NULL thì ta phải điền dữ liệu của các dòng ở cột mới thêm vào là một giá trị mặc định nào đó để tránh giá trị Null.

Xóa các cột

```
ALTER TABLE <TableName>
DROP COLUMN <Column name>[,...n]
```

Ví dụ:

```
ALTER TABLE Employees
DROP COLUMN Address2
```

Lưu ý: Lệnh trên sẽ không thực hiện được vì khi tạo cột Address2 ta đã khai báo giá trị mặc định nên SQL Server đã tạo ra một đố tượng 'Defaul Constraint'. Do đó, muốn xóa cột thì phải xóa tất cả các đối tượng liên quan đến cột cần xóa rồi mới xóa cột đó.

Ví dụ:

```
ALTER TABLE Employees
DROP CONSTRAINT DF_Employees_Addre_1372D2FE
ALTER TABLE Employees
DROP COLUMN Address2
```

[1] Bạn tham khảo cú pháp lệnh đầy đủ trong Books-Online

Lưu ý: *DF_Employees_Addre_1372D2FE là tên của Defaul Constraint do SQL Server tự đặt.*

Thay đổi kiểu dữ liệu cho cột

ALTER TABLE <TableName>

ALTER COLUMN <Column Name NewDatatype>

Ví dụ:

```
ALTER TABLE Employees
ALTER Address  NVARCHAR(20)
```

Xoá toàn bộ dữ liệu trong Table

TRUNCATE TABLE <Name Table>

Ví dụ:

```
TRUNCATE TABLE Employees
```

Lưu ý: Nếu bảng muốn xóa là một bảng con (child table) thì bạn có thể xóa dữ liệu của nó bất kỳ lúc nào bạn thích, nhưng nếu nó là một bảng cha (Parent Table) thì bạn phải xóa dữ liệu ở bảng con trước, kế tiếp xóa khoá ngoại (Foreign key constraint) giữa 2 bảng, cuối cùng mới xóa dữ liệu ở bảng cha.

5.2.3 Xóa bảng khỏi cơ sở dữ liệu.

Xóa một bảng là gỡ bỏ định nghĩa bảng và tất cả dữ liệu, cũng như các quyền (permission) định cho bảng đó.

Trước khi xóa một bảng, bạn phải gỡ bỏ tất cả các phụ thuộc giữa bảng đó và những đối tượng khác.

DROP TABLE <Name Table>

Ví dụ:

```
DROP TABLE Employees
```

5.3 Bảng tạm (Temporary Tables).

Bạn cũng có thể tạo các bảng tạm. Các bảng tạm tương tự như các bảng bình thường, ngoại trừ việc các bảng tạm thời được chứa trong CSDL TempDb và được xóa một cách tự động khi không còn sử dụng nữa.

Có hai loại bảng tạm: bảng tạm cục bộ (Local) và bảng tạm tổng thể (global). Chúng khác nhau về tên, tính hiển thị, và tính có sẵn.

Bảng tạm cụ bộ: bảng tạm cục bộ có một dấu # là ký tự đầu tiên trong tên của chúng; chúng chỉ hiển thị đối với nối kết hiện hành dành cho người sử dụng, và chúng được xóa khi người dùng ngắt nối kết với các thể hiện của SQL Server.

Ví dụ: Tạo bảng tạm tên là #MyLocalTempTable

```
CREATE TABLE #MyLocalTempTable
(    ID   INT PRIMARY KEY,
     ColA VARCHAR(30) NULL,
     ColB VARCHAR(30) NULL,
     ColC VARCHAR(30) NULL    )
```

Nếu một bảng tạm được tạo trong tiến trình của một thủ tục thì bảng tạm đó sẽ bị xóa khi thủ tục hoàn tất. Trong trường hợp một tiến trình A gọi thủ tục B mà B có tạo một bảng tạm thì chỉ có B mới dùng được bảng tạm đó còn tiến trình A không nhìn thấy bảng tạm đó. Trong trường hợp thủ tục B được gọi cùng một lúc bởi nhiều tiến trình khác nhau thì nó sẽ có nhiều bảng tạm giống nhau khi đó SQL Server sẽ giải pháp giải quyết bằng cách thêm một hậu tố (suffix) vào tên của các bảng tạm cùng tên sao cho các bảng này có tên là khác nhau (tên của bảng tạm tối đa là 116 ký tự)

Bảng tạm toàn cục: Các bảng tạm toàn cục thì có 2 dấu ## là ký tự đầu tiên trong tên của chúng; chúng hiển thị đối với bất kỳ người sử dụng nào sau khi chúng được tạo; và chúng được xóa khi tất cả những người sử dụng đang tham chiếu table ngắt kết nối với SQL Server.

Ví dụ:

```
CREATE TABLE ##MyGlocalTempTable
(    ID   INT PRIMARY KEY,
     ColA VARCHAR(30) NULL,
     ColB VARCHAR(30) NULL,
     ColC VARCHAR(30) NULL    )
```

BÀI 6:
TOÀN VẸN DỮ LIỆU

6.1 Giới thiệu toàn vẹn dữ liệu (data Integrity)

Toàn vẹn dữ liệu là đề cập đến trạng thái của tất cả các giá trị dữ liệu lưu trữ trong CSDL là đúng. Nếu dữ liệu không đúng mà đã được lưu trữ trong CSDL thì được gọi là vi phạm toàn vẹn dữ liệu.

Các bảng trong CSDL của SQL Server có một số loại toàn vẹn dữ liệu khác nhau. Ví dụ định nghĩa NOT NULL, định nghĩa DEFAULT, thuộc tính IDENTITY, CONSTRAINTS, RULES, TRIGGERS, INDEXES.

Xác định đúng kiểu dữ liệu của cột hoặc biến cũng là một cách thúc ép tính toàn vẹn dữ liệu. Ví dụ bạn không thể chấp nhật giá trị của cột CustomName là một giá trị dạng ngày giờ cũng như ngược lại. Để tạo hoặc thêm các ép thỏa toàn vẹn dữ liệu, chúng ta có thể thực hiện trong các câu lệnh.

Định nghĩa ràng buộc:

Create Table … : Định nghĩa trong lúc thiết kế cấu trúc bảng.

Alter Table… : Định nghĩa trong khi hiệu chỉnh bảng.

Để kiểm tra hoặc xem các toàn vẹn dữ liệu

```
Sp_HelpConstraint <Tên Table>
```

Hoặc

Bật cửa số **Object Browser** của **Query Analyzer**, mở nhánh Constraint của từng bảng.

Xóa các toàn vẹn dữ liệu

```
ALTER TABLE <Tên Table>
DROP CONSTRAINT <Tên Constraint>
```

6.2 Tìm hiểu các toàn vẹn dữ liệu.

6.2.1 Định nghĩa NULL/NOT NULL

Một giá trị không biết, chưa xác định chúng ta quy là giá trị **NULL**. Khả năng null của một cột được xem là khả năng của cột chấp nhận hoặc không chấp nhận giá trị null.

- Bạn có thể định nghĩa giá trị của một cột không là null.
- Một giá trị null không đồng nhất với giá trị 0, khoảng trắng, chuỗi rỗng.
- Null có nghĩa là không có thao tác nhập nào thực hiện được.
- Sự tồn tại của Null thường cho biết rằng giá trị chưa được biết rõ hay chưa xác định. Chẳng hạn, một giá trị Null trong cột *price* của bảng *Items* không có nghĩa là mặt hàng này không có giá hoặc là giá bằng 0.

Nói chung, hãy tránh chấp nhận giá trị Null bởi vì chúng gây ra nhiều phức tạp hơn trong các truy vấn cũng như cập nhật dữ liệu. Việc chỉ định một cột không chấp nhận giá trị Null có thể giúp duy trì tính toàn vẹn dữ liệu.

Thông thường để khai báo một cột có thể chấp nhận giá trị null, chúng ta sẽ khái báo trong khi định nghĩa hoặc hiệu chỉnh cột. Tức là dùng trong câu lệnh Create Table/Alter Table.

Ví dụ:

```
USE SalesDb
GO
DROP TABLE Product_Info
GO
CREATE TABLE Product_Info
(    Product_ID     smallint NOT NULL,
     Product_Name   char(20) NOT NULL,
     Description    char(30) NULL,
     Price          smallmoney NOT NULL
)
GO
```

6.2.2 Giá trị mặc định (Default Values)

Trong số các đặc tính của cột, chúng ta xét thấy giá trị có thể null và giá trị mặc định. Cả hai đặc tính này định ra giá trị chèn vào một cột khi nó không được chỉ định trong câu lệnh INSERT. Các trường hợp này có thể xảy ra khi giá trị cột không được đưa vào trong câu lệnh INSERT:

- Khi cột được định nghĩa như là chấp nhận giá trị NULL và không có giá trị mặc định, giá trị của cột là NULL.
- Khi cột được định nghĩa không chấp nhận giá trị NULL và không có giá trị mặc định, một lỗi sẽ xảy ra.
- Khi cột có một giá trị mặc định

Như vậy, mỗi một cột trong một mẫu tin của bảng đều phải chứa một giá trị, ngay cả khi giá trị đó là NULL. Có những trường hợp bạn cần phải tải một hàng dữ liệu vào một bảng nhưng bạn không biết giá trị dành cho cột hay giá trị này không tồn tại. Nếu cột chấp nhận các giá trị Null, bạn có thể tải hàng có giá trị Null. Thông thường, các cột chấp nhận giá trị Null có thể không phải là các cột cần thiết nên giải pháp tốt hơn hết là ấn định một giá trị mặc nhiên (không nhập giá trị vào thì cột sẽ chấp nhận giá trị mặc định). Việc đó chính là định nghĩa DEFAULT cho cột ở những nơi thích hợp. Chẳng hạn, người ta thường chỉ định 0 là giá trị mặc định cho các cột số, hoặc N/A là giá trị mặc định cho các cột chuỗi khi không có giá trị nào được chỉ định).

Khi bạn nhập vào một mẫu tin của bảng có một định nghĩa Default dành cho một cột bạn đang gián tiếp hướng dẫn SQL Server nhập một giá trị mặc định trong cột khi bạn không chỉ định một giá trị cho cột đó.

SQL Server 2000 có hai cách để triển khai các giá trị mặc định cho các cột: Default Constraint và Default Object.

6.2.2.1 Default Constraint

Default constraint có thể được tạo tại thời điểm tạo bảng, thêm sau khi bảng được tạo.

- Giá trị Default được dùng để gán giá trị hằng số cho một cột.
- Chỉ có một giá trị Default có thể được tạo cho một cột.

- Các cột TIMESTAMP, IDENTITY và ROWGUIDCOL không thể có default constraint, vì giá trị của chính đã tự động xác định.
- Giá trị default có thể là một hằng số; một hàm hệ thống, chẳng hạn Getdate(); một biến toàn cục, như @@trancount; hoặc một hàm do người dùng định nghĩa.

Khai báo default constraint

- **Định nghĩa Default constraing trong khi tạo bảng**

  CREATE TABLE *tablename*(
  ***columnname datatype* [NULL | NOT NULL]**
  [CONSTRAINT *constraintname*] DEFAULT *expression*
  [,...])

- **Định nghĩa Default constraint đối với một bảng đã tồn tại.**

  ALTER TABLE *tablename*
  ADD [CONSTRAINT *constraintname*] DEFAULT *expression* FOR *columnname*

Ví dụ 1: Tạo bảng Events với các default constraint

```
CREATE TABLE Events
( EventID int IDENTITY (1, 1) NOT NULL ,
  EventType nvarchar (10) NOT NULL,
  EventTitle nvarchar (100) NULL ,
  EventDescription ntext NULL ,
  EventLanguage nvarchar (2) NULL ,
  EventDate smalldatetime NULL DEFAULT GETDATE(),
  EventEndDate smalldatetime NULL DEFAULT DATEADD(day,
1,
GETDATE()),  EventCreator nvarchar (50) NOT NULL
DEFAULT SYSTEM_USER
 )
```

Ví dụ 2:

- Tạo bảng Events không có default constraint

```
CREATE TABLE Events
( EventID int IDENTITY (1, 1) NOT NULL,
  EventType nvarchar (10) NOT NULL,
  EventTitle nvarchar (100) NULL,
  EventDescription ntext NULL,
  EventLanguage nvarchar (2) NULL,
  EventDate smalldatetime NULL,
  EventEndDate smalldatetime NULL,
  EventCreator nvarchar (50) NOT NULL
 )
```

- Thêm các default constraint cho bảng Events

```
ALTER TABLE Events
ADD DEFAULT 'Party' FOR EventType
---------
ALTER TABLE Events
ADD CONSTRAINT EVentDate_DF DEFAULT GETDATE() FOR
EventDate
```

Kiểm tra constraint

```
Sp_helpConstraint   Events
--- chèn một mẫu tin trống vào bảng Events
INSERT Events DEFAULT VALUES
SELECT * FROM Events
```

Kết quả

	EventID	EventType	EventTitle	EventDescription	EventLanguage
1	1	Party	NULL	NULL	NULL

EventDate	EventEndDate	EventCreator
2004-02-15 01:28:00	2004-02-16 01:28:00	sa

Xoá default constraint

```
ALTER TABLE Events
DROP CONSTRAINT DF__Events__EventTyp__7E6CC920


ALTER TABLE Events
DROP CONSTRAINT EVentDate_DF
```

6.2.2.2 Default Object

Default object là một cách khác để định nghĩa một giá trị mặc định cho một cột. Các Default Object được gọi là "default" có đầu tiên trong phiên bản 2000 của SQL Server. Các Default không là một phần của toàn vẹn khai báo bởi vì chúng không là một của cấu trúc bảng; chúng thực sự là một phần của lược đồ CSDL.

Quá trình khai báo một Default như sau:

- Định nghĩa Default.
- Kết Default vào cột của bảng hoặc kiểu dữ liệu.
- Nếu muốn dùng Default thì sẽ gỡ bỏ khỏi cột hoặc kiểu dữ liệu.
- Không cần Default nữa thì xoá khỏi CSDL

Định nghĩa default

CREATE DEFAULT default AS constant_expression

Kết đính default với cột:

sp_binddefault defaultname, tablename.columnname

Kết đính default với user-defined datatype

sp_binddefault *defaultname*, *datatypename* [, futureonly]

Futureonly chỉ định rằng các cột đã tồn tại có liên quan đến kiểu dữ liệu sẽ không kế thừa giá trị mặc định mới. Cờ này chỉ có thể dùng kkhi kết giá trị mặc định cho kiểu dữ liệu.

Ví dụ:

```
CREATE DEFAULT CalifDef AS 'CA'
GO
sp_bindefault 'CalifDef', 'Orders.ShipRegion'
```

Gở bỏ kết đính một deault với cột

sp_unbindefault tablename.columnname

Gở bỏ kết đính một deault với User-defined datatype

sp_unbindefault *datatypename* [, futureonly]

Xóa một Default

DROP DEFAULT *defaultname*

Lưu ý: Chỉ xóa được những Defaut không được kết với cột hoặc kiểu dữ liệu.

6.2.3 Thuộc tính Identity:

- Identity là một thuộc tính của cột, nó không là một constraint. Tuy nhiên, Identity dùng để ràng buộc sự tồn tại dữ liệu.
- Một bảng chỉ có duy nhất một cột kiểu Identity.
- Kiểu dữ liệu của cột Identity phải là bigint, int, smallint, hoặc tinyint.
- Giá trị của cột Identity sẽ tự động tăng.

Một cột Identity được tạo khi bảng được tạo bằng lệnh Create Table, hoặc khi hiệu chỉnh cột trong bảng bằng lệnh Alter table.

Ví dụ:

```
CREATE TABLE  Table1
( ID INT IDENTITY,
  FirstName VARCHAR(30)  NOT NULL,
  LastName VARCHAR(30)   NOT NULL
)
GO
---Chèn dữ liệu vào Table1, không cần đưa giá trị
cho cột ID
INSERT Table1 (FirstName, LastName)
VALUES ('Minh', 'Thu')
```

Tuy nhiên, đôi khi bạn cần chỉ định giá trị cho cột có định nghĩa Identity, bạn thực hiện tuần tự các bước sau: Bật chế độ chèn dữ liệu cho cột Identity cho bảng, chèn dữ liệu, tắt chế độ chèn cho cột Identity nếu cần.

Ví dụ:

```
SET IDENTITY_INSERT Table1 ON
INSERT Table1 (ID, FirstName, LastName) VALUES
(99, 'Thuy','Tien')
SELECT *  FROM Table1
```

6.2.4 Check

Kiểu dữ liệu và giá trị default ép thỏa ràng buộc miền giá trị. Các Check giới hạn các giá trị có thể đưa vào cột. Chúng sẽ xác định các giá trị nào là hợp lệ.

- Một cột có thể có nhiều hơn một check constraint, chúng được lượng giá theo thứ tự được tạo.
- Các check constraint giới hạn các giá trị được phép bằng cách định nghĩa:
  - Một vùng hoặc nhiều vùng các giá trị có thể chấp nhận được.
  - Danh sách các giá trị.
  - Một mẫu định trước.

Bạn có thể qui định nhiều constraint check cho một cột đơn, chúng được lượng giá theo thứ tự được tạo.

6.2.4.1 Check Constraint

Check Constraint là một của định nghĩa bảng. Chúng có thể được định nghĩa trong khi tạo bảng, hiệu chỉnh bảng, và có thể xóa bất kỳ lúc nào. Chúng có thể được vô hiệu hoá (disabled) hoặc làm có hiệu lực (enabled) khi cần. Một cột có thể có nhiều hơn một check constraint. Chúng được lượng giá theo thứ tự được tạo. Check constraint:

- ✓ Lượng giá thành một biểu thức logic, như là biểu thức của mệnh đề WHERE.
- ✓ Có thể tham chiếu đến các cột khác trong cùng một bảng.

- Định nghĩa Check Constraint khi tạo bảng
  - Định nghĩa ở mức cột

    CREATE TABLE tablename (*columname datatype* [CONSTRAINT *constraintname*] CHECK [NOT FOR REPLICATION] (*logical_expression*)
  - Định nghĩa ở mức bảng

    CREATE TABLE *tablename* (*columname datatype* [,...], [CONSTRAINT *constraintname*] CHECK [NOT FOR REPLICATION] (*logical_expression*)

- Định nghĩa CHECK CONSTRAINT với bảng đã tồn tại

  ALTER TABLE *tablename*
  [WITH CHECK | WITH NOCHECK] ADD
  [CONSTRAINT *constraintname*]
  CHECK [NOT FOR REPLICATION] (*logical_expression*)

Ví dụ:

```
ALTER TABLE Chucvu
ADD CONSTRAINT NV_HSPC_Chk CHECK
          (HSPC>=0.1 AND HSPC<0.5)
```

6.2.4.2 RULE

Rule là một tính năng tương thích ngược để định nghĩa các qui tắc hợp lệ mà có thể kết buộc vào các cột của bảng hoặc các kiểu dữ liệu do người dùng định nghĩa. Giống như đối tượng Default, Rule được tạo trên chính nó trước khi được kết buộc vào đối tượng khác. Để tạo một Rule, ta sẽ dùng lệnh CREATE RULE.

Một cột chỉ có thể có một Rule được kết buộc, bạn có thể kết buộc một Rule với cột đã có định nghĩa Check Constraint. Cả hai đều có giá trị nhưng sẽ ưu tiên Check Constraint.

Định nghĩa Rule

CREATE RULE rulename AS condition_expression

Kết buộc Rule vào một cột

sp_bindrule rulename, tablename.columnname

Kết buộc Rule vào user-defined datatype

sp_bindrule *rulename*, *datatypename* [, futureonly]

Futureonly chỉ định rằng các cột tồn tại sẳn mà có dùng kiểu dữ liệu này thì không kế thừa rule mới. Cờ này chỉ sử dụng khi kết Rule với kiểu dữ liệu, đối với cột thì không.

Ví dụ: Tạo một rule kiểm tra ngày và kết nó vào cột OrderDate cho bảng Orders

```
CREATE RULE ActiveDate AS
@Date BETWEEN '01/01/70' AND GETDATE()
AS
sp_bindrule ActiveDate, 'Orders.OrderDate'
```

Các biểu thức dùng trong Rule giống các nguyên tắc như các điều kiện của check costraint và cũng tương tự biểu thức trong mệnh đề Where, ngoài bạn không thể tham chiếu đến các cột CDSL khác trong các Rule. Nếu bạn so sánh cú pháp của lệnh Check và biểu thức rule, hai sự khác biệt chính là:

- Biểu thức Rule dùng một biến (bắt đầu với một ký hiệu @) mà sẽ được thay thế bởi giá trị cột khi được đính vào cột.
- Một biểu thức Rule không thể tham chiếu đến các cột của bảng.

Điểm thứ hai là hành vi khác biệt lớn nhất giữa Check constraint và các Rule: Rule chỉ tương tương với Check constraint ở mức cột.

6.2.5 Primary key Constraint

Một bảng thường có một hay nhiều cột với các giá trị riêng để nhận biết mỗi hàng trong bảng. Các cột này được gọi là khóa chính (Primary key) của bảng và bảo đảm tính toàn vẹn thực thể của bảng.

- SQL Server tự động tạo một chỉ mục cho bảng ứng với các cột tham gia primary key constraint.
- Một bảng chỉ có một constraint Primay key.
- Một cột nằm trong constraint Primay key không thể chấp nhận giá trị Null, trùng lắp. Bởi vì Primay key constraint bảo đảm tính duy nhất của dữ liệu nên chúng thường được ấn định cho cột nhận dạng (identity column).
- Nếu một Primay key constraint được ấn định trên nhiều cột, các giá trị có thể được lặp lại trong một cột, nhưng mỗi sự kết hợp giá trị từ tất cả các cột trong Primay key constraint phải là sự kết hợp duy nhất.

Tao Primary Key Constraint

Có thể tạo constraint Primay key trong khi tạo table hoặc thêm constraint Primay key cho table có sẵn

Để hiệu chỉnh constraint Primay key của một table thì bạn phải xóa constraint Primay key và tạo lại.

- **Định nghĩa Primary Key Constraint khi tạo bảng:**
  - Định nghĩa ở mức cột

    CREATE TABLE *tablename*
    (*columname datatype* [CONSTRAINT *constraintname*]
    PRIMARY KEY [CLUSTERED | NONCLUSTERED]
    [WITH FILLFACTOR = *fillfactor*]
    [ON {*filegroup* | DEFAULT}] [,...]

  - Định nghĩa ở mức bảng

    CREATE TABLE *tablename*
    (*columname datatype* [,...], [CONSTRAINT *constraintname*]
    PRIMARY KEY [CLUSTERED | NONCLUSTERED]
    { (*column* [ASC | DESC] [,...*n*]) }
    [WITH FILLFACTOR = *fillfactor*]
    [ON { *filegroup* | DEFAULT }]

Ví dụ 1: Vừa tạo bảng vừa định nghĩa Primary Key Constraint

Cách 1: Định nghĩa ở mức cột

```
CREATE TABLE Table1
     (col1 INT NOT NULL PRIMARY KEY,
      col2 VARCHAR(30)
      )
-- Kiểm tra constraint
EXEC Sp_helpconstraint Table1
```

Cách 2: Định nghĩa ở mức bảng

```
CREATE TABLE Table2
     (col1 INT NOT NULL,
      col2 VARCHAR(30)
      CONSTRAINT Table2_PK PRIMARY KEY(col1)
     )
EXEC Sp_helpconstraint Table1
```

- **Định nghĩa Primary Key Constraint với bảng đã tồn tại:**
  - Định nghĩa ở mức cột

    ALTER TABLE tablename ADD columname datatype
    [CONSTRAINT constraintname] PRIMARY KEY
    [ON {filegroup | DEFAULT}] [,...]

  - Định nghĩa ở mức bảng

    ALTER TABLE *tablename* ADD[CONSTRAINT *constraintname*]
    PRIMARY KEY { (*column* [ASC | DESC] [,...n]) }
    [ON { *filegroup* | DEFAULT }]

Ví dụ 2: Thêm một PRIMARY KEY constraint cho một bảng có sẵn

```
CREATE TABLE Table3
     ( col1 INT NOT NULL,
       col2 VARCHAR(30)
      )
ALTER TABLE Table3
ADD CONSTRAINT table3_PK PRIMARY KEY (col1)
EXEC Sp_helpconstraint Table3
```

Lưu ý: Khi một Primary Key Constraint được thêm vào một bảng với cột có sẵn trong bảng thì SQL Server sẽ kiểm tra dữ liệu hiện có có tuân theo các qui tắc của một Primary key hay không: **Không Null, Không trùng lắp**. Nếu không thỏa qui tắc thì sẽ không tạo được Primary Key Constraint.

Xóa một Primary Key Constraint

```
ALTER TABLE Table2
DROP CONSTRAINT table2_PK
```

Lưu ý: Không thể xóa một Primary Key constraint nếu nó được tham chiếu bởi một Foreign key Constraint của một bảng khác, muốn xóa thì phải xóa Foreign key Constraint trước.

6.2.6 Unique Constraints

Unique Constraints dùng để đảm bảo không có giá trị trùng ở các cột. Mặc dù cả Unique constraint và Primary key constraint đều tuân theo tính duy nhất, nhưng hãy sử dụng Unique contarint khi bạn muốn bảo đảm tính duy nhất của:

- Một cột, hay sự kết hợp giữa các cột, vốn không phải là khóa chính.
- Một cột chấp nhận giá trị null, trong khi đó constraint primary key không thể ấn định trên cột này.
- Một bảng có thể có nhiều Unique constraint.

Định nghĩa Unique Constraint:

- **Định nghĩa UNIQUE CONSTRAINT khi tạo bảng**
  - Định nghĩa ở mức cột

    CREATE TABLE *tablename* (*columname datatype*
    [CONSTRAINT *constraintname*] UNIQUE [CLUSTERED | NONCLUSTERED]
    [ON {*filegroup* | DEFAULT}] [,...]

  - Định nghĩa ở mức bảng

    CREATE TABLE *tablename*
    (*columname datatype* [,...], [CONSTRAINT *constraintname*]
    UNIQUE [CLUSTERED | NONCLUSTERED]
    { (column [ASC | DESC] [,...n]) }
    [ON { *filegroup* | DEFAULT }])
- **Định nghĩa UNIQUE CONSTRAINT với bảng đã tồn tại**
  - Định nghĩa ở mức cột

    ALTER TABLE *tablename*
    ADD *columname datatype* [CONSTRAINT *constraintname*]
    UNIQUE [CLUSTERED | NONCLUSTERED]
    [ON {*filegroup* | DEFAULT}] [,...]

  - Định nghĩa ở mức bảng

    ALTER TABLE *tablename*
    ADD [CONSTRAINT *constraintname*]
    UNIQUE [CLUSTERED | NONCLUSTERED]
    { (column [ASC | DESC] [,...n]) }
    [ON { *filegroup* | DEFAULT }]

Ví dụ:

```
ALTER TABLE Nhanvien
ADD Scmnd CHAR(15) CONSTRAINT NV_CMND_UNIQUE
UNIQUE
```

6.2.7 Foreign Key Constraint

Một khóa ngoại (Foreign key) là một cột hay sự kết hợp của nhiều cột được thiết lập và tuân theo một liên kết giữa các dữ liệu trong hai bảng. Một liên kết được tạo ra giữa hai bảng bằng cách bổ sung một hay nhiều cột có chứa giá trị khóa primay key của một bảng vào một bảng khác. Các cột này trở thành khóa ngoại của bảng thứ hai.

Ta có thể quy định khóa ngoại bằng các ấn định một constraint Foreign key khi bạn tạo hay thay đổi một bảng.

Một constraint Foreign key không bắt buộc phải liên kết với chỉ một constraint Foreign key trong một bảng khác, nó cũng có thể được ấn định để tham chiếu các cột của một constraint Foreign Unique trong một bảng khác.

Mặc dù mục đích chính của một constraint Foreign key là điều khiển dữ liệu có thể được chứa trong bảng khóa ngoại, nhưng nó cũng điều khiển các thay đổi đối với bảng khóa chính. Chẳng hạn, nếu một mẫu tin phòng ban bị xóa ra khỏi bảng Phongban, và mã phòng ban được sử dụng cho các nhân viên trong bảng NhanVien, tính toàn vẹn trong mối quan hệ giữa hai bảng này sẽ bị phá vỡ. Các dòng nhân viên có mã phòng ban bị xóa sẽ nằm mồ côi trong bảng NhanVien mà không có liên kết với dữ liệu trong bảng PhongBan.

Một constraint Foreign key sẽ ngăn chặn tình trạng mồ côi dữ liệu. Constraint Foreign key sẽ bảo đảm không cho phép bạn xóa dữ liệu trong bản chính nếu các dữ liệu này có sự liên kết với dữ liệu trong bảng khóa ngoại.

- **Định nghĩa FOREIGN KEY CONSTRAINT khi tạo bảng**
  - Định nghĩa ở mức cột

CREATE TABLE *tablename*
(*columname datatype* [CONSTRAINT *constraintname*]
[FOREIGN KEY] REFERENCES *ref_table* [(*ref_column*)]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION] [,...]

  - Định nghĩa ở mức bảng

CREATE TABLE *tablename*
(*columname datatype* [,...],
[CONSTRAINT *constraintname*]
FOREIGN KEY [(*column* [,...*n*])]
REFERENCES *ref_table* [(*ref_column* [,...n])]
[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION]

Ví dụ:

```
CREATE TABLE PhongBan
     ( MaPb INT,
       TenPb VARCHAR(30)
       CONSTRAINT Pb_Pk (MaPb) PRIMARY KEY(MaPb)
      )
GO
CREATE TABLE NhanVien
     ( MaNv INT,
       TenNV VARCHAR(30),
       MaPB int,
       MaCv int
       CONSTRAINT Nv_Pk PRIMARY KEY(MaNv)
       CONSTRAINT Nv_Fk FOREIGN KEY (MaPb) REFERENCES
      PhongBan(MaPb)
     )
GO
-- Xem các constraint
SP_HELPCONSTRAINT NHANVIEN
GO
```

- **Định nghĩa Foreign Key Constraint với bảng đã tồn tại:**

  ALTER TABLE *tablename*
  [WITH CHECK | WITH NOCHECK] ADD
  [CONSTRAINT *constraintname*]
  FOREIGN KEY [(*column* [,...*n*])]
  REFERENCES *ref_table* [(*ref_column*[,...*n*])]
  [ON DELETE { CASCADE | NO ACTION }]
  [ON UPDATE { CASCADE | NO ACTION }]
  [NOT FOR REPLICATION]

Trong đó:

WITH CHECK: trước kho tạo ràng buộc, SQL Server sẽ kiểm tra dữ liệu hiện có có vi phạm ràng buộc hay không, nếu có sẽ không tạo constraint.

WITH NOCHECK: Tạo Constraint mà không cần kiểm tra dữ liệu hiện có có vi phạm ràng buộc hay không

Ví dụ:

```
CREATE TABLE ChucVu
     ( MaCv INT PRIMARY KEY,
       TenCv VARCHAR(30),
       HSPC Real
```

```
        )
    GO
    ALTER TABLE Nhanvien
    ADD CONSTRAINT Nv_Cv_Pk FOREIGN KEY (Macv) REFERENCES
ChucVu(MaCv)
```

Luru ý:

ON DELETE CASCADE| NO ACTION

ON UPDATE CASCADE| NO ACTION

BÀI 7:
TRUY XUẤT CƠ SỞ DỮ LIỆU CỦA SQL SERVER

Mục đích chính của CSDL trong SQL Server là lưu trữ dữ liệu sao cho dữ liệu dễ dàng được khai thác bởi người sử dụng. Bạn cũng có thể truy cập dữ liệu thông qua một ứng dụng hoặc các trình tiện ích để gửi yêu cầu nhập dữ liệu hoặc hiệu chỉnh dữ liệu đến SQL Server. Nhằm mục đích tìm hiểu ta dùng SQL Query Analyzer như là một công cụ chính để truy xuất và hiệu chỉnh dữ liệu trong một CSDL của SQL Server.

7.1 Câu lệnh SELECT

Câu lệnh Select được sử dụng một cách thường xuyên và là cách cơ bản để truy vấn dữ liệu.

SELECT [ALL | DISTINCT] [TOP n [WITH TIES]]*select_list*
[INTO *new_table*]
FROM *table_source*
[WHERE *search_condition*]
[GROUP BY *group_by_expression*]
[HAVING *search_condition*]
[ORDER BY *order_expression* [ASC | DESC]]

Products(ProductID,ProductName, SupplierID, CategoryID, UnitPrice, …)

Customers(CustomerID, CompanyName, Address, City, Region, Country, …)

Employees(EmployeeID, LastName, FirstName, BirthDate, City, …)

Orders(OderID, CustomerID, EmployeeID, OrderDate,…)

Order Details(OrderID, ProductID, UnitPrice, Quantity, Discount)

Mệnh đề SELECT

```
USE NorthWind
SELECT * FROM Products

SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE UnitPrice >40
ORDER BY UnitPrice DESC
```

Từ khóa **DISTINCT**: Loại bỏ các mẫu tin trùng trong tập kết quả.

```
SELECT DISTINCT City, Region
FROM Customers
ORDER BY Region
```

Từ khóa **TOP n :** Lấy ra n mẫu tin đầu tiên

-- Chỉ lấy đúng 5 mẫu tin đầu tiên có số lượng bán cao nhất.

```
SELECT TOP 5 orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
```

```
--- WITH TIES có nghĩa là các mẫu tin ngang bằng giá trị
trong cột ORDER BY được liệt kê.
    SELECT TOP 5 WITH TIES orderid, productid, quantity
    FROM [order details]
    ORDER BY quantity DESC
    Từ khóa AS: Đặt bí danh cho bảng, cột

    SELECT p.Productid, p.Productname
    FROM Products As p
```

Mệnh đề SELECT INTO: Sinh thêm một bảng mới mà dữ liệu được lấy từ các bảng khác.

```
    SELECT FirstName, LastName
    INTO EmployeeNames
    FROM Northwind.dbo.Employees
```

Dùng với các hàm SUM, MAX, MIN, AVG, COUNT

```
    --AVG
    USE Northwind
    SELECT AVG(UnitPrice)
    FROM dbo.Products
    -- SUM
    SELECT SUM(Quantity)
    FROM dbo.[Order Details]
    --- Phân biệt Count va Count(*); Count(*) đếm tất cả
    các mẫu tin có trong bảng
    SELECT COUNT(*)
    FROM dbo.Employees
    --- Count(Reportsto) đếm tất cả các giá trị có trong
    cột ReportTo, nếu giá trị của Reportsto là NULL thì SQL
    Server sẽ bỏ qua không đếm
    USE Northwind
    SELECT COUNT(ReportsTo)
    FROM dbo.Employees
```

Mệnh đề WHERE, GROUP BY, và HAVING: Where và Having dùng để lọc dữ liệu trong câu Select. **Select với Where. Lọc dữ liệu thỏa điều kiện. Các phép toán có thể dùng trong where: >, >=, <, <=, =, AND, OR, BETWEEN … AND, !=, IS NULL, NOT IS NULL**

Cho biết danh sách các mặt hàng đã bán trong hóa đơn số 2

```
SELECT OrderID, Quantity
FROM Orderhist
WHERE OrderID=2
```

Select với Group by: Kết nhóm

Cho biết tổng lượïng hàng đã bán ứng với mỗi ProductId

```
SELECT productID,SUM(quantity) AS Total_quantity
FROM [Order details]
GROUP BY productID
```

Cho biết tổng lượng hàng đã bán ứng với ProductId=2

```
SELECT ProductID, SUM(Quantity) AS Total_quantity
FROM [Order Details]
WHERE ProductID=2
GROUP BY productID
```

Cho biết danh sách các mã hàng có tổng số lượng >=30

```
SELECT ProductID,SUM(Quantity) AS Total_quantity
FROM [Orde details]
GROUP BY ProductID
HAVING SUM(quantity)>=30
```

Lưu ý: *Having để giới hạn kết quả do Group By sản sinh ở các hàm Where giới hạn trước khi Group By*

Cho biết tổng lượng hàng đã bán ứng với mỗi mã nhóm hàng

```
SELECT Categoryid, SUM(Quantity) AS Total_quantity
FROM [order details] JOIN products ON [order
details].productId=products.productId
GROUP BY Categoryid
```

Tổng số lượng bán ứng với mỗi hóa đơn

```
SELECT orderid, SUM(quantity) AS Total_quantity
FROM [order details]
GROUP BY orderid
HAVING SUM(quantity)>=250
```

Sử dụng Group By với toán tử ROLLUP.

ROLL UP Sẽ chèn thêm các dòng Total nằm trước các nhóm mẫu tin được phân theo GROUP BY.

Ví dụ:

/* Tổng số lượng đã được đặt hàng cho mỗi Product ứng mỗi Order cho các Order có OrderID<1025

```
*/
use northwind
SELECT     productID,orderid,     SUM(quantity)     AS
Total_quantity
FROM [Order details]
WHERE orderid<10250
GROUP BY OrderId,productId
ORDER BY OrderId, productID
```

Kết quả

Không có With Rollup

ProductID	Orderid	Total_quantity
11	10248	12
42	10248	10
72	10248	5
14	10249	9
51	10249	40

Có With Rollup

ProductID	Orderid	Total_quantity
NULL	NULL	76
NULL	10248	27
11	10248	12
42	10248	10
72	10248	5
NULL	10249	49
14	10249	9
51	10249	40

Ví dụ khác: Danh sách Order ứng với tổng quantity của product có mã ProductID là 8 hoặc 9.

```
SELECT ProductID,Orderid, SUM(quantity) AS
Total_quantity
FROM [Order Details]
WHERE ProductID IN (8,9)
GROUP BY ProductID, OrderId
WITH ROLLUP
ORDER BY ProductID, OrderId

/* total quantity for: each product for each order. all
products for each order
*/
SELECT ProductID, Orderid, SUM(quantity) AS
Total_quantity
FROM [Order details]
WHERE orderid<10250
GROUP BY OrderId,productId
WITH ROLLUP
ORDER BY OrderId, productID
```

Sử dụng mệnh đề Group By với toán tử CUBE: Tương tự như Rollup nhưng thêm các dòng Total của mỗi sự kết hợp có thể có giữa các cột

Ví dụ:

```
SELECT ProductID, Orderid, SUM(quantity) AS Total_quantity
FROM [Order details]
WHERE orderid<10250
GROUP BY OrderId,productId
WITH CUBE
ORDER BY OrderId, productID
```

Kết quả

NULL	**NULL**	**76**
11	**NULL**	**12**
14	**NULL**	**9**
42	**NULL**	**10**
51	**NULL**	**40**
72	**NULL**	**5**
NULL	**10248**	**27**
11	10248	12
42	10248	10
72	10248	5
NULL	**10249**	**49**
14	10249	9
51	10249	40

Ví dụ khác:

```
SELECT ProductID, Orderid, SUM(quantity) AS Total_quantity
FROM [Order Details]
GROUP BY OrderId, ProductId
WITH CUBE
ORDER BY ProductID, OrderId
```

Dùng hàm GROUPING: Hàm Grouping sẽ trả về một giá trị là 0 hoặc 1 để xác định dòng kết quả đó là dòng tổng (1) hay là dòng chi tiết (0)

Ví dụ:

```
SELECT ProductID, grouping(productID)as a, Orderid, grouping(orderid) as b,SUM(quantity) AS Total_quantity
FROM [Order details]
WHERE orderid<10250
```

```
GROUP BY OrderId,productId
WITH CUBE
ORDER BY OrderId, productID
```

ProductID	a	Orderid	b	Total_quantity
NULL	1	NULL	1	76
11	0	NULL	1	12
14	0	NULL	1	9
42	0	NULL	1	10
51	0	NULL	1	40
72	0	NULL	1	5
NULL	1	10248	0	27
11	0	10248	0	12
42	0	10248	0	10
72	0	10248	0	5
NULL	1	10249	0	49
14	0	10249	0	9
51	0	10249	0	40

Dùng toán tử COMPUTE và COMPUTE BY: Thông thường dùng để kiểm tra số liệu, dùng kèm với các hàm thống kê SUM, AVG, MAX, MIN…. COMPUTE … BY … :có kết nhóm

```
SELECT productID,orderid, quantity
FROM [Order Details]
ORDER BY productID,OrderId
COMPUTE SUM(quantity)

SELECT productID,orderid, quantity
FROM [Order Details]
ORDER BY productID,OrderId
COMPUTE SUM(quantity) BY productid
COMPUTE SUM(quantity)
```

7.2 Sử dụng JOINS để truy xuất dữ liệu

Bằng JOIN, chúng ta có thể lấy dữ liệu từ hai hoặc nhiều bảng dựa trên mối quan hệ giữa các bảng. Tuy nhiên nếu ta không thích dùng Join để lấy dữ liệu thì bạn cũng có thể viết các câu truy vấn bằng dạng truy vấn con (Subqueries). Dùng Joins thì tốc độ thực hiện của câu truy nhanh hơn SubQueries nhưng lại khó hiểu hơn. Trong một truy vấn tham chiếu đến nhiều table, thì tất cả các cột phải được chỉ rõ một các tường minh là cột đó lấy từ table nào.

Thông thường thì điều kiện kết nối là dùng phép so sánh bằng, nhưng trong SQL Server ta có thể định nghĩa mối quan hệ giữa trên các toán tử khác như <>, >, >=, <, <= …

Các cột được dùng để kết nối không nhất thiết phải cùng tên hay cùng kiểu dữ liệu. Nếu kiểu dữ liệu khác nhau thì có thể dùng các hàm để chuyển đổi kiểu dữ liệu.

Gồm hai loại Joins chính: INNER JOINS và OUTER JOINS. Tuynhiên ta còn có thể tạo một dạng Join khác như CROSS-JOINS và SELF-JOINS.

Inner Joins: Liên kết Inner Joins chỉ trả về những mẫu tin đều hiển hữu ở cả hai bảng quan hệ và phải thỏa mãn điều kiện kết.

Ví dụ:

```
SELECT Em.EmployeeID,LastName+ ' '+FirstName AS EmployeeName,
COUNT(OrderID) AS TotalOrders
FROM Employees AS Em INNER JOIN Orders AS O ON
Em.EmployeeID=O.EmployeeID
WHERE MONTH(OrderDate)=8 AND YEAR(Orderdate)=1996
GROUP BY Em.EmployeeID,LastName+ ' '+FirstName
ORDER BY Em.EmployeeID
```

Điều kiện kết nằm ở mệnh đề Where

```
SELECT P.ProductID, S.SupplierID, S.CompanyName
FROM Suppliers AS S, Products AS P
WHERE S.SupplierID = P.SupplierID
 AND P.UnitPrice > $10
 AND S.CompanyName LIKE N'F%'
```

Điều kiện kết nằm ở mệnh đề From

```
SELECT P.ProductID, S.SupplierID, S.CompanyName
FROM Suppliers AS S JOIN Products AS P
     ON (S.SupplierID = P.SupplierID)
WHERE P.UnitPrice > $10
  AND S.CompanyName LIKE N'F%'
```

Outer Joins: trả về tất cả những mẫu tin nằm ít nhất một bảng nào đó trong các bảng tham gia kết nối và cũng phải thoả điều kiện kết. Outer Joins cung cấp 3 kiểu outer Join: Left, Right, và Full. Trả về tất cả các dòng từ bảng bên trái mà được tham chiếu từ Left Outer Joins. Trả về tất cả các dòng từ bảng bên phải mà được tham chiếu từ Right Outer Joins. Trả về tất cả các dòng từ cả 2 bảng mà được tham chiếu từ Full Outer Joins.

Ví dụ : Dùng câu truy vấn sau và lần lượt thay đổi kiểu Join (Inner, Left, Right, Full), cho thi hành và cho nhận xét kết quả của câu truy vấn. Tự rút ra kết luận

```
SELECT O.OrderID, O.CustomerID, c.ContactName, C.City
```

```
FROM Orders O LEFT JOIN Customers C
     ON O.CustomerID = C.CustomerID AND
O.ShipCity=C.City
ORDER BY O.OrderID
```

Cross Jions: Mỗi dòng trong tất cả các dòng của bảng bên trái sẽ kết hợp với tất cả các dòng của bảng bên phải. Giả sử X, Y là số dòng của bảng bên trái và bên phải, thì tập kết quả sau khi Cross Join có X*Y dòng.

Sefl Joins: Tự liên kết

7.3 Dùng Sub-Queries

Subqueries là một câu lệnh Select mà nó trả về một giá trị đơn hoặc một tập các giá trị và nó được nằm trong các câu lệnh như Select, Insert, Update, or Delete. Một SubQuery có thể được dùng bất kỳ nơi nào trong biểu thức được cho phép. Một SubQuery cũng được gọi trong một truy vấn khác, hoặc một câu lệnh select có subquery cũgn có thể được gọi từ một câu Select khác (32 cấp).

- ✓ Ngắt câu lệnh phức tạp thành những đoạn truy vấn đơn giản.
- ✓ Trả lời một truy vấn từ một truy vấn khác.
- ✓ Các Sub Query hầu như có thể viết bằng Join và SQL Server luôn luôn thi hành những câu lệnh Join nhanh hơn SubQueries.
- ✓ Không thể dùng SubQueries trên cột hình ảnh.

```
/* SELECT statement built using a subquery. */
```

Lọc ra những Product Name có UnitPrice bằng với Unitprice của ProductName là 'Sir Rodney''s Scones'

```
SELECT ProductName
FROM Products
WHERE UnitPrice =
      (SELECT UnitPrice
       FROM Products
       WHERE ProductName = 'Sir Rodney''s Scones')

/* SELECT statement built using a join that returns the same
result set. */
SELECT ProductName
FROM Products AS Prd1
     JOIN Products AS Prd2
       ON (Prd1.UnitPrice = Prd2.UnitPrice)
WHERE Prd2.ProductName = 'Sir Rodney''s Scones'
```

Dùng từ khóa IN trong subquery.

Lọc ra những OrderId, EmployeeId do những Employee ở City Seattle Order.

```
SELECT OrderId, EmployeeID AS EmpID
FROM Orders
```

```
WHERE EmployeeID    IN (
                    SELECT EmployeeId
                    FROM Employees
                    WHERE City='Seattle'
                   )
ORDER BY OrderID
```

Dùng các toán tử so sánh trong Subquery: Các toán tử bao gồm =, >, >=, <, <=, <>, >=ALL, > ANY, ….

Lọc ra những OrderId, ProductId, UnitPrice mà có Unitprice lớn hơn nhữ Unitprice được bán bởi EmployeeID=5

```
SELECT OrderId, [Order Details].ProductId, [Order
Details].UnitPrice
FROM [Order Details]
WHERE Unitprice  > ALL ( SELECT [Order
Details].UnitPrice
                    FROM [Order Details] JOIN Orders
                    ON [Order Details].OrderId=
Orders.OrderId
                    where Orders.EmployeeID=5
                    )
ORDER BY [Order Details].UnitPrice,OrderID)
ORDER BY UnitPrice.OrderID
```

Dùng từ khóa EXISTS, NO EXISTS trong subquery:

Lọc ra những OrderId ứng với các CustomerId mà những Customer này ở London City.

```
SELECT OrderId, CustomerId
FROM Orders
WHERE EXISTS ( SELECT * FROM Customers
               WHERE
Customers.CustomerId=Orders.CustomerId
               AND City='LonDon'
               )
ORDER BY OrderID
```

7.4 Hiệu chỉnh dữ liệu trong cơ sở dữ liệu của SQL SERVER

7.4.1 Chèn (INSERT) dữ liệu vào CSDL.

Câu lệnh Insert được dùng để thêm một hoặc nhiều dòng dữ liệu vào bảng hoặc một View. Một số lưu ý:

- ✓ Cố gắng chèn dữ liệu đúng kiểu dữ liệu của cột, tránh đưa giá trị Null vào cột không chấp nhận giá trị Null.
- ✓ Tuân thủ đúng các toàn vẹn dữ liệu, bẩy lỗi.
- ✓ Chỉ định một danh sách các giá trị (values) ứng với danh sách các cột, nếu không chỉ định thì insert sẽ theo thứ tự cấu trúc của table.
- ✓ Cột có thuộc tính Identity thì không cần chỉ định giá trị. Nếu cột có định nghĩa Default và chúng ta chấp nhận giá trị default thì ta có thể bỏ qua việc chỉ định giá trị của cột này.
- ✓ Khi chèn dữ liệu thì tại một thời điểm cũng như một câu lệnh chỉ có thể chèn vào một bảng duy nhất.

7.4.1.1 Chèn mẫu tin từ danh sách các giá trị được chỉ định (Insert … Values)

INSERT <TableName> [(field1, field2, ...)] VALUES (Value1, Value2,...)

```
CREATE TABLE NewProducts
(
ProductID INT NOT NULL,
     ProductName NVARCHAR(40),
     CategoryID INT NULL,
     UnitPrice MONEY NULL,
     SupplierID INT NULL
)
GO
INSERT NewProducts (ProductID, ProductName)
       VALUES (123, 'Ice Tea')
```

7.4.1.2 Chèn dữ liệu bằng các giá trị từ các bảng khác (Insert … Select)

INSERT <TableName> [(field1, field2, ...)]
SELECT (Value1, Value2,...)

```
INSERT NewProducts (ProductID, ProductName)
       SELECT ProductID, ProductName
       FROM Products
     WHERE CategoryID=2
INSERT NewProducts (Od.ProductID, P.ProductName)
     SELECT OD.ProductID, ProductName
     FROM Products as P INNER JOIN [Order Details]
AS Od
          ON P. ProductID = Od. ProductID
     WHERE CategoryID<>2
```

7.4.1.3 Chèn mẫu tin từ một stored Procedure (trình bày sau)

7.4.1.4 Xây dựng một Table bằng SELECT INTO

```
SELECT DISTINCT C.CustomerID, C.CompanyName
INTO NewCustomers
FROM Customers AS C INNER JOIN Orders
ON C.CustomerID= orders.CustomerID
AND orders.EmployeeID=2
```

7.4.2 Cập nhật (UPDATE) dữ liệu vào CSDL.

SQL Server cung cấp 3 phương pháp để thay đổi/cập nhật dữ liệu trong một bảng có sẵn. Đó là câu lệnh Update, giao diện lập trình ứng dụng (Application Programming Interfaces - APIs) và dùng con trỏ (Cursors). Trong phần bài học này chỉ trình bày câu lệnh Update.

UPDATE <table_name>
SET <column_name> = <expression>
[FROM <table_list>]
[WHERE <search_condition>
WHERE CURRENT OF <Variable_Cursor>

Ví dụ:

```
USE Northwind
UPDATE dbo.Products
SET UnitPrice=UnitPrice*1.1
GO
UPDATE [Order Details]
SET Discount=Discount+0.05
WHERE Discount<>0 AND ProductId=2
GO
UPDATE [Order Details]
SET UnitPrice=
(SELECT UnitPrice+ UnitPrice*0.2
     FROM Products
     WHERE = ProductId=2
          )
               WHERE ProductId=2
```

7.4.3 Xóa dữ liệu trong cơ sở dữ liệu.

7.4.3.1 Dùng câu lệnh DELETE.

Xóa một hoặc nhiều dòng trong một bảng hay một truy vấn.

DELETE table_or_view FROM table_sources WHERE search_condition

```
DELETE Orders
```

```
FROM Orders
WHERE EmployeeID IN ( SELECT EmployeeId
                        FROM Employees
                        WHERE City='Seattle'
                      )
```

7.4.3.2 Dùng APIs và cursors để xoá dữ liệu (trình bày sau).

7.4.3.3 Dùng câu lệnh TRUNCATE TABLE.

- Để xóa toàn bộ dữ liệu trong một bảng.
- Về phần chức năng, hoàn toán giống như câu lệnh Delete.
- Nhanh hơn câu lệnh Delete
- Không bật các bẩy lỗi (trigger)

Cú pháp:

TRUNCATE TABLE *Tên Table*

Ví dụ:

```
TRUNCATE TABLE NewProducts
```

BÀI 8:
KHUNG NHÌN - VIEW

8.1 Giới thiệu về View.

View là một bảng ảo mà nội dung được định nghĩa bởi một truy vấn (câu Select). Giống như một bảng thực, một view bao gồm một tập các cột và các dòng dữ liệu. Tuy nhiên, một view không là nơi lưu trữ dữ liệu. Các dòng và cột của dữ liệu được tham chiếu từ các bảng trong một truy vấn mà định nghĩa View và là kết quả động khi View được tham chiếu.

Dùng view để:

- Chỉ cho User xem những gì cần cho xem.
- Đơn giản hóa việc truy cập dữ liệu.
- Dùng để lựa chọn những dữ liệu cần thiết ứng với mỗi user.
- Dùng View để Import, Export.
- Kết hợp các dữ liệu khác nhau.

Hạn chế khi định nghĩa View:

- Không bao gồm các mệnh đề COMPUTER hoặc COMPUTER BY.
- Không bao gồm từ khóa INTO.
- Chỉ được dùng ORDER BY chỉ khi từ khóa TOP được dùng.
- Không thể tham chiếu quá 1024 cột.
- Không thể kết hợp với câu lệnh T-SQL khác trong một cùng một bó lệnh.
- Không thể định nghĩa chỉ mục full text trên View.

Partitioned Views: Một partition View kết nối theo chiều dọc các dữ liệu phân tán từ một tập các bảng ở một hay nhiều server, các dữ liệu sẽ hiện lên như thể là chúng được lấy từ một bảng. Có hai lạoi: Local partition view là view có tham chiếu các table và các view khác nằm trong cùng một serever. Distributed partition view có ít nhất một bảng nằm ở server khác.

8.2 Tạo, hiệu chỉnh, xóa View

Tạo View

```
CREATE VIEW [ < database_name > . ] [ < owner > . ] view_name [ (column [,...n ]) ]
[ WITH < view_attribute > [,...n ] ]
AS
select_statement
[ WITH CHECK OPTION ]
< view_attribute > ::=
  { ENCRYPTION | SCHEMABINDING }
```

Giải thích

view_name: là tên của View. Tên View phải tuân thủ các qui tắc định danh.

***Column*:** Tên được dùng cho cột trong view. Tên cột chỉ dùng trong trường hợp cột được phát sinh từ một biểu thức, hàm, một hằng, các cột trong các table trùng tên. Tuy nhiên tên cột cũng có thể được ấn định trong câu lệnh Select. Không chỉ định tên cột chính là tên các cột trong câu lệnh select

select_statement: Là câu lệnh select để định nghĩa View. Nó có thể tham chiếu một hoặc nhiều bảng hoặc các View khác với một câu Select phức tạp.

Lưu ý: Để tạo một view, bạn phải có quyền dành riêng trên các bảng hoặc view tham chiếu trong định nghĩa view.

WITH CHECK OPTION: Bắt buộc tất cả các câu lệnh hiệu chỉnh dữ liệu thực thi dựa vào View phải tuyệt đối tôn trọng triệt để đến tập tiêu chuẩn trong câu lệnh Select. Nếu bạn dùng từ khóa này, các dòng không thể được hiểu chỉnh trong cách mà tại sao chúng hiện trong view. Bất kỳ hiệu chỉnh nào mà sẽ gây ra tình trạng thay đổi đều bị hũy bỏ, và một lỗi được hiện ra.

WITH ENCRYPTION: Mã hóa câu lệnh Select tạo ra view

SCHEMABINDING: Kết view với một giản đồ. Khi SCHEMABINDING được chỉ định, câu lệnh Select phải chỉ rõ chủ quyền của các bảng, các view. Các hàm được tham chiếu View hay bảng tham gia trong view được tạo với schema không thể xóa trừ phi view đó bị xóa hoặc thay đổi cơ chế này. Câu lệnh Alter Table trên bảng tham gia trong view cũng bị lỗi.

Ví dụ 1: Xem danh sách các InvoiceNo và CustNo

```
CREATE VIEW vwSim
AS
SELECT InvoiceNo, CustNo
From tblSim
Where InvoiceDate = Getdate()
ORDER BY InvoiceNo
```

Xem nội dung View

```
Select * From vwSim
```

Ví dụ 2:

```
CREATE VIEW vwSales
AS
Select c.CustNo,c.CustName, sim.InvoiceNo,
sim.InvoiceDate, sid.ItemNo, sid.Quatity
From TblCustomer c INNER JOIN TblSim sim
ON c.CustNo=sim.CustNo INNER JOIN TblSid sid
ON sim.InvoiceDate = sid.InvoiceDate
```

Xem nội dung View

```
Select * From vwSales
```

8.3 Tạo Partition view

- Các bảng tham gia trong Partition view phải có cấu trúc giống nhau.
- Có một cột có check contraint, với phạm vi của Check constraint ở mỗi bảng là khác nhau.
- Tạo view bằng cách kết các dữ liệu bằng từ khóa UNION ALL

Ví dụ: Ta có 3 table tương ứng dùng để lưu trữ các khách hàng ở 3 miền Bắc, Trung, Nam có cấu trúc tạo như sau:

```
Create Table KH_BAC
```

```
        (Makh int primary key,
        TenKh Nchar(30),
        Khuvuc Nvarchar(30) CHECK (Khuvuc='Bac bo')
        )
    Create Table KH_TRUNG
        (Makh int primary key,
        TenKh Nchar(30),
        Khuvuc Nvarchar(30) CHECK (Khuvuc='Trung bo')
        )
    Create Table KH_NAM
        (Makh int primary key,
        TenKh Nchar(30),
        Khuvuc Nvarchar(30) CHECK (Khuvuc='Nam bo')
        )
```

Tạo một partition View gộp 3 bảng trên lại với nhau:

```
        Create View Khachhang
        AS
            Select * From KH_BAC
            UNION ALL
            Select * From KH_TRUNG
            UNION ALL
            Select * From KH_NAM
```

8.4 Truy xuất dữ liệu thông qua View.

(Tương tự như truy xuất dữ liệu trên bảng)

8.4.1 Xem dữ liệu thông qua view.

Dùng câu lệnh Select (Ví dụ ở trên)

8.4.2 Hiệu chỉnh dữ liệu thông qua View.

Thao tác hiệu chỉnh dữ liệu giống thao tác hiệu chỉnh dữ liệu trên một bảng. Tuy nhiên, view phải thỏa mãn điều kiện sau:

- View chỉ tham chiếu duy nhất một bảng.
- Thao tác Delete không bao giờ được phép thực hiện trên nhiều bảng trong View.
- Các hàm kết hợp Group By, Union, Distinct, Top không dùng trong danh sách chọn trong câu Select của View.
- Không có các cột tính toán.

Ví dụ:

```
        INSERT INTO Khachhang VALUES (11,'ddddd','Nam bo')
```

Hiệu chỉnh dữ liệu thông qua partitioned View

Khi dùng lệnh Insert và update phải tôn trọng các qui tắc sau:

- Tất cả các cột phải có giá trị ngay cả cột chấp nhận null và cột có giá trị defaul.
- Từ khóa Defaul không được sử dụng trong câu Insert, update.
- Phải có giá trị đúng của cột có check constraint.
- Câu lệnh insert không cho phép nếu bảng thành viên có cột có thuộc tính identity, cột timestamp.
- Không insert hoặc Update nếu có một kết self-join trong cùng view hay bảng thành viên.

Khi dùng lệnh delete, ta có thể xóa các mẫu tin trong bảng thành viên thông qua view. Lệnh Delete không thực thi nếu có liên kết Sefl-join

BÀI 9:
CHUYỂN ĐỔI DỮ LIỆU

9.1 Khái niệm chuyển đổi và biến đổi dữ liệu.

Sau khi bạn tạo CSDL của bạn, bạn cần nhập các mẫu dữ liệu. Thông thường, bạn thường đưa dữ liệu vào (importing data) hoặc hoặc chuyển dữ liệu (transfering) có sẵn từ một hoặc nhiều nguồn dữ liệu khác đến hoặc đi từ SQL Server 2000. Trong bài này chúng ta nghiên cứu import dữ liệu từ nguồn dữ liệu khác, đồng thời cũng giới thiệu các công cụ (Tool) chính sử dụng để importing data và biến đổi dữ liệu (transforming data). Các công cụ đó là DTS, Bcp, và lệnh BULK INSERT.

9.1.1 Import/Export dữ liệu.

Import dữ liệu là quá trình đưa dữ liệu có sẵn từ nguồn dữ liệu khác hoặc chính SQL Server vào trong SQL Server. Export là quá trình ngược lại với import, đưa dữ liệu của SQL Server ra ngoài nguồn dữ liệu bên ngoài. Nguồn dữ liệu đó có thể là một CSDL hảng thứ ba, bảng tính, tập tin văn bản (Text). Tuy nhiên trước khi bạn import/export dữ liệu này vào, bạn phải thực hiện các tác vụ chuẩn bị để ước lượng dữ liệu bên ngoài và quyết định các bước sẽ phải thực hiện trong tiến trình import/export. Các bước chuẩn bị này cũng sẽ giúp bạn chọn công cụ thích hợp để dùng

- Quyết định tính nhất quán (consistency) của dữ liệu hiện đã có trong nguồn dữ liệu bên ngoài/bên trong.
- Quyết định những cột được đưa vào/đưa ra.
- Quyết định dạng dữ liệu (Format) của dữ liệu có sẵn nên hiệu chỉnh để nó nhất quán trong CSDL đích đến (Ví dụ: Cần đổi dạng ngày hoặc chuyển giá trị số sang giá trị chuỗi như 1, 2, 3 chuyển thành nghèo, trung bình, khá).
- Quyết định cột dữ liệu có sẵn nên hiệu chỉnh.
- Quyết định import/export dữ liệu sẽ là một tác vụ thực hiện một lần hay một tác vụ thực hiện định kỳ.
- Quyết định các truy xuất dữ liệu có sẵn là truy xuất trực tiếp hay gián tiếp.
- ….

9.1.2 Biến đổi dữ liệu (Data Transformations)

Sau khi bạn ước lượng dữ liệu trong nguồn dữ liệu bên ngoài/bên trong, bạn cần quyết định cách tiến hành. Đôi khi, những thay đổi dữ liệu có thể thực hiện ngay trong nguồn dữ liệu bên ngoài nhưng thông thường những thay đổi này không có thể thực hiện trong nguồn dữ liệu bên ngoài mà không hoặc dừng ứng dụng hiện có (ví dụ: thêm cột hoặc thay đổi định dạng cột) hoặc tiêu tốn quá nhiều thời gian (ví dụ: Cố gắng thúc ép nhất quán dữ liệu tại nơi sự nhất quán chưa có. Những thay đổi này có thể hoặc là sau khi dữ liệu được import vào SQL Server, sử dụng các bảng tạm và sử dụng câu lệnh Transact-SQL để lọc và tinh chế dữ liệu, hoặc có thể được thực hiện trong tiến trình import vào chính bảng. Những thay đổi đến dữ liệu tạo trong tiến trình import và export được nói đến như các biến đổi dữ liệu. Một biến đổi xảy ra khi một hoặc nhiều thao tác hoặc chức năng được áp dụng tương phản tới dữ liệu trước khi dữ liệu được chuyển đến đến đích đến. Dữ liệu tại nguồn thì không thay đổi. Biến đổi dữ liệu thực hiện một cách dễ dàng để thực thi tinh chế dữ liệu, chuyển đổi và xác nhận tính hợp lệ dữ liệu phức tạp trong suốt tiến trình import và export.

9.1.3 Các công cụ chuyển đổi dữ liệu (Data transfer tools)

SQL Server 2000 cung cấp số công cụ dành cho việc import và export dữ liệu. Các công cụ này có những khả năng khác nhau để trích lọc tập các dữ liệu từ nguồn dữ liệu có sẵn và chuyển đổi dữ liệu. Bảng dưới đây mô tả ngắn gọn các công cụ chính và khả năng của nó.

Các công cụ chuyển đổi dữ liệu và chức năng của chúng

Công cụ	Mô tả
DTS	DTS là một công cụ đồ họa dùng để import, export, và transform dữ liệu. DTS có thể làm việc trực tiếp các nguồn dữ liệu đa dạng. DTS tạo các gói (package) mà có thể lập biểu. DTS cũng có thể import và export các lược đồ đối tượng CSDL giữa các thể hiện (instance) của SQL Server.
Bcp	Bcp là một lệnh tiện ích tại dấu nhắc được dùng để sao chép dữ liệu từ một tập tin văn bản thành một bảng hoặc View của SQL Server thông qua ODBC. Khả năng biến đổi dữ liệu của Bcp bị giới hạn và qui định dạng dạng tập tin khó hiểu. Làm việc với CSDL của MicroSoft hoặc hảng thứ 3 là một tiến trình 2 bước.
Lệnh BULK INSERT trong Transact-SQL	BULK INSERT là một lệnh Transact-SQL dùng để sao chép dữ liệu từ một tập tin văn bản ASCII thành một một bảng hoặc View của SQL Servers thông qua OLEDB. Câu lệnh BULK INSERT cung cấp chức năng tương tự như Bcp (và cũng hạn chế) trong một câu lệnh Transact-SQL và có thể nhúng trong một gói DTS.

9.2 Dịch vụ chuyển đổi dữ liệu DTS (Data Transformation Services - DTS)

DTS là một tập các công cụ mạnh mà bạn có thể dùng để import, export, và transform dữ liệu đến và đi từ một nguồn và đích dữ liệu đa dạng.

9.2.1 DTS Package.

Một DTS package là một một tập hợp có tổ chức của các DTS Connection, DTS Task, DTS Package Workflow, DTS transformation.

Mỗi package gồm có một hoặc nhiều bước mà được thực thi một cách tuần tự hoặc song song khi mà package được chạy.

9.2.2 DTS Connections.

DTS Connection là các kết nối đến dữ liệu nguồn hay đích đến.

***Data source connection*:** Là một kết nối đến một CSDL chuẩn (như là SQL Server, Access, dBase,…), một kết nối OLE DB đến một nguồn dữ liệu ODBC,..

***File connection*:** Là một kết nối đến một tập tin văn bản

Data *link connection*: Một kết nối đến một tập tin trung gian mà nó lưu trữ một chuỗi kết nối để tạo một kết nối OLE DB mà được thực hiện tại thời điểm chạy.

9.2.3 DTS Tasks.

Là một tập các chức năng rời rạc, được thực thi như là một bước đơn trong một package. Mỗi tác vụ (task) định rõ một mục công việc là:

- Thực thi một câu lệng T-SQL
- Thực thi một script
- Khởi động một ứng dụng mở rộng.

- Sao chép các đối tượng SQL Server.
- Thực thi hoặc lấy các kết quả từ một DTS package.

Các tác vụ có thể trong DTS Designer

Loại	Tác vụ	Mô tả
Các tác vụ mà sao chép và quản lý dữ liệu và biến đổi dữ liệu	**Bulk Insert Task**	Dùng để chạy câu lệnh T-SQL BULK INSERT từ trong một DTS package. Tác vụ này hỗ trợ phương cách nhanh nhất để sao chép thông tin vào một bảng or view, nhưng nó không ghi nhận lại (log) các dòng gây lỗi. Nếu bạn cần giữ lại các dòng gây lỗi vào một tập tin, bạn nên sử dụng tác vụ Transform Data task để thay thế.
	Execute SQL task	Dùng để chạy các câu lệnh Transact-SQL trong suốt việc thực thi package. Bạn có thể thực hiện một số thao tác bao gồm việc xóa một bảng và chạy một thủ tục bằng Execute SQL task.
	Copy SQL Server Objects task	Dùng để sao chép các đối tượng của CSDL của SQL Server (siêu dữ liệu - meta data) từ một thể hiện của SQL Server đến thể hiện khác. Tác vụ này có thể chuyển các đối tượng từ một thể hiện của SQL Server 7.0 đến thể hiện khác; từ một thể hiện của SQL Server 7.0 đến SQL Server 2000; hoặc từ thể hiện của SQL Server 2000 đến thể hiện khác của SQL Server 2000.
	Transfer Database Objects tasks	Một tập hợp các task mà chép thông tin server-wide (Copy SQL Server Objects task chỉ sao chép thông tin đặc biệt của CSDL) từ một thể hiện của SQL Server đến thể hiện khác. Những tác vụ này bao gồm Transfer Database task, Transfer Error Messages task, Transfer Logins task, Transfer Jobs task, và Transfer Master Stored Procedures task. Các tác vụ này được sử dụng bởi Copy Database Wizard.
Các tác vụ mà transform data	**Transform Data task**	Sao chép, biến đổi và chèn dữ liệu từ một data source đến một data destination. Tác vụ này thì hầu hết thực hiện cơ bản của động cơ bom dữ liệu (data pump engine) trong DTS.
	Data Driven Query task	Chọn, tùy chọn, và thực thi một của vài thao tác Transact-SQL (như là update hoặc delete) trên một dòng dựa trên dữ liệu trong dòng. Sử dụng tác vụ này nếu Transform Data task và Bulk Insert task không phù hợp với các yêu cầu của ứng dụng của bạn.
Các tác vụ mà chức năng như là các công việc	**ActiveX Script task**	Để chạy một ActiveX script. Bạn có thể dùng tác vụ này để viết code để thực hiện các chức năng mà không có sẵn trong DTS Designer.
	Dynamic Properties task	Truy xuất dữ liệu trừ một nguồn bên ngoài và gán giá trị truy xuất được cho các thuộc tính package được chọn. External sources có thể là một tập tin .INI, tập tin

Loại	Tác vụ	Mô tả
		dữ liệu, truy vấn, biến toàn cục, biến môi trường, hoặc một hằng số.
	Execute Package task	Dùng chạy DTS package khác như là một phần của workflow. Không dùng tác vụ này một cách đệ quy bởi vì nó có thể sinh ra tràn stack, mà đều này có thể dẫn đến MMC bị shut down.
	Execute Process task	Dùng để chạy một chương trình hoặc tập tin bó lệnh có khả năng thực thi. Tác vụ này có thể được dùng để mở bất kỳ application chuẩn nào đó, như là Microsoft Excel, nhưng nó được sử dụng chính là để chạy các tập tin bó lệnh hoặc các ứng dụng thương mại mà nó làm việc với một data source.
	File Transfer	Dùng để Download dữ liệu từ một từ một server từ xa bởi Protocol task hoặc một Internet location sử dụng FTP. FTP task và Ftp.exe dùng cùng một phương pháp kết nối.
	Send Mail task	Dùng để gửi một thông báo e-mail message như là một tác vụ. Ví dụ, thông báo có thể được gửi đến nhà quản trị về sự thành công hay thất bại của một thao tác dự phòng (backup). Để sử dụng tác vụ này, bạn cần cài đặt một MAPI client ở thể hiện của SQL Server mà bạn đang chạy.

9.2.4 DTS Package Workflow.

Workflow là lưu đồ làm việc của DTS Package. Bạn có thể định rõ thứ tự thực thi của các bước trong một package với ràng buộc độ ưu tiên.

- **Unconditional:** Nếu Task 2 được kết nối với task 1 bởi một ràng buộc unconditional thì Task 2 sẽ phải đợi cho đến khi Task 1 hoàn tất và sau đó nó sẽ được thực thi, bất chấp sự thành công hay thất bại của Task 1
- **On Success**: Nếu Task 3 được kết nối với Task 2 bởi ràng buộc On Success thì Task 2 sẽ đợi cho đến khi Task 2 hoàn tất, và sau đó sẽ thực thi nếu Task 2 hoàn tất một cách thành công.
- **On Failure** Nếu Task 4 liên kết với Task 2 bởi một ràng buộc On Failure, Task 4 sẽ đợi cho đến khi Task 2 hoàn tất và sau đó sẽ chỉ thực thi nếu Task 2 failed thì hoàn tất một cách thành công.

Các Task không có các ràng buộc độ ưu tiên thực thi song song

9.2.5 DTS Package Storage.

Bạn có thể lưu một DTS package vào SQL Server 2000, SQL Server 2000 Meta Data Services, một tập tin Microsoft Visual Basic, hay một tập tin lưu trữ có cấu trúc. Khi bạn lưu một DTS package thì tất cả DTS connections, tasks, transformations, và workflow đều được lưu lại.

9.3 Thực hiện việc biến đổi và chuyển đổi dữ liệu bằng công cụ đồ họa DTS.

9.3.1 DTS Import/Export Wizard

Là một cách đơn giản để tạo một DTS packages để sao chép dữ liệu giữa các nguồn dữ liệu nhưng nó bị giới hạn đối với những biến đổi dữ liệu phức tạp, thêm nhiều task và các lưu đồ tác phụ phức tạp. DTS Import/Export Wizard được sẵn sàng trong Enterprise Manager.

9.3.2 DTS Designer

Cho phép bạn tạo mới hoặc hiệu chỉnh package bằng các đối tượng đồ họa đề hỗ trợ xây dựng DTS package kể cả các lưu đồ phức tạp. DTS Designer sẵn dùng trong Data Transformation Services chứa trong Enterprise Manager.

Các bước tạo một package

- Mở mới một package.
- Khai báo các connection.
- Tạo các tác vụ (task) cần thực hiện.
- Qui định các Wokflow giữa các tác vụ nếu cần.
- Lưu, thực hiện package nếu cần.

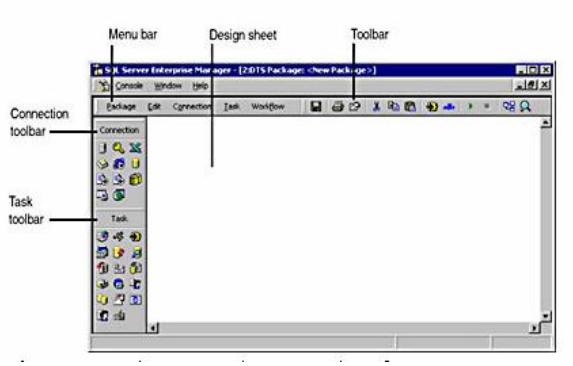
9.3.2.1 Thao tác mở một Package

Mở mới một Package: Trong **Enterprise Manager**, nhấn nút phải chuột tại **Data Transformation Services** và chọn **New Package.**

Mở một Package có sẵn: Khi mở (open) một package có sẵn trong Data Transformation Services tuỳ thuộc vào cách mà DTS pakage đó được lưu trữ như thế nào.

- Package lưu trong tập tin có cấu trúc:
  - R_Click tại Data Transformation Services
  - Chọn Open Package để mở tập tin từ đĩa.
- Package lưu trong SQL Server.
  - Click tại hộp Local Packages trong Data Transformation Service.
  - Double-click vào tên của DTS package cần mở.
- Package lưu trong Meta data Services Package.
  - Click tại nhánh Meta data Services Package trong Data Transformation Service.
  - Double-click vào tên của DTS package cần mở.

Trong cửa sổ DTS Designer cho phép bạn tạo các kết nối đồ họa đến nguồn và đích đến dữ liệu. Cấu hình các DTS tasks, thực hiện các DTS transformation, và chỉ định các ràng buộc ưu tiên. Bạn dùng phương pháp drag và drop và bạn tất các hộp thoại cho các đối tượng để tạo các DTS package trong trang thiết kế. Hình bên dưới hiển thị giao diện người dùng ứng với DTS Designer



Hình 27: Giao diện thiết kế DTS

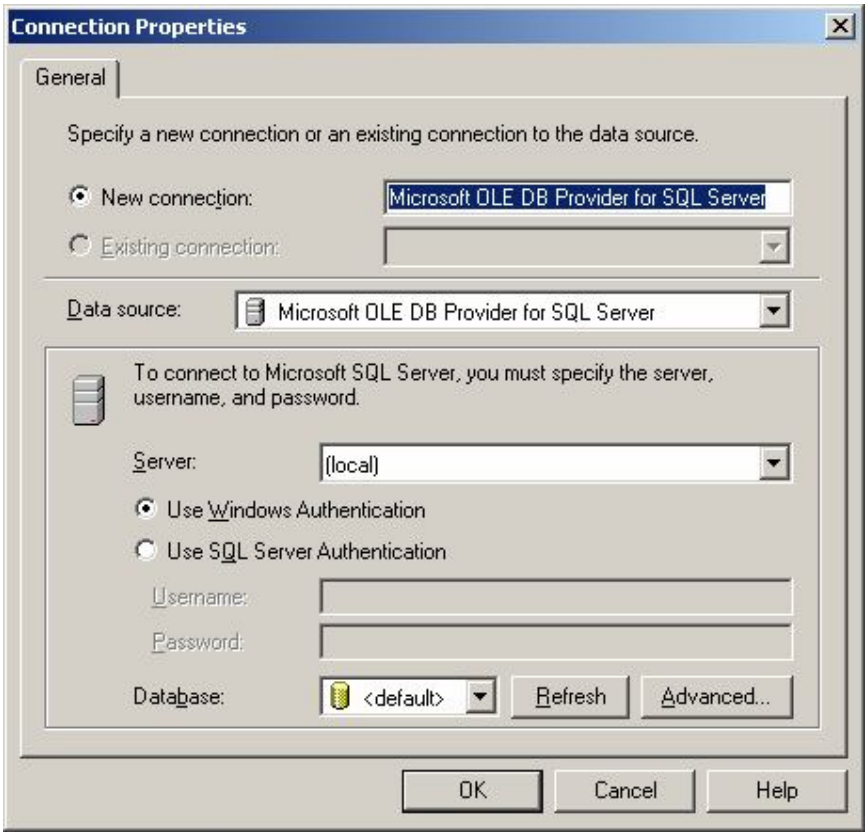
9.3.2.2 Khai báo connection

Bước 1: Chọn nguồn dữ liệu (Data Source)

- Drag một đối tượng data source từ thanh công cụ Connection vào cửa sổ thiết kế hoặc chọn Connection → loại data source

  → Hộp thoại Connection Properties xuất hiện khác nhau tùy vào sự chọn lựa data source.

- Hoàn tất hộp thoại để chỉ định data source.

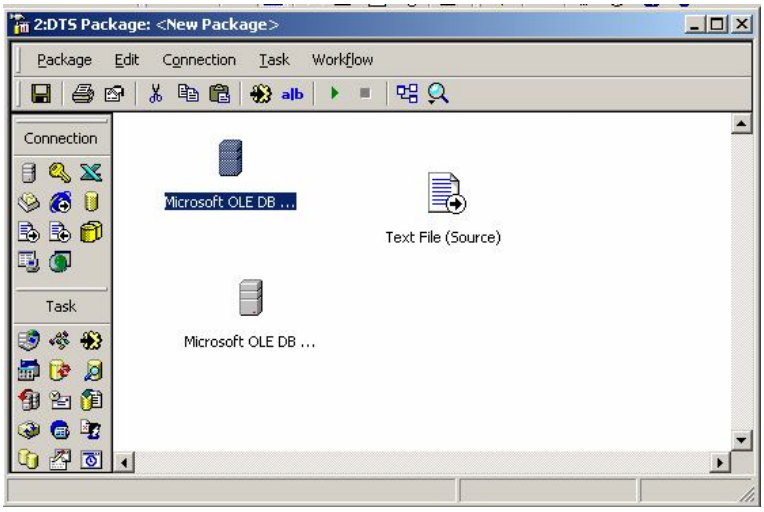


Hình 28: Kết nối đến SQL Server bằng Microsoft OLE DB Provider for SQL Server

Bước 2: Chọn lựa và cấu hình một data destination.

Tương tự như cấu hình một Data Source

Hình 3 hiển thị một bảng thiết kế gồm 3 data source: 2 kết nối đến Microsoft OLE DB Provider for SQL Server và một kế nối đến một tập tin văn bản (Source).



Hình 29: Cài đặt một data Destination.

Lưu ý:

Một Connection đến tập tin văn bản thì phải xác định rõ là một **Text file (Source)** hoặc **Text file (Destination)**.

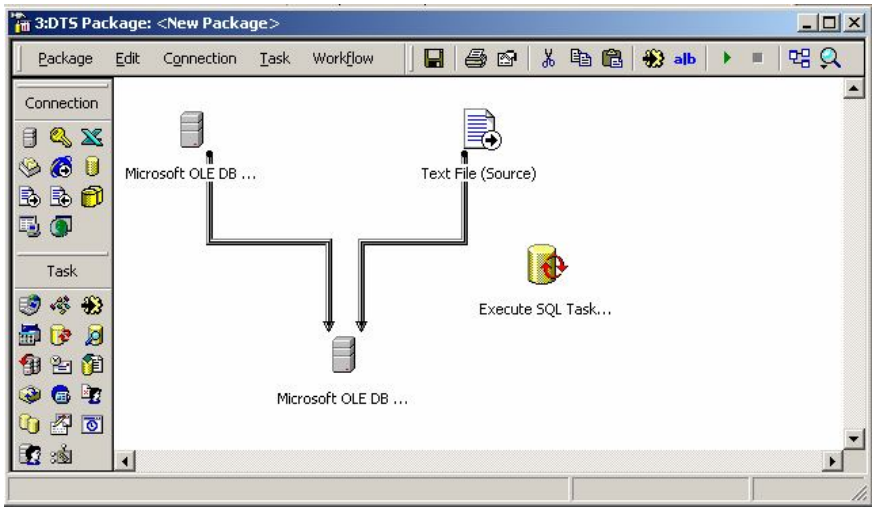
9.3.2.3 Định nghĩa các tác vụ (Task) mà bạn muốn thực hiện

Sử dụng hoặc thực đơn Task hoặc thanh công cụ Task.

Lưu ý:

- Nếu bạn chọn tác vụ Transform dữ liệu, bạn được nhắc nhở xác định data source và data destination. Khi đó một mũi tên màu xám xuất hiện trỏ từ data source đến data destination.
- Nếu bạn chọn bất kỳ tác vụ khác thì một hộp thoại sẽ xuất hiện để nhắc nhở bạn cấu hình thuộc tính của tác vụ. Tác vụ xuất hiện trong cửa sổ thiết kế như là một biểu tượng.

Hình bên dưới hiển thị 3 tác vụ Transform Data và một tác vụ SQL thực hiện (Execute) tạo một bảng.

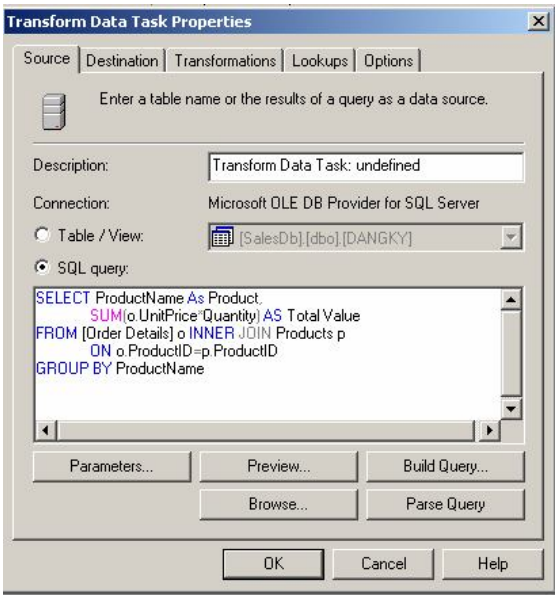


Hình 30: Một DTS với 3 tác vụ transform data và một tác vụ Execute SQL

9.3.2.4 Hiệu chỉnh tác vụ.

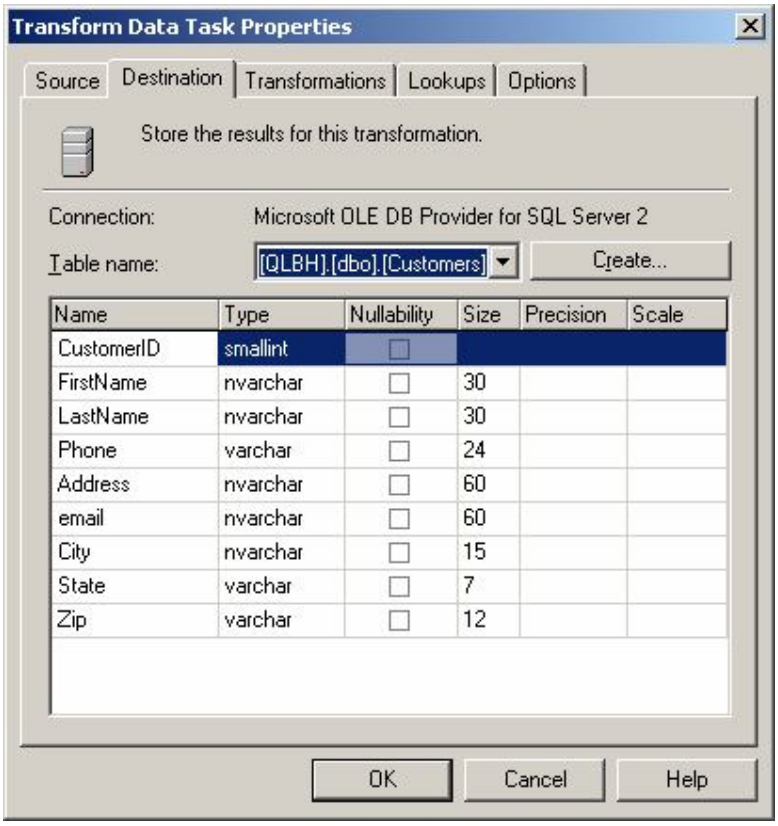
Để hiệu chỉnh và tùy chọn một tác vụ transform dữ liệu, double-click vào mũi tên xám giữa data source và data destination để mở hộp thoại cho tác vụ này.

Trang Source: nếu data source là một CSDL, bạn có thể lọc các dữ liệu được sao chép từ các bảng hoặc View cụ thể được chọn hoặc sử dụng một truy vấn Transact-SQL. Hình 5 hiển thị một truy vấn Transact-SQL được dùng để lọc dữ liệu được import.



Hình 31: Sử dụng một truy vấn để lọc dữ liệu import.

Trang Destination: bạn có thể xác định thông tin về dữ liệu được import (như là các cột đích đến). Những sự lựa chọn của bạn sẽ biến đổi tùy vào data destination. Nếu data destination là một CSDL, bạn có thể tạo và định nghĩa một bảng mới hoặc chọn một bảng có sẵn để import dữ liệu vào.



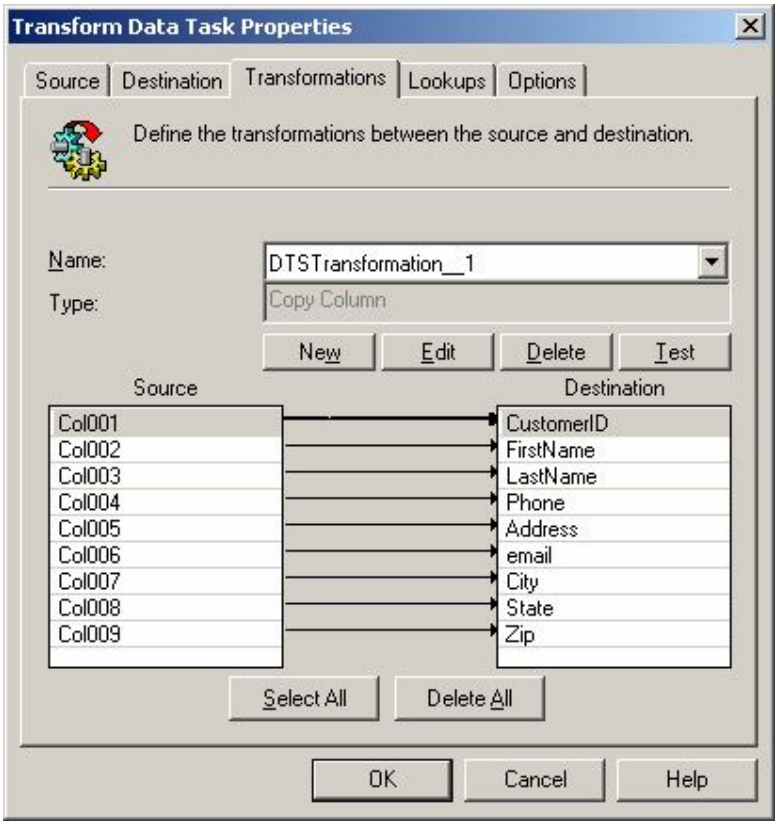
Hình 32: Trang Destination của hộp thoại thuộc tính của tác vụ transform data

Trang Transformations: bạn có thể thiết lập sự biến đổi dữ liệu theo ý người dùng. Theo mặc nhiên, các cột nguồn sẽ được sao chép đến các cột đích mà không cần hiệu chỉnh.

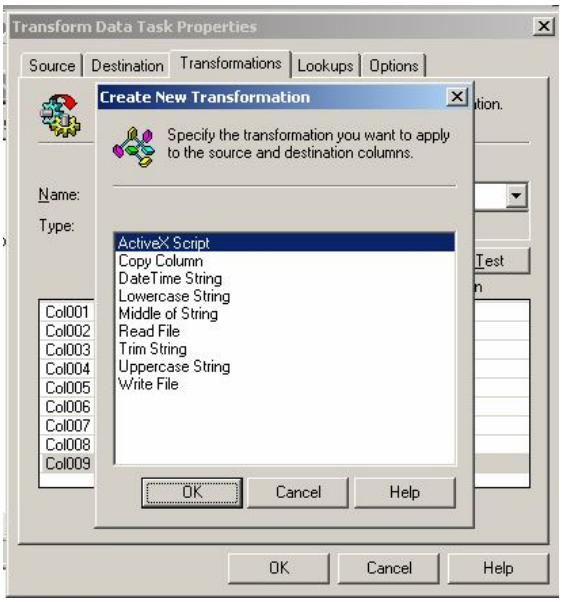
Tạo mới hoặc hiệu chỉnh sự chuyển dữ liệu giữa data source và data destination:

- Chọn cột mà bạn muốn có sự hiệu chỉnh trong danh sách tên đổ xuống hoặc bằng cách click vào mũi tên giữa source và destination.
- Click vào nút **New** hoặc **Edit** để tạo một transformation mới hoặc hiệu chỉnh một transform có sẵn. (double-click vào mũi tên màu đen để hiệu chỉnh một transforrm).

Nếu bạn click vào nút New, bạn có thể chọn kiểu transform mà bạn muốn từ danh sách các transform sẵn sàng trong việc tạo mới một transform.



Hình 33: Trang Tranformations của hộp thoại thuộc tính của tác vụ transform data



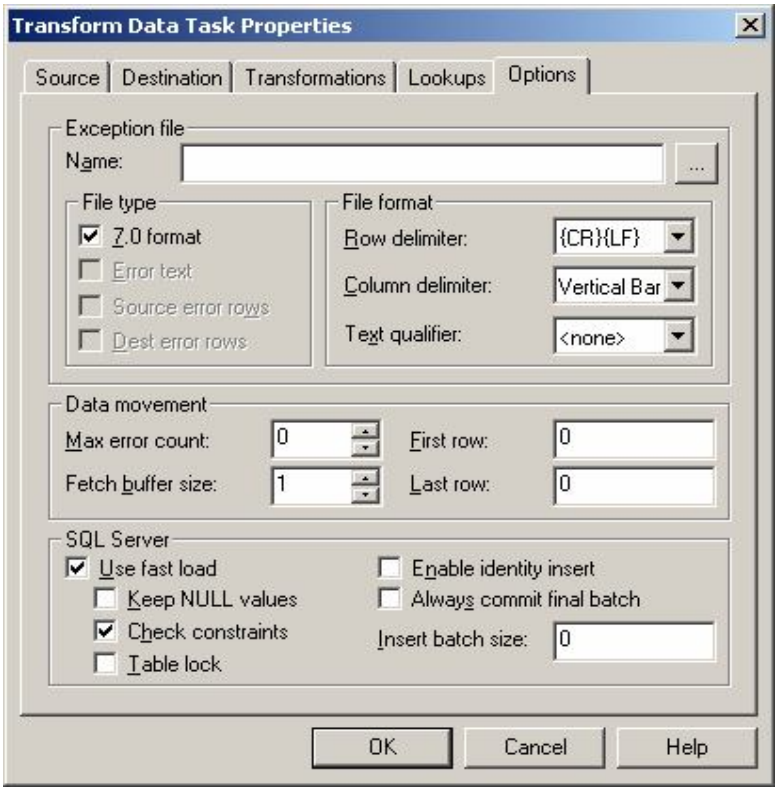
Hình 34:Tạo một transform mới

Nếu bạn chọn ActiveX Script từ hộp thoại Create New Transformation, bạn có thể tạo một transform script mới để thực thi các transform phức tạp hơn xem hình 10

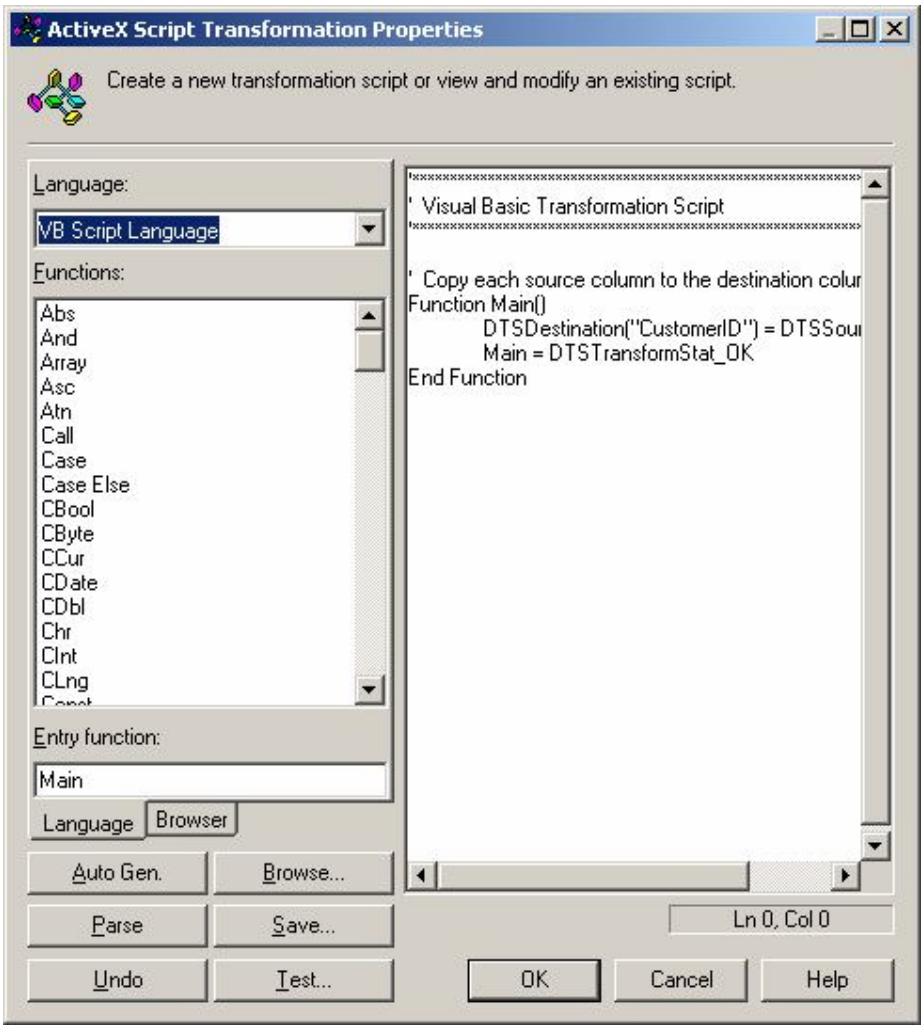
Trang Lookups: bạn có thể định nghĩa một truy vấn dò tìm. Một truy vấn dò tìm đòi hỏi một data connection để chạy các truy vấn và các thủ tục (stored procedure) tương ứng với một data source và data destination. Sử dụng một truy vấn dò tìm để tìm ra thông tin dạng

danh sách, thực thi các cập nhật song song trên hệ thống nhiều CSDL, xác nhận tính hợp lệ nhập vào trước khi tải dữ liệu, gọi các thủ tục trong sự trả lời đến điều kiện nhập dữ liệu và sử dụng các giá trị biến toàn cục như là các tham số của truy vấn.

Trang Options: bạn định nghĩa một số thuộc tính thêm vào cho transform. Bạn có thể định nghĩa một hoặc nhiều tập tin ngoại lệ (exception file) dùng để ghi nhận lại các exception record trong suốt quá trình thực thi package. Tập tin này có thể ở trên ổ đĩa cục bộ hoặc mạng. Bạn có thể tách các lỗi của source và destination trong các tập tin riêng lẽ. Bạn cũng có thể định số lỗi tối đa cho phép trước khi việc thực thi package bị dừng. Cuối cùng, bạn cũng có thể xác định các thuộc tính thực thi đặc biệt khi destination connection là Microsoft OLE DB Provider for SQL Server. Các thuộc tính này bao gồm việc chỉ định thực thi bulk-copy với tốc độ cao, kiểm tra ràng buộc trong suốt quá trình thực thi package, kiểu khóa, kích cở của khối, và nhận dạng các thuộc tính chèn….



Hình 35: Trang khai báo một số option



Hình 36: Tạo một transformation script mới

Một khi bạn có cài đặt một tác vụ Data Transformation task và bất kỳ các tác vụ DTS khác thì DTS package sẽ thực thi, bạn phải cấu hình các ràng buộc quyền ưu tiên. Trong ví dụ, chúng ta có 2 data source riêng lẻ để chép đến data destination. Chúng ta cũng có một tác vụ tạo bảng (Create Table task). Bạn dùng ràng buộc quyền ưu tiên để xác định thứ tự của việc thực thi mỗi tác vụ. Để thiết lập độ ưu tiên của khối công việc, hãy chọn 2 hay nhiều bảng theo thứ tự tác vụ sẽ được thực thi và sau đó chọn loại workflow từ thực đơn Workflow. Ví dụ: Nếu tác vụ Create Table phải thực thi trước khi sao chép dữ liệu đến data destination, Hãy chọn ràng buộc ưu tiên là On Success từ thực đơn Workflow. Xem hình.

Bạn có thể tạo một tác vụ Send Mail và cấu hình ràng buộc ưu tiên On Failure precedence giữa tác vụ tạo bảng (Create Table) và tác vụ gửi mail (Send Mail). Điều này có nghĩa là sẽ gửi một thông báo e-mail đến nhà quản trị nếu tác vụ Create Table bị faile. Khi bạn đang dùng tự đồng hoàn toàn hay các DTS packages được lập lịch để thực thi các quá trình hoạt động của CSDL thì thông báo lỗi là cần thiết

9.4 Dùng BULK COPY (BCP) và BULK INSERT

Bulk copy là một giao diện chương trình ứng dụng (Application Programming Interface – API) được xây dựng trong SQL Server mà cho phép xuất nhập dữ liệu một các hiệu quả. Ta dùng BCP trong trường hợp trong một lần ta muốn chuyển một số lượng lớn dữ liệu từ

các hệ quản trị CSDL khác vào trong SQL Server. BCP là một phương pháp cơ bản chuyển đổi dữ liệu vào ra SQL Server.

BCP là một API được định nghĩa bởi SQL Server 2000. Hiện nay, giao diện này chỉ được đưa ra thông qua ODBC (Open Database Connectivity- Hệ thống kết nối CSDL mở) và DB-LIB (DataBase Library). SQL Server cung cấp một tiện ích (BCP.EXE) mà hầu hết người dùng quen nghĩ là BCP. Ta có 2 phương pháp để xuất nhập dữ liệu là CBP command-line utility, và BULK INSERT Transact-SQL statement.

Dạng dữ liệu

Dạng dữ liệu	bcp Switch	Lệnh Bulk Insert	Ý nghĩa
Native	-n	DataFiletype='native'	Kiểu dữ liệu là tự nhiên, khi cập nhật phải tốn nhiều thời gian chuyển đổi kiểu dữ liệu
Character	-c	DataFiletype='char'	Kiểu dữ liệu là CHAR cho tất cả các cột
Unicode character	-w	DataFiletype='widechar'	Kiểu dữ liệu là Unicode char cho tất cả các dữ liệu, có thể một số dữ liệu sẽ bị mất nếu kiểu dữ liệu của Source và Destination không trùng nhau.
Unicode Native	-N	DataFiletype='widenative'	Kiểu dữ liệu là Native cho những dữ liệu không là kiễu Char, và dùng Unicode cho nhữ dữ liệu kiểu char (char, Varchar, nchar, text…)

BCP Command-line Utily: là một hàm đơn giản của Bulk Copy, thực thi tại dấu nhắc lệnh, dùng để sau chép dữ liệu vào SQL Server hoặc xuất dữ liệu ra thành một tập tin dữ liệu. Muốn dùng được bcp, thì trước tiên dữ liệu trong chương trình nguồn (một DBMS khác) phải được sang một tập tin dữ liệu vào sau đó mới chuyển dữ liệu từ tập tin data vào table cùa SQL. Ngược lại, bcp sẽ chuyển dữ liệu từ một table thành một tập tin dành cho chương trình khác chẳng hạn như Excel.

Data file: là một tập tin hệ thống hoạt động, dùng chứa dữ liệu để Import vào SQL Server (gọi là Source) hoặc tập tin chứa dữ liệu khi dùng Export (gọi Destination) .

bcp [*database_name*.][*owner*.]*table_name* in | out *data_file* [-c | -n | w | N] [-T]

Ví dụ: bcp Northwind..customers out d:\data\kh.txt -c –T

-T: chỉ định là Connect trực tiếp với Server đang log on

```
SELECT * INTO Cust FROM Customers
TRUNCATE TABLE Cust
bcp Northwind..cust IN d:\data\kh.txt -c –T
```

CÂU LỆNH BULK INSERT: dùng để chuyển một dữ liệu vào bảng của SQL Server từ một tập tin dữ liệu khác. Câu lệnh Bulk insert không thể đưa dữ liệu từ bảng của SQL Server ra thành tập tin dữ liệu dành cho chương trình khác. Tuy vậy, nó lại là một câu lệnh trong T-SQL nên có thể được dùng một cách thuận tiên trong các hàm hoặc thủ tục trong SQL Sever.

BULK INSERT '*database_name*' ['*owner*' .] '*table_name*' FROM '*data_file*'
[WITH (DATAFILETYPE = 'char' | 'native'| 'widechar' | 'widenative')]

```
BULK INSERT dbo.Employees FROM 'C:\DATA\Employees.TXT'
     WITH (DATAFILETYPE ='widechar')
BULK INSERT dbo.Customers FROM 'C:\DATA\Customers.TXT'
     WITH (DATAFILETYPE ='widechar')
```

Lưu ý:

- Số cột hoặc số thứ tự của các cột của file text và của bảng không tương thích thì không chèn dữ liệu vào được. Cách giải quyết?
- Nếu trong file Text có giá trị của cột Identity thì sẽ chấp nhận giá trị đó, nếu không có thì nó tự phát sinh.
- Khi Export vào một tập tin, tập tin sẽ bị ghi dè nếu như tập tin đó hiện tồn tại.
- Khi BCP đang import dữ liệu vào bảng, người dùng khác vẫn có thể đang dùng bảng đó. Những dòng được Import chỉ hiện hữu khi tập lệnh import kết thúc.
- Khi BCP đang import dữ liệu, người dùng khác vẫn truy xuất dữ liệu bình thường.
- Khi đang import dữ liệu vào bảng, có 2 phương pháp có thể sử dụng: Logged và no-logged BCP.

BÀI 10:
CƠ BẢN VỀ LẬP TRÌNH BẰNG TRANSACT- SQL

10.1 Khái niệm cơ bản.

10.1.1 Định danh -IDENTIFIERS.

Tên của tất cả các đối tượng đều được gọi là định danh. Mọi thứ trong SQL Server đều có một định danh, trong đó bao gồm Servers, Databases, và các đối tượng (Object) của CSDL như bảng, Views, cột, chỉ mục, ràng buộc,…. Có những đối tượng bắt buộc phải qui định một định danh, ngược lại có một số đối tượng không cần định danh (SQL Server sẽ tự động định danh)

Ví dụ:

```
CREATE TABLE Table1
(Keycol INT PRIMARY KEY, Description NVARCHAR(30))
```

Qui tắc định danh:

- Tối đa 128 ký tự.
- Bắt đầu là một ký tự thông thường A -> Z
- Bặt đầu là một ký hiệu (@, #) sẽ có một ý nghĩa khác.
- Những định danh nào có khoảng trắng ở giữa thì phải kẹp trong dấu [ ] hoặc " "
- Đặt các định danh sao cho ngắn gọn, đầu đủ ý nghĩa, phân biệt giữa các đối tượng với nhau, không trùng lắp, không trùng với từ khóa của T-SQL.

10.1.2 Tham chiếu đến các đối tượng trong SQL Server.

- Tên đầy đủ: **Server.database.ower.object**
- Tên ngắn: Nếu là local server thì ta khỏi chỉ Server, nếu ở CSDL hiện hành thì không cần chỉ Database, Owner mặc định là user name trong Database.

Nếu Tham chiếu tường minh thì tăng tốc.

Ví dụ:

```
CREATE TABLE Northwind.dbo.orederhist
CREATE TABLE Northwind..orederhist
```

Database Owner (dbo)

Dbo là một người dùng mà có đầy đủ các quyền thao tác trong CSDL. Bất kỳ một thành viên thuộc nhóm **sysadmin** đóng vai trò người dùng CSDL được gọi là **dbo**. Cũng như bất kỳ một đối tượng nào được tạo bởi bất kỳ thành viên thuộc nhóm **sysadmin** thì mặc nhiên thuộc **dbo.**

Ví dụ: nếu người dùng **Andrew** là thành viên của nhóm **sysadmin** tạo bảng **T1**, thì **T1** thuộc chủ quyền **dbo** và nó thuộc dbo (**dbo.T1)**, không thuộc **Andrew (**không là **Andrew.T1**). Ngược lại, nếu **Andrew** không là thành viên của **sysadmin** nhưng chỉ là một thành viên của **db_owner** và tạo bảng **T1**, **T1** thuộc về **Andrew** và đủ tư cách gọi là Andrew**.T1**.

User **dbo** không bao giờ bị xóa và nó luôn luôn hiện hữu trong mỗi CSDL. Chỉ có những đối tượng được tạo bởi thành viên của sysadmin (hoặc bởi user dbo) thì thuộc về dbo.

10.1.3 Kiểu dữ liệu (DATA TYPE).

Kiểu dữ liệu là một định nghĩa để xác định loại dữ liệu mà đối tượng có thể chứa đựng. Cột, tham số, biến, giá trị trả về của hàm, thủ tục, tất cả đều phải có kiểu dữ liệu.

Có 2 loại kiểu dữ liệu: **System-Supplied datatype** và **User-defined data types**

(Đã được trình bày trong bài trước)

10.1.4 Batch

Batch là một tập các phát biểu T-SQL nằm liên tiếp và kết thúc bởi phát biểu GO, và được biên dịch đồng thời bởi SQL Server.

Ví dụ:

```
USE pubs
GO /* Signals the end of the batch */

CREATE VIEW auth_titles
AS
SELECT *
FROM authors
GO /* Signals the end of the batch */

SELECT *
FROM auth_titles
GO /* Signals the end of the batch */
```

Lưu ý:

- Các phát biểu trong 1 batch được biên dịch thành một nhóm.
- Nếu một trong phát biểu của batch bị lỗi thì batch cũng xem như lỗi.
- Các phát biểu Create bị ràng buộc trong một batch đơn, tức trong batch đó chỉ có phát biểu Create. Các phát biểu đó là: Create DataBase, Create Table, Create Index,…

10.1.5 Kịch bản - SCRIPT

Một Script là một tập của một hay nhiều bó lệnh được lưu lại thành một tập tin .SQL

10.2 Biến (VARIABLES)

Biến là một đối tượng trong tập lệnh T-SQL mà nó dùng để lưu trữ dữ liệu. Sau khi biến đã được khai báo hoặc định nghĩa, một câu lệnh trong tập lệnh sẽ gán giá trị cho biến và

cũng có thể một câu lệnh khác sẽ lấy giá trị của biến ra dùng. Phải được khai báo trước khi dùng.

Biến được dùng để:

- Đếm số lần lặp được thực hiện hoặc dùng để điều khiển vòng lặp
- Dùng lưu giá trị dữ liệu được kiểm tra một số lệnh điều khiển
- Lưu trữ giá trị trả về từ một store Procedure
- ……

Các loại biến: có 2 loại biến là biến cục bộ (local) và biến toàn cục (Global). Biến kiểu Global được SQL Server đưa ra và bạn có thể dùng bất cứ khi nào và cũng không cần khai báo (được xem như là những hàm chuẩn của SQl Server).

Local variable

- Được khai báo trong phần thân của một bó lệnh hoặc một thủ tục.
- Phạm vi hoạt động của biến bắt đầu từ điểm mà nó được khai báo cho đến khi kết thúc một lô (batch) hoặc stored procedure hoặc Function mà nó được khai báo.
- Tên của biến bắt đầu @

Khai báo:

```
DECLARE @var_name var_type
```

Gán giá trị cho biến:

Nếu biến vừa khai báo xong thì biến mặc nhiên được gán giá trị là NULL.

Để gán biến ta dùng lệnh SET hoặc dùng câu lệnh SELECT

```
SET @var_name = expression
SELECT { @var_name = expression } [,...n ]
```

Ví dụ 1:

```
USE Northwind
DECLARE @EmpIDVar INT
SET @EmpIDVar=3
SELECT * FROM [Orders]
WHERE
```

Ví dụ 2:

```
DECLARE MyVariable INT
SET @MyVariable = 1
GO – điểm kết thúc một lô batch.
-- @MyVariable đã vượt quá phạm vi và nó đã hết tồn tại
-- Câu lệnh SELECT sau sẽ nhận lỗi sai cú pháp bởi vì nó không tham chiếu được biến @MyVariable.
SELECT * FROM [Orders]
WHERE
```

Ví dụ 3:

```
USE Northwind
```

```
GO
-- Khai báo 2 biến.
DECLARE @FirstNameVariable NVARCHAR(20),
@RegionVariable NVARCHAR(30)

-- Gán giá trị cho 2 biến.
SET @FirstNameVariable = N'Anne'
SET @RegionVariable = N'WA'

-- Dùng chúng trong mệnh đề WHERE của lệnh SELECT.
SELECT LastName, FirstName, Title
FROM Employees
WHERE FirstName = @FirstNameVariable
   OR Region = @RegionVariable
GO
```

Ví dụ 4:

```
USE Northwind
GO
-- Khai báo 1 biến
DECLARE @EmpIDVariable INT
-- Gán giá trị biến bằng câu lệnh Select
SELECT @EmpIDVariable = MAX(EmployeeID)
FROM Employees
GO
```

-- Nếu câu Select trả về là một tập giá trị thì biến sẽ nhận giá trị sau cùng.

Ví dụ 5:

```
DECLARE @ProdIDVariable int
SELECT @ProdIDVariable = ProductID
FROM Northwind..Products
```

Ví dụ 6:

```
USE Northwind
GO
DECLARE @EmpIDVariable INT
SELECT @EmpIDVariable = EmployeeID
FROM Employees
ORDER BY EmployeeID DESC
SELECT @EmpIDVariable
GO
```

Global variable trong SQL Server 2000 gọi là System Function: Từ SQL Server 7.0 biến Global được định nghĩa như là hàm hệ thống. Mỗi kết nối sẽ tạo ra một session, SQL

Server tạo ra sẵn một số biến có sẵn rất tiện ích trong việc lập trình và quản trị hệ thống. Các biến này không có kiểu, tên bắt đầu @@.

Một hàm hệ thống thường dùng:

@@VERSION	phiên bản của SQL Server và hệ điều hành
	`SELECT @@VERSION`
@@TRANCOUNT	Xem coi có bao nhiêu transaction đang mơ

	`IF (@@TRANCOUNT > 0)`
	`BEGIN`
	`RAISERROR('Task cannot be executed within a transaction.', 10, 1)`
	`RETURN`
	`END`

@@ROWCOUNT	Trả về số dòng bị ảnh hưởng đối với lệnh thực thi gần nhất

	Ví dụ 1:
	`USE Northwind`
	`UPDATE Employees SET LastName = 'Brooke'`
	`WHERE LastName = 'Brook'`
	`IF (@@ROWCOUNT = 0)`
	`BEGIN`
	`PRINT 'Warning: No rows were updated'`
	`RETURN`
	`END`
	Ví dụ 2:
	`UPDATE Customers`
	`SET Phone =’030’ + Phone`
	`WHERE Country=’Germany’`
	`PRINT @@ROWCOUNT`

@@IDENTITY	trả về số Identity phát sinh sau cùng

	`CREATE TABLE TABLE_HD`
	`( mahd int Identity Primary Key,`
	`Ghichu varchar(20) )`
	`CREATE TABLE TABLE_CTHD`
	`( Mahd int, Masp char(10), Soluong int)`
	`declare @maso Int`
	`Insert into Table_HD Values ('Mau tin 1')`
	`Insert into Table_HD Values ('Mau tin 2')`
	`set @maso= @@IDENTITY`
	`Insert into Table_CTHD`
	`values (@maso, 'sp001',5)`

	Insert into Table_CTHD values (@maso, 'sp002',10) -- Kiểm tra SELECT * FROM Table_HD SELECT * FROM Table_CTHD
@@ERROR	Trả về lỗi số (STT lỗi) của lệnh sau cùng mà SQL thực thi, là 0 có nghĩa là câu lệnh thực thi hoàn thành.
@@FETCH_STATUS	Trả về trạng thái của lệnh Fetch của biến con trỏ có thành công hay không (0: Thành công, -1: bị lỗi hoặc vượt quá phạm vi; -2: Thất bại)

Một số hàm thường dùng:

GetDate()	Lấy ngày, giờ hiện hành của hệ thống
Month(Date); Year(Date)	Lấy tháng, năm của ngày Date
DateAdd(Datepart, Number, Date)	Cộng thêm Date một giá trị số
DATEDIFF (datepart, startdate, enddate)	Khoảng chênh lệch giữa startdate và enddate
DATEPART (***datepart, date)***	Trả về số nguyên biểu diễn datepart nào đó của ngày được chỉ định
CAST (**expression AS data_type)**	Dùng để chuyển đổi kiểu dữ liệu
CONVERT (***data_type*** **[** (***length)*** **],** ***expression*** **[,** ***style*** **])**	Dùng để chuyển đổi kiểu dữ liệu
LOWER (***character_expression)***	Chuyển sang chữ thường
UPPER (***character_expression)***	Chuyển sang chữ hoa
REPLACE ('**string_expression1**', '**string_expression2**', '**string_expression3**')	Thay thế chuỗi biểu thức
DIFFERENCE (**character_expression, character_expression)**	So sánh 2 biểu thức

Ví dụ:

```
USE pubs
SELECT 'The price is ' + CAST(price AS
varchar(12))
FROM titles
WHERE price > 10.00
GO

-- Use CONVERT.
USE pubs
SELECT SUBSTRING(title, 1, 30) AS Title, ytd_sales
FROM titles
WHERE CONVERT(char(20), ytd_sales) LIKE '3%'
GO
```

10.3 Cấu trúc điều khiển.

T-SQL cung cấp một số cấu trúc điều kiển cơ bản để bạn có thể thực thi một khối lệnh dựa trên kết quả của một phép so sánh. Nó cũng tương tự như một số ngôn ngữ lập trình khác.

10.3.1 Khối BEGIN … END

Nếu bạn cần nhiều phát biểu được thực thi với nhau thì ta đặt các phát biệu trong cặp Begin … End. Nó được hữu dụng trong các cấu trúc điều khiển.

10.3.2 Phát biểu PRINT

Phát biểu PRINT: Dùng để in thông tin ra màn hình kết quả của SQL

PRINT 'any ASCII text' | @local_variable | @@FUNCTION | string_expr

Ví dụ:

```
PRINT ‘Hello!’
PRINT N’Chào bạn’
PRINT @@VERSION
```

10.3.3 Cấu trúc điều khiển IF … ELSE

Là một cấu trúc điều kiển, cho phép thực thi một hoặc nhiều phát biểu tùy thuộc vào một điều kiện nào đó. câu lệnh để thực thi một khối các câu lệnh theo một điều kiện nào đó.

Cú pháp:

IF condition
{statements}
[ELSE [Condition 1]
{statements}]

Condition: là một biểu thức logic, có giá trị *True* hoặc *False.*Tùy thuộc vào *condition,* một trong hai khối lệnh sẽ được thực thi.

Ví dụ 1: Kiểm tra xem trong Customers của NorthWind có chứa các khách hàng đến từ Germany không?

```
USE NorthWind
IF (SELECT COUNT(*) FROM Customers
     WHERE Country='Germany') > 0
     BEGIN
       Print ' Có tồn tại các khách hàng từ
Germany ở trong cơ sở dữ liệu.'
       Print ' statements to process German
customers'
     END
```

```
ELSE
     BEGIN
       PRINT ' Không có khách hàng đến tử Germany
trong cơ sở dữ liệu.'
     END
```

Ví dụ 2:

```
USE pubs
GO
DECLARE @msg varchar(255)
IF ( SELECT COUNT(price) FROM titles
WHERE title_id LIKE 'TC%' AND price BETWEEN 10 AND
20) > 0
BEGIN
   SET NOCOUNT ON
   SET @msg = 'Có vài quyển sách có giá từ $10 đến
$20. Các sách đó là:'
   PRINT @msg
   SELECT title FROM titles
    WHERE title_id LIKE 'TC%' AND price BETWEEN 10
AND 20
END
ELSE
BEGIN
   SET NOCOUNT ON
   SET @msg = 'Không có quyển sách nào có giá từ
$10 đến $20. Bạn nên tham khảo các quyễn sách có
giá nhỏ hơn $10 sau đây.'
   PRINT @msg
   SELECT title FROM titles
   WHERE title_id LIKE 'TC%' AND price < 10
END
```

Ví dụ 3:

```
USE pubs
IF (SELECT AVG(price) FROM titles WHERE type =
'mod_cook') < $15
BEGIN
     PRINT 'The following titles are excellent
mod_cook books:'
     PRINT ' '
     SELECT SUBSTRING(title, 1, 35) AS Title
     FROM titles WHERE type = 'mod_cook'
```

```
END
ELSE
     IF (SELECT AVG(price)
           FROM titles WHERE type = 'mod_cook')>
     $15
     BEGIN
         PRINT 'The following titles are expensive
     mod_cook books:'
         PRINT ' '
         SELECT SUBSTRING(title, 1, 35) AS Title
         FROM titles WHERE type = 'mod_cook'
     END
```

10.3.4 Biểu thức CASE.

Biểu thức CASE là một biểu thức điều kiện được áp dụng bên trong một phát biểu khác. Case trả về các giá trị khác nhau tuỳ thuộc vào điều kiện hoặc một điều khiển nào đó.

Dạng 1:

```
CASE input_expression
  WHEN when_expression THEN result_expression
    [ ...n ]
  [
    ELSE else_result_expression
  ]
END
```

Dạng 2:

```
CASE
     WHEN Boolean_expression THEN result_expression
    [ ...n ]
  [
    ELSE else_result_expression
  ]
END
```

Ví dụ 1: Cho 2 số a và b, so sánh 2 số đó, số nào lơn hơn số nào?

```
DECLARE @a As int, @b As int, @ketqua as nvarchar(30)
SET @a=3
SET @b=5
SET @ketqua = CASE
               WHEN @a<@b THEN N'A nhỏ hơn B'
               -- When chỉ dùng trong case
               WHEN @a<@b THEN N'A lớn hơn B'
              ELSE N'A bằng B'
         END  -- END của CASE
```

```
PRINT @ketqua
```

Ví dụ 2: Dựa vào price của các title cho biết price đó như thế nào?

```
Use Pub
SELECT title, price,
'classification'=CASE
    WHEN price < 10.00 THEN 'Low Priced'
    WHEN price BETWEEN 10.00 AND 20.00 THEN 'Moderately
 Priced'
    WHEN price > 20.00 THEN 'Expensive'
    ELSE 'Unknown'
    END
FROM titles
```

Ví dụ 3: Cho biết ý nghĩa của đoạn lệnh sau:

```
USE NorthWind
SELECT ProductID, Quantity, UnitPrice, [discount%]=
     CASE
          WHEN Quantity <=5 THEN 0.05
          WHEN Quantity BETWEEN 6 and 10 THEN 0.07
          WHEN Quantity BETWEEN 11 and 20 THEN 0.09
     ELSE
     0.1
END
FROM [Order Details]
ORDER BY Quantity, ProductId
```

Ví dụ 4: Ý nghĩa của đoạn lệnh sau?

```
SELECT title, pub_id,
    CASE WHEN price IS NULL THEN (SELECT MIN(price)
                                  FROM titles)
    ELSE price
    END
FROM titles
```

10.3.5 Cấu trúc vòng lập WHILE …

Là phát biểu điều khiển vòng lặp. Vòng lặp sẽ thực hiện cho đến khi biểu thức điều kiện (*Boolean_expression*) trong While mang giá trị False. Biểu thức điều kiện có thể là một câu SELECT.

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

{ sql_statement | statement_block }
[CONTINUE]

[BREAK] : Dùng để kết thúc vòng lặp khi gặp một trường hợp cụ thể nào đó.

[CONTINUE] : Lặp lại đầu vòng lặp.

Thông thường 2 từ khóa Break và Continue phải nằm trong trong cấu trúc If … Else..

Ví dụ 1: Trong khi đơn giá trung bình vẫn còn nhỏ hơn $30 thì cập nhật các đơn giá tăng lên gấp đôi đơn giá cũ.

```
USE pubs
GO
WHILE (SELECT AVG(price) FROM titles) < $30
BEGIN
   UPDATE titles
      SET price = price * 2
   SELECT MAX(price) FROM titles
   IF (SELECT MAX(price) FROM titles) > $50
      BREAK
   ELSE
      CONTINUE
END
PRINT 'Too much for the market to bear'
```

10.3.6 Lệnh RETURN

RETURN [integer_expression]

Nếu gặp phát biểu Return, quá trình xử lý sẽ kết thúc

Đôi khi ta dùng RETURN trong thủ tục để thủ tục có thể trở thành hàm như các ngôn ngữ khác.

10.3.7 Lệnh WAITFOR

Là một chỉ thị cho SQL Server tạm dừng một thời gian trước khi xử lý tiếp các phát biểu sau đó.

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

‘Time’: được viết thao dạng hh:mm:ss, tối đa là 24 giờ

DEPLAY ‘Time’: Hệ thống tạm dừng trong khoảng thời gian ‘Time’.

TIME ‘Time’: Hệ thống được tạm dựng đền thời gian ‘Time’ chỉ ra.

```
WAITFOR   ’02:10’
WAITFOR   ’02:10’
```

10.3.8 Lệnh RAISERROR

Phát sinh lỗi của người dùng. Người dùng có thể phát sinh các lỗi từ bảng sysmessage hoặc xây dựng lỗi động tùy thông tin của người dùng. Sau khi một lỗi được định nghĩa thì nó được gửi đến người dùng như là một lỗi hệ thống.

RAISERROR (**{** ***msg_id*** **|** ***msg_str*** **}** **{,** ***severity, state*** **}**
[, ***argument*** **[,...***n***]])**
[WITH ***option*** **[,...***n***]]**

msg_id: là mã thông báo, nó được lưu trong bảng sysmessage. Mã thông báo của người dùng định nghĩa phải được bắt đầu từ trên 50000

msg_str : Nội dung thông báo, tối đa 400 ký tự.

Để truyền tham số vào trong thông báo thì dùng dạng %<Loai ký tự>

Loại ký tự là d, i, o, x, X, hoặc u

Các ký tự	**Mô tả**
d hoặc I	Biểu hiện là số nguyên (integer)
O	Octal không dấu
P	Con trỏ
S	Chuỗi
U	So nguyên không dấu
x or X	Hexadecimal không dấu

Lưu ý: số **float**, double, char không được hỗ trợ

severity: Độ nghiêm khắt của thông báo

Severity Levels: Mức lỗi của một thông báo lỗi cung cấp một sự biểu thị loại vấn đề mà SQL Server gặp phải.

- Mức lỗi **10** là lỗi về thông tin và biểu thị nguyên nhân do thông tin nhập vào.
- Mức lỗi từ **11 đến 16** thì thông thường là do các user.
- Mức từ **17 đến 25** do lỗi phần mềm hoặc phần cứng. Bạn nên báo cho nhà quản trị hệ thống bất cứ khi nào sự cố xảy ra. Nhà quả trị hệ thống phải giải quyết sự cố đó và theo dõi chúng thường xuyên. Khi mức lỗi 17,18,19 xảy ra, bạn có thể tiếp tục làm việc mặc dù bạn không thể thực thi lệnh đặc biệt.
- **Mức lỗi 17:** Những thông báo này cho biết rằng câu lệnh nguyên nhân SQL Server cạn kiệt tài nguyên (Ví dụ như lock hoặc không gian đĩa cho CSDL) hoặc vượt quá tập giới hạn bởi nhà quản trị
- Người quản trị hệ thống nên giám sát tất cả các sự cố mà được phát ra mức trầm trọng từ **17 đến 25** và in ra giải thích lỗi mà bao gồm các thông tin để quay lại từ lỗi.

Nếu sự cố ảnh hưởng đến toàn bộ một CSDL, bạn dùng DBCC CHECKDB (Database) để xác định phạm vi của sự thiệt hại. DBCC có thể xác định một vài đối tượng mà phải bị di chuyển và sẽ tùy ý phục hồi sự thiệt hại. Nếu thiệt hại là lớn, CSDL có thể không phục hồi được. Trong trường hợp đặc biệt, người dùng định nghĩa thông báo lỗi với RAIERROR, dùng mã lỗi trên 50000 và mức lỗi trầm trọng từ 0 đến 8. Chỉ có nhà quản trị hệ thống có thể phát hành lỗi với mức trầm trọng từ 19 đến 25.

- Mức lỗi từ **20 đến 25** chỉ ra sự cố hệ thống. Đó là lỗi không tránh được, mà có nghĩa là tiến trình không còn đang chạy. Tiến trình tê liệt trước khi nó dừng, ghi nhận thông tin về cái gì xảy ra, và sau đó kết thúc.

state: Là một số nguyên tùy ý từ 1 đến 127 mà nó mô tả thông tin diễn giải về trạng thái lỗi.

Argument: Là tham số dùng trong việc thay thế cho biến để định nghĩa thông báo lỗi hoặc thông báo tương ứng với mã lỗi msg_id. Có thể không hoặc có nhiều tham số. Tuy nhiên, không được quá 20. Mỗi tham số thay thế có thể là một biến local hoặc bất kỳ một trong các kiểu dữ liệu int, char, varchar, binary, varbinary. Các kiểu khác không được cung cấp.

Thêm một lỗi mới của người dùng định nghĩa:

Sp_addMessage msg_ID, *severity*, '*msg*' [, '*language*'] [, '*with_log*'] [, '*replace*']

Xóa một lỗi của người dùng

sp_dropmessage Msg-ID

Giải thích

msg_id: là mã số của lỗi mới, là một số int, không được trùng các mã đã có sẵn, bắt đầu là 50001.

severity: là mức lỗi của lỗi, là một số smallint. Mức hợp lệ là từ 1 đến 25. Chỉ có người quản trị CSDL mới có thể phát sinh thêm một thông báo lỗi mới từ 19 đến 25.

'*msg*': là một chuỗi thông báo lỗi, tối đa 255 ký tự.

'*language*': là ngôn ngữ của thông báo lỗi, không chỉ định tì mặc định là ngôn ngữ của phiên kết nối.

'*with_log*': thông báo lỗi có được gi nhận vào nhật ký của ứng dụng khi nó xảy ra hay không, mặc định là FALSE. Nếu là **true**, thì lỗi luôn luôn được ghi vào nhật ký ứng dụng. Chỉ có những thành viên thuộc **sysadmin** server role mới có thể sử dụng tham số này.

'*replace*': nếu được chỉ định chuỗi **REPLACE**, thì thông báo lỗi đã tồn tại được ghi đè bởi chuỗi thông báo mới và mức lỗi mới. Tham số này phải chỉ định nếu *msg_id* đã có.

Lưu ý: *nếu trả về 0 tức là thêm vào thành công, 1 thất bại.*

BÀI 11:
PROCEDURES, FUNCTIONS

11.1 STORED PROCEDURES.

11.1.1 Giới thiệu Stored proccedures.

Stored procedure (thủ tục) là một tập các lệnh T-SQL và một số cấu trúc điều kiển, được lưu với một tên và được thực thi như là một đơn vị công việc đơn (single unit of work). Trong các ngôn ngữ khác như C, pascal, Basic, một thủ tục thông thường là một tập các câu lệnh với mục đích hoàn tất một mục đích nào đó và có thể được gọi từ một chương trình như là một lệnh đơn.

- Thủ tục trong SQL Server được lưu trữ tại server khi nó được tạo ra. Vì vậy, khi thủ tục được thi hành thì nó được chạy tại Server. Có thể gọi thủ tục chạy bằng một lệnh đơn giản và trong thủ tục có thể chứa rất nhiều lệnh của T-SQL.
- Trước khi thủ tục được tạo, SQL Server sẽ kiểm tra tính đúng đắn của các cú pháp lệnh. Nếu không có lỗi về cú pháp thì thủ tục được tạo, tên của thủ tục được lưu trong bảng hệ thống **SysObjects** và nội dung được lưu trong bảng hệ thống **SysCommanes**.
- Trước khi thủ tục được thực thi, một kế hoạch thực thi (Execution plain) được tạo ra và thủ tục được biên dịch. Từ đó trở về sau tiến trình dịch thủ tục nhanh hơn bởi vì SQL Server sẽ không kiểm tra tính đúng đắng của các câu lệnh nữa, chỉ tạo lại execution plan và biên dịch lại thủ tục.
- Security (an toàn): thủ tục có một đặc tính quan trọng là nó có thể được nâng cao an toàn thông tin thông qua isolation (cô lập) hoặc encryption (mã hóa). Người dùng CSDL có thể được cho quyền thực thi thủ tục nhưng sẽ không có quyền trực tiếp truy xuất các đối tượng của thủ tục. Một thủ tục có thể được mã hóa ngay khi được tạo hoặc chỉnh sửa vì thế người sử dụng không thể đọc được các câu lệnh trong thủ tục.

Phân loại:

- **System sp:** được lưu trữ trong CSDL master và được đặt tên với tiếp đầu ngữ là **sp**. Chúng đóng vai trò khác nhau của các tác vụ được cung cấp trong SQL Server.

  Ví dụ: Sp_help, Sp_helpConstraint, ….
- **Local sp:** được lưu trữ trong các CSDL của người dùng, nó thực thi các tác vụ (Task) trong CSDL chứa nó. Một Local sp có thể được người sử dụng tạo hoặc từ các sp hệ thống.
- **Temporary sp:** giống như là một local sp, nhưng nó chỉ hiện hữu cho đến khi kết nối tạo ra nó bị đóng. Nó được nằm trong CSDL TempDB. Có 3 loại temporary sp: local (private), Global, sp tạo trực tiếp trong TempDB.
- **Extended sp:** Là một thủ tục được tạo từ các ngôn ngữ lập trình khác (không phải SQL Server) và nó được triển khai tính năng của một thủ tục trong SQL Server. Các thủ tục này có tên với tiếp đầu ngữ là xp.
- **Remote sp:** là một thủ tục được gọi thực thi từ một server từ xa.

11.1.2 Tạo, thực thi, hiệu chỉnh, xóa stored procedures.

- Khi thực thi thủ tục, bạn phải cung cấp các giá trị của tham số của các thủ tục nếu có. Một thủ tục có thể được gọi thực thi hoặc tự động thực khi SQL Server khởi động. Gọi thực thi bằng từ khóa EXECUTE.

- Khi cần thêm một thàm số (parameter) hoặc thay đổi một vài phần trong đoạn mã thì ta dùng lệnh ALTER để hiệu chỉnh.
- Xóa một thủ tục dùng lệnh DROP
- Các thủ tục có thể được tạo trước khi các đối tượng mà thủ tục tham chiếu, đặt tính này gọi là tính trì hoãn.

Tạo thủ tục

Cách 1: Dùng Enterprice Manager

R-Click tại Store procudure trong CSDL, chọn New Store procudure

Cách 2: Tạo Stored procedure Wizard

Tool → Wizard, click vào DataBase, chọn Create Store Procedure Wizard

Cách 3: Bằng lệnh Create proceduce

CREATE PROC [EDURE] *procedure_name* [; *number*]
[{ *@parameter data_type* } [VARYING] [= *default*] [OUTPUT]
] [,...*n*] [WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION }] [
FOR REPLICATION]
AS
***sql_statement* [...*n*]**

[VARYING]: Chỉ dùng với biến Cursor

Ví dụ 1: Tạo thủ tục để liệt kê các order có ngày giao hàng đã quá hạn theo yêu cầu

```
CREATE PROC dbo.overdueOrders  – quá hạn
AS
     SELECT *
     FROM dbo.orders
     WHERE RequiredDate < GETDATE() and shippeddate is
     null
```

Kiểm tra sự tồn tại của Stored procedures

sp_helptext proc_name

Thực thi một Stored procedures

[[EXEC [UTE]]
{
[*@return_status* =]
{ *procedure_name* [;*number*] | *@procedure_name_var*
}
[[*@parameter* =] { *value* | *@variable* [OUTPUT] | [DEFAULT]]
[,...*n*]
[WITH RECOMPILE]

Đơn giản hơn

EXECUTE ProductName [; *number*][<parameter>[, ...n][OUTPUT]]

```
Ví dụ:    EXECUTE  dbo.overdueOrders
```

Thực thi sp ngay khi SQL Server khởi động: Store procedure phải nằm ở CSDL **Master.**

Cách 1: Dùng thủ tục **Sp_procoption** để gán thuộc tính tự động thực thi

```
USE Master
EXECute Sp_procoption [ @ProcName = ] 'procedure'
, [ @OptionName = ] Startup
, [ @OptionValue = ] True
```

Ví dụ:

```
EXECute Sp_procoption dbo.overdueOrders Startup, True
```

Cách 2: Dùng **Enterprice Manager**

R-Click tại tên thủ tục →Properties → Execute whenever SQL Server Start

Hiệu chỉnh một stored procedures

```
USE Northwind
GO
ALTER PROC dbo.overdueOrders
AS
    SELECT CONVERT(CHAR(8), RequiredDate,1)
    RequiredDate, CONVERT(CHAR(8), orderDate,1)
    orderDate, orderId, Customerid, EmployeeID
    FROM dbo.orders
    WHERE RequiredDate<GETDATE()and shippeddate is
    null
    ORDER BY RequiredDate
```

Xóa một stored procedures

```
DROP ProcedureName
```

11.1.3 Tham số và biến trong Stored procedures.

Tham số và biến là phần cơ bản để tạo nên sự uyển chuyển của thủ tục.

- **Input parameter**: tham số nhập, đưa giá trị của tham số để thông báo cho thủ tục nên làm gì trong CSDL
- **Output parameter**: tham số xuất chứa giá trị trả về của thủ tục.

Khi dùng tham số phải khai báo tham số (Tên tham số, kiểu dữ liệu, Giá trị mặc nhiên nếu có, có chỉ dẫn tham số OUT PUT không)

```
@parameter_name [AS] datatype
[=default | NULL] [VARYING] [OUTPUT]
```

Ví dụ 1: Tạo thủ tục dùng để chèn một mẫu tin vào bảng Customer, với các tham số dạng Input

```
USE SalesDB
GO
```

```
CREATE PROC Sp_InsertCust
@No_para VARCHAR(10),
@Name_para NVARCHAR(50),
@Address_para NVARCHAR(50),
@Phone_para VARCHAR(24),
@Fax_para VARCHAR(24),
@Mail_para VARCHAR(50)
AS
INSERT INTO tblCustomer
(CustNo, CustName, Address, Phone, Fdax, Mail)
VALUES (@No_para, @Name_para, @Address_para,
 @Phone_para, @Fax_para,@Mail_para)
```

Thực thi thủ tục có tham số

```
USE SalesDB
GO
Sp_InsertCust  'CDS',  'Trường Tin học ABC','12
Nguyễn Văn Bảo', '2352344',234652','cds@yahoo.com'
```

Kiểm tra việc chèn dữ liệu

```
SELECT CustMo, CustName, Address
FROM tblCustomer
```

Ví dụ 2:

```
USE Northwind
GO
IF EXISTS (SELECT name
           FROM  sysobjects
           WHERE  name = 'GetUnitPrice' AND
type = 'P')
          DROP PROCEDURE GetUnitPrice
GO

CREATE PROCEDURE GetUnitPrice @prod_id int, @unit_
price money OUTPUT
AS
     SELECT @unit_price = UnitPrice
     FROM  Products
     WHERE ProductID = @prod_id
     GO
```

Thực thi: Bạn phải khai báo một biến trước khi gọi thủ tục thực thi

```
DECLARE @price money
EXECUTE GetUnitPrice 77, @unit_price = @price OUTPUT
PRINT CONVERT(varchar(6), @price)
GO
```

Ví dụ 3: Tạo thủ tục InsertRows như sau:

```
USE SalesDB
GO
CREATE TABLE mytable
     ( column1 int, column2 char(10) )
GO
CREATE PROCEDURE InsertRows @start_value int
AS
DECLARE @loop_counter int, @start_val int
SET   @start_val = @start_value - 1
SET   @loop_counter = 0
WHILE (@loop_counter < 5)
   BEGIN
      INSERT INTO mytable VALUES (@start_val + 1, "new row")
       PRINT (@start_val)
       SET @start_val = @start_val + 1
       SET @loop_counter = @loop_counter + 1
   END
GO
```

Thực thi: Hãy thực thi thủ tục với giá trị khởi tạo là 1

```
EXECUTE InsertRows 1
GO
```

Kiểm tra kết quả

```
     SELECT * FROM mytable
```

Ví dụ 4:

```
--- Tạo thủ tục Count_tables có 2 tham số Output
USE Pubs
GO
CREATE PROC count_tables
     @authorcount INT OUTPUT,
     @titlecount INT OUTPUT
```

```
AS
SELECT * FROM authors
SET @authorcount=@@rowcount

SELECT * FROM Titles
SET @titlecount=@@rowcount

---- Thực thi thủ tục Count_tables
DECLARE @a_count INT, @t_count INT
EXECUTE count_tables @a_count OUTPUT, @t_count OUTPUT
SELECT authorcount=@a_count, titlecount=@t_count
```

Ví dụ 5: Viết một thủ tục tính giai thừa của một số nằm trong khoảng 0 đến 12

Dùng T-SQL để tính giai thừa bằng đệ quy, của các số từ 0 đến 12. Các tham số có giá trị lớn hơn 12 sẽ không cho phép bởi vì kết quả trả về sẽ vượt quá phạm vi của dữ liệu kiểu int

```
CREATE PROC Giaithua @so1 int
AS
DECLARE @So1Giam1 int, @Ketqua int
IF (@so1 < 0 OR @so1 > 12)
    BEGIN
        -- Giá trị tham số không hợp lệ.
        RETURN -1
    END

IF (@so1=0 or @so1=1)
    SELECT @Ketqua=1
ELSE
    BEGIN
        SET @ So1Giam1=@So1 - 1
        EXEC @Ketqua=Giaithua @  So1Giam1 – Đệ qui gọi
lại chính nó
        IF (@Ketqua= -1)
            BEGIN
                RETURN -1
            END

        SET @Ketqua=@Ketqua * @so1
        IF (@@ERROR <> 0)
            RETURN -1
```

```
    END

     RETURN(@Ketqua)
```

Thực thi: Tính giai thừa cho 3

```
DECLARE @kq  INT
EXEC @kq = giaithua 3
PRINT @kq
```

Khi thủ tục Giathua đã tồn tại thì chúng ta có thể dùng bó lệnh sau để hiển thị giai thừa của tất cả các số từ 0 đến 12:

```
DECLARE @Ketqua int, @n int
SET @Ketqua =0
SET @n =0
WHILE (@n <= 12) BEGIN
  EXEC @Ketqua = Giaithua @n
    IF (@Ketqua = -1) BEGIN
        RAISERROR('Error executing factorial procedure.
', 16, -1)
        RETURN
    END
PRINT CONVERT(varchar, @n) + '! = ' + CONVERT(varchar(5
0), @Ketqua)
SET @n=@n + 1
END
```

11.2 FUNCTIONS.

Hàm thực sự tương tự như Stored procedure của SQl Server, nội dung bao gồm các phát biểu T-SQL kết hợp tạo thành hàm, có thể gọi thực thi các hàm như là một đơn vị độc lập.

Hàm được dùng trong:

- Danh sách chọn của một câu lệnh Select để cho ra một giá trị.
- Một điều kiện tìm kiếm của mệnh đề Where trong các câu lệnh T-SQL

Buil-in functions: Những hàm này hoạt động như là một định nghĩa trong T-SQL và không thể hiệu chỉnh. Nó có thể chỉ được tham chiếu trong các câu lệnh T-SQL. Giá trị trả về của hàm có thể là một Rowset (tập các dòng), argergate và scalar (vô hướng).

Nên tìm hiểu và tận dụng tối đa các hàm Buil-in function của SQL Server

User-defined function hay còn gọi là UDFs: Những hàm này do người dùng tự định nghĩa để đáp ứng một mục tiêu nào đó. Một số hạn chế so với thủ tục là các tham số truyền vào không được mang thuộc tính OUTPUT, nghĩa là giá trị của tham số không được truyền ra bên ngoài hàm UDF, thay vào đó ta phải sử dụng giải pháp là trả về giá trị cho hàm bằng phát biểu RETURN. Giá trị trả về của hàm có thể là một giá trị vô hướng (Scalar valued) hoặc bảng (Table-valued)

Scalar Function. Một hàm vô hướng trả về một giá trị đơn và có thể được dùng bất cứ nơi nào biểu thức hay biến có thể được dùng (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

Table-valued Function. Một hàm có giá trị bảng trả về một tập kết quả và có thể được dùng bất cứ nơi nào bảng hay view được dùng. Hàm giá trị bảng có thể được tham chiếu trong mệnh đề FROM của câu lệnh SELECT. Các hàm người dùng có thể có nhiều phức tạp hơn viêu và có thể có tham số.

Với các ngôn ngữ lập trình khác, nếu tham số không truyền vào thì xem như là hàm sẽ lấy giá trị default của các tham số, nhưng với SQL Server thì phải truyền giá trị Default vào.

11.2.1 Scalar Functions

```
CREATE FUNCTION [ owner_name. ] function_name
  ([ { @parameter_name [AS] scalar_parameter_data_type [ = default ] } [,...n ] ])
RETURNS scalar_return_data_type
[ WITH < function_option> [ [,] ...n] ]
[ AS ]
BEGIN
  function_body
  RETURN scalar_expression
END
```

Ví dụ:

```
CREATE FUNCTION TotalAmount
(@UnitPrice money, @Quantity smallint, @Discount
real)
RETURNS money
AS
BEGIN
RETURN (@UnitPrice*@Quantity)*(1-@discount)
END
```

Sử dụng:

```
SELECT ProductID, Total=dbo.TotalAmount(UnitPrice,
Quantity, Discount)
FROM [Order details]
WHERE OrderID=10250
```

11.2.2 Table-valued Functons

Được chia thành hai loại nhỏ: inline table-value và multistatement table-valued.

Một hàm *inline table-valued:* Nó có thể được xem như là một View có tham số. Chúng thực thi một câu lệnh Select như trong một view nhưng có thể bao gồm các tham số, giống như thủ tục.

CREATE FUNCTION [*owner_name.*] *function_name*
([{ *@parameter_name* [AS] *scalar_parameter_data_type* [= *default*] } [,...*n*]])
RETURNS TABLE
[WITH < function_option > [[,] ...*n*]]
[AS]
RETURN [(] *select-stmt* [)]

Ví dụ 1:

```
CREATE FUNCTION SalesByCategory(@Categoryid Int)
RETURNS TABLE
AS
RETURN
(SELECT c.CategoryName, P. ProductName,
         SUM(Quantity) AS TotalQty
  FROM Categories c
  INNER JOIN Products p ON c.CategoryID= p.
CategoryID
  INNER JOIN [Order Details] od ON p.ProductID =
od.ProductID
  WHERE c.CategoryID= @Categoryid
GROUP BY c. CategoryName,p.ProductName)
```

Hàm Multistatement Table-valuesd là dạng phức tạp nhất. Loại hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh SELECT.

CREATE FUNCTION [owner_name.]function_name
([{@parameter_name [AS] data_type [=default]} [,...n]])
RETURNS @return_variable
TABLE ({column_definition | table_constraint} [,...n])
[WITH { ENCRYPTION | SCHEMABINDING } [[,] ...n]]
[AS]
BEGIN
function_body
RETURN
END

Ví dụ:

```
CREATE FUNCTION Contacts(@suppliers bit=0)
RETURNS @Contacts TABLE (ContactName nvarchar(30),
Phone nvarchar(24), ContactType nvarchar(15))
AS
BEGIN
```

```
INSERT @Contacts
SELECT ContactName, Phone, 'Customer' FROM
Customers
INSERT @Contacts
SELECT FirstName + ' ' + LastName, HomePhone,
'Employee'
FROM Employees
IF @Suppliers=1
INSERT @Contacts
SELECT ContactName, Phone, 'Supplier‘
FROM Suppliers
RETURN
END
```

Sử dụng

```
SELECT * FROM CONTACTS(1) ORDER BY ContactName
```

BÀI 12:
TRANSACTIONS – LOCK

SQL Server dùng các transaction và các lock để đảm bảo tính vững chắc và toàn vẹn dữ liệu, không chấp nhận các lỗi xảy ra trong hệ thống.

12.1 TRANSACTIONS

Transaction: Một transactin có thể được định nghĩa như là một chuỗi các thao tác thực thi cùng với nhau như là một khối thống nhất đơn của công việc. Một khối thống nhất của công việc phải có bốn đặc điểm được gọi là *ACID* (*Atomicity, Consistency, Isolation,* và *Durability*).

- **Atomicity**: đặc tính này thể hiện rằng hoặc là tất cả các hiệu chỉnh dữ liệu được thực hiện hoặc là tất cả chúng đều không được thực hiện.
- **Consistency**: đây là một trạng thái mà tất cả các dữ liệu ở trong tình trạng nhất quán sau khi một transaction được hoàn tất một cách thành công. Tất cả các qui tắc (rules) trong một CSDL quan hệ phải được thoả mãn đối với các hiệu chỉnh trong một transaction nhằm duy trì toàn bộ các toàn vẹn dữ liệu.
- **Isolation**: đặc tính này thể hiện rằng bất kỳ sự hiệu chỉnh dữ liệu thực hiện bởi các transaction đồng thời phải độc lập với các hiệu chỉnh khác của các transaction đồng thời khác. Nói một cách đơn giản hơn, một transaction hoặc là truy xuất dữ liệu ở trạng thái mà trước khi transaction đồng thời thực hiện hiệu chỉnh hoặc là truy xuất dữ liệu sau khi transaction thứ hai được hoàn tất.
- **Durability**: Đặc tính này thể hiện rằng bất kỳ thay đổi trong dữ liệu bởi một transaction đã hoàn tất giữ nguyên ảnh hưởng trong hệ thống. Vì vậy, bất kỳ sự thay đổi bởi một transaction hoàn tất vẫn còn thậm chí trong sự kiện hệ thống bị fail. Đặc tính này được đảm bảo bởi sự dự phòng và phục hồi transaction log.

Ba loại Transaction:

- **Implicip Transactions (Transaction ngầm định):** Khi một connection đang mở trong chế độ implicip, SQL Server bắt đầu một transaction mới một cách tự động sau khi transaction hiện hành hoàn tất hoặc Roll back. Bạn không cần phát họa bắt đầu một transaction; bạn chỉ cần commit hoặc Rollback mỗi transaction. Chế độ Implicit transaction phát sinh một chuỗi các transaction liên tục

  Sau khi chế độ transactin implicit đã được bật ON cho một kết nối, SQL Server tự động bắt đầu một transaction khi nó thực thi bất kỳ các lệnh sau: Alter Table, Creat, Delete, Drop, Fetch, Grant, Insert, Open, Revoke, Select, Trucate Table, Update.
- **Explicip Transactions: (Transaction tường minh):** Là một transaction mà chúng ta phải định nghĩa bắt đầu một transaction (Begin transaction) và kết thúc một transaction (Commit Transaction)

Bắt đầu một transaction

```
BEGIN TRAN [ SACTION ] [ transaction_name | @tran_name_variable]
```

Hoàn Tất Transaction

```
COMMIT [ TRAN [ SACTION ] [ transaction_name | @tran_name_variable ] ]
```

Lưu vị trí Transaction

```
SAVE TRAN [ SACTION ] { savepoint_name | @savepoint_variable }
```

Hũy một Transaction

```
ROLLBACK [ TRAN [ SACTION ]
   [ transaction_name | @tran_name_variable
   | savepoint_name | @savepoint_variable ] ]
```

@@TRANCOUNT: Trả về số thứ tự mà trasaction được mở, tối đa lồng 32 cấp, không nên lồng.

- **Distributed Transaction:** là một loại explicip Trangsaction nhưng giao tác của nó liên quan nhiều server. Sự quản lý phải được kết hợp giữa các nhà quản lý tài nguyên của các server và điều này gọi là transaction manager. Các transaction trong một server những tham chiếu từ nhiều database, thực ra cung là một distributed transaction.

Transaction log là một tranction dùng lock để ngăn chặn người dùng hiệu chỉnh dữ liệu ảnh hưởng từ các transaction chưa hoàn tất.

Công dụng transaction log

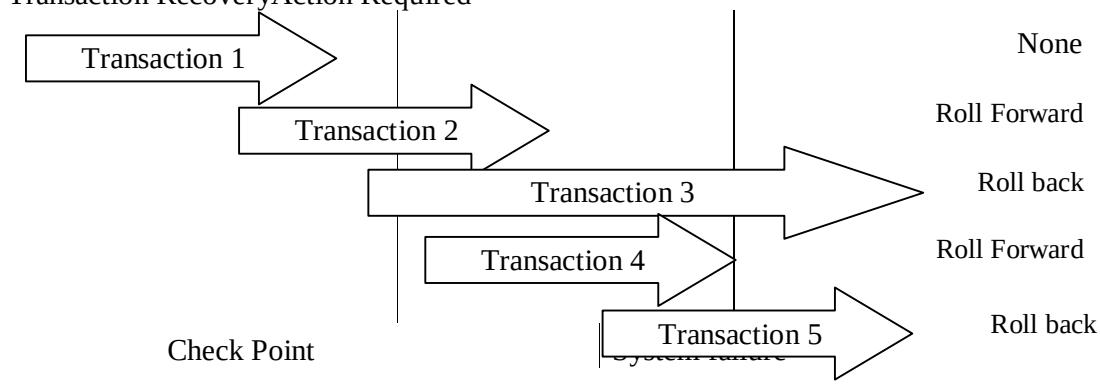
- **Phục hồi các transaction đặc biệt**: Khi một application đưa ra lệnh ROLL BACK hoặc SQL Server nhận ra một lỗi, thì bảng ghi log được dùng để roll back bất kỳ hiệu chỉnh trong suốt quá trình của transaction chưa hoàn tất.
- **Phục hồi tất cả các transaction chưa hoàn tất khi Sql server được bắt đầu:**
- **Hoàn trả lại database lại đến một thời điểm bị lỗi:** Nhằm đảm bảo không phát sinh mâu thuẩn sau khi có sự cố.

Write-ahead Transaction Log: Dùng Write – ahead log đảm bảo rằng không có dữ liệu hiệu chỉnh nào được cất vào đĩa trước khi cất trong log record.

Check point : Là một hành động thực hiện định kỳ trong CSDL, nó sẽ ghi lại tất cả các transaction liên quan đến data lên transaction log, nhằm phục vụ cho việc recovery data.

Check point xãy ra khi:

- Gặp câu lệnh Check point.
- Có sự hiệu chỉnh trên CSDL
- Trước ngay khi SQL Server Shutdow
- Áp định định kỳ.



12.2 LOCK

Lock: là cơ cấu ngăn chặn các xung đột do các user không thể đọc hoặc hiệu chỉnh các dữ liệu mà các dữ liệu này hiện đang trong một tiến trình xử lý khác. Tuy nhiên, bạn vẫn có thể thao tác trên những đối tượng còn phụ thuộc vào chuyển tác mà user khác đang thực hiện. Khi đó hệ thống sẽ kiểm soát tiến trình của bạn có tương thích với quá trình trước đó hay không.

Các vấn đề đồng thời (Concurrency problem)

- **Lost Updates (cập nhật mất dữ liệu):** Lost updates xảy ra khi 2 hoặc nhiều transaction chọn cùng một dữ liệu và sau đó cập nhật dòng dựa trên giá trị cũ. Mỗi transaction không biết những transaction khác. Thao tác cập nhật cuối cùng ghi đè lên những thao tác cập nhật khác mà kết quả dẫn đến mật dữ liệu.
- **Uncommitted Dependency (Dirty Read – đọc dữ liệu sai)**: Uncommitted Dependency xảy ra khi transaction thứ 2 chọn một dòng mà đang sẳn sàng cập nhật bởi một transaction. Transaction thứ 2 đang sẳn sàn đọc dữ liệu mà chưa được hoàn tất và có thể bị thay đổi bởi transaction cập nhật
- **Inconsistent Analysis (Nonrepeatable Real – đọc hai lần mẫu tin)**: Xảy ra khi transaction thứ hai truy xuất cùng một dữ liệu với vài lần và đọc lên những dữ liệu khác nhau ở mỗi lần đọc. Inconsistent Analysis tương tự với Uncommitted Dependency trong trường hợp transaction thứ nhất đang hiệu chỉnh dữ liệu thì một trasaction thứ hai đọc dữ liệu. Tuy nhiên, dữ liệu đọc bởi transaction thứ hai đã được commited bởi transaction update
- **Phantom Reads (đọc các mẫu tin ma)**: Xảy ra khi hành động insert hoặc delete được thi hành trên một dòng dữ liệu mà nó thuộc vùng dự liệu đọc của một transaction khác.

Kiểu locks

- **Share locks:** được dùng cho những thao tác mà không làm thay đổi hay cập nhật dữ liệu (thao tác chỉ đọc), như là một câu select
- **Exclusive locks:** được dùng cho nhữ thao tác hiệu chỉnh dữ liệu, như là Insert, Update, hay Delete. Đảm bảo rằng nhiều cập nhật không thể được thực hiện trong cùng tài nguyên tại cùng một thời điểm.
- **Update locks:** được dùng trên những tài nguyên mà có thể được cập nhật. Ngăn chặn một dạng thông thường của deadlock mà xảy ra khi nhiều session đang độc, đang lock, và có khả năng cập nhật tài kuyên sau này.
- **Intent locks:** Dùng để thiết lập một lock kế thừa. Kiểu intent lock là : Intent shared (IS), Intent exclusive (IX), và share with intent exclusive (SIX)
- **Schema locks**: được dùng khi thao tác thùy thuộc vào giản đồ của table là đang thực thi. Kiểu schema locks là schema modification (Sch-M), schema stability (Sch-S)
- **Bulk Update (BU) locks**: Cho phép chia sẽ cho Bulk-copy thi hành.

Deadlock

Deadlock xảy ra khi có một sự phụ thuộc chu trình giữa hai hay nhiều luồng cho một tập hợp tài nguyên nguyên nào đó. SQL Server sẽ tự giải quyết trường hợp deadlock bằng cách RoolBack một trong các transaction, và ưu tiên rollback những transaction có thời gian ít hơn. Để giảm bớt deadlock, bạn nên:

- ✓ Truy xuất các object theo thứ tự.
- ✓ Tránh sự tướng tác người dùng trong thời gian transaction.
- ✓ Cố giữ transaction càng ngắn càng tốt.
- ✓ Dùng mức cô lập thấp nhất.
- ✓ Dùng giới hạn các connection.

BÀI 13:
SỬ DỤNG CURSORS ĐỂ TRUY XUẤT DỮ LIỆU

13.1 Khái niệm

Cursor là một đối tượng của CSDL mà nó hỗ trợ cho phép truy xuất và thao tác dữ liệu trong một tập kết quả (result set). Sau khi cursor được định vị trên một dòng, các hoạt động có thể thực hiện trên dòng đó hoặc khối các dòng bắt đầu từ vị trí đó.

Dùng Cursor để:

- Định vị một dòng đặc biệt trong tập kết quả.
- Truy xuất một hoặc khối dòng bắt đầu từ vị trí cursor trong tập kết quả.
- Cung cấp thao tác hiệu chỉnh dữ liệu cho các dòng tại vị trí của cursor trong tập kết quả.
- Cung cấp các mức độ khác nhau của tính tường minh trong dự thay đổi được tạo bởi những người dùng khác đến tập kết quả.
- Cung cấp việc truy cập dữ liệu trong tập kết quả cho các câu lệnh T-SQL trong scripts, Stored procedure, và triggers.

Các thao tác cần thực hiện trong khi sử dụng cursor trong SQL Server:

- Cursor cần phải khai báo và các thuộc tính của nó cũng cần được xác định.
- Mở cursor.
- Phải lấy (fetch) các dòng cần thiết từ cursor.
- Dữ liệu trong dòng hiện hành có thể được hiệu chỉnh nếu cần thiết.
- Tạm thời không dùng cursor thì phải đóng cursor lại.
- Cursor cần phải được giải phóng (deallocate) khi không cần dùng nữa.

Cursor:

- **T-SQL Server cursors**: Cursor này được dựa trên câu lệnh khái báo cursor, nó được dùng chủ yếu trong các script, strore procedure, triggers. Nó được thi hành trên server và được quản lý bởi các câu lệnh T0SQL gửi từ client đến server. Khi làm việc với cursor thì phải khai báo, mở, truy xuất, xử lý, đóng, giải phóng.
- **API Server cursors (API-Application Program interface)**: Nó được thực thi trên server và được quản lý bởi một hàm cursor API. API Server cursors được cung cấp bởi hàm cursor API trong OLE DB, ODBC, và DB-Library. Mỗi lần một ứng dụng của client gọi một hàm cursor API, SQL server OLE DB provider, ODBC driver, hoặc DB-Library DLL gửi các yêu cầu đến server cho hành động ứng với các hàm cursor API. Nó chỉ khác với T-SQL cursor ở syntax, còn về bản chất tương tự.
- **Client Cursors**: SQL server ODBC driver, DB-Library DLL, và ADO API DLL giúp thi hành cursor client một cách nội tại. cursor client được thi hành bằng cách nắm giữ tập kết quả dữ liệu của clients. Mỗi lần Application của client gọi hàm cursor API, thì SQL server OLE DB provider, ODBC driver, hoặc DB-Library DLL thực thi tính toán ngay trên tập kết quả dữ liệu được giữ trên client. Ta chỉ dùng client cursor để làm giảm bớt sự giới hạn mà server consor không được cung cấp các câu lênh T-SQL. Nếu con trỏ static hoặc Scroll thì ta có thể dùng client cursor.

Loại cursor

Static

- Tập kết quả của contrỏ loại Static được xây dựng trong tempdb khi con trỏ được mở.
- Một static con trỏ luôn luôn hiện tập kết quả giống như tập kết quả có được ngay sau khi con trỏ mở.
- Con trỏ không phản ánh được bất kỳ sự thay đổi nào trong database ngay cả khi những dòng dữ liệu có thay đổi, các dòng mới được insert bởi các transaction khác cũng vẫn không hiện lên mặc dù chúng thỏa điều kiện lọc dữ liệu.
- Thao tác Insert, Update, Delete đều không có tác dụng khi dùng static cursor

Keyset-driven:

- Thành viên và thứ tự của các dòng trong một keyset-driven cursor là cố định khi cursor được mở. Con trỏ được điều khiển bởi một tập giá trị nhận dạng goi là Keyset. Keyset được xây dựng từ một tập các cột mà dùng để nhận dạng các dòng trong tập kết quả. Keyset được xây dựng trong Tempdb khi con trỏ được mở
- Cho phép hiệu chỉnh (update, delete) dữ liệu trên cột không là keyset (bởi chủ cursor hoặc từ user khác) khi user duyệt thông qua con trỏ.
- Có thể thêm (insert) vào bảng nếu như cursor thể chèn dữ liệu vào bảng.

Dynamic: Trái ngược với static cursor.

- Dynamic cursors phản ánh được toàn bộ sự thay đổi của các dòng dữ liệu trong tập kết quả khi duyệt con trỏ.
- Giá trị dữ liệu, thứ tự, và thành viên của các dòng trong tập kết quả có thể thay đổi ứng với mỗi lần duyệt con trỏ.
- Tất cả các lệnh Insert, Update, Delete của các user đều hữu hiệu thông qua con trỏ.
- Sự Update hữu hiệu ngay tức thời nếu chúng được update thông qua qua con trỏ ứng với tại mẫu tin hiện thời, còn nếu update bên ngoài con trỏ thì nó không hữu hiệu cho đến khi nó hoàn tất.

Fast Forward only: Tương tự như Dynamic cursor nhưng nó chỉ có thể duyệt con trỏ thào một chiiều từ First đến Last

Cusrsor type	Membership	Order	Values
Forward-only	Dynamic	Dynamic	Dynamic
Static	Fixed	Fixed	Fixed
Dynamic	Dynamic	Dynamic	Dynamic
Keyset-driven	Fixed	Fixed	Dynamic

13.2 Làm việc với T-SQL server cursors

Khai báo cursor

Bằng câu lệnh declare ở sau từ khóa AS trong stored procedures hoặc functions

```
DECLARE cursor_name CURSOR
[ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
```

[STATIC | KEYSET | DYNAMIC | FAST_FORWARD]
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
[TYPE_WARNING]
FOR *select_statement*
[FOR UPDATE [OF *column_name* [,...*n*]]]

- *select_statement:* là câu lệnh truy vấn để định nghĩa tập kết quả của cursor. Từ khóa COMPUTE, COMPUTE BY, FOR BROWSE, and INTO Không cho phép trong *select_statement* này.
- READ ONLY Không cho phép Update trong cursor này.
- UPDATE [OF column_name [,...n]]: Quy định cột cho phép được update khi dùng cursor. Nếu OF column_name [,...n] được chỉ định rõ ràng thì chỉ có các cột được chỉ định mới được cho phép hiệu chỉnh, nếu không có column list, all columns có thể update.

Mở cursor

Mở cursor trước khi dùng (Cursor phải được khai báo rồi)

OPEN {cursor_name }

Lấy mẫu tin hoặc điều hướng cursor (FETCH)

Truy xuất từng dòng dữ liệu. Kiểm tra phạm vi con trỏ bằng **@@Fetch_status**

FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n]]
FROM cursor_name[INTO @variable_name [,...n]]

@@Fetch_status: Trả về giá trị 0 hoặc 1. Trả về **1** có nghĩa là con trỏ đã được dời đến quá cuối tập kết quả. Trả về **0** vẫn còn trong phạm vi của tập kết quả. Khi đó dùng vòng lặp While để duyệt cả tập kết quả của Cursor.

Xử lý dữ liệu:

Có thể dùng các câu lệnh Upadte hoặc Delete để hiệu chỉnh dữ liệu

- Sử dụng dữ liệu của mẫu tin hiện hành: dữ liệu được lấy lên và gán cho các biến tương tứng trong câu lệnh Fetch
- Cập nhật dữ liệu thông qua cursor: thực chất là dữ liệu được hiệu chỉnh trực tiếp vào trong bảng

Cập nhật giá trị cho cột

UPDATE <Ten Table>
SET <TenColumn>= <Value> [,..n]
WHERE CURRENT OF <Cur_Name>

Xoá dữ liệu thông qua cursor: Thực sự là dữ liệu xóa trên bảng

DELETE <Ten Table>
WHERE CURRENT OF <Cur_Name>

Đóng Cursor

Kết thúc hành động của cursor cho lần mở (open), nó vẫn hiện hữu cho đến khi gặp một lệnh Open khác hoặc gặp lệnh Close cursor.

CLOSE cursor_name

Giải phóng Cursor

Giải phóng cursor, huy bỏ tham chiếu đến con trỏ từ session hiện hành. Tiến trình này làm cho tài nguyên trở về trạng thái sẵn sàng truy xuất.

DEALLOCATE cursor_name

13.3 Ví dụ.

Giả sử người quản lý cần một bảng báo cáo lịch sử khách hàng theo dạng sau:

Customer:ALFKI - Alfreds Futterkiste

Order:10643 (Aug 25 1997)

Order:10692 (Oct 3 1999)

Order:10702 (Oct 13 1999)

Customer:ANATR - Ana Trujillo Emparedados y helados

Order:10625 (Aug 8 1997)

Order:10759 (Nov 28 1999)

Customer:ANTON - Antonio Moreno Taquería

Order:10507 (Apr 15 1997)

Order:10535 (May 13 1999)

Order:10573 (Jun 19 1999)

Order:10677 (Sep 22 1999)

Order:10682 (Sep 25 1999)

Đoạn Batch thực hiện báo cáo như sau:

```
--- Khai báo contrỏ
DECLARE rpt CURSOR FOR
SELECT c.CustomerID, c.CompanyName, o.OrderID,
o.OrderDate
FROM Customers c, Orders o
WHERE c.CustomerID = o.CustomerID AND c.CustomerID LIKE
'A%' AND DatePart( year, o.OrderDate) = 1997

DECLARE @cid char( 8), @cname char( 40),
@ordid char( 8), @orddt datetime, @old char( 8)
--- Mở contrỏ
OPEN rpt
--- Lấy dữ liệu của mẫu tin đầu tiên vào các biến
FETCH NEXT FROM rpt INTO @cid, @cname, @ordid, @orddt
SELECT @old = ' ‘
```

```
WHILE @@fetch_status = 0
BEGIN
     IF @old = @cid
          BEGIN
          PRINT ' Order:' + rtrim( @ordid) + ' (' +
     cast(@orddt as CHAR( 10)) + ')'
         END
     ELSE
         BEGIN
          PRINT 'Customer:' + rtrim( @cid) + ' - ' +
     rtrim(@cname)
          PRINT ' Order:' + rtrim( @ordid) + ' (' +
     cast(@orddt as CHAR( 11)) + ')'
          SELECT @old = @cid
         END
     FETCH NEXT FROM rpt INTO @cid, @cname, @ordid,
@orddt
END
--- Đóng con trỏ
CLOSE rpt
--- Giải phóng con trỏ
DEALLOCATE rpt
```

Ví dụ 2:

```
DECLARE MyCursor CURSOR FOR
SELECT c.CustomerID,c.Companyname,c.contactname,
     o.OrderID,o.OrderDate
FROM Customers c, Orders o WHERE c.CustomerID =
o.CustomerID
FOR UPDATE
OPEN MyCursor
DECLARE @cid VARCHAR( 8), @c VARCHAR( 80), @o INT,
     @od DATETIME, @cn VARCHAR( 80)
FETCH NEXT FROM MyCursor INTO @cid, @c, @cn, @o, @od
SELECT @cid
BEGIN TRANSACTION
UPDATE Customers SET CompanyName = 'q'
WHERE CURRENT OF Mycursor
DEALLOCATE MyCursor
SELECT *  FROM Customers
```

```
ROLLBACK TRANSACTION
```

BÀI 14:
BẪY LỖI - TRIGGER

14.1 Giới thiệu về trigger

Chất lượng của một CSDL được đánh giá một phần bởi tính nhất quán và độ chính xác của dữ liệu trong CSDL. Để đảm bảo tính toàn vẹn dữ liệu ta có nhiều phương pháp, trigger là một phương pháp hữu hiệu.

- Trigger là một loại stored procedure đặc biệt, nó được định nghĩa để tự động thực thi khi có một câu lệnh Update, Insert, hoặc Delete được phát ra trên bảng hoặc View.
- Trigger là một công cụ mạnh mà nó có thể dùng để ràng buộc các qui tắc quản lý một các tự động khi dữ liệu bị hiệu chỉnh.
- Trigger cũng có thể nới rộng các tính toàn vẹn kiểm soát logic của SQL Server.
- Trigger tự động thực thi, không thể gọi một trigger thi hành một cách trực tiếp.

Dùng trigger khi:

- Ràng buộc toàn vẹn dữ liệu cho phù hợp với mô hình quan hệ CSDL.
- Kiểm soát dữ liệu hiện tại khi có thay đổi đến giá trị trong mẫu tin trong bảng.
- Kiểm tra dữ liệu nhập vào phù hợp với mối quan hệ dữ liệu giữa các bảng.
- Định nghĩa thông báo lỗi của người dùng.
- So sánh trạng thái của dữ liệu trước và sau hiệu chỉnh

Đặc điểm và giới hạn của trigger

- Trigger (After trigger) là Reactive; constraints và instead of trigger là proactive. (Reactive: khi delete/insert một dòng vào table, thì sau khi insert thì trigger mới được tự động thực thi thì gọi là reactive, proactive là kiểm tra trước khi Insert/Delete)
- Các constraint được kiểm tra trước, sau đó mới tới trigger.
- Các bảng có thể có nhiều trigger cho bất kỳ hành động nào. Tuy nhiên không nên dùng quá nhiều trigger trong cùng một bảng. SQl Server chỉ cho phép chỉ định trigger nào thi hành đầu tiên, thi hành cuối cùng, còn các trigger khác thứ tự thi hành không xác định. Vì vậy, nếu có quá nhiều trigger trên 1 đối tượng có thể sẽ gặp nhiều rắt rối khi có nhiều trigger không xác định thứ tự.
- Không thể tạo trigger trên các đối tượng ở temporary.
- Nên thiết kế trigger không trả về tập kết quả nhằm đảm bảo tính chất chuyển tác giữa các user và lập trình.
- Các trigger có thể xử lý hành động trên nhiều dòng.
- Trigger không ngăn ngừa thay đổi cấu trúc, trigger chỉ quan tâm đến sự thay đổi dữ liệu trong bảng. Khi bạn xóa đối tượng trong CSDL với các bước hợp lý, SQL Server sẽ cho phép xóa đối tượng đó và trigger không thể kiểm soát được.
- **Trigger Events:** Có 3 biến cố mà trigger sẽ tự động thực thi khi biến cố xảy ra, đó là **Insert, Update, Delete**. Trigger không thể được gọi một cách trực tiếp.

Cơ cấu thực thi trigger:

Khi insert hoặc update dữ liệu của bảng bật trigger, trigger sẽ lưu trữ dòng dữ liệu mới hoặc dòng dữ liệu đã hiệu chỉnh vào một bảng có tên là **Inserted** trong bộ

nhớ cash, khi xóa dữ liệu của bảng bật trigger lên, trigger sẽ lưu trữ dòng dữ liệu bị xóa vào bảng có tên là **Deleted** trong bộ nhớ cash. Các bảng tồn tại trong bộ nhớ và được truy vấn bởi các lệnh T-SQL trong các trigger. Bạn có thể sử dụng thông tin trong bảng inserted và Deleted để so sánh, lưu trữ, rollback,… nếu cần, khi đó bạn không cần tạo ra các biến để lưu trữ thông tin và tốc độ truy xuất nhanh.

Hai loại trigger trong SQL Server 2000: INSTEAD OF và AFTER (nếu chỉ nói trigger có nghĩa là nói đến AFTER Trigger)

- **FOR triggers và AFTER triggers:** các trigger này chỉ được thực thi khi tất cả các thao tác Insert, Update hay Delete thực hiện xong. Tất cả các hành động tham chiếu và kiểm tra constraint cũng phải được thực hiện xong trước khi trigger thi hành. Loại trigger này chỉ cài đặt được trên bảng, không cài đặt được trên View.

  Khi tạo trigger và không chỉ định rõ thì mặc định là AFTER, FOR chỉ là từ khóa tương thích ngược với các phiên bản trước của SQL Server.
- **INSTEAD OF triggers:** Trigger này chỉ có trong SQL Server 2000. Trigger này sẽ thi hành thay cho các câu lệnh Insert, Delete, Update. Như vậy khi tạo trigger kiểu này bạn phải viết lại các lệnh insert, Delete, Update đối với dữ liệu. Có thể áp dụng cho cả bảng và View, tuy nhiên nó không cho phép áp dụng với các view có lựa chọn WITH CHECK OPTION.

Nested trigger: có nghĩa là bảng Table1 có trigger, Table2 có trigger khác. Nếu ta thao tác trên Table1 thì trigger của nó sẽ thực thi, nếu thao tác này có liên quan đến Table2 thì trigger2 ở bảng Table2 thực thi. Gọi là lồng các trigger, bạn có thể lồng tối đa là 32 cấp.

14.2 Tạo và quản lý các trigger

Một trigger có thể tạo và quản lý bằng cách sử dụng Query Analyzer hoặc Enterprice Manager. Tạo một trigger trên đối tượng thì phải có quyền Owner đối với đối tượng.

14.2.1 Tạo trigger

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
  { { FOR | AFTER | INSTEAD OF } { [DELETE] [,] [ INSERT ] [, ] [ UPDATE ] }
    [ WITH APPEND ]
    [ NOT FOR REPLICATION ]
    AS
    sql_statement [ ...n ]
  }
}
```

Giải thích:

- *trigger_name* : Tên của trigger. Nếu trong 1 bảng có nhiều trigger thì tên của các trigger phải là duy nhất.
- ON { *table* | *view* }: Chỉ định table/View được áp dụng trigger, chỉ có instead of được áp dụng cho cả view và table.
- [WITH ENCRYPTION]: Trigger được mã hóa. Thông tin mã hóa năm trong bảng syscomment
- **{** FOR | AFTER | INSTEAD OF **}:** Xác định loại trigger cho thao tác DELETE, INSERT, UPDATE.

Delete trigger: trigger sẽ được thực thi khi có mẫu tin bị xóa khỏi bảng, SQL Server tạo ra bảng mang tên DELETED để cất mẫu tin bị xóa, trong trigger ta có thể tham khảo đến mẫu tin này.

Insert trigger: trigger sẽ được thực thi khi có mẫu tin chèn vào bảng, SQL server tạo ra bảng mang tên INSERTED để cất mẫu tin chèn, trong trigger ta có thể tham khảo đến mẫu tin này.

Update trigger: Mỗi khi có mẫu tin nào đó được cập nhật, giá trị những cột có liên quan đến trigger sẽ được kiểm tra trước khi cập nhật. Mẫu tin bị cập nhật sẽ được sao lưu trong bảng Inserted (chứa giá trị mới) và Deleted (chứa giá trị cũ).

- [NOT FOR REPLICATION]: Trigger sẽ không thực hiện khi bảng có liên quan đến kỹ thuật sao chép nhân bản (relication)

Lưu ý:

- Một hành động (Insert hoặc Delete hoặc Update) có thể kích hoạt cùng lúc nhiều trigger khác nhau.
- Trigger sẽ thi hành cho dù thao tác của người sử dụng có tác động thực sự trên các mẫu tin hay không, do đó dùng @@ROWCOUNT để kiểm tra trước khi cho các hành động trong trigger thi hành.
- Các phát biểu sau không thể có trong trigger: các phát biểu Create, Drop, Alter Table, Alter DataBase, Truncate Table, Grant/Revoke, Reconfigure, Load DataBase, Transaction, Update statistics, Select Into, Disk.

14.2.2 Quản lý trigger

Alter Trigger [2] Hiệu chỉnh trigger

Drop Trigger : Xóa trigger

Sp_rename : đổi tên.

Sp_helptrigger, Sp_heltext: Xem code trigger

DISABLE TRIGGER/ ENABLE TRIGGER trong câu lệnh Alter Table

14.3 Vài ví dụ về trigger.

Ví dụ 1: Viết một trigger cho thao tác Insert, điểm kiểm tra ngày lập hoá đơn thì luôn luôn lớn hơn ngày giao.

```
CREATE TRIGGER Trg_NgayLap_NgayGiaoHD
ON Hoadon AFTER INSERT
AS
DECLARE @NgayLapHD DateTime, @NgayGiao DateTime
SELECT @NgayLapHD=hd.NGayLapHD,NgayGiao=hd.NgayGiaoNhan
FROM HoaDon hd INNER JOIN Inserted i ON hd.MaHD=i.Mahd
If @NgayGiao<@NgayLapHD
     BEGIN
          RAISERROR(500103,10,1)
```

[2] Tham thảo cú pháp lệnh trong Books-Online

```
            ROLLBACK TRANSACTION
        END
-------------------------------
INSERT HoaDon VALUES (1003,'1/1/2004','N','TP.
HCM',111,‘12/24/2003‘)
```

Ví dụ 2: Tạo một trigger có tên là Msg_InstUpd_Kh, trigger này thực hiện chi 2 thao tác Insert, Update của bảng KhachHang. Trigger sẽ thông báo “Có n dong da duoc hieu chinh”.

```
Use SalesDB
GO
CREATE TRIGGER Msg_InstUpd_Kh
On KhachHang
FOR INSERT, UPDATE
AS
RAISEORROR(‘”Có %d dòng đã được hiệu chỉnh””,
0,1,@@Rowcount)
RETURN
```

Kiểm chứng: bằng cách chèn hoặc cập nhật 1 hoặc nhiều mẫu tin vào bảng KhachHang.

```
INSERT INTO KhachHang …..
```

Ví dụ 3: Hai bảng HoaDon và CT_HoaDon có quan hệ 1-n, tức là khi thêm một chi tiết hóa đơn trong bảng chi tiết hoá đơn thì hóa đơn này phải đã được phát sinh trong bảng hóa đơn. Vì thế ta phải viết một trigger Insert cho bảng CT_HoaDon. Nếu ta có trigger này thì khi ta vi phạm SQL Server cũng sẽ báo lỗi vì vi phạm ràng buộc toàn vẹn khóa ngoại. Tuy nhiên ta hãy viết một trigger để thực hiện thông bao khi có lỗi này xuất hiện.

```
Use SalesDB
GO

CREATE TRIGGER Trigger_Ins_CT_HD
ON HoaDon
FOR INSERT
/* Insert trigger cho bảng hóa đơn*/
AS
IF NOT EXISTS
(Select * From Inserted I inner join HoaDon hd ON
i.Mahd = hd.Mahd)
/* Nếu Mahd được chèn vào bảng CT_HoaDon không tồn tại
trong bảng hóa đơn thì không chèn được */
BEGIN
```

```
RAISERROR(60000,16,1,'MaHd', 'CT_HoaDon','Mahd','Hoa
Don')
     ROLLBACK
END
```

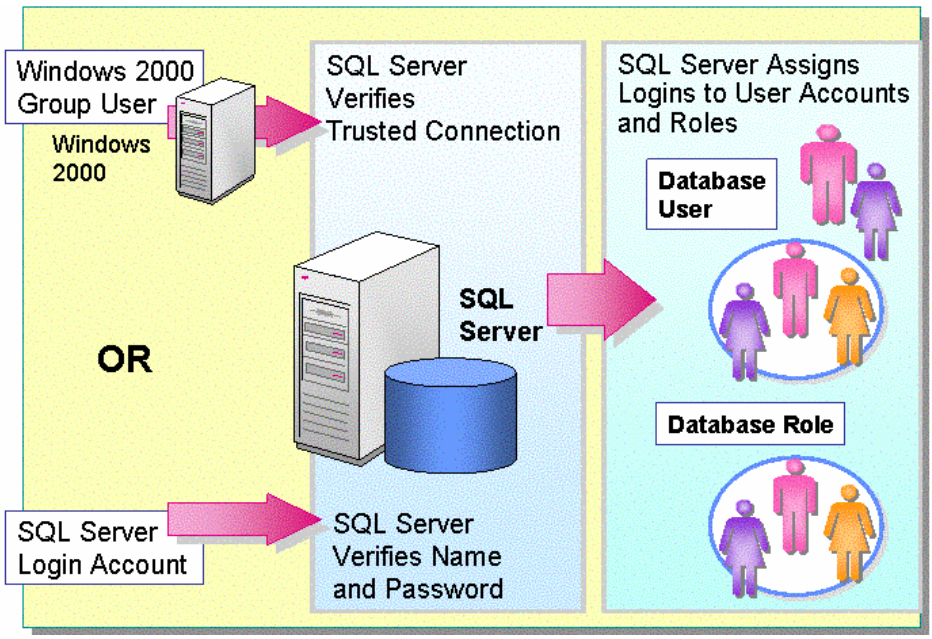
Ví dụ 4: Price trong CT_HoaDon luôn luôn lớn hơn hay bằng GiaGoc trong bảng sản phẩm. Để bắt ràng buộc này bạn thường dùng Check Constraints, tuy nhiên bạn cũng có thể bẩy lỗi bằng trigger

```
Create Trigger MGT_Gia
ON CT_HoaDon
FOR UPDATE
AS
IF EXISTS (Select * From INSERTED I where i.price <
tblItem.OriginalPrice)
     Begin
     RAISERROR('Khong thể cập nhật vì giá không hợp
lệ',16,1)
     ROLLBACK TRAN
     End
```

BÀI 15:
BẢO MẬT TRONG SQL SERVER

15.1 Khái niệm về bảo mật.

15.1.1 Mô hình truy cập bảo mật của SQL Server.



Hình 37: Các lớp kiểm tra bảo mật chứng thực của SQL Server

Việc kết nối đến SQL Server 2000 thật đơn giản. Sự bảo mật được kiểm tra ở ba nơi khác nhau: có thể bị kiểm hợp lệ bởi Windows 2000, bản thân SQL Server, mức CSDL riêng lẽ. Ngay sau khi bạn kết nối vào SQL Server bạn chưa thật sự truy cập được bất kỳ một đối tượng CSDL nào, bạn cần phải được cấp quyền (permissions) truy cập đến đối tượng.

15.1.2 Các chế độ bảo mật.

SQL Server 2000 cung cấp hai chế độ bảo mật:

Hai loại chứng thực

Windows Authentication

SQL Server kiểm tra nhận dạng của user và sau đó cho phép hay từ chối đăng nhập truy xuất dựa trên cơ sở tên của User mà không cần tên đăng nhập và password riêng biệt. Điều này gọi là kết nối tin tưởng. Khi bạn kết nối đến SQL Server theo cách này thì có nghĩa là bạn trình bày với SQL Server một ủy nhiệm bảo mật của Windows (như là một thẻ bài truy cập của bạn). Bạn xây dựng các ủy nhiệm này trong quá trình đăng nhập vào mạng windows 2000. Các ủy nhiệm bảo mật này được truyền âm thầm cho bạn, vì thế bạn không cần làm bất cứ điều gì đặc biệt để vượt qua việc kiểm tra bảo mật.

SQL Server Authentication

Người quản trị CSDL có thể tạo ra các tài khoản và password đăng nhập SQL Serevr. Các tài khoản này hoàn toàn không tuỳ thuộc vào các tài khoản hay nhóm người dùng hệ điều hành. Nếu có một kết nối chỉ định chứng thực SQL Server thì

SQL Server 2000 thực thi chứng thực chính nó bằng cách kiểm tra xem tài khoản đang nhập có tồn tại hay không và mật khẩu chỉ định có khớp với một ghi nhận trước đây trong SQL Server 2000 không.

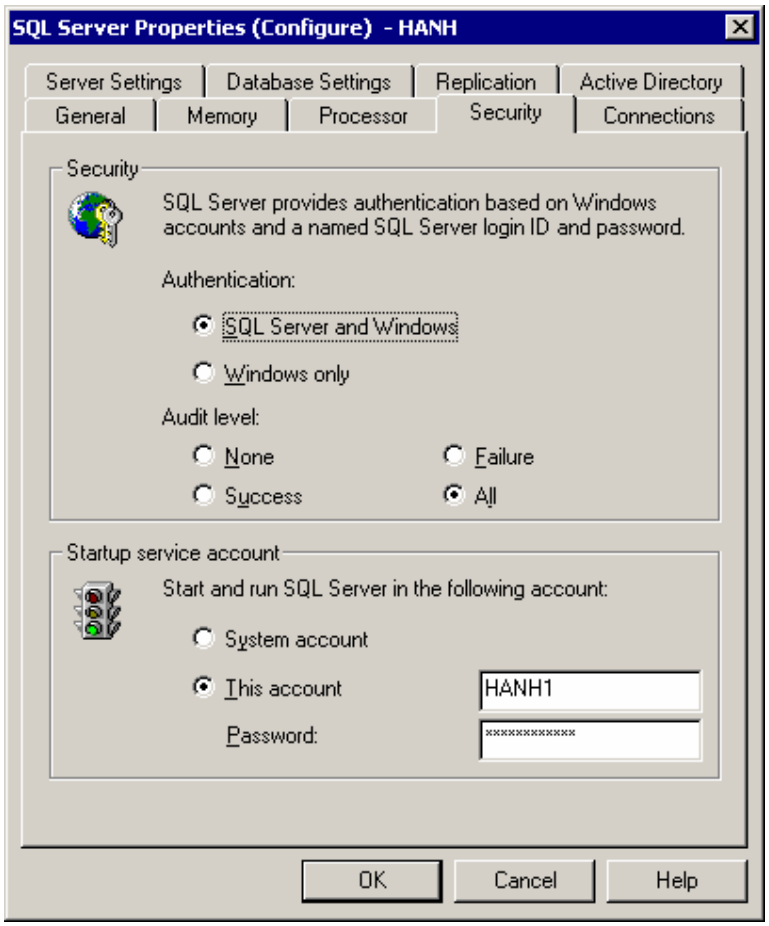
Chế độ chứng thực:

Windows Authentication mode: Người sử dụng chỉ có thể kết nối với SQL Server 200 bằng Windows Authentication (Kết nối tin tưởng)

Mixed mode: Người dùng có thể kết nối với SQL Server 200 bằng cách dùng cả Windows Authentication và SQL Server Authentication

Chuyển đổi chế độ chứng thực:

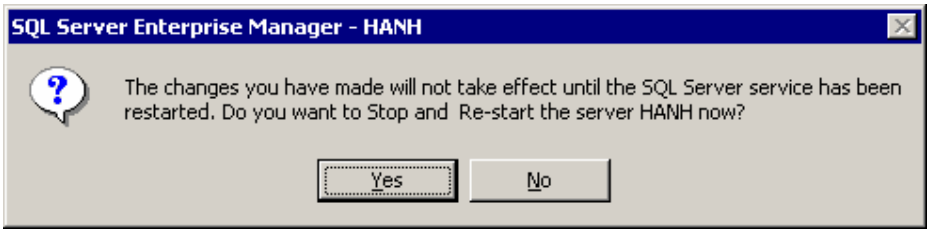
Sau khi cài đặt, bạn có thể sử dụng Enterprise Manager để chuyển đổi qua lại giữa các chế độ. Tại cửa sổ Enterprise Manager, nhắp nút phải chuột tại instance và chọn Properties



Hình 38: cấu hình Security cho SQL Server

Ở trang Security click chọn hoặc SQL Server And Windows hoặc Windows Only để đổi chế độ chứng thực.

Sau khi bạn chuyển chế độ, bạn phải stop và sau đó restart dịch vụ của SQL Server service để sự thay đổi này có tác dụng. Enterprise Manager sẽ hỏi ý kiến bạn.



Hình 39: Hộp thoại hỏi ý kiến người dùng có muốn Stop và Restart Server

15.1.3 Tìm hiểu các Server-Wide Permission.

SQL Server 2000 hỗ trợ một số server role được định nghĩa trước được kết hợp với các quyền quản trị. Các server role này cấp các quyền server-wide để thực hiện các tác vụ khác nhau và bao gồm cac quyền mà bạn có thể cấp cho những người dùng thông qua việc sử dụng các server role này. Bạn không thể xoá các server role và không thể thay đổi các quyền của chúng. Để cấp các quyền cho một người dùng, thì bạn thêm đăng nhập của chúng server role. Với Transact-SQL, bạn có thể thêm các người dùng hay nhóm các người dùng vào một server role

Lưu ý: Các Server role về cơ bản giống như các group trong Windows 2000. Trong SQL Server 2000 có 8 server role

Server Role	Thành viên của Server Role có thể …
sysadmin	Thực thi bất cứ thao tác nào trong một thể hiện SQL Server 2000 và trong bất kỳ CSDL. Mặc nhiên, tất cả các thành viên của nhóm Windows built-in Administrators, tài khoản người dùng sa thuộc vào server role này.
serveradmin	Cấu hình SQL Server 2000 bằng cách dùng thủ tục hệ thống sp_configure và có thể kết thúc các sevice. Các thành viên của nhóm điều hành viên built-in của Windows là rất tốt để nhận server role này.
setupadmin	Cài đặt và cấu hình linked server, remote server, và replication. Có thể chỉ định một stored procedure được thực thi lúc khởi động (startup), như là sp_serveroption. Các thành viên của nhóm điều hành viên built-in của windows là rất tốt nhận server role này.
securityadmin	Thực hiện tất cả các thao tác liên quan đến security trong SQL Server 2000, kể cả quản lý các quyền câu lệnh CREATE DATABASE, điều khiển server logins, và đọc error log. Giúp đỡ ở các nhân viên văn phòng. Thành viên của nhóm điều hành viên built-in của windows là rất tốt nhận server role này.
processadmin	Quản lý các tiến trình chạy các instance của SQL Server. Có thể ngắt (kill) tiến trình của các user, các truy vấn.
dbcreator	Có thể tạo, hiệu chỉnh, và xóa các CSDL. Những nhà quản trị CSDL lâu năm đảm trách server role này tốt.
Diskadmin	Có thể quản trị các tập và các thiết bị dự phòng. Nói chung Role này dùng để tương thích ngược với SQL Server 6.x.
bulkadmin	Có thể thực hiện các câu lệnh BULK INSERT. Cho phép các thành viên của sysadmin server role làm đại diện các tác vụ BULK INSERTmà không

Server Role	Thành viên của Server Role có thể …
	cần gán các quyền sysadmin. Hãy cẩn thận bởi vì các thành viên cũng phải truy xuất đọc đến bất kỳ dữ liệu được chèn và quyền INSERT trên bất kỳ bảng mà dữ liệu sẽ được chèn.

Lưu ý: Một thành viên của bất kỳ server role nào đều có thể thêm các user khác có server role đó.

15.1.4 Tìm hiểu các quyền (Permission) chỉ định trên cở sở dữ liệu.

Khi Truy xuất đến SQL Server 2000 thì bạn chưa có quyền truy xuất đến các CSDL. Ngoại trừ các thành viên trong sysadmin role, thành viên trong một server role có săn quyền truy xuất CSDL. Các quyền truy xuất CSDL phải được cấp một cách rõ ràng bởi một system administrator hoặc thành viên của administrator role trong CSDL. Các quyền có thể được cấp (grant), từ chói (deny), cởi bỏ (revoke) và bao gồm các quyền tạo đối tượng, quản trị CSDL, thực thi các câu lệnh T-SQL, chèn dữ liệu vào bảng, xem dữ liệu bằng view. SQL Server 2000 có một số cơ chế để cấp các quyền cụ thể cho các user trong một CSDL.

Các quyền cụ thể trên CSDL

Quyền	Mô tả
Database owner	User có thể được chỉ định như là chủ (owner) của CSDL và có thể thực hiện bất kỳ hành động liên quan đến CSDL.
DBO role	Tất cả các thành viên của sysadmin server role thì tự động là thành viên của dbo role trong mỗi CSDL, và có thể thực hiện bất kỳ thao tác liên quan đến CSDL.
User	Các user và group có thể được cấp user truy xuất đến CSDL theo tài khoản bảo mật của Windows 2000 hoặc SQL Server 2000. Sau đó một giấy phép người dùng CSDL được cấp các quyền trong CSDL thông qua database role, role chung, và chỉ định cấp các quyền câu lệnh và đối tượng.
Guest user	Một user mà có thể truy xuất đến 1 instance của SQL Server 2000 (nhưng người này không có tài khoản truy người dùng để truy xuất đến CSDL cụ thể) có thể được cho phép truy xuất đến CSDL như là một người khách (guest user). Tài khoản guest có thể được cấp các quyền cụ thể trong CSDL (đề đọc dữ liệu). Theo mặc nhiên, một CSDL không có tài khoản guest user.
Public role	Tất cả các user được phép truy xuất đến CSDL trở thành thành viên của public role trong mỗi CSDL. Public role có thể được cấp các quyền cụ thể (Tổng quát các quyền cần thiết cho tất cả các user của CSDL).
Fixed database role	Cho phép các user có thể được thêm vào các fixed database role trong một CSDL. Các Fixed database role chứa các quyền định trước trong CSDL để thi hành các hoạt động của database-wide.
User-defined database role	Cho phép các user có thể thêm vào user-defined database role trong một CSDL. Các role này có thể được tạo bởi administrator và cấp một cách cụ thể các quyền đưa ra hoặc các quyền trong CSDL.

Quyền	Mô tả
Statement permissions	Quyền thực thi các câu lệnh quản trị (như CREATE PROCEDURE) có thể được cấp, huỹ bỏ, từ chói các users, groups, và roles.
Object permissions	Quyền truy xuất đến các đối tượng CSDL (như là bảng hay view) có thể được cấp, huỹ bỏ, từ chối các users, groups, và roles.
Application role	Quyền thực thi các hành động trong một CSDL có thể được cấp cho một application, thậm chí cấp cho user. Một application kết nối đến một CSDL và kích hoạt application role. Các User truy xuất đến một CSDL thông qua sự kết nối này để dành lấy các quyền kết hớp với application role trong suốt quá trình kết nối. Các quyền phát hành đến một user cụ thể thì không liên quan khi user đang truy xuất CSDL thông qua application role.

15.1.5 Fixed Database Roles.

Mỗi CSDL có 9 role về CSDL được định nghĩa trước với các quyền kết hợp với database-wide để thực hiện các tác vụ khác nhau. Bạn không thể xóa những database role này và cũng không thể thay đổi các quyền của chúng. Để gán một người dùng các quyền này trong một CSDL, bạn thêm tài khoản của người dùng vào database role. Nếu những fixed database role này không gán tổng hợp các quyền mà bạn cần thì bạn có thể tạo các role với các quyền người dùng (thông thường nhiều quyền hạn chế).

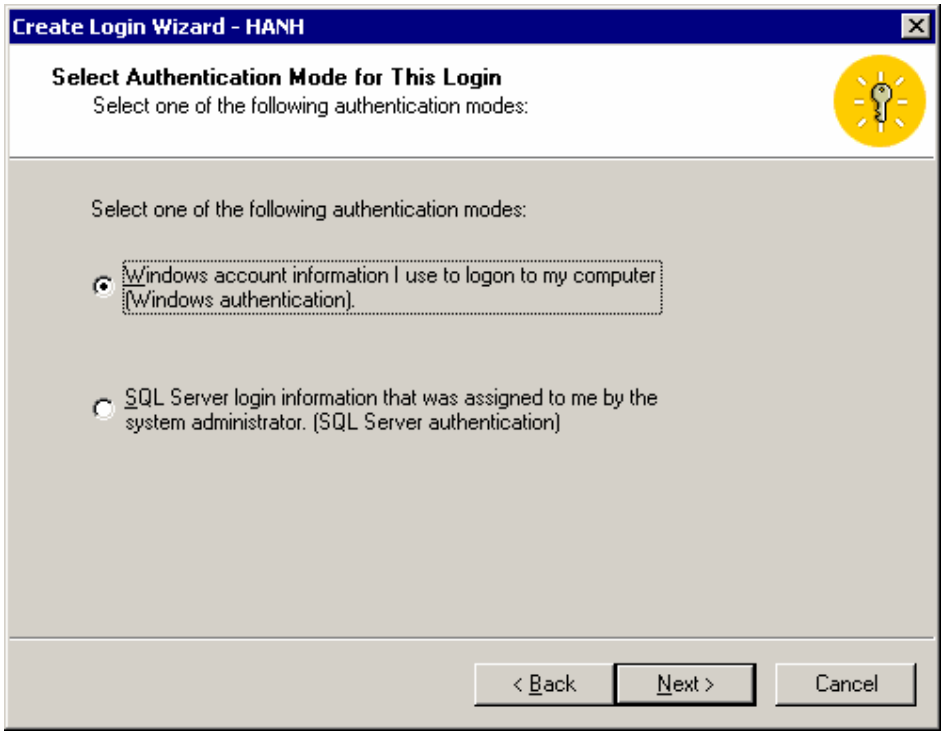
9 fixed database role có thể có trong SQL Server 2000.

Database Role	Thành viên của CSDL này có thể …
db_owner	Thực hiện bất kỳ tác vụ trong CSDL của SQL Server 2000. Các thành viên của role này có cùng quyền như là chủ của CSDL là các thành viên của dbo role.
db_accessadmin	Thêm hay xóa các user và group của Windows 2000 hoặc Win NT4.0 và các user trong một CSDL (dùng thủ tục hệ thống sp_grantdbaccess).
db_securityadmin	Quản lý tất cả các permission, role, role membership, và chuyển ower (ownership) trong một CSDL (sử dụng lệnh GRANT, REVOKE, và DENY).
db_ddladmin	Thêm, hiệu chỉnh, xóa các đối tượng trong CSDL (sử dụng lệnh CREATE, ALTER, và DROP).
db_backupoperator	Chạy các lệnh DBCC, phát hành checkpoint, và dự phòng CSDL (sử dụng các câu lệnh T-SQL: DBCC, CHECKPOINT, và BACKUP).
db_datareader	Đọc dữ liệu từ bất kỳ các bảng hoặc view của người dùng trong CSDL (bạn có quyền SELECT đối với tất cả table và view).
Db_datawriter	Hiệu chỉnh hoặc xóa dữ liệu từ các bảng hay view của người dùng trong CSDL (bạn phải có quyền INSERT, UPDATE, và DELETE đối với tất cả các table và view).
Db_denydatareader	Không đọc dữ liệu từ bất kỳ bảng trong CSDL (bạn không có quyền SELECT đối với bất kỳ đối tượng). Có thể permission on any objects). Có thể được sử dụng với role db_ddladmin để cho phép tạo các đối tượng làm chủ bằng dbo role, nhung không có thể đọc nhạy cảm chứa trong các đối tượng đó.
Db_denydatawriter	Không hiệu chỉnh hay xóa dữ liệu từ các bảng của người dùng trong CSDL (bạn không có quyền INSERT, UPDATE, và DELETE đối với các đối tượng)

15.2

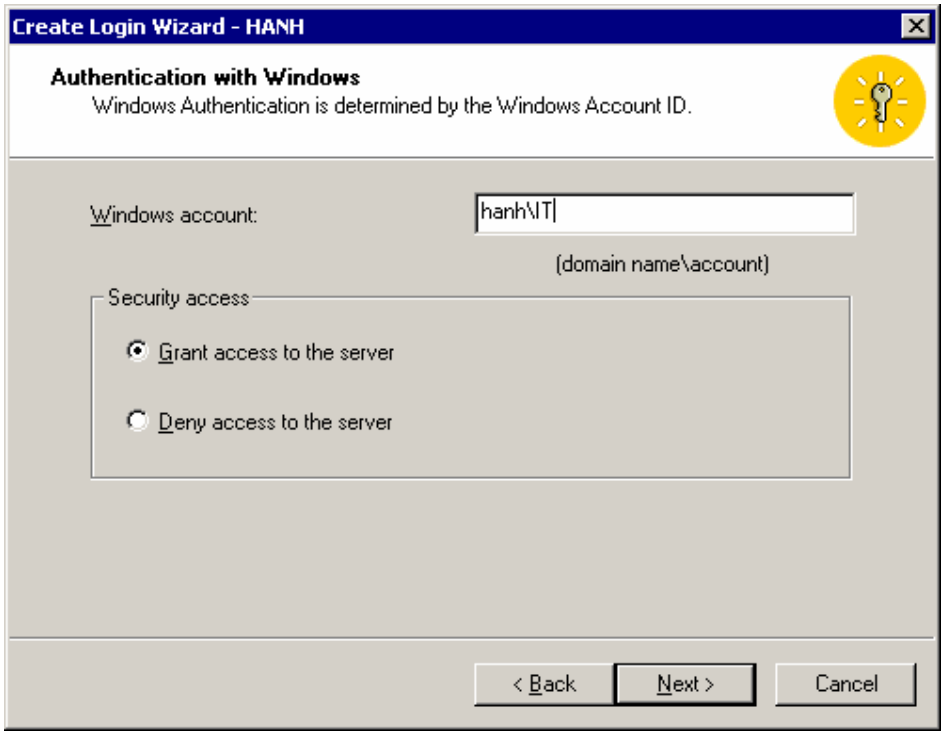
15.3 Tạo tài khoản đăng nhập (Login).

15.3.1 Dùng Create Login Wizard.



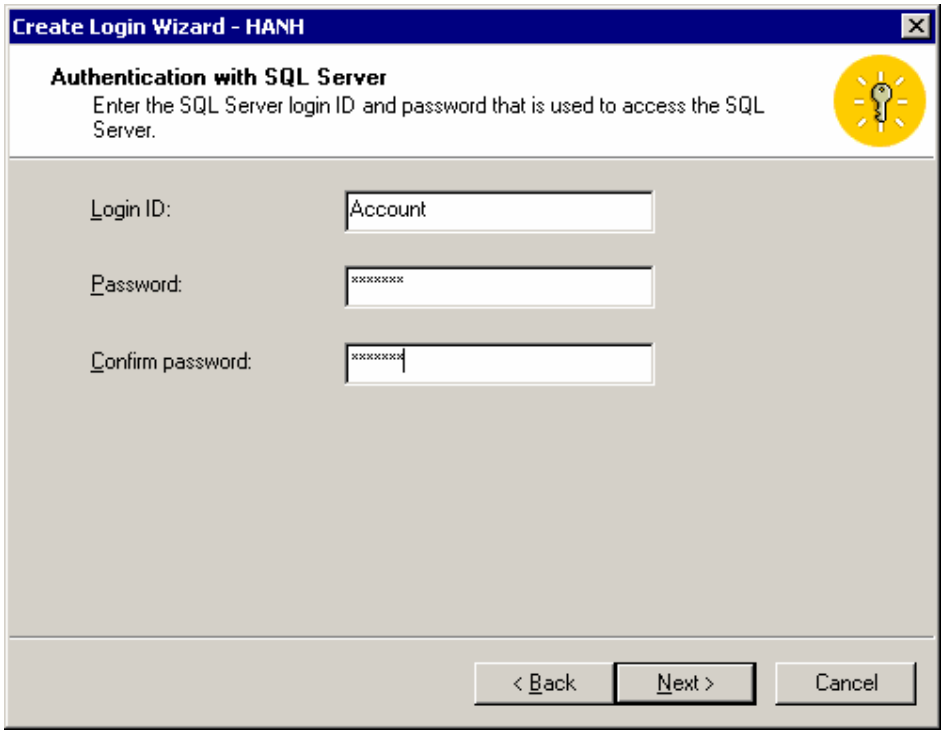
Hình 40: Chọn chế độ đăng nhập cho login đang tạo

Nếu bạn chọn Windows authentication thì bạn phải liên kết login ID này với một user hay group của Windows 2000 có sẵn. Khai báo như trong hình sau



Hình 41: Xác định 1 tài khoản của Windows 2000 và xác định cho phép hoặc từ chối login mới truy xuất đến Server

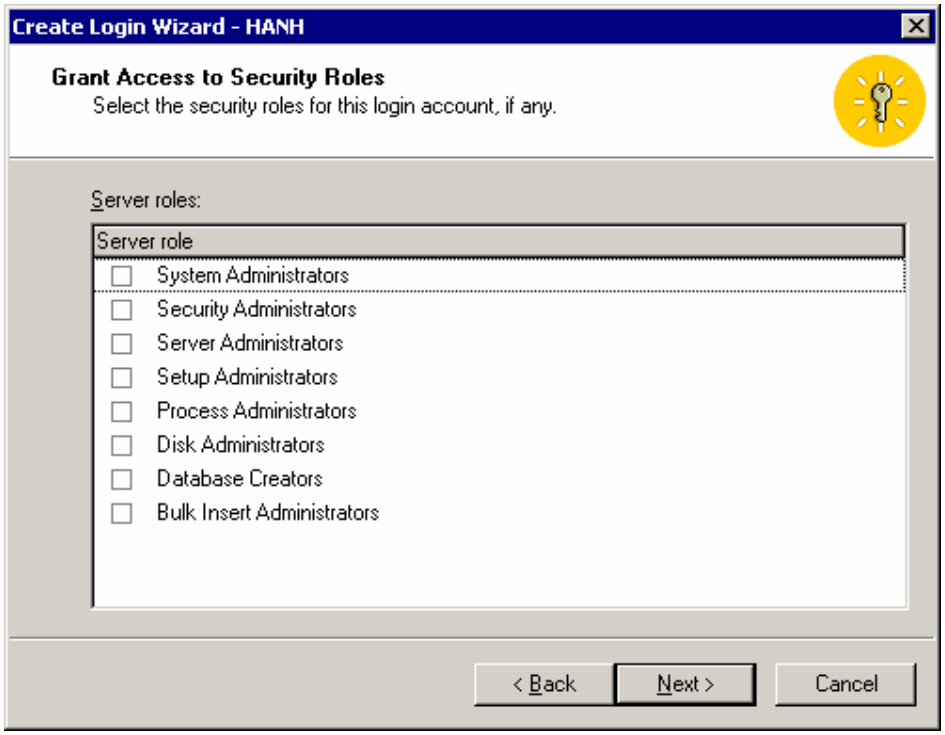
Nếu bạn chọn SQL Server authentication, bạn sẽ tạo một tài khoản bảo mật của SQL Server 2000 như trong hình kế.



Hình 42: Khai báo LogiID và Password

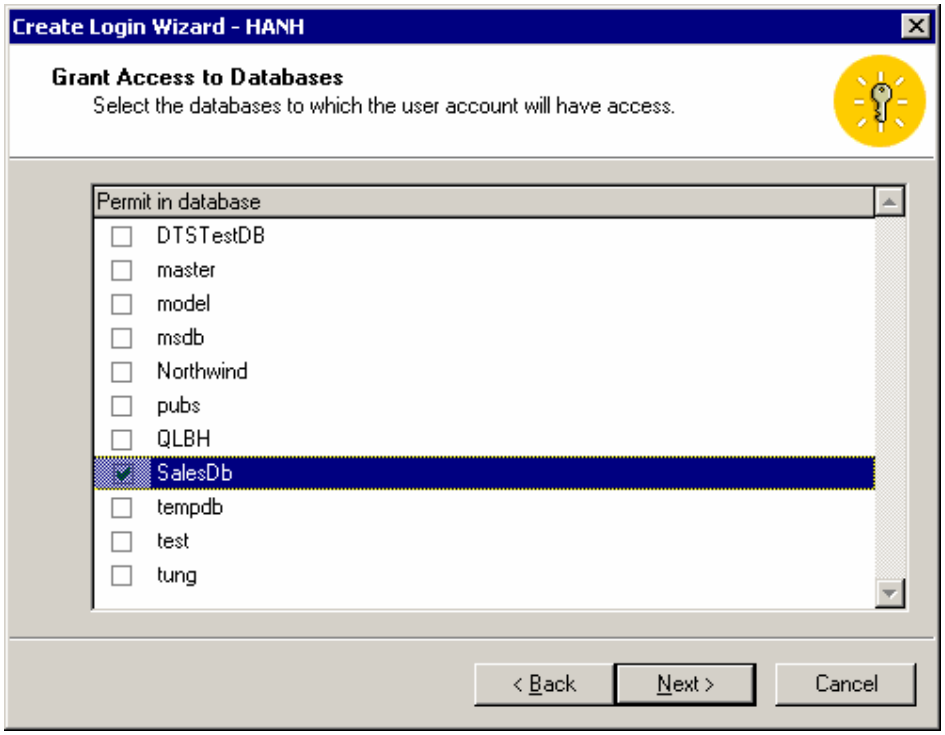
Phải chỉ định rõ tên login và Password. Để ngăn chặn (deny) một login, đơn giản bạn chỉ cần xóa bỏ login từ nơi chứa login trong Enterprise Manager (hoặc từ bảng sysxlogins trong CSDL master).

Sau khi bạn xác định kiểu đăng nhập và liên kết hay tạo tài khoản bảo mật, bạn chỉ định server role (nếu cần) cho login này trong hộp thoại kế. Nếu user sẽ không là một server-wide administrator, thì không cần chọn các server role.



Hình 43: Chỉ định server role cho login đang được tạo

Kế tiếp bạn chỉ định CSDL (nếu cần) để cho user này có thể truy xuất trong hộp thoại kế tiếp. Nhớ rằng hầu hết các server role không cung cấp CSDL truy xuất



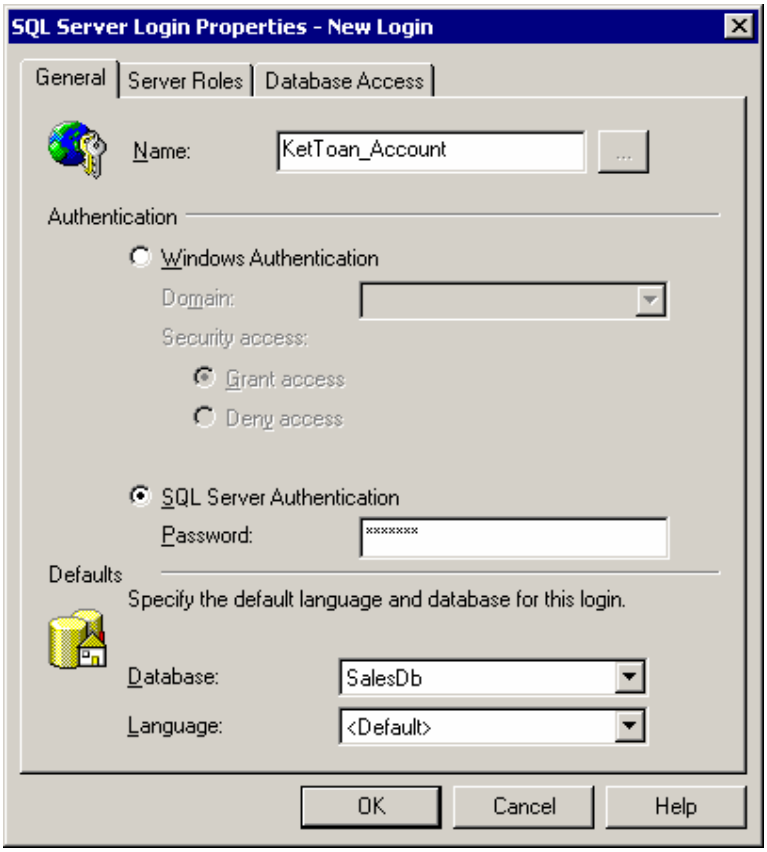
Hình 44: Chỉ định 1 hay nhiều CSDL để login này được truy xuất

Cuối cùng, bạn được SQL Server cho xem trước các lựa chọn mà bạn đã thực hiện trước khi login mới được tạo thực sự. Click vào nút Finish để tạo login.

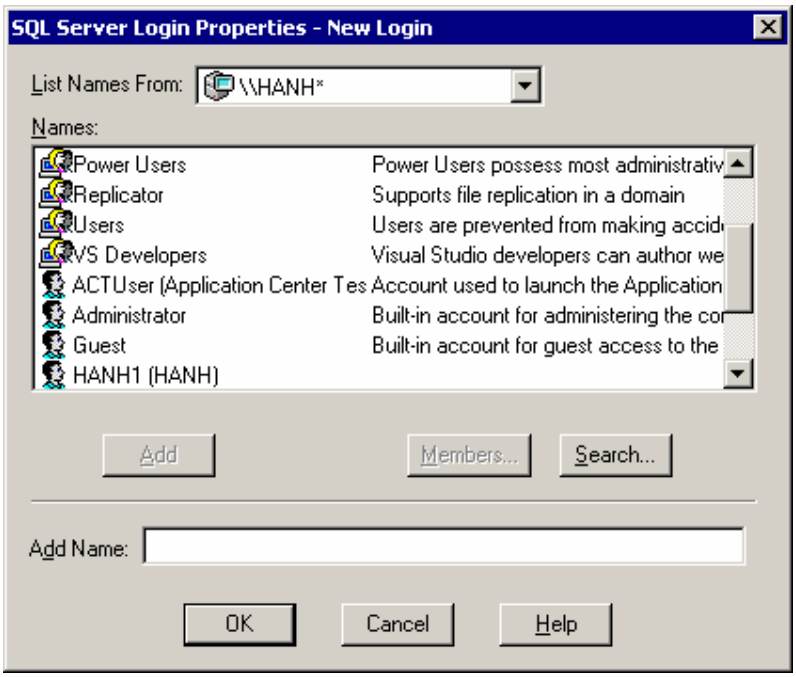
Hình 45: Chọn nút Finish để hoàn tất

15.3.2 Dùng Enterprise Manager để tạo một Login.

Để tạo một login bằng Enterprise Manager, nhấp nút phải chuột tại Security/Login của 1 instance, và chọn New Login. Thông thường, các trang Server Role, và Database Access trong hợp thoại Propertie của login



Hình 46: Chỉ định Name, chế độ chứng thực, chọn CSDL mặc định

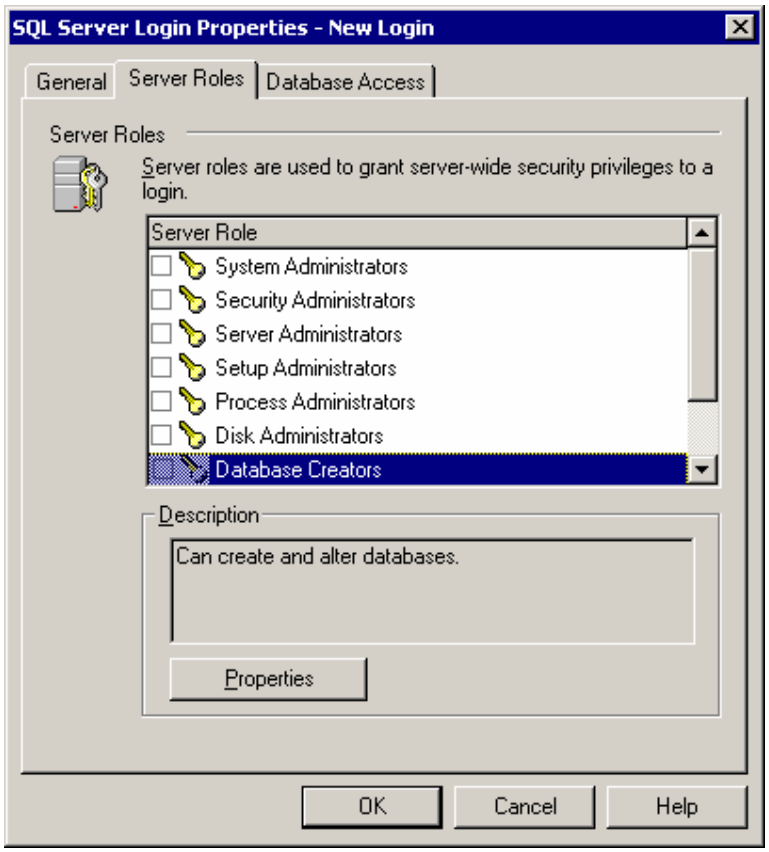


Hình 47: Chọn tài khoản người dùng của Windows

Bạn cũng chọn một CSDL mặc định và ngôn ngữ trong trang General. CSDL mặc định sẽ là CSDL hiện hành khi mà User đăng nhập vào. Mặc định không chọn là CSDL Master nhưng bạn nên chọn 1 CSDL khác làm mặc định. Ngôn ngữ mặt định là ngôn ngữ mặc định của instance hiện hành.

Lưu ý:

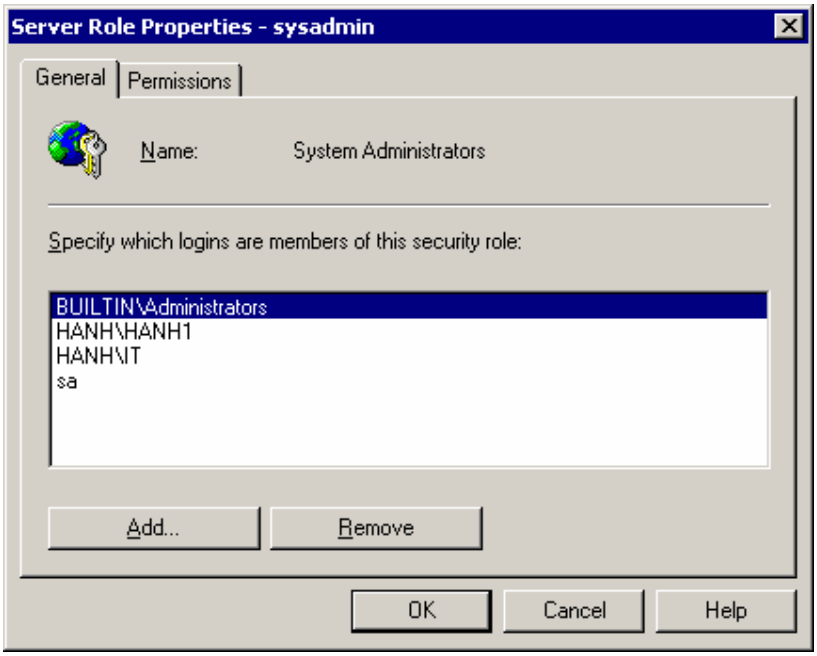
Bạn cũng có thể từ chối sự truy xuất của một User hay group của Windows. Điều này được áp đặt đến việc truy xuất khác của user hay group (kể cả thành viên của một nhóm khác mà có login khác).



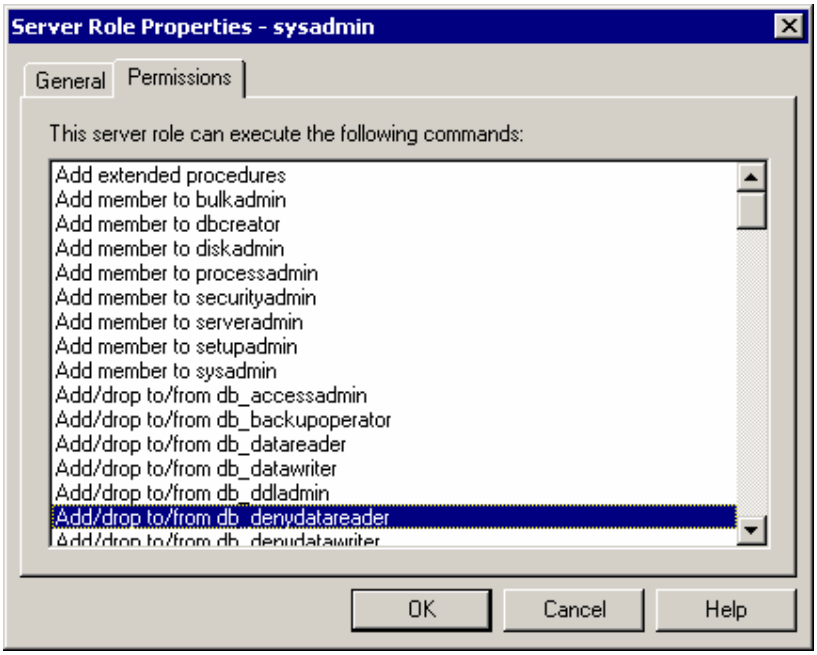
Hình 48: Gán login thuộc server role nào đó nếu cần

Lưu ý rằng nếu System Administrators server role được chọn, thì bạn có thể xem nhóm built-in Administrator group, và login sa của the SQL Server là những thành viên của sysadmin server role.

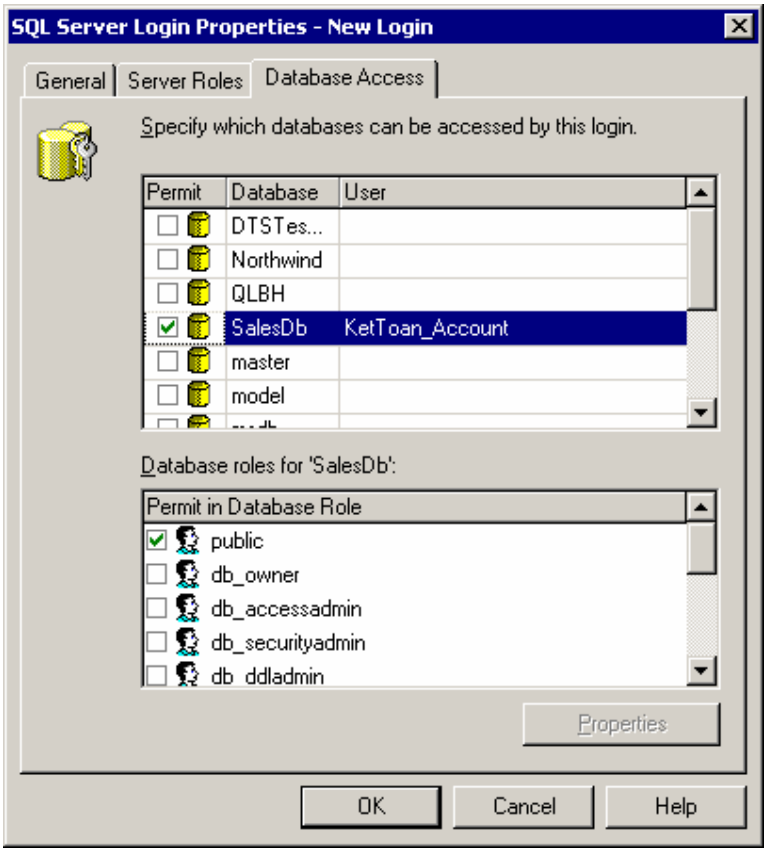
Chọn nút properties để xem một số propererties của Server role



Hình 49:Thành viên của server role



Hình 50: Các quyền hạn cụ thể của Server role



Hình 51: Chỉ định CSDL được truy xuất tới và các quyền cụ thể cho login

Khi bạn chọn 1 CSDL thì login là thành viên của public role trong mỗi CSDL. Bạn không thể xóa (remove) một login ra khỏi public role.

15.3.3 Tạo Login bằng T-SQL.

Bạn cũng có thể tạo các Login bằng T-SQL

Windows Logins

Các thủ tục hệ thống dùng để cấp, hũy, từ chối, hiệu chỉnh một login cho một user hay group của Windows. Chỉ có những thành viên của sysadmin hoặc securityadmin server roles mới có thể thực thi các thủ tục hệ thống này

Thủ tục hệ thống	Mô tả
Sp_grantlogin *'login'*	Tạo một login cho một user hay group của Windows 2000.
Sp_revokelogin *'login'*	Hũy login từ SQL Server đối với các user hay group của Windows 2000 (hoặc Windows NT 4.0). Điều này không rõ ràng lắm vì nó vẫn có thể được cấp quyền truy xuất đến SQL Server 2000.
Sp_denylogin *'login'*	Ngăn chặn một user hay thành viên của Windows 2000 (hoặc Windows NT 4.0) kết nối đến SQL Server 2000. Ngăn chặn các user hay group của thôgn qua một login khác kết với user hay group của Windows.
Sp_defaultdb *'login'*, *'database'*	Thay đổi CSDL mặc định cho một login.

Thủ tục hệ thống	Mô tả
Sp_defaultlanguage *'login', 'language'*	Thay đổi ngôn ngữ mặc định của một login.

Ví dụ:

```
Sp_grantlogin 'CDCN4\Bill'
```

SQL Server Logins

Các thủ tục hệ thống sau cho phép cấp, hũy, từ chối, hiệu chỉnh một login kết với một tài khoản người dùng SQL Server. Chỉ có các thành viên của sysadmin hoặc securityadmin server roles mới có thể thực thi các thủ tục hệ thống này

Thủ tục hệ thống	Mô Tả
Sp_addlogin *'login'*, ['password', 'database', 'language', 'sid', encryption_option']	Tạo một login SQL Server mới. Password là NULL nếu không chỉ định. CSDL mặc định là master nếu không chỉ định. Ngôn ngữ mặc định là ngôn ngữ của server hiện hành nếu không chỉ định. Mặc định, password ở trong csdl master.
Sp_droplogin *'login'*	Xóa một SQL Server login.
Sp_password 'old_password', 'new_password', 'login'	Thêm hoặc thay đổi password cho SQL Server login.
Sp_defaultdb 'login', 'database'	Thay đổi CSDL mặc định
Sp_defaultlanguage *'login', 'language'*	Thay đổi ngôn ngữ mặc định.

Ví dụ:

```
Sp_addlogin 'Joe', 'Joe123', 'Northwind'
```

Tạo một SQL Server login mới có tên là **Joe**, với password là **Joe123** và CSDL mặc định Northwind.

Server Roles

Các thủ tục hệ thống sau được dùng để thêm hay xóa một login vào một server role. Chỉ có thành viên của sysadmin server role mới có thể thêm các login vào bất kỳ server role. Các thành viên của server role cũng có thể thêm các login vào server role đó.

Thủ tục hệ thống	Mô tả
Sp_ addsrvrolemember *'login', 'role'*	Thêm login như là tàhnh viên của server role.
Sp_dropsrvrolemember *'login', 'role'*	Xoá login không là thành viên của một server role.

Ví dụ:

```
Sp_addsrvrolemember 'Joe', 'securityadmin'
```

Thêm login có tên là Joe vào server role Security Administrator.

Database Access

Các thủ tục sau đây được dùng để thêm hay xóa một login (Windows hoặc SQL Server) hiện hữu được quyền truy xuất trong CSDL hiện hành. Không giống như SQL Server Enterprise Manager, bạn có thể cấp một nhóm của Windows 2000 (hoặc Windows NT 4.0) group truy xuất đến CSDL mà không cần tạo login trước một cách tường minh trong

bảng sysxlogins. Chỉ có các thành viên của sysadmin server role, và db_accessadmin và db_owner fixed database role mới có thể thực thi các thủ tục hệ thống này.

Thủ tục hệ thống	Mô tả
Sp_grantdbaccess 'login', 'name_in_db'	Thêm một login như là một user trong CSDL hiện hành. Mặc dù tên user name trong CSDL có thể khác với tên login, điều này không khuyến cáo.
Sp_revokedbaccess *'name'*	Bỏ một login như là một user trong CSDL hiện hành.

Ví dụ:

```
USE Northwind
EXEC Sp_grantdbaccess 'Joe'
```

Cho phép login tên là Joe truy xuất đến CSDL hiện hành, dùng user name là Joe trong CSDL Northwind.

Database Roles

Các thủ tục hệ thống sau đây được dùng để thay đổi database owner, thêm hoặc xóa một tài khoản bảo mật vào một database role có sẵn, hoặc tạo hoặc xóa một user-defined database role.

```
Use Northwind
EXEC Sp_addrolemember 'db_securityadmin', ' CDCN4\KeToan'
```

Thêm tài khoản CDCN4\KeToanvào db_securityadmin database role trogn CSDL Northwind.

Thủ tục hệ thống	Mô tả
Sp_changedbowner 'login', remap_alias_flag	Thay đổi owner của một CSDL người dùng. Chỉ có những thành viên của sysadmin server role hoặc owner database hiện hành mới có thể thau đổi owner của CSDL.
Sp_addrolemember 'role', 'security_account'	Thêm một tài khoản vào một database role trong CSDL hiện hành. Bạn có thể thêm một user-defined database role vào fixed hoặc user-defined database role. Chỉ có những thành viên của sysadmin server role và db_owner and db_security fixed database roles mới có thể thêm thành viên vào database role. Thành viên của database role có thể thành viên vào cho database role đó.
Sp_droprolemember 'role', 'security_account'	Xóa một tài khoản từ một CSDL vào CSDL hiện hành. Chỉ có những thành viên của sysadmin server role và db_owner và db_security fixed dababase roles mới có thể xóa các thành viên ra khỏi database role. Các thành viên của database role mới có thể xóa các thành viên ra khỏi database role.
Sp_addrole 'role', 'owner'	Thêm một user-defined database role mới trong CSDL hiện hành. Mặc dù bạn có thể chỉ định một owner của role, sử dụng mặc định là dbo là không được khuyến cáo. Các thành viên của sysadmin server role và db_securityadmin và db_owner fixed database roles mới có thể tạo user-defined database roles.
Sp_droprole *'role'*	Xóa một user-defined database role ở CSDL hiện hành. Các tàhnh

Thủ tục hệ thống	Mô tả
	viên của sysadmin server role, db_securityadmin và db_owner fixed database roles mới có thể xóa user-defined database roles.

Để xem các thông tin về login, dùng các thủ tục hệ thống sau:

Thủ tục hệ thống	Mô tả
Sp_helplogins [*'login'*]	Trả về các thông tin của tất cả các login hoặc một login được chỉ định, kể cả các CSDL mà login có truy xuất đến và các database roles mà login là thành viên.
Sp_helpsrvrolemember [*'role'*]	Trả về thông tin về tất cả các server roles và những thành viên của chúng hoặc tất cả các thành viên trong một server role chỉ định.
Sp_helpuser ['security_account']	Trả về thông tin về tất cả các user hoặc user chỉ định trong CSDL hiện hành, kể cả tất cả các hội thành viên của database role.
Sp_helprolemember [*'role'*]	Trả về các thông tin của tất cả các database role hoặc tất cả các thành viên trong database role chỉ định trong CSDL hiện hành.
Sp_helpntgroup [*'name'*]	Trả về các thông tin về các nhóm hoặc 1 nhóm chỉ định của Windows 2000 (hoặc Windows NT 4.0) trong CSDL hiện hành.

Gán các quyền Database

Các view và các stored procedure có thể được tạo trên các table. Khi một user cố gắng lấy thông tin thông qua view hoặc procedure, thì SQL Server 2000 phải kiểm tra user có được phép lấy dữ liệu hay không. Nếu view hoặc procedure được làm chủ bởi một user và của các bảng cơ sở lại là của một user khác thì SQL Server 2000 phải kiểm tra các quyền trên mỗi object trong dây chuyền đó. Khi một chuỗi các ownership kéo dài thì điều này sẽ ảnh hưởng đến việc thực hiện. Nhưng có lẽ quan trọng hơn là nó có thể bị gắt gối cho nhà quản trị vạch ra và gỡ lỗi các từ việc đưa ra bảo mật

Các quyền về lệnh

Câu lệnh Transact-SQL	Quyền để thực thi câu lệnh Transact-SQL
CREATE DATABASE	Thừa kế bởi thành viên của sysadmin và dbcreator server roles. Mặc dù sysadmin và securityadmin server roles có thể cấp quyền một cách trực tiếp cho các tài khoản để thực hiện câu lệnh này, Tóm lại các tài khoản bảo mật sử dụng dbcreator server role nếu system administrator đại diện quyền. Quyền này chỉ tồn tại trong CSDL master.
BACKUP DATABASE BACKUP LOG	Kế thừa bởi các thành viên của sysadmin server role và db_owner và db_backupoperator fixed database roles. Mặc dù bạn có thể cấp quyền để chạy những câu lệnh này một cách trực tiếp đến các tài khoản bảo mật, một cách tổng quát bạn sẽ sử dụng db_backupoperator fixed database role.

Câu lệnh Transact-SQL	Quyền để thực thi câu lệnh Transact-SQL
CREATE TABLE CREATE VIEW CREATE PROCEDURE CREATE DEFAULT CREATE RULE CREATE FUNCTION	Kế thừa các thành viên của sysadmin server role và db_owner và db_ddladmin fixed database roles. Quyền cho phép tạo những đối tượng thì được cấp trực tiếp đến nhà lập trình trong suốt thời gian triển khai. Theo mặc định các object được làm chủ bởi người tạo ra đối tượng (mặc dù các đối tượng tạo bởi các thành viên của sysadmin server role thì chủ sẽ là dbo role). Các thành viên của db_owner hoặc db_ddladmin fixed database roles có thể được chỉ định dbo role như là owner của đối tượng được tạo. Ngoài ra, các thành viên của sysadmin server role hoặc db_owner hoặc db_ddladmin fixed database role có thể chỉ định bất kỳ user như là chủ của object mà chúng tạo ra. Tuy nhiên, các user mà không là thành viên của một trong các role này thì không thể chỉ định user khác hoặc dbo role làm chủ của object chúng tạo ra.
CREATE TRIGGER	Kế thừa bởi chủ của table mà của sysadmin server role, và db_owner and db_ddladmin fixed database roles. Những thành viên này không thể cấp quyền để chạy câu lệnh này cho những tài khoản bảo mật khác..

Ví dụ:

```
CREATE TABLE Northwind.dbo.CustomerTable
 (CustID nchar (5), CustomerName nvarchar (40))
```

Cho phép tạo một bảng, cấp ownership cho dbo role. Chỉ có những thành viên sysadmin server role và db_owner hoặc db_ddladmin fixed database roles có thể thực hiện một cách thành công câu lệnh này.

Thay đổi Ownership của Object

Một thành viên của db_owner, db_ddladmin, hoặc db_securityadmin fixed database role, hoặc a member of the sysadmin server role có thể thay đổi ownership của bất kỳ object trong bằng cách chạy thủ tục sp_changeobjectowner.

```
sp_changeobjectowner 'CDCN4\KeToan.Customer', 'dbo'
```

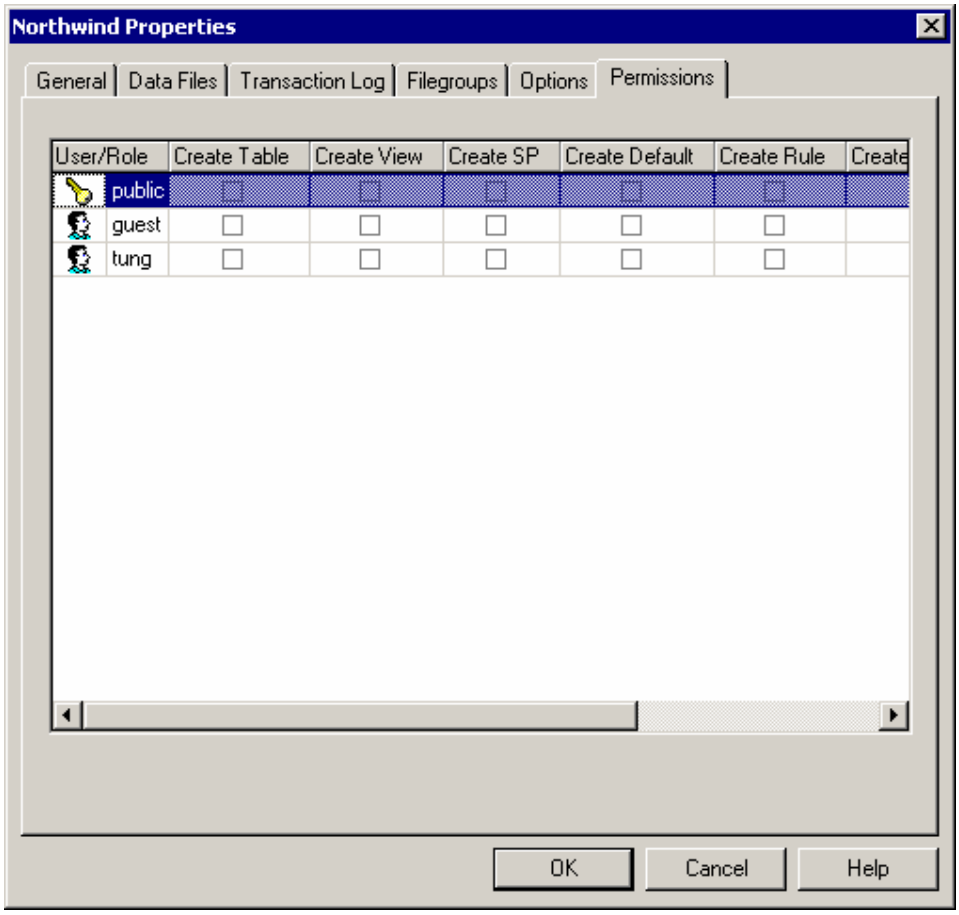
Đổi ownership của table Customer từ CDCN4\KeToanBill cho dbo role.

Lưu ý

Thay đổi owner của một object thì sẽ xó tất cả các quyền hiện có trên đối tượng. Nếu bạn cần giữ lại các quyền thì nên tạo Scrip trước khi đổi owner.

Dùng Enterprise Manager đề Grant, Deny, or Revoke quyền lệnh

Nhắp nút phải chuột tại CSDL cần thực hiện cấp/xoá/từ chối/xem các quyền



Hình 52: Trang Permission của hộp thoại thuộc tính của CSDL

Dùng Transact-SQL để Grant, Deny, or Revoke các quyền

Dùng câu lệnh GRANT CREATE TABLE TO ….

```
GRANT CREATE TABLE TO Joe, SalesManagers,
      [CDCN4\SQLServerAdmins]
```

Cấp quyền lệnh CREATE TABLE cho Joe (một SQL Server login), SalesManagers (a user-defined database role), và CDCN4\SQLServerAdmins (một Windows group).

Xem các quyền bằng Transact-SQL : Dùng sp_helprotect

```
EXEC sp_helprotect NULL, NULL, NULL, 's'
```

Xem tất cả các quyền câu lệnh trong CSDL hiện hành.