Mục lục

1 Giới thiệu về SQL Server 2005

SQL Server 2005 là một hệ thống quản lý cơ sở dữ liệu (Relational Database Management System (RDBMS)) sử dụng Transact-SQL để trao đổi dữ liệu giữa Client computer và SQL Server computer. Một RDBMS bao gồm databases, database engine và các ứng dụng dùng để quản lý dữ liệu và các bộ phận khác nhau trong RDBMS.

SQL Server 2005 được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến Tera-Byte và có thể phục vụ cùng lúc cho hàng ngàn user. SQL Server 2005 có thể kết hợp "ăn ý" với các server khác như Microsoft Internet Information Server (IIS), E-Commerce Server, Proxy Server....

Các phiên bản của SQL Server 2005:

Enterprise: Hỗ trợ không giới hạn số lượng CPU và kích thước Database. Hỗ trợ không giới hạn RAM (nhưng tùy thuộc vào kích thước RAM tối đa mà HĐH hỗ trợ) và các hệ thống 64bit.

Standard: Tương tự như bản Enterprise nhưng chỉ hỗ trợ 4 CPU. Ngoài ra phiên bản này cũng không được trang bị một số tính năng cao cấp khác.

Workgroup: Tương tự bản Standard nhưng chỉ hỗ trợ 2 CPU và tối đa 3GB RAM

Express: Bản miễn phí, hỗ trợ tối đa 1CPU, 1GB RAM và kích thước Database giới hạn trong 4GB.

Chi tiết có thể tham khảo tại địa chỉ:

http://www.microsoft.com/sql/prodinfo/features/compare -features.mspx

1.1 Cài đặt SQL Server 2005 Express Edition

1.1.1 Các yêu cầu cho hệ thống 32bit

Express Edition System Requirements	
32-bit	
Processor	PIII 600MHZ hoặc cao hơn Tốt nhất: 1GHZ hoặc cao hơn
Framework	Microsoft .NET Framework 2.0
Operating System	• Windows XP with Service Pack 2 hoặc cao hơn • Microsoft Windows 2000 Professional SP4

Express Edition System Requirements	
	32-bit
	• Microsoft Windows 2000 Server Service Pack 4 hoặc cao hơn • Windows Server 2003 Standard, Enterprise, or Datacenter editions with Service Pack 1 hoặc cao hơn • Windows Server 2003 Web Edition SP1 • Windows Small Business Server 2003 with Service Pack 1 hoặc cao hơn • Vista Home Basic và các phiên bản cao hơn (SQL Express SP1 and SQL Express Advanced SP2) • Windows XP Embedded SP2 Feature Pack 2007 • Windows Embedded for Point of Service SP2
Memory	192 MB RAM hoặc cao hơn; tốt nhất: 512 MB hoặc cao hơn
Hard Disk	• 350 MB ổ cứng cho các cài đặt cơ bản • 425 MB ổ cứng cho các cài đặt SQL Server Books Online, SQL Server Mobile Books Online, và sample databases
Drive	CD-ROM or DVD-ROM drive
Display	Super VGA (1,024x768) hoặc cao hơn
Other Devices	Mouse, Keyboard
Other Requirements	Microsoft Internet Explorer 6.0 SP1 hoặc cao hơn

Chi tiết yêu cầu hệ thống cho các phiên bản Microsoft SQL Server 2005 có thể tham khảo tại địa chỉ:

http://www.microsoft.com/sql/prodinfo/sysreqs/default.mspx

Download và cài đặt Microsoft .NET Framework 2.0: Để cài đặt thành công SQL Server Express Edition hay các phiên bản SQL Server 2005 khác, Microsoft .NET Framework 2.0 phải được cài đặt trước.

Gỡ bỏ các phiên bản Beta, CTP hoặc Tech Preview của SQL Server 2005, Visual Studio 2005 và Microsoft .NET Framework 2.0.

Download và cài đặt

Cài đặt SQL Server 2005 Express Edition: Microsoft SQL Server 2005 Express Edition l à phiên bản miễn phí, dễ sử dụng và "nhẹ" của Microsoft SQL Server 2005. Microsoft SQL Server 2005 Express Edition đư ợc tích hợp trong Visual Studio 2005 tạo ra sự dễ d àng trong việc phát triển các ứng dụng h ướng CSDL. SQL Server 2005 Express Edition đư ợc tự do sử dụng trong các ứng dụng th ương mại và dễ dàng cập nhật lên các phiên bản cao hơn khi cần thiết.

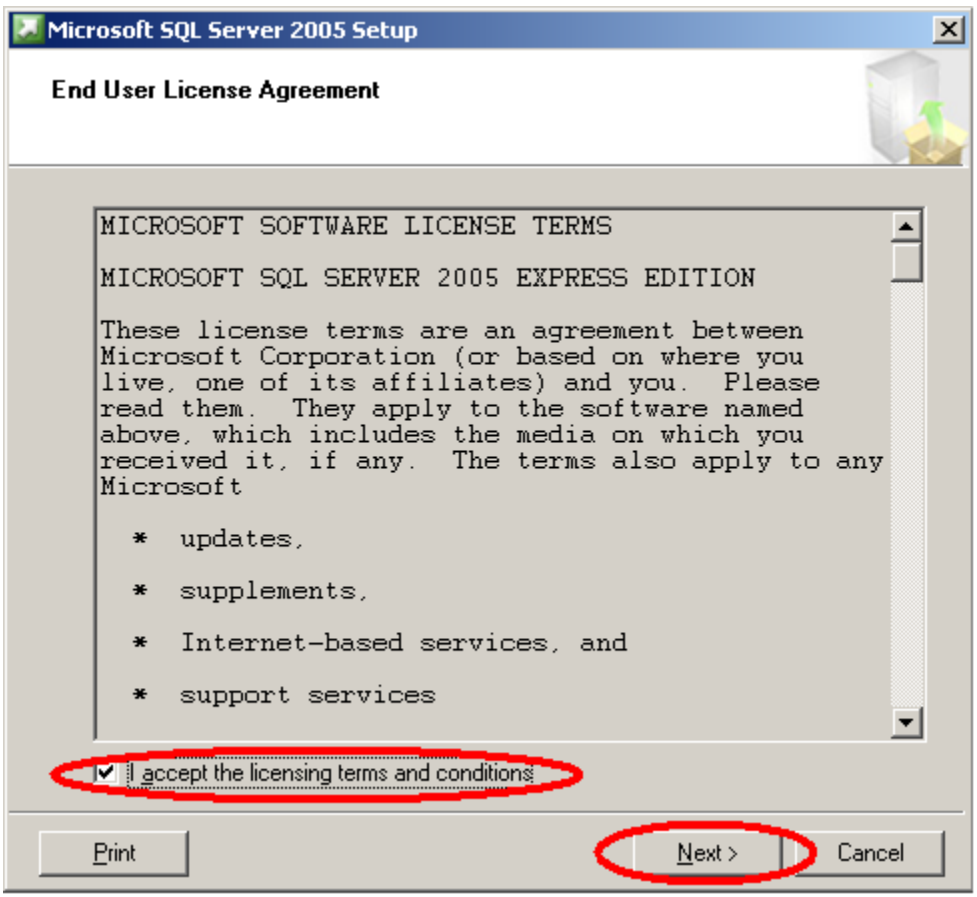
Cài đặt SQL Server Management Studio Express: SQL Server Managemen t Studio Express cung cấp giao diện để người dùng dễ dàng tương tác với các thành phần của Microsoft SQL Server 2005 Express Edition. Trước khi cài đặt SQL Server Management Studio Express, MSXML 6.0 phải được cài đặt

Download tại địa chỉ:

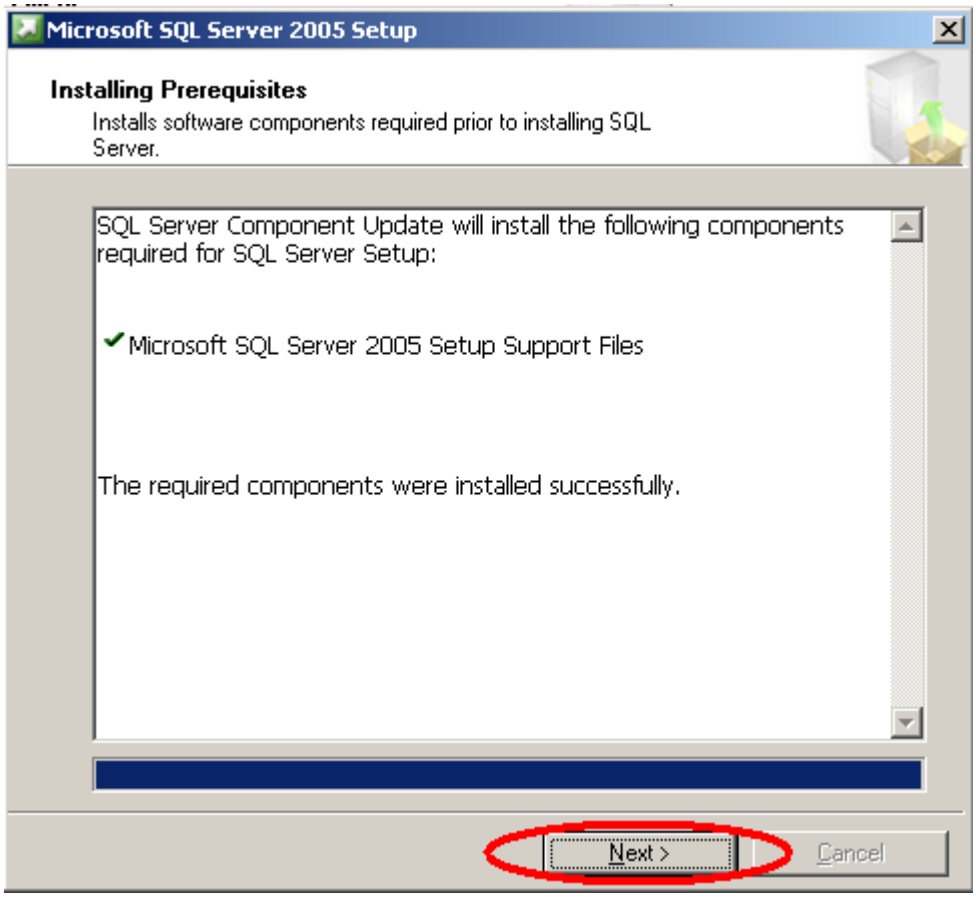
http://www.microsoft.com/express/sql/download/default.aspx

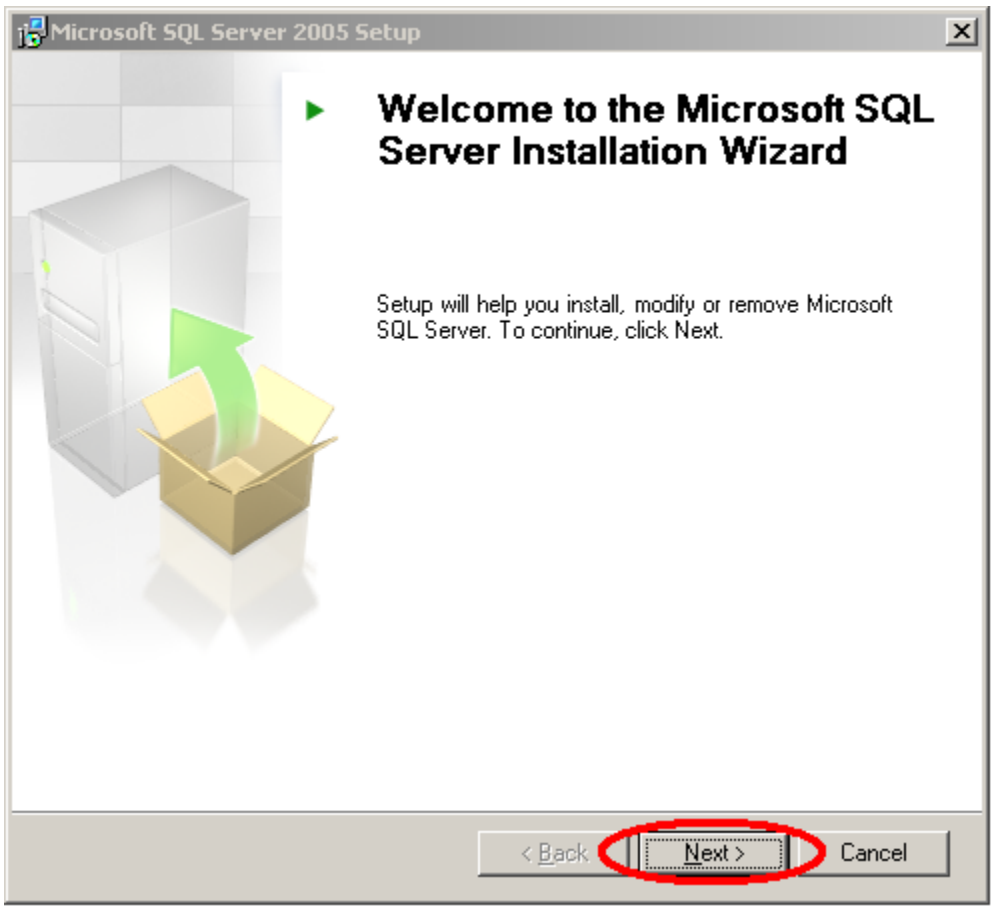
1.1.2 Các bước cài đặt SQL Server 2005 Express Edition

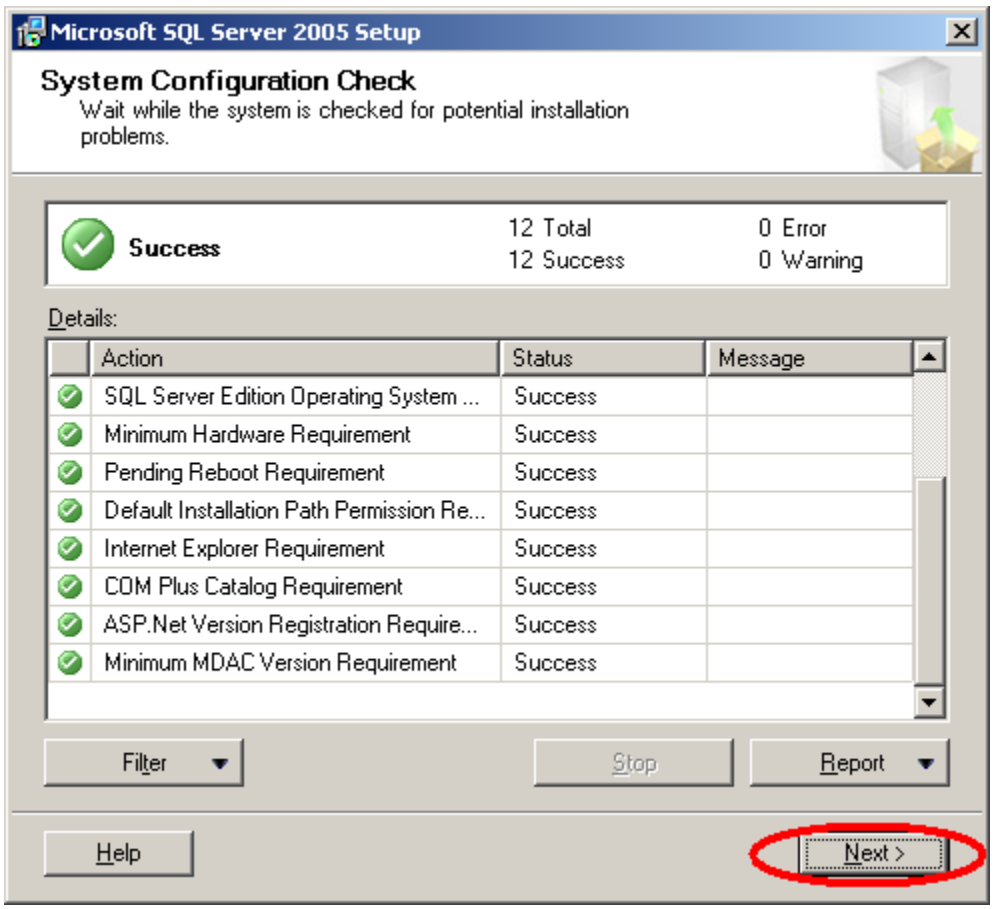
Double click vào file cài đặt Microsoft SQL Server Express Edition .

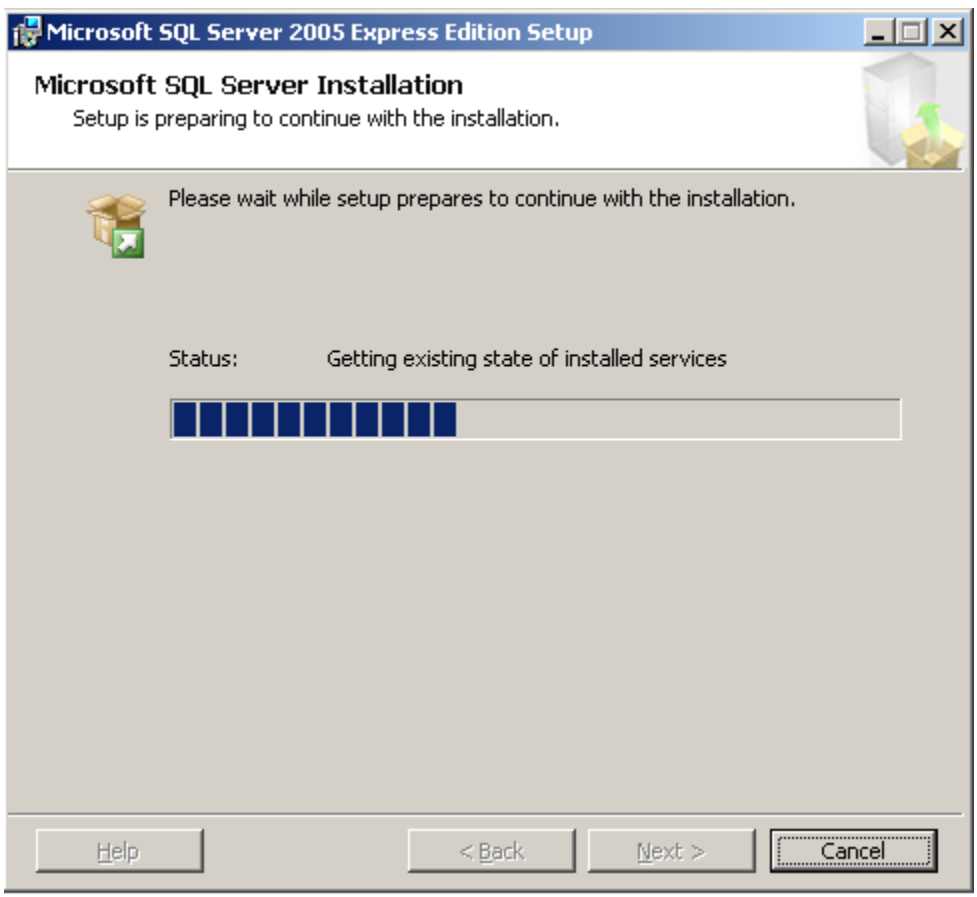


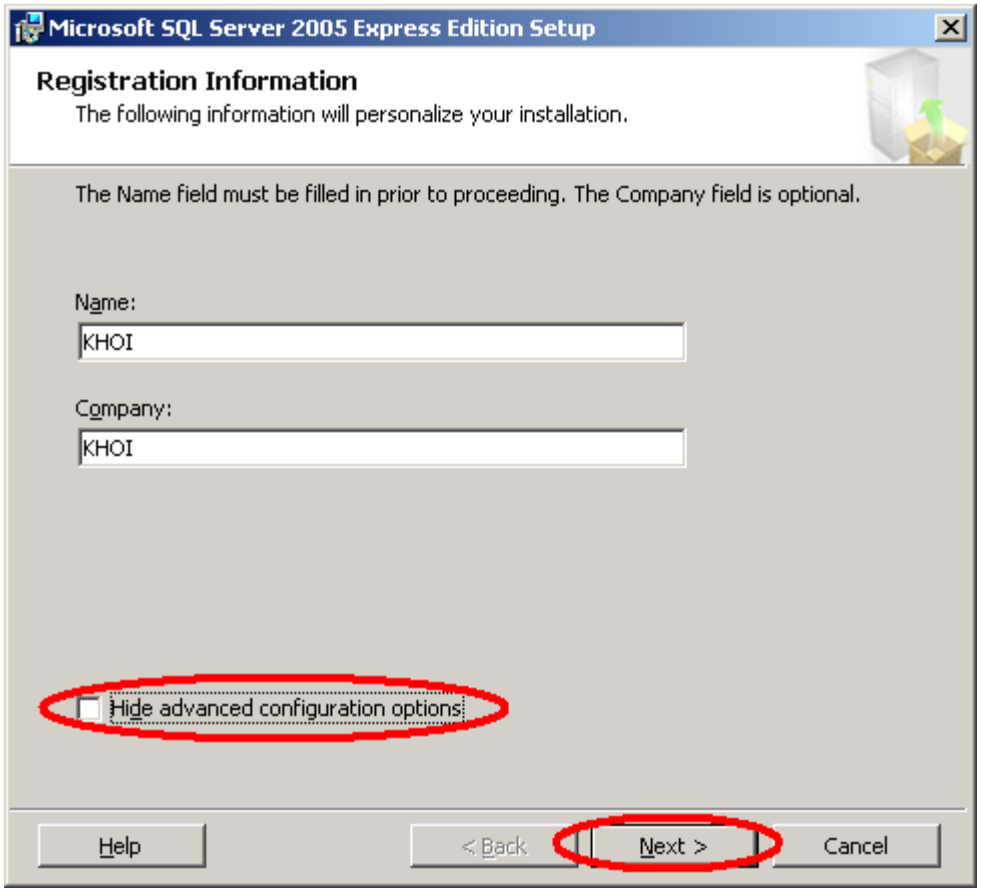
Click Next:

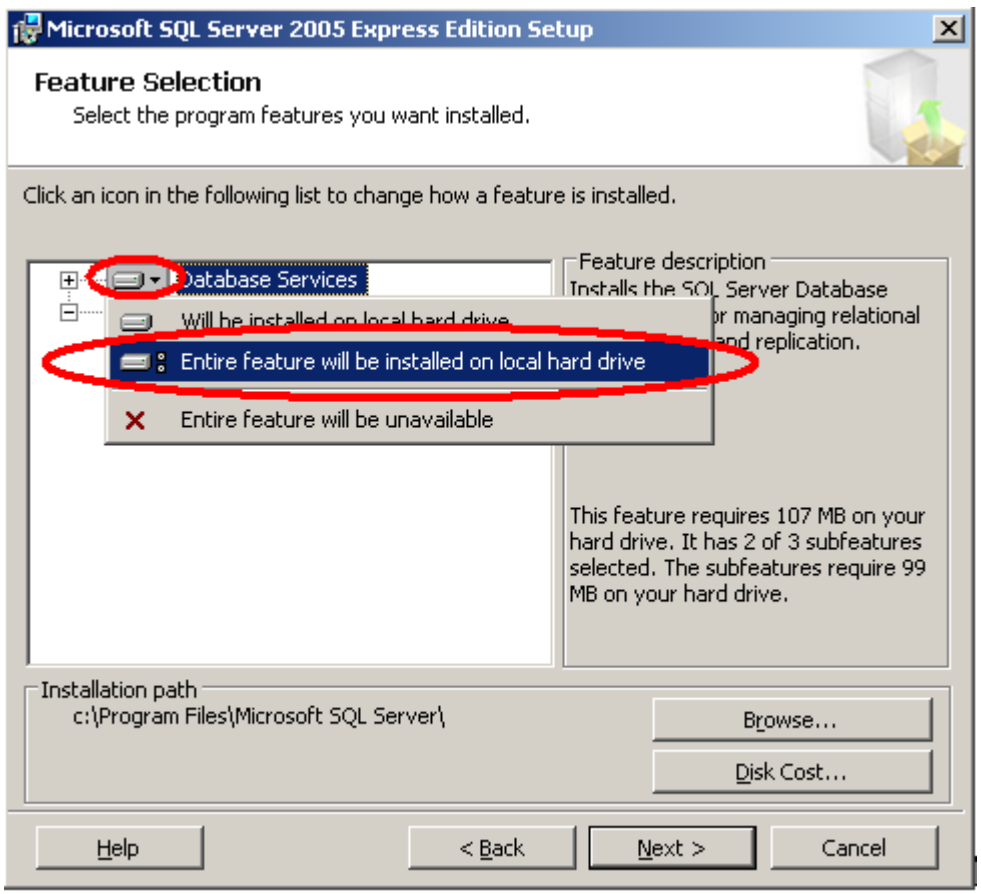

Microsoft SQL Server 2005 Setup
Installing Prerequisites
Installs software components required prior to installing SQL Server.
SQL Server Component Update will install the following components required for SQL Server Setup:
Microsoft SQL Server 2005 Setup Support Files
The required components were installed successfully.
Next >
Cancel

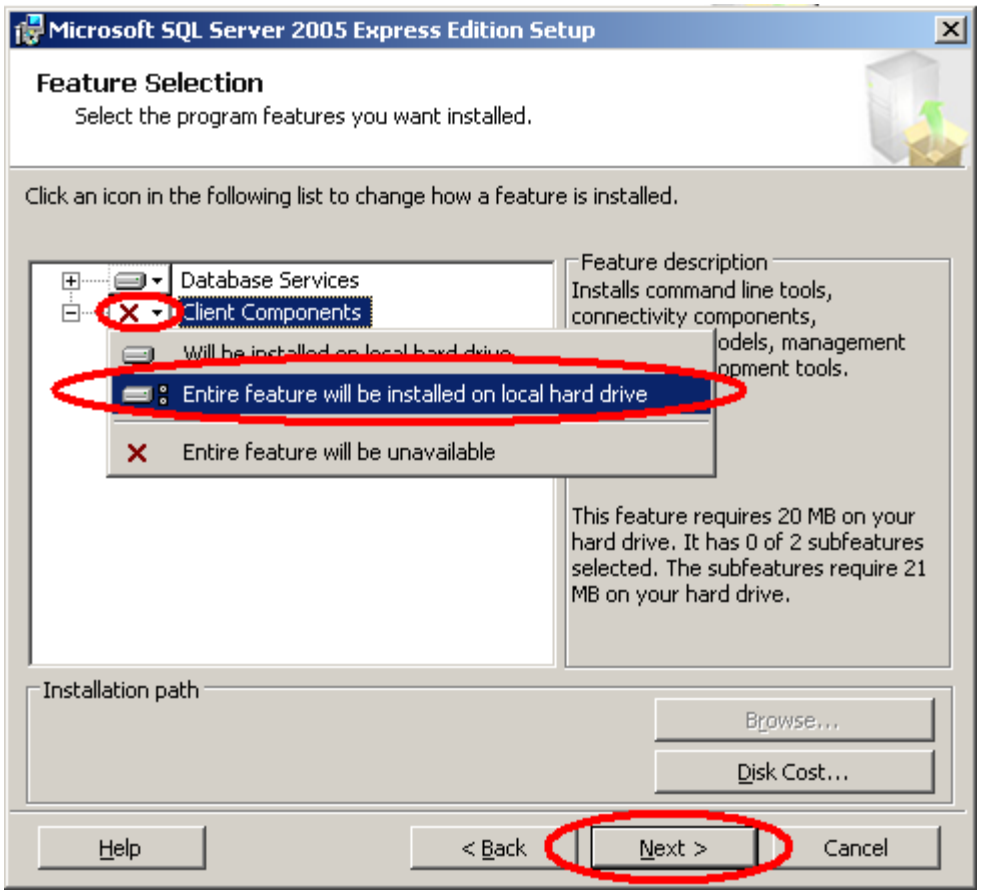

Microsoft SQL Server 2005 Setup
Welcome to the Microsoft SQL Server Installation Wizard
Setup will help you install, modify or remove Microsoft SQL Server. To continue, click Next.
< Back
Next >
Cancel

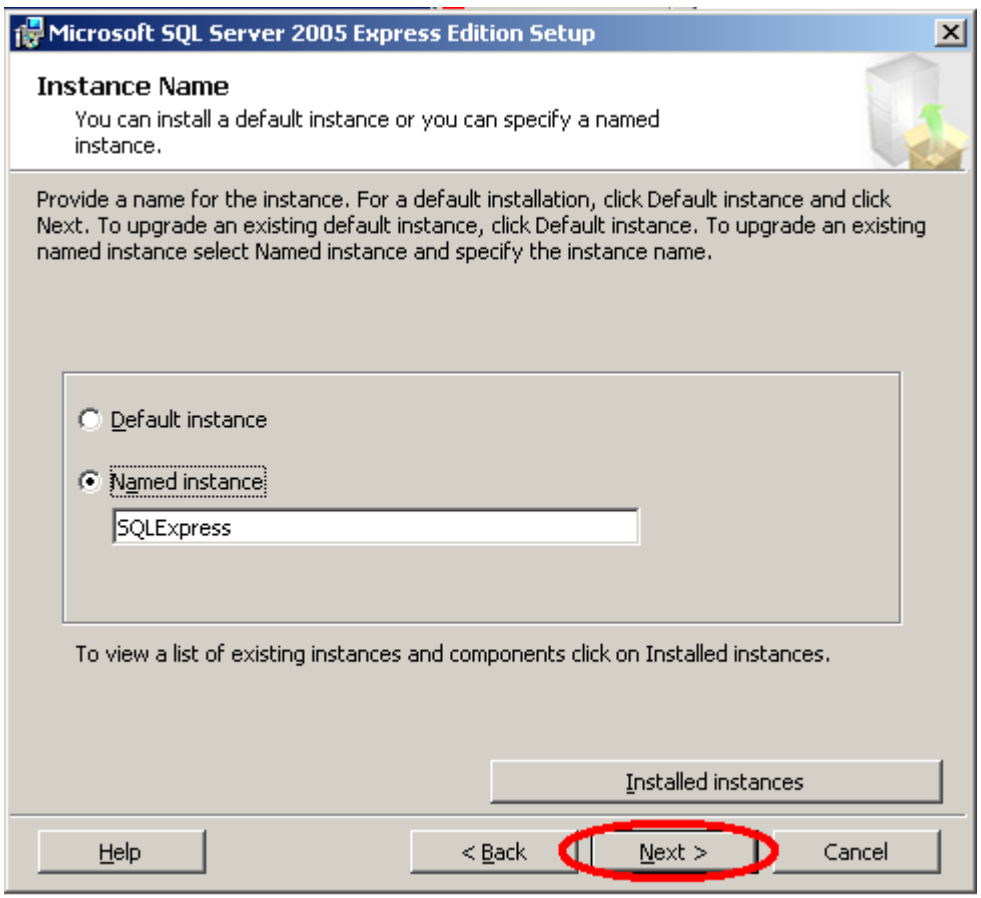

Microsoft SQL Server 2005 Setup
System Configuration Check
Wait while the system is checked for potential installation problems.
Success
12 Total
12 Success
0 Error
0 Warning
Details:
Action
Status
Message
SQL Server Edition Operating System ...
Success
Minimum Hardware Requirement
Success
Pending Reboot Requirement
Success
Default Installation Path Permission Re...
Success
Internet Explorer Requirement
Success
COM Plus Catalog Requirement
Success
ASP.Net Version Registration Require...
Success
Minimum MDAC Version Requirement
Success
Filter
Stop
Report
Help
Next >

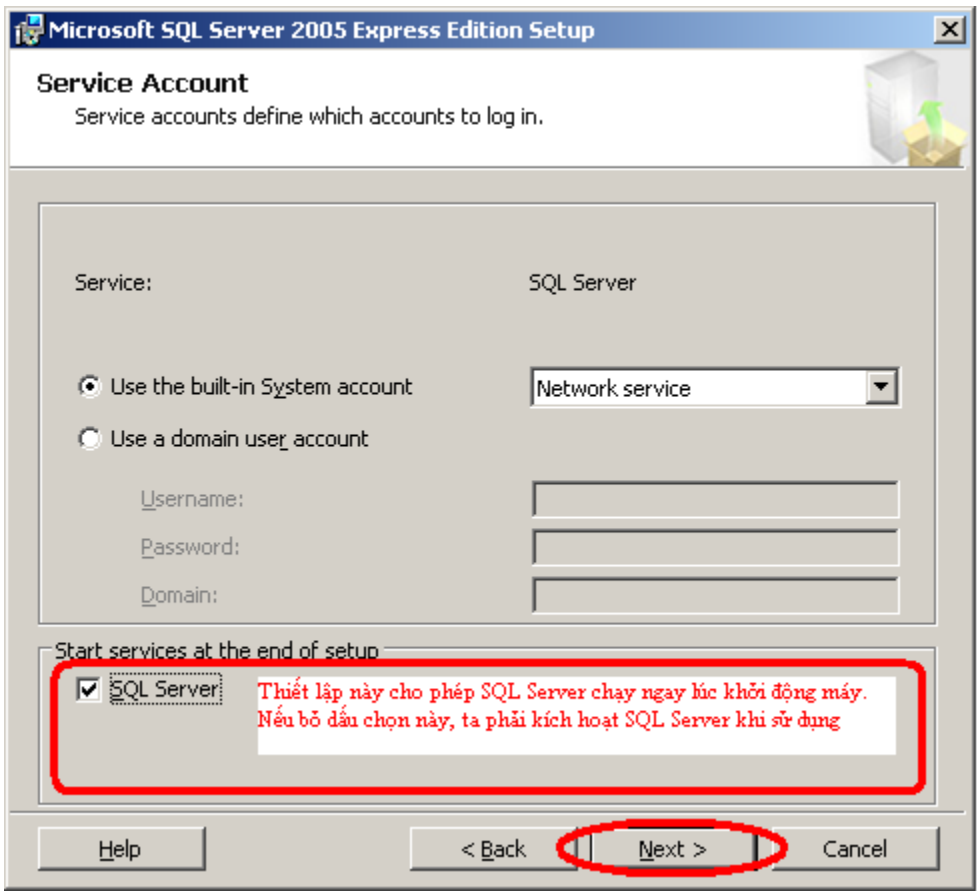

Microsoft SQL Server 2005 Express Edition Setup
Microsoft SQL Server Installation
Setup is preparing to continue with the installation.
Please wait while setup prepares to continue with the installation.
Status:
Getting existing state of installed services
Help
< Back
Next >
Cancel

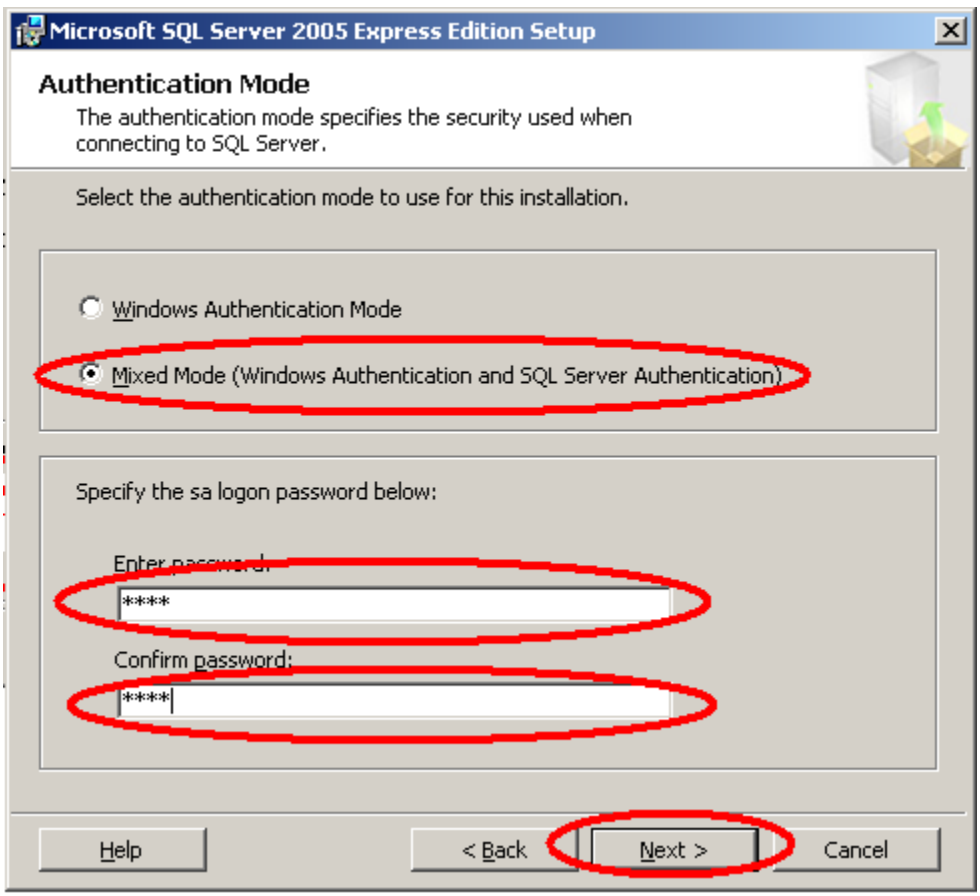

Microsoft SQL Server 2005 Express Edition Setup
Registration Information
The following information will personalize your installation.
The Name field must be filled in prior to proceeding. The Company field is optional.
Name:
KHOI
Company:
KHOI
Hide advanced configuration options
Help
< Back
Next >
Cancel


Microsoft SQL Server 2005 Express Edition Setup
Feature Selection
Select the program features you want installed.
Click an icon in the following list to change how a feature is installed.
Database Services
Will be installed on local hard drive
Entire feature will be installed on local hard drive
Entire feature will be unavailable
Feature description
Installs the SQL Server Database
r managing relational
nd replication.
This feature requires 107 MB on your hard drive. It has 2 of 3 subfeatures selected. The subfeatures require 99 MB on your hard drive.
Installation path
c:\Program Files\Microsoft SQL Server\
Browse...
Disk Cost...
Help
< Back
Next >
Cancel


Microsoft SQL Server 2005 Express Edition Setup
Feature Selection
Select the program features you want installed.
Click an icon in the following list to change how a feature is installed.
Database Services
Client Components
Entire feature will be installed on local hard drive
Entire feature will be unavailable
Feature description
Installs command line tools,
connectivity components,
odels, management
opment tools.
This feature requires 20 MB on your hard drive. It has 0 of 2 subfeatures selected. The subfeatures require 21 MB on your hard drive.
Installation path
Browse...
Disk Cost...
Help
< Back
Next >
Cancel


Microsoft SQL Server 2005 Express Edition Setup
Instance Name
You can install a default instance or you can specify a named instance.
Provide a name for the instance. For a default installation, click Default instance and click Next. To upgrade an existing default instance, click Default instance. To upgrade an existing named instance select Named instance and specify the instance name.
Default instance
Named instance
SQLExpress
To view a list of existing instances and components click on Installed instances.
Installed instances
Help
< Back
Next >
Cancel


Microsoft SQL Server 2005 Express Edition Setup
Service Account
Service accounts define which accounts to log in.
Service:
SQL Server
Use the built-in System account
Network service
Use a domain user account
Username:
Password:
Domain:
Start services at the end of setup
SQL Server
Thiết lập này cho phép SQL Server chạy ngay lúc khởi động máy. Nếu bỏ dấu chọn này, ta phải kích hoạt SQL Server khi sử dụng
Help
< Back
Next >
Cancel


Microsoft SQL Server 2005 Express Edition Setup
Authentication Mode
The authentication mode specifies the security used when connecting to SQL Server.
Select the authentication mode to use for this installation.
Windows Authentication Mode
Mixed Mode (Windows Authentication and SQL Server Authentication)
Specify the sa logon password below:
Enter password:
****
Confirm password:
****
Help
< Back
Next >
Cancel

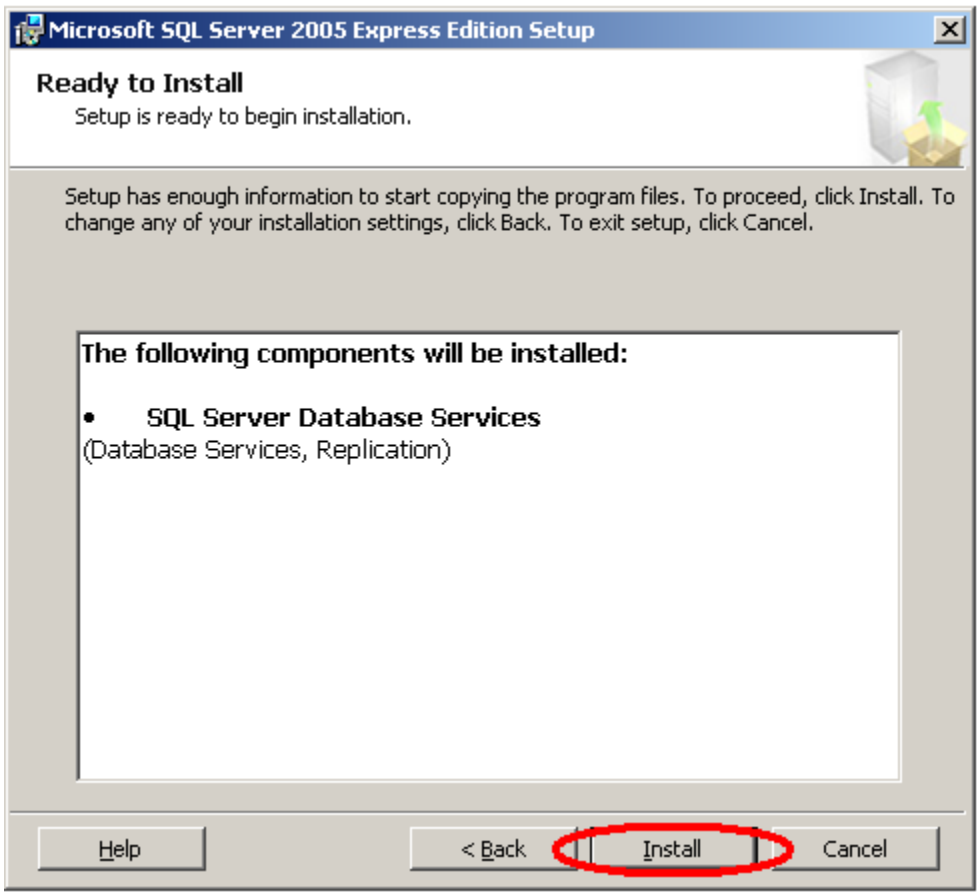
Lưu ý: SQL Server 2005 có hai kiểu authentication (kiểm tra người dùng).

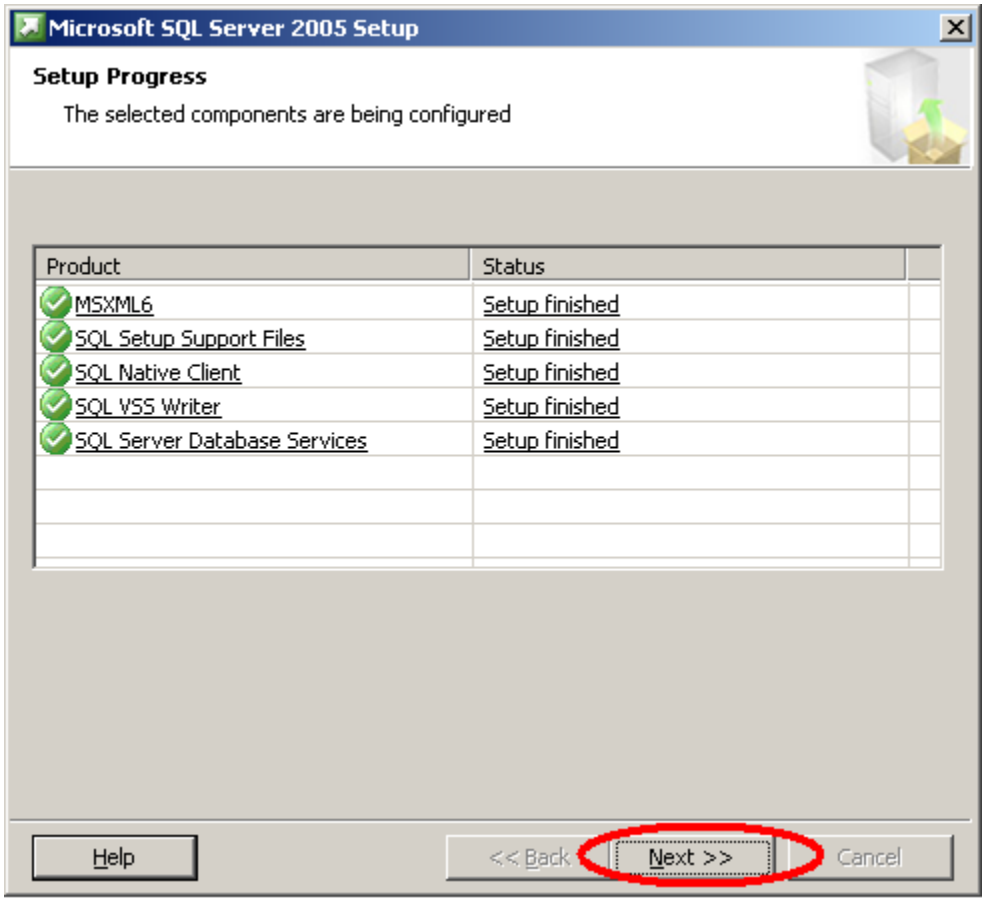
Windows authentication mode: Việc kiểm tra người dùng của SQL Server 2005 sẽ phụ thuộc vào việc kiểm tra người dùng của Windows. Khi người dùng có quyền đăng nhập vào Windows, người dùng đó sẽ có quyền đăng nhập vào SQL Server. Kiểu kiểm tra người dùng này thường được sử dụng khi ứng dụng khai thác dữ liệu và SQL Server được cài trên cùng một máy tính.

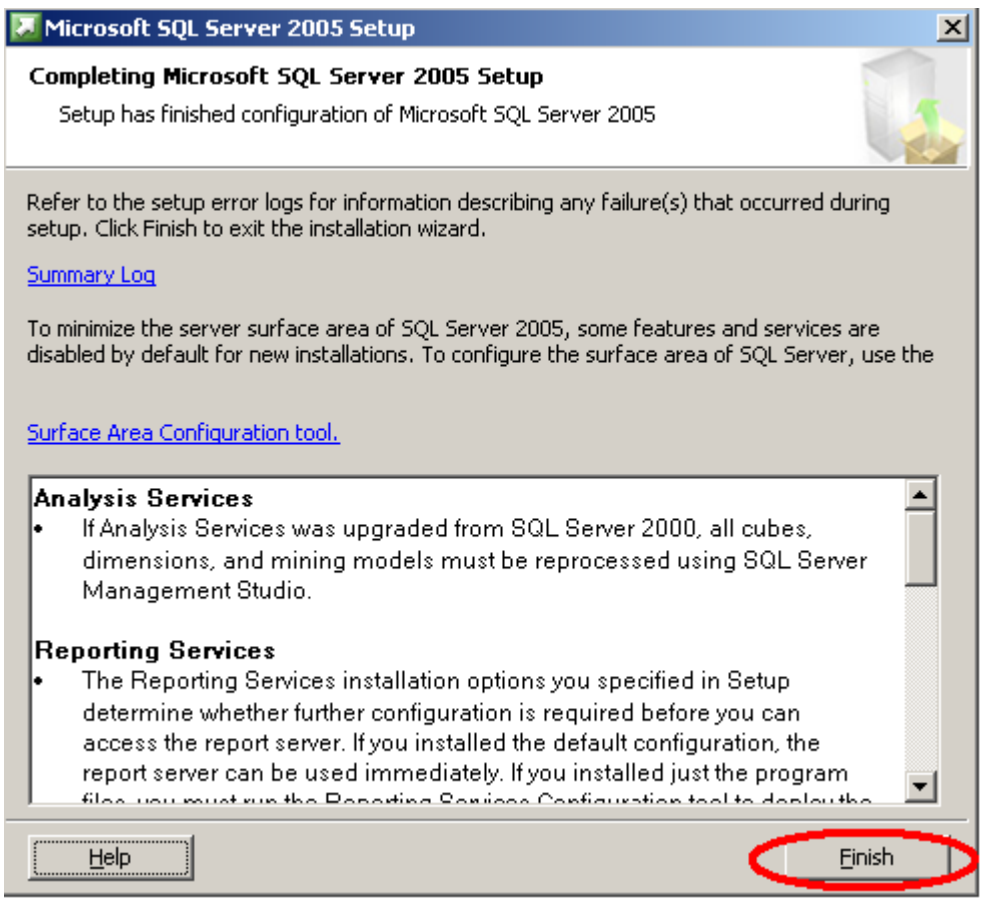
SQL Server authentication mode: Việc kiểm tra người dùng của SQL Server 2005 sẽ không phụ thuộc vào việc kiểm tra người dùng của Windows. Khi người dùng có quyền đăng nhập vào Windows, người dùng đó chưa chắc sẽ có quyền đăng nhập vào SQL Server. Để đăng nhập vào SQL Server, người dùng này phải có một bộ username và password do SQL Server quản lý. Kiểu kiểm tra người dùng này thường được sử dụng khi ứng dụng khai thác dữ liệu và SQL Server không được cài trên cùng một máy tính.

Khi chọn Mixed mode, SQL Server có thể dùng bất kỳ kiểu kiểm tra người dùng nào khi cần thiết. Đây là một thiết lập thực sự rất hữu ích khi xây dựng các ứng dụng CS DL. Ngoài ra, ta cũng phải đánh password vào hai ô bên dưới để có thể đăng nhập vào SQL Server khi ta xây dựng một ứng dụng truy xuất vào CSDL ở máy này khi ta đang ở máy khác.

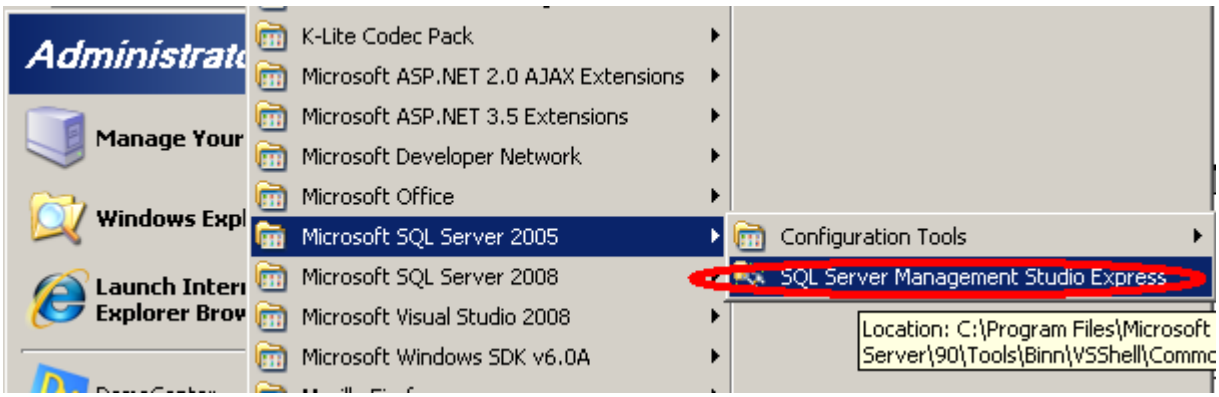
Click Next ba lần:




Microsoft SQL Server 2005 Setup
Setup Progress
The selected components are being configured
Product
Status
MSXML6
Setup finished
SQL Setup Support Files
Setup finished
SQL Native Client
Setup finished
SQL VSS Writer
Setup finished
SQL Server Database Services
Setup finished
Help
<< Back
Next >>
Cancel


Microsoft SQL Server 2005 Setup
Completing Microsoft SQL Server 2005 Setup
Setup has finished configuration of Microsoft SQL Server 2005
Refer to the setup error logs for information describing any failure(s) that occurred during setup. Click Finish to exit the installation wizard.
Summary Log
To minimize the server surface area of SQL Server 2005, some features and services are disabled by default for new installations. To configure the surface area of SQL Server, use the
Surface Area Configuration tool.
Analysis Services
If Analysis Services was upgraded from SQL Server 2000, all cubes, dimensions, and mining models must be reprocessed using SQL Server Management Studio.
Reporting Services
The Reporting Services installation options you specified in Setup determine whether further configuration is required before you can access the report server. If you installed the default configuration, the report server can be used immediately. If you installed just the program
Help
Finish

Cài đặt SQL Server Management Studio Express. Sau khi cài đặt, đăng nhập vào SQL Server 2005 Express Edition như sau:



Khi đăng nhập có thể chọn Windows Authentication hoặc SQL Server Authentication. Nếu chọn SQL Server Authentication thì phải nhập password. Password này được thiết lập trong quá trình cài đặt SQL Server 2005 Express Edition.

Nếu trong quá trình cài đặt SQL Server 2005 Express Edition chúng ta không cho phép SQL Server kích hoạt ngay khi khởi động máy, bấm nút Connect sẽ gây ra lỗi. Để khắc phục vào Start->Run đánh services.msc->Enter.

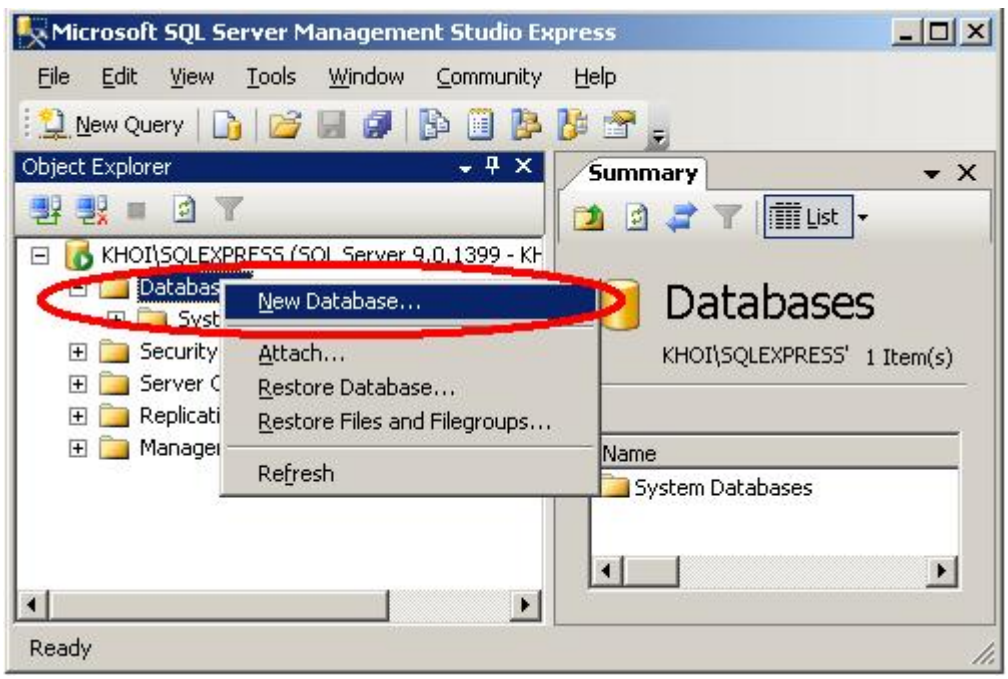
Tìm service SQL Server (SQLExpress), double click và trong comboxbox Startup type chọn Automatic -> Apply - >Start -> OK.

Giao diện sau khi đăng nhập thành công

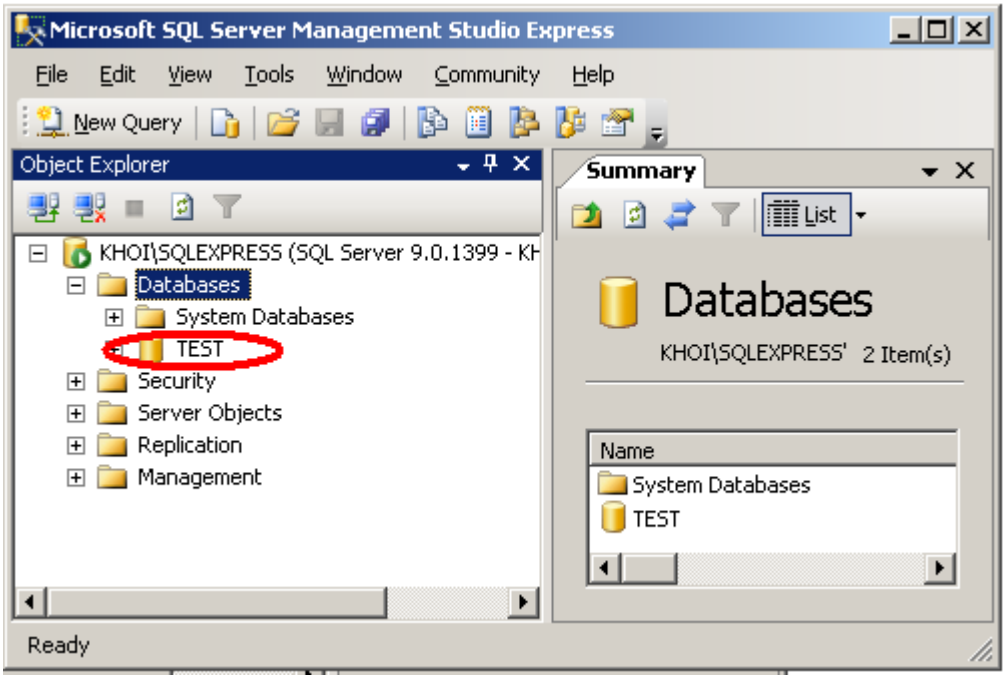
1.2 Một số thao tác cơ bản trên SQL Server 2005 Express Edition.

Microsoft SQL Server Management Studio cung cấp một giao diện thân thiện giúp cho người dùng thực hiện các thao tác một cách dễ dàng. Một số các thao tác cơ bản bao gồm: tạo CSDL mới, xóa CSDL, tạo bảng, xóa bảng…Cũng cần lưu ý rằng các thao tác thực hiện thông qua giao diện thì đều có thể được thực hiện được bằng các câu lệnh SQL.

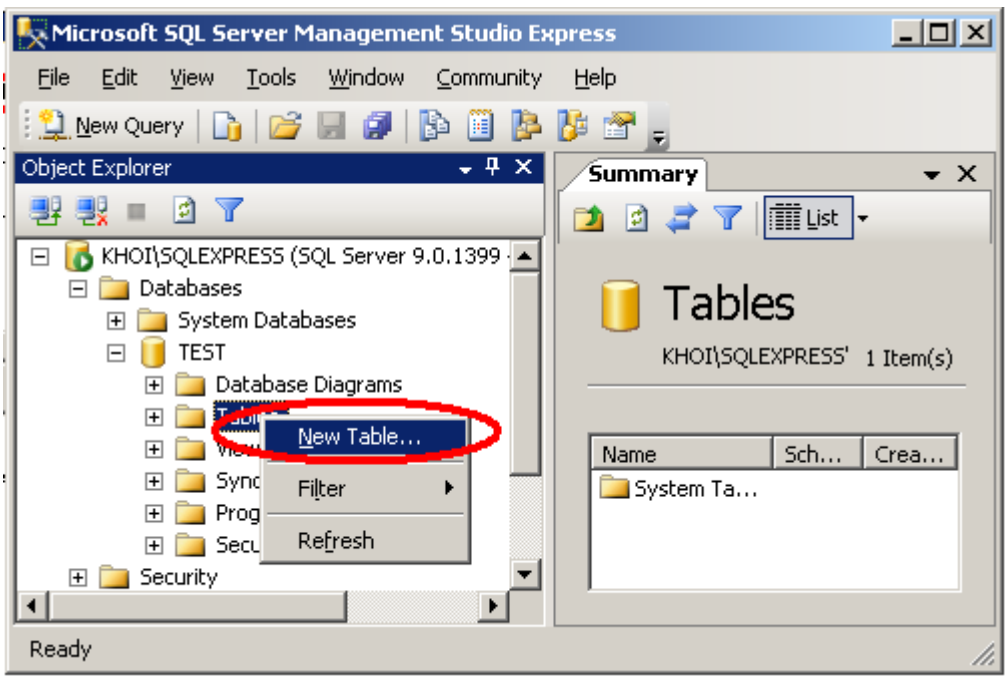
1.2.1 Tạo một CSDL mới

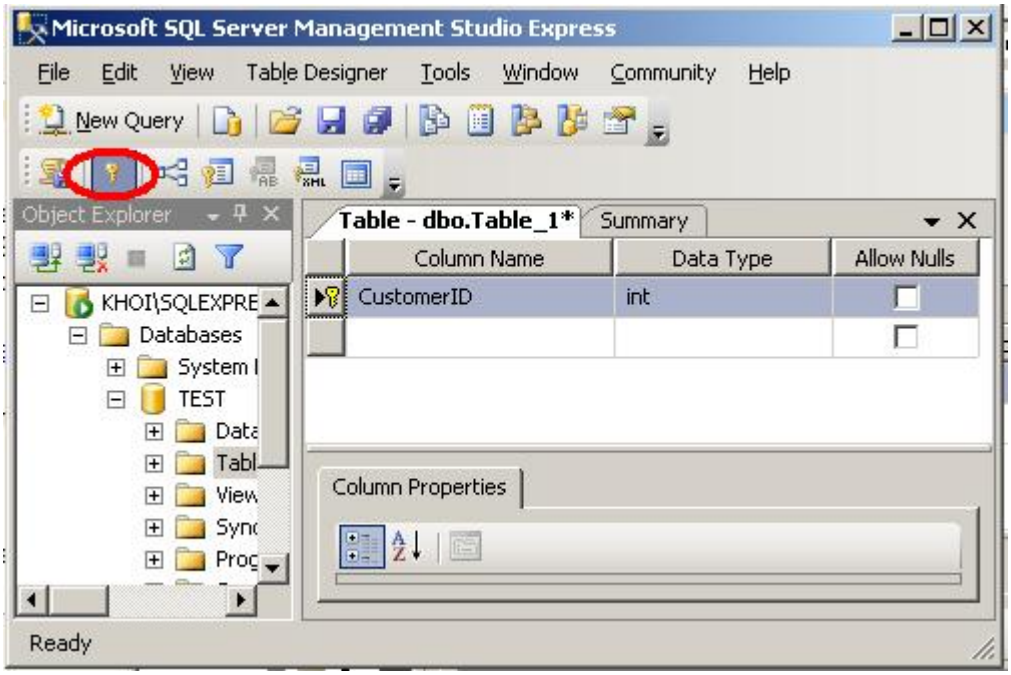


Đặt tên Database trong Textbox Database Name, click OK .

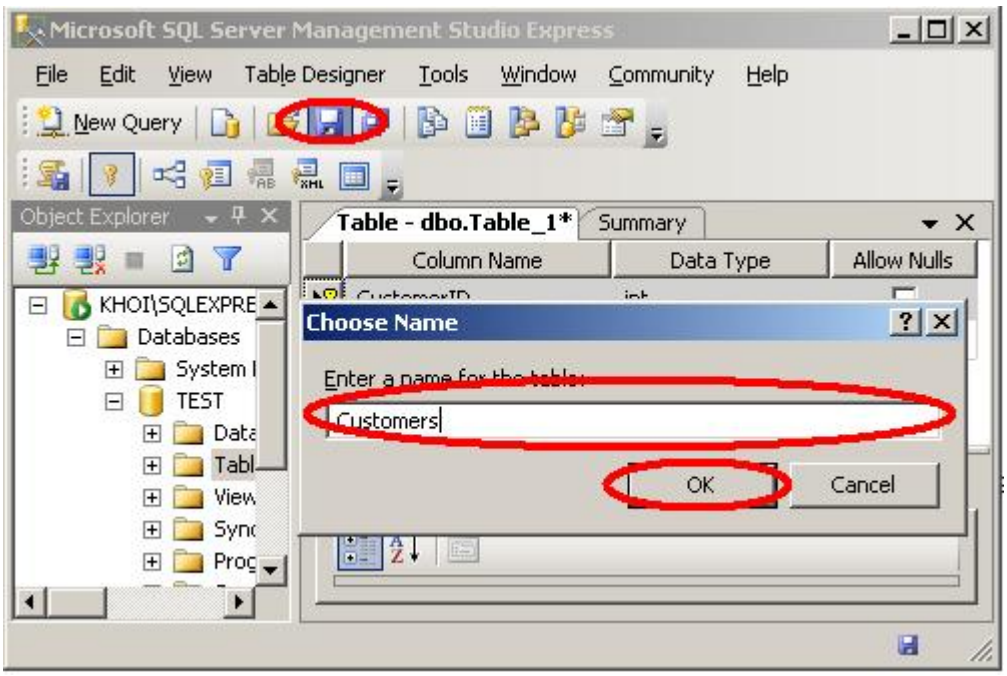


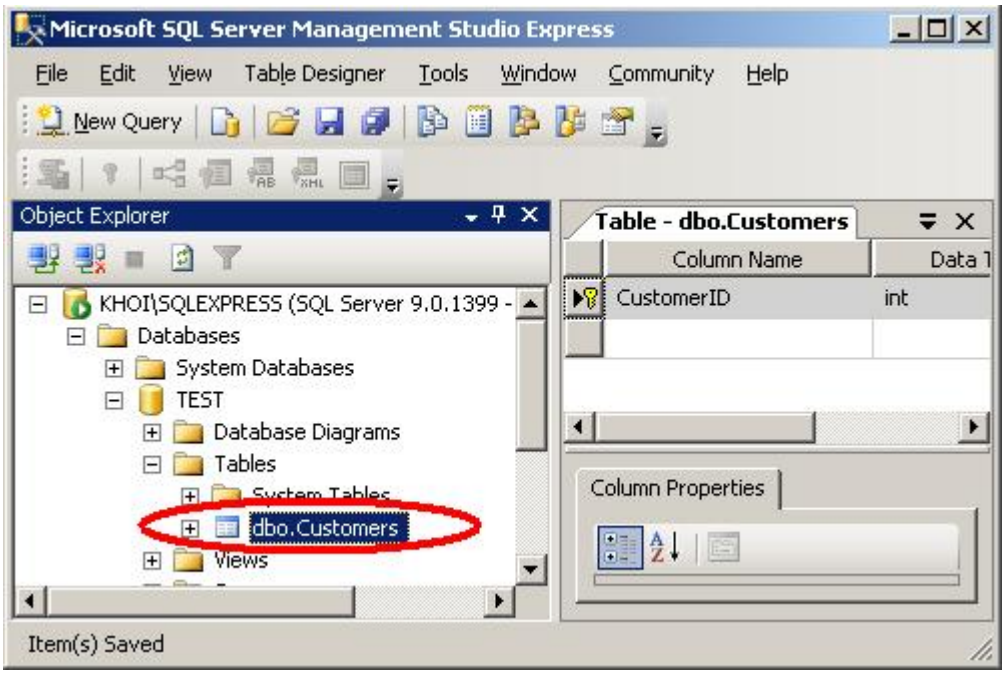
1.2.2 Tạo bảng mới





Bảng gồm các các cột. Mỗi cột gồm tên cột (Column Name), kiểu dữ liệu (Data Type) và một giá trị cho biết cột đó có thể chứa giá trị NULL hay không. Trong bảng sẽ có ít nhất một cột làm khóa chính (primary key). Cột làm khóa chính sẽ có biểu tượng chìa khóa trước tên cột. Sau khi tạo xong tất cả các cột của bảng, tiến hành Save -> OK

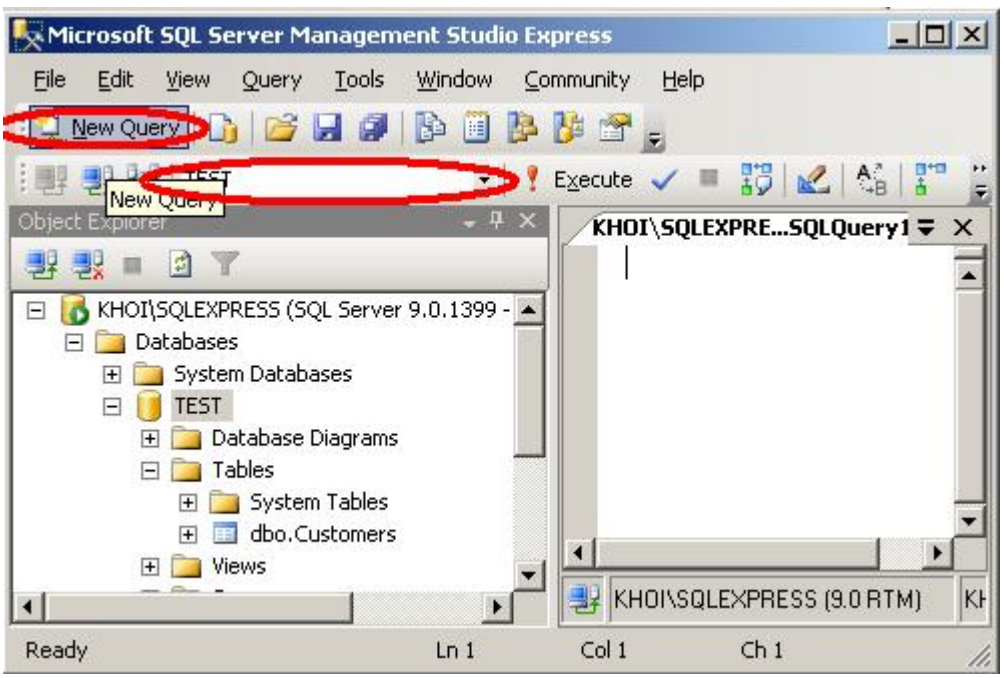




1.2.3 Xóa bảng, xóa CSDL

Click chuột phải lên bảng hay CSDL muốn xóa -> Delete - >OK. Trong trường hợp xóa một CSDL, nên chọn dấu tích vào Close existing connections. Khi đó SQL Server 2005 sẽ ngắt tất cả các kết nối vào CSDL này và việc xóa sẽ không gây báo lỗi.

1.2.4 Mở một query editor để viết câu lệnh SQL



Cần chú ý là câu lệnh SQL sẽ có tác dụng trên CSDL đang được chọn trong ComboBox. Do đó cần chú ý lựa chọn đúng CSDL cần tương tác.

2 Structured Query Language (SQL)

2.1 SQL là ngôn ngữ của cơ sở dữ liệu quan hệ

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

Khả năng của SQL vượt xa so với một công cụ truy xuất dữ liệu, mặc dù đây là mục đích ban đầu khi SQL được xây dựng nên và truy xuất dữ liệu vẫn còn là một trong những chức năng quan trọng của nó. SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

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

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

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

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

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

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

2.2 Vai trò của SQL

Bản thân SQL không phải là một hệ quản trị cơ sở dữ liệu, nó không thể tồn tại độc lập. SQL thực sự là một phần của hệ quản trị cơ sở dữ liệu, nó xuất hiện trong các hệ quản trị cơ sở dữ liệu với vai trò ngôn ngữ và là công cụ giao tiếp giữa người sử dụng và hệ quản trị cơ sở dữ liệu.

Trong hầu hết các hệ quản trị cơ sở dữ liệu quan hệ, SQL có những vai trò như sau:

SQL là ngôn ngữ hỏi có tính tương tác: Người sử dụng có thể dễ dàng thông qua các trình tiện ích để gởi các yêu cầu dưới dạng các câu lệnh SQL đến cơ sở dữ liệu và nhận kết quả trả về từ cơ sở dữ liệu

SQL là ngôn ngữ lập trình cơ sở dữ liệu: Các lập trình viên có thể nhúng các câu lệnh SQL vào trong các ngôn ngữ lập trình để xây dựng nên các chương trình ứng dụng giao tiếp với cơ sở dữ liệu

SQL là ngôn ngữ quản trị cơ sở dữ liệu: Thông qua SQL, người quản trị cơ sở dữ liệu có thể quản lý được cơ sở dữ liệu, định nghĩa các cấu trúc lưu trữ dữ liệu, điều khiển truy cập cơ sở dữ liệu,...

SQL là ngôn ngữ cho các hệ thống khách/chủ (client/server) : Trong các hệ thống cơ sở dữ liệu khách/chủ, SQL được sử dụng như là công cụ để giao tiếp giữa các trình ứng dụng phía máy khách với máy chủ cơ sở dữ liệu.

SQL là ngôn ngữ truy cập dữ liệu trên Internet: Cho đến nay, hầu hết các máy chủ Web cũng như các máy chủ trên Internet sử dụng SQL với vai trò là ngôn ngữ để tương tác với dữ liệu trong các cơ sở dữ liệu.

SQL là ngôn ngữ cơ sở dữ liệu phân tán: Đối với các hệ quản trị cơ sở dữ liệu phân tán, mỗi một hệ thống sử dụng SQL để giao tiếp với các hệ thống khác trên mạng, gởi và nhận các yêu cầu truy xuất dữ liệu với nhau.

SQL là ngôn ngữ sử dụng cho các cổng giao tiếp cơ sở dữ liệu: Trong một hệ thống mạng máy tính với nhiều hệ quản trị cơ sở dữ liệu khác nhau, SQL thường được sử dụng như là một chuẩn ngôn ngữ để giao tiếp giữa các hệ quản trị cơ sở dữ liệu.

2.3 Giới thiệu sơ lược về Transact SQL (T-SQL)

Transact-SQL là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institute) được sử dụng trong SQL Server khác với P-SQL (Procedural-SQL) dùng trong Oracle.

SQL chuẩn bao gồm khoảng 40 câu lệnh. Trong các hệ quản trị cơ sở dữ liệu khác nhau, mặc dù các câu lệnh đều có cùng dạng và cùng mục đích sử dụng song mỗi một hệ quản trị cơ sở dữ liệu có thể có một số thay đổi nào đó. Điều này đôi khi dẫn đến cú pháp chi tiết của các câu lệnh có thể sẽ khác nhau trong các hệ quản trị cơ cơ sở dữ liệu khác nhau.

T-SQL được chia làm 3 nhóm:

2.3.1 Ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL)

Đây là những lệnh dùng để tạo (create), thay đổi (alter) hay xóa (drop) các đối tượng trong CSDL. Các câu lệnh DDL thường có dạng:

Create object

Alter object

Drop object

Trong đó object có thể là: table, view, storedprocedure, function, trigger…

Ví dụ: Câu lệnh Create sau sẽ tạo một bảng mới có tên là Nhanvien trong CSDL Test.

Bảng Nhanvien này gồm có ba cột: manv, tennv, diachi.

Lưu ý: Nếu trong SQL Server 2005 Express Edition chưa có CSDL Test, hãy tạo một CSDL có tên Test theo hướng dẫn trong Chương 1.

```
create table Nhanvien
(
        manv int primary key,
        tennv nvarchar(50) not null,
        diachi nvarchar(50) not null
)
```

Để chạy câu lệnh SQL trên, mở một Query Editor, copy câu lệnh vào Query Editor, bôi đen toàn bộ câu lệnh và bấm F5.

Tiếp theo, dùng lệnh alter để thay đổi cấu trúc bảng Nhanvien.Cụ thể là một thêm một cột mới có tên ghichu vào bảng Nhanvien.

```
alter table Nhanvien
add ghichu nvarchar(50) not null
```

Cuối cùng, dùng lệnh drop để xóa hoàn toàn bảng Nhanvien ra khỏi CSDL, nghĩa là toàn bộ định nghĩa bảng và các dữ liệu bên trong đều bị xóa.

```
drop table Nhanvien
```

Lưu ý: Lệnh drop khác với lệnh delete. Lệnh delete chỉ xóa các dòng dữ liệu có trong bảng

2.3.2 Ngôn ngữ điều khiển dữ liệu (Data control language – DCL)

Đây là các lệnh quản lý quyền truy cập lên các object (table, view, storedprocedure…). Bao gồm:

Grant

Deny

Revoke

Ví dụ: Lệnh grant sẽ cấp quyền Select trên bảng Nhanvien trong CSDL Test cho các Users thuộc Role public

grant select

on nhanvien

to public

Sau khi thực hiên lệnh này, có Users trong Role public có thể thực hiện câu lệnh Select trên bảng Nhanvien trong CSDL Test.

Dùng lệnh deny để từ chối quyền select trên bảng Nhanvien trong CSDL Test của các Users thuộc Role public

deny select

on nhanvien

to public

Sau khi thực hiện lệnh này, có Users trong Role public sẽ không thể thực hiện câu lệnh Select trên bảng Nhanvien trong CSDL Test.

Dùng lệnh revoke để xóa bỏ các quyền được cấp hay từ chối trước đó.

revoke select

on nhanvien

to public

Sau khi thực hiện lệnh này, các quyền được gán hay từ chối của Users trong Role public trên bảng Nhanvien trong CSDL Test sẽ được "xóa" hoàn toàn.

2.3.3 Ngôn ngữ thao tác dữ liệu (Data manipulation language – DML)

Đây là các lệnh phổ biến dùng để xử lý dữ liệu. Bao gồm:

Select

Insert

Update

Delete

Ví dụ: Câu lệnh sau sẽ lọc ra các nhân viên có tên bắt đầu bằng chữ A trong bảng Nhanvien.

*select **

from Nhanvien as nv

where nv.tennv like 'A%'

Dấu * hàm ý là lựa chọn tất cả các cột của bảng Nhanvien. Toán tử like và ký tự đại diện sẽ được nói trong phần sau.

Câu lệnh sau sẽ thêm dữ liệu về một nhân viên mới vào trong bảng Nhanvien.

insert into Nhanvien

values(1, N'Nguyễn Văn An', N'22 Nguyễn Thiện Thuật')

Câu lệnh sau sẽ cập nhật lai địa chỉ của nhân viên có manv là 1

update Nhanvien

set diachi = N'22 Nguyễn Thị Minh Khai'

where manv = 1

Câu lệnh sau sẽ xóa thông tin của nhân viên có manv là 1 trong bảng Nhanvien

delete Nhanvien

where manv = 1

2.3.4 Cú pháp của T-SQL

Các đối tượng trong cơ sở dữ liệu dựa trên SQL (table, view, index, storedprocedure…) được xác định thông qua tên của đối tượng (hay còn gọi là identifier). Tên của các đối tượng là duy nhất trong mỗi cơ sở dữ liệu. Tên được sử dụng nhiều nhất trong các truy vấn SQL và được xem là nền tảng trong cơ sở dữ liệu quan hệ là tên bảng và tên cột.

Có hai loại Identifiers một loại thông thường (Regular Identifier) và một loại gọi là Delimited Identifier, loại này cần có dấu "" hay dấu [] để ngăn cách. Loại Delimited được dùng đối với các chữ trùng với từ khóa của SQL Server (reserved keyword) hay các chữ có khoảng trống.

Ví dụ:

Select *

From “My table”

Where [sum] = 10

Trong các cơ sở dữ liệu lớn với nhiều người sử dụng, khi ta chỉ định tên của một bảng nào đó trong câu lệnh SQL, hệ quản trị cơ sở dữ liệu hiểu đó là tên của bảng do ta sở hữu (tức là bảng do ta tạo ra). Thông thường, trong các hệ quản trị cơ sở dữ liệu này cho phép những người dùng khác nhau tạo ra những bảng trùng tên với nhau mà không gây ra xung đột về tên. Nếu trong một câu lệnh SQL ta cần chỉ đến một bảng do một người dùng khác sở hữu (hiển nhiên là phải được phép) thì tên của bảng phải được viết sau tên của người sở hữu và phân cách với tên người sở hữu bởi dấu chấm:

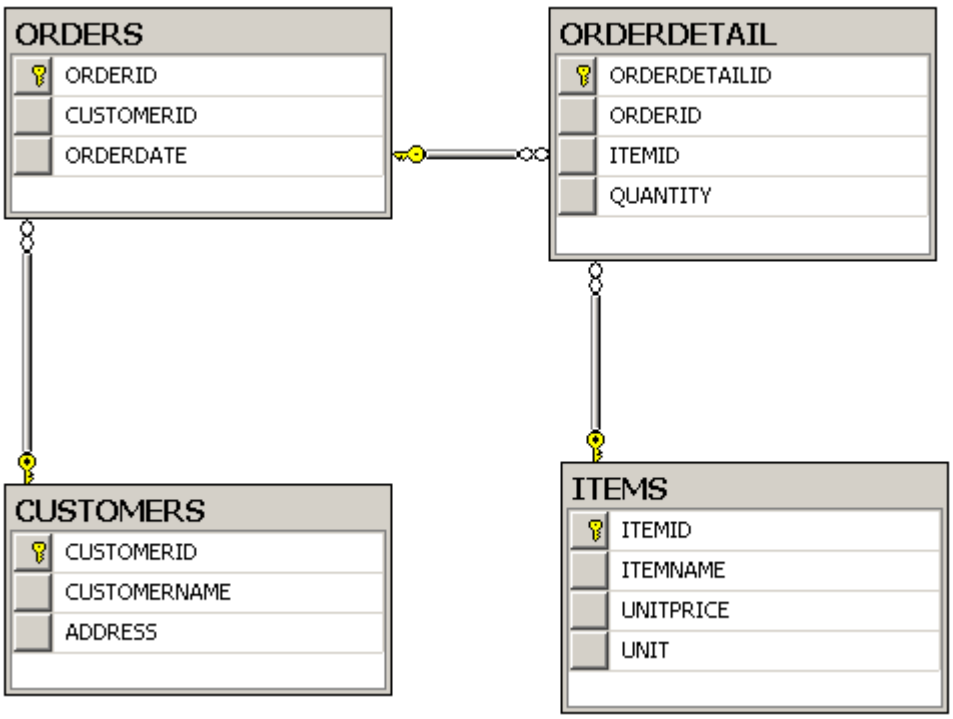
tên_người_sở_hữu.tên_bảng

Một số đối tượng cơ sở dữ liệu khác (như khung nhìn, thủ tục, hàm), việc sử dụng tên cũng tương tự như đối với bảng.

Ta có thể sử dụng tên cột một cách bình thường trong các câu lệnh SQL bằng cách chỉ cần chỉ định tên của cột trong bảng. Tuy nhiên, nếu trong câu lệnh có liên quan đến hai cột trở

lên có cùng tên trong các bảng khác nhau thì bắt buộc phải chỉ định thêm tên bảng trước tên cột; tên bảng và tên cột được phân cách nhau bởi dấu chấm

Ví dụ: Giả sử chúng ta có CSDL như sau:



Để tìm ra khách hàng có tên Nguyễn Văn An đã đặt hàng vào ngày nào, câu truy vấn như sau:

Select orderid, orderdate
from orders, customers
where orders.customerid = customers.customerid
and customername = N'Nguyễn Văn An'

2.3.5 Các kiểu dữ liệu

Bảng dưới đây liệt kê một số kiểu dữ liệu thông dụng được sử dụng trong SQL.

Char(n)	Kiểu chuỗi với độ dài cố định
Nchar(n)	Kiểu chuỗi với độ dài cố định hỗ trợ UNICODE
Varchar(n)	Kiểu chuỗi với độ dài chính xác
Nvarchar(n)	Kiểu chuỗi với độ dài chính xác hỗ trợ UNICODE
Int	Số nguyên có giá trị từ -2^{31} đến $2^{31} - 1$
Tinyint	Số nguyên có giá trị từ 0 đến 255.
Smallint	Số nguyên có giá trị từ -2^{15} đến $2^{15} - 1$

Bigint	Số nguyên có giá trị từ -263 đến 263-1
Numeric	Kiểu số với độ chính xác cố định.
Decimal	Tương tự kiểu Numeric
Float	Số thực có giá trị từ -1.79E+308 đến 1.79E+308
Real	Số thực có giá trị từ -3.40E + 38 đến 3.40E + 38
Money	Kiểu tiền tệ
Bit	Kiểu bit (có giá trị 0 hoặc 1)
Datetime	Kiểu ngày giờ (chính xác đến phần trăm của giây)
Smalldatetime	Kiểu ngày giờ (chính xác đến phút)
Binary	Dữ liệu nhị phân với độ dài cố định (tối đa 8000 bytes)
Varbinary	Dữ liệu nhị phân với độ dài chính xác (tối đa 8000 bytes)
Image	Dữ liệu nhị phân với độ dài chính xác (tối đa 2,147,483,647 bytes)
Text	Dữ liệu kiểu chuỗi với độ dài lớn (tối đa 2,147,483,647 ký tự)
Ntext	Dữ liệu kiểu chuỗi với độ dài lớn và hỗ trợ UNICODE (tối đa 1,073,741,823 ký tự)

Ví dụ: Mỗi cột trong bảng sẽ chứa những dữ liệu thuộc về duy nhất một kiểu dữ liệu trong SQL Server. Cột nào chứa những dữ liệu thuộc kiểu nào sẽ được quy định lúc định nghĩa bảng.

```
Create table Nhanvien
(
MANV   NVARCHAR(10)      NOT NULL,
HOTEN NVARCHAR(30)      NOT NULL,
GIOITINH      BIT,
NGAYSINH      SMALLDATETIME,
NOISINH       NCHAR(50),
HSLUONG       DECIMAL(4,2),
MADV   INT
)
```

2.3.6 Biến (Variables)

Biến trong T-SQL cũng có chức năng tương tự như trong các ngôn ngữ lập trình khác nghĩa là cần khai báo trước loại dữ liệu trước khi sử dụng. Biến được bắt đầu bằng dấu @ (Đối với các biến toàn cục - global variable - thì có hai dấu @@)

Ví dụ: Ví dụ dưới đây khai báo một biến có tên @numberOfCustomers thông qua từ khóa declare. Biến này lưu số khách hàng đếm được thông qua hàm count. Sau đó in ra giá trị của biến.

declare @numberOfCustomers int
select @numberOfCustomers = count(*)
from Customers
print @numberOfCustomers

2.3.7 Hàm (Function)

Có 2 loại hàm: một loại là được xây dựng sẵn trong SQL Server 20005 Express Edition (built-in) và một loại do người dùng tự định nghĩa (user-defined)

Các hàm Built-In được chia làm 3 nhóm:

Rowset Functions : Loại này thường trả về một object và được đối xử như một table. Ví dụ như hàm OPENQUERY sẽ trả về một recordset và có thể đứng vị trí của một table trong câu lệnh Select.

Aggregate Functions : Loại này làm việc trên một số giá trị và trả về một giá trị đơn hay là các giá trị tổng. Ví dụ như hàm AVG sẽ trả về giá trị trung bình của một cột.

Scalar Functions : Loại này làm việc trên một giá trị đơn và trả về một giá trị đơn. Trong loại này lại chia làm nhiều loại nhỏ như các hàm về toán học, về thời gian, xử lý kiểu dữ liệu String....Ví dụ như hàm MONTH('2002-09-30') sẽ trả về tháng 9.

Các hàm User-Defined (được tạo ra bởi câu lệnh CREATE FUNCTION và phần body thường được gói trong cặp lệnh BEGIN...END) cũng được chia làm các nhóm như sau:

Scalar Functions : Loại này cũng trả về một giá trị đơn bằng câu lệnh RETURNS.

Table Functions : Loại này trả về một table

2.3.8 Các toán tử (Operators)

Trong SQL Server các biểu diễn (expression) có thể xuất hiện nhiều toán tử. Độ ưu tiên của toán tử sẽ quyết định thứ tự thực hiện của các phép tính. Thứ tự thực hiện ảnh hưởng rất lớn đến kết quả.

Bảng dưới đây mô tả các toán tử trong SQL Server 2005 Express Edititon và mức độ ưu tiên của các toán tử đó.

Level	Operators
1	* (Multiply), / (Division), % (Modulo)
2	+ (Positive), - (Negative), + (Add), (+ Concatenate), - (Subtract),

3	=, >, <, >=, <=, <>, !=, !>, !< (Comparison operators)
4	NOT
5	AND
6	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
7	= (Assignment)

2.3.9 Các thành phần điều khiển (Control of flow)

Như BEGIN...END, BREAK, CONTINUE, GOTO, IF...ELSE, RETURN, W HILE…

2.3.10 Chú thích (Comment)

T-SQL dùng kí hiệu -- để chú thích cho một dòng đơn và kí hiệu /*…*/ để chú thích cho một nhóm dòng

Ví dụ:

/* Minh họa chú thích

Chú thích cho một dòng đơn và một nhóm các dòng*/

DECLARE @MyNumber int ***-- khai báo biến***

SET @MyNumber = 4 - 2 + 27

-- kết quả là 29

SELECT @MyNumber

2.3.11 Giá trị NULL

Một cơ sở dữ liệu là sự phản ánh của một hệ thống trong thế giới thực, do đó các giá trị dữ liệu tồn tại trong cơ sở dữ liệu có thể không xác định được. Một giá trị không xác định được xuất hiện trong cơ sở dữ liệu có thể do một số nguyên nhân sau:

Giá trị đó có tồn tại nhưng không biết.

Không xác định được giá trị đó có tồn tại hay không.

Tại một thời điểm nào đó giá trị chưa có nhưng rồi có thể sẽ có.

Giá trị bị lỗi do tính toán (tràn số, chia cho không,...)

Những giá trị không xác định được biểu diễn trong cơ sở dữ liệu quan hệ bởi các giá trị NULL. Đây là giá trị đặc biệt và không nên nhầm lẫn với chuỗi rỗng (đối với dữ liệu kiểu chuỗi) hay giá trị không (đối với giá trị kiểu số). Giá trị NULL đóng một vai tr ò quan trọng trong các cơ sở dữ liệu và hầu hết các hệ quản trị cơ sở dữ liệu quan hệ hiện nay đều hỗ trợ việc sử dụng giá trị này.

3 Ngôn ngữ thao tác dữ liệu – DML

SQL được xem như là công cụ hữu hiệu để thực hiện các yêu cầu truy vấn và thao tác trên dữ liệu. Trong chương này, ta sẽ bàn luận đến nhóm các câu lệnh trong SQL được sử dụng cho mục đích này. Nhóm các câu lệnh này được gọi chung là ngôn ngữ thao tác dữ liệu (DML: Data Manipulation Language) bao gồm các câu lệnh sau:

SELECT: Sử dụng để truy xuất dữ liệu từ môt hoặc nhiều bảng.

INSERT: Thêm dữ liệu.

UPDATE: Cập nhật dữ liệu

DELETE: Xoá dữ liệu

Trong số các câu lệnh này, có thể nói SELECT là câu lệnh tương đối phức tạp và được sử dụng nhiều trong cơ sở dữ liệu. Với câu lệnh này, ta không chỉ thực hiện các yêu cầu truy xuất dữ liệu đơn thuần mà còn có thể thực hiện được các yêu cầu thống kê dữ liệu phức tạp. Cũng chính vì vậy, phần đầu của chương này sẽ tập trung tương đối nhiều đến câu lệnh SELECT. Các câu lệnh INSERT, UPDATE và DELETE được bàn luận đến ở cuối chương

3.1 Câu lệnh SELECT

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

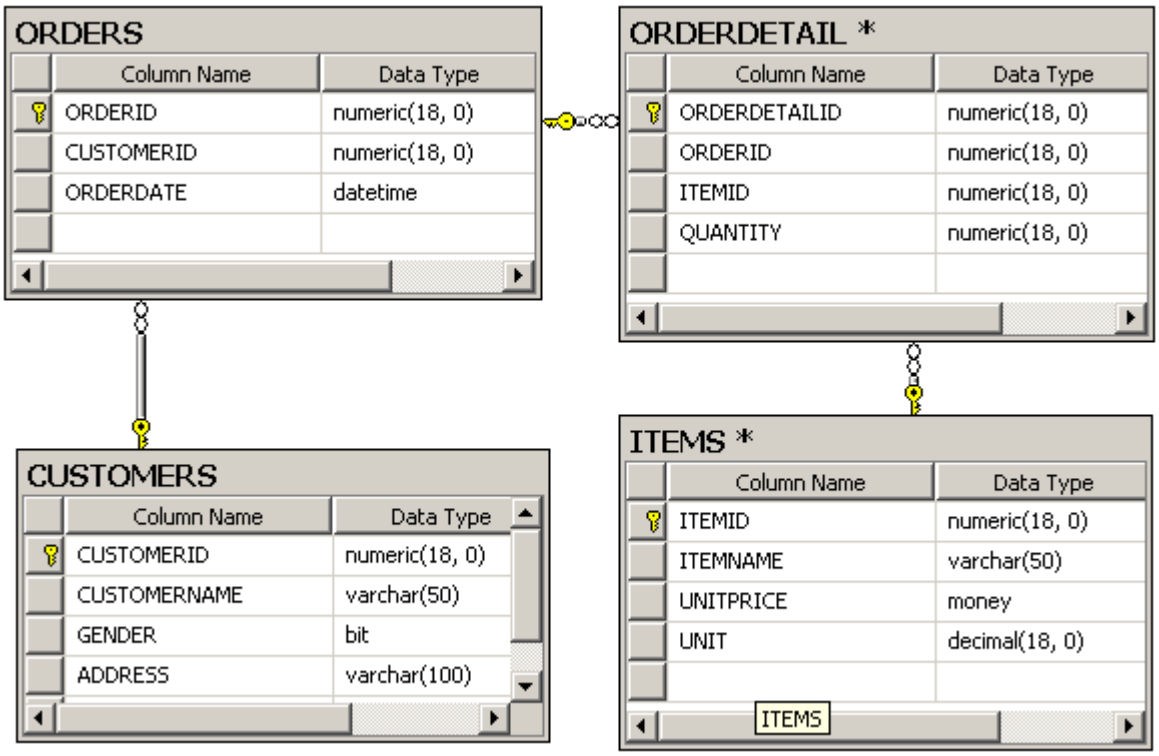
Cú pháp chung của câu lệnh SELECT có dạng:

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

Điều cần lưu ý đầu tiên đối với câu lệnh này là các thành phần trong câu lệnh SELECT nếu được sử dụng phải tuân theo đúng thứ tự như trong cú pháp. Nếu không, câu lệnh sẽ được xem là không hợp lệ.

Câu lệnh SELECT được sử dụng để tác động lên các bảng dữ liệu và kết quả của câu lệnh cũng được hiển thị dưới dạng bảng, tức là một tập hợp các dòng và các cột (ngoại trừ trường hợp sử dụng câu lệnh SELECT với mệnh đề COMPUTE).

Ví dụ:



Ví dụ dưới đây hiển thị tên khách hàng và địa chỉ các khách hàng hiện có.

select customername, gender, address

from customers

CustomerName	Address
Cao Van Trung	33 Nguyen Trung Truc
Tran Van Phuc	99 Nguyen Thi Minh Khai
Tran Viet Cuong	45/2B Da Tuong
Nguyen Van Dai	76 Tran Phu
Le Thi Hoa	56 Le Hong Phong
Nguyen Thanh Thai	29A Phuong Sai
Cao Van Trung	12 Nguyen Thien Thuat

3.1.1 Danh sách chọn trong câu lệnh SELECT

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

Chọn tất cả các cột: Như đã nói trong chương 1, chúng ta dùng dấu * trong câu lệnh Select để hàm ý chọn hết tất cả các cột. Trong trường hợp này, các cột được hiển thị trong kết quả truy vấn sẽ tuân theo thứ tự mà chúng đã được tạo ra khi bảng được định nghĩa.

Ví dụ:

*Select * from Customers*

CUSTOMERID	CUSTOMERNAME	GENDER	ADDRESS
6	Cao Van Trung	1	33 Nguyen Trung Truc
2	Tran Van Phuc	1	99 Nguyen Thi Minh Khai
3	Tran Viet Cuong	1	45/2B Da Tuong
4	Nguyen Van Dai	1	76 Tran Phu
7	Le Thi Hoa	0	56 Le Hong Phong
8	Nguyen Thanh Thai	0	29A Phuong Sai
9	Cao Van Trung	1	12 Nguyen Thien Thuat

Chọn một số cột cụ thể: Trong trường hợp cần chỉ định cụ thể các cột cần hiển thị trong kết quả truy vấn, ta chỉ định danh sách các tên cột trong danh sách chọn. Thứ tự của các cột trong kết quả truy vấn tuân theo thứ tự của các trường trong danh sách chọn.

Ví dụ:

Select CUSTOMERNAME, ADDRESS

From Customers

CUSTOMERNAME	ADDRESS
Cao Van Trung	33 Nguyen Trung Truc
Tran Van Phuc	99 Nguyen Thi Minh Khai
Tran Viet Cuong	45/2B Da Tuong
Nguyen Van Dai	76 Tran Phu
Le Thi Hoa	56 Le Hong Phong
Nguyen Thanh Thai	29A Phuong Sai
Cao Van Trung	12 Nguyen Thien Thuat

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

tên_bảng.tên_trường

Thay đổi tiêu đề các cột:

Trong kết quả truy vấn, tiêu đề của các cột mặc định sẽ là tên của các trường tương ứng trong bảng. Tuy nhiên, để các tiêu đề trở nên thân thiện hơn, ta có thể đổi tên các tiêu đề của các cột. Để đặt tiêu đề cho một cột nào đó, ta sử dụng cách viết:

tiêu_đề_cột = tên_trường hoặc

tên_trường AS tiêu_đề_cột hoặc

tên_trường tiêu_đề_cột

Ví dụ:

select [Mã khách hàng] = Customerid,

customername as [Tên khách hàng],

address [Địa chỉ]

from Customers

Mã khách hàng	Tên khách hàng	Địa chỉ
6	Cao Van Trung	33 Nguyen Trung Truc
2	Tran Van Phuc	99 Nguyen Thi Minh Khai
3	Tran Viet Cuong	45/2B Da Tuong
4	Nguyen Van Dai	76 Tran Phu
7	Le Thi Hoa	56 Le Hong Phong
8	Nguyen Thanh Thai	29A Phuong Sai
9	Cao Van Trung	12 Nguyen Thien Thuat

Sử dụng cấu trúc CASE…WHEN:

Cấu trúc CASE được sử dụng trong danh sách chọn nhằm thay đổi kết quả của truy vấn tuỳ thuộc vào các trường hợp khác nhau. Cấu trúc này có cú pháp như sau:

CASE biểu_thức

WHEN biểu_thức_kiểm_tra THEN kết_quả

[...]

[ELSE kết_quả_của_else]

END

hoặc:

CASE

WHEN điều_kiện THEN kết_quả

[...]

[ELSE kết_quả_của_else]

END

Ví dụ: Câu lệnh SQL dưới đây sẽ hiện thị giới tính của khách hàng tùy theo giá trị thực được lưu trong CSDL. Nếu giá trị trong CSDL là FALSE-> hiện thị giới tính NỮ, nếu giá trị là TRUE-> hiện thị giới tính NAM.

```
select CUSTOMERNAME, ADDRESS,
  case GENDER
        when 1 then 'NAM'
        else  N'NỮ'
  end as [GIỚI TÍNH]
from customers
```

Câu lệnh trên cũng có thể viết như sau:

select CUSTOMERNAME, ADDRESS,

case

when GENDER = 1 then 'NAM'

else N'NỮ'

end as [GIỚI TÍNH]

from customers

CUSTOMERNAME	ADDRESS	GIỚI TÍNH
Cao Van Trung	33 Nguyen Trung Truc	NAM
Tran Van Phuc	99 Nguyen Thi Minh Khai	NAM
Tran Viet Cuong	45/2B Da Tuong	NAM
Nguyen Van Dai	76 Tran Phu	NAM
Le Thi Hoa	56 Le Hong Phong	NỮ
Nguyen Thanh Thai	29A Phuong Sai	NỮ
Cao Van Trung	12 Nguyen Thien Thuat	NAM

Loại bỏ các dòng dữ liệu trùng nhau:

Từ khóa DISTINCT sẽ loại bỏ các dòng dữ liệu giống nhau. Trong ví dụ trên, có hai khách hàng có tên Cao Van Trung. Nếu ta chỉ truy vấn tên khách hàng, để loại bỏ sự trùng lắp ta dùng từ khóa DISTINCT

select ***distinct*** CUSTOMERNAME

from customers

CUSTOMERNAME
Cao Van Trung
Le Thi Hoa
Nguyen Thanh Thai
Nguyen Van Dai
Tran Van Phuc
Tran Viet Cuong

Lựa chọn một số lượng giới hạn các dòng:

Từ khóa TOP n sẽ trả về chỉ n dòng dữ liệu

Ví dụ: ví dụ sau chỉ trả về duy nhất hai dòng dữ liệu

select top 2 Customername

from customers

Customername
Cao Van Trung
Tran Van Phuc

Nếu sử dung TOP n PERCENT thì sẽ trả về n % số dòng dữ liệu hiện có trong CSDL.

3.1.2 Mệnh đề FROM

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

Ví dụ: Câu lệnh sau hiển thị thông tin khách hàng

*Select * from Customers*

CUSTOMERID	CUSTOMERNAME	GENDER	ADDRESS
6	Cao Van Trung	1	33 Nguyen Trung Truc
2	Tran Van Phuc	1	99 Nguyen Thi Minh Khai
3	Tran Viet Cuong	1	45/2B Da Tuong
4	Nguyen Van Dai	1	76 Tran Phu
7	Le Thi Hoa	0	56 Le Hong Phong
8	Nguyen Thanh Thai	0	29A Phuong Sai
9	Cao Van Trung	1	12 Nguyen Thien Thuat

Trong mệnh đề FROM có thể sử dụng bí danh (alias) nhằm làm cho câu truy vấn dễ nhìn hơn.

Ví dụ:

*Select * from Customers* ***c***

Where c.CustomerID = 1

3.1.3 Mệnh đề WHERE - điều kiện truy vấn dữ liệu

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

Ví dụ: Lọc ra thông tin các khách hàng có mã

*Select **

From Customers

Where CustomerID > 3

CUSTOMERID	CUSTOMERNAME	GENDER	ADDRESS
6	Cao Van Trung	1	33 Nguyen Trung Truc
4	Nguyen Van Dai	1	76 Tran Phu
7	Le Thi Hoa	0	56 Le Hong Phong
8	Nguyen Thanh Thai	0	29A Phuong Sai
9	Cao Van Trung	1	12 Nguyen Thien Thuat

Trong mệnh đề WHERE thường sử dụng:

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

Các toán tử so sánh

Kiểm tra giới hạn của dữ liệu (BETWEEN/ NOT BETWEEN)

Tập hợp

Kiểm tra khuôn dạng dữ liệu.

Các giá trị NULL

Các toán tử so sánh

Toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ: Ví dụ dưới đây lấy tên, ngày sinh theo định dạng dd/MM/yyyy và địa chỉ của những khách hàng có tên Le Thi Hoa và tuổi các khách hàng này lớn hơn 20.

```
select    CUSTOMERNAME,
convert (varchar, BIRTHDAY, 103) as BIRTHDAY, ADDRESS
from Customers
where Customername = 'Le Thi Hoa'
and year(getdate()) - year(BIRTHDAY) > 20
```

CUSTOMERNAME	BIRTHDAY	ADDRESS
Le Thi Hoa	05/04/1982	56 Le Hong Phong

Kiểm tra giới hạn của dữ liệu

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

Mệnh đề	Ý nghĩa
variable BETWEEN a AND b	a <= variable <=b
variable NOT BETWEEN a AND b	variable <a hoặc variable > b

Ví dụ: ví dụ này tương tự ví dụ ở trên nhưng điều kiện là độ tuổi nằm trong khoảng từ 20 đến 30 tuổi.

```
select    CUSTOMERNAME,
convert (varchar, BIRTHDAY, 103) as BIRTHDAY, ADDRESS
from Customers
where Customername = 'Le Thi Hoa'
and year(getdate()) - year(BIRTHDAY) between 20 and 30
```

Toán tử làm việc trên tập hợp (IN/ NOT IN)

Từ khoá IN/ NOT IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu lệnh SELECT là một danh sách các giá trị. Sau IN/ NOT IN có thể là một danh sách các giá trị hoặc là một câu lệnh SELECT khác.

Ví dụ: Câu lệnh dưới đây lấy ra các thông tin của khách hàng có mã là 5,6 hoặc 7

```
select CUSTOMERID, CUSTOMERNAME,
convert(varchar,BIRTHDAY, 103) as BIRTHDAY, ADDRESS
from Customers
where CUSTOMERID in (5,6,7)
```

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	ADDRESS
6	Cao Van Trung	12/06/1959	33 Nguyen Trung Truc
2	Tran Van Phuc	02/03/1970	99 Nguyen Thi Minh Khai
3	Tran Viet Cuong	01/01/1980	45/2B Da Tuong
4	Nguyen Van Dai	04/03/1955	76 Tran Phu

Ví dụ: Ví dụ này minh họa một câu lệnh SELECT khác đứng sau mệnh đề IN/ NOT IN

```
select CUSTOMERID, CUSTOMERNAME,
convert(varchar,BIRTHDAY, 103) as BIRTHDAY, ADDRESS
from Customers
where CUSTOMERID not in
( select CUSTOMERID from customers where customerid >= 7)
```

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	ADDRESS
6	Cao Van Trung	12/06/1959	33 Nguyen Trung Truc
2	Tran Van Phuc	02/03/1970	99 Nguyen Thi Minh Khai
3	Tran Viet Cuong	01/01/1980	45/2B Da Tuong
4	Nguyen Van Dai	04/03/1955	76 Tran Phu

Toán tử LIKE/ NOT LIKE và ký tự đại diện (WildCard)

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

Ký tự đại diện	Ý nghĩa
%	Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự
_	Một ký tự bất kì
[]	Một ký tự nằm trong giới hạn được chỉ định. Ví dụ:[a-f] hàm ý chỉ một trong các ký tự: a, b, c, d, e, f.
[^]	Một ký tự không nằm trong giới hạn được chỉ định. Ví dụ:[^a-f] hàm ý chỉ một ký tự khác tất cả các ký tự: a, b, c, d, e, f.

Ví dụ: Ví dụ dưới đây tìm ra các khách hàng có tên bắt đều bằng Nguyen

*select **

from customers

where customername like 'Nguyen%'

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS
4	Nguyen Van Dai	1955-03-04 00:00:00.000	1	76 Tran Phu
8	Nguyen Thanh Thai	1940-03-01 00:00:00.000	0	29A Phuong Sai

Giá trị NULL

Dữ liệu trong một cột cho phép NULL sẽ nhận giá trị NULL trong các tr ường hợp sau:

Nếu không có dữ liệu được nhập cho cột và không có mặc định cho cột hay kiểu dữ liệu trên cột đó.

Người sử dụng trực tiếp đưa giá trị NULL vào cho cột đó.

Một cột có kiểu dữ liệu là kiểu số sẽ chứa giá trị NULL nếu giá trị được chỉ định gây tràn số.

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không, ta sử dụng cách viết:

WHERE tên_cột IS NULL

hoặc:

WHERE tên_cột IS NOT NULL

Ví dụ:

*select **

from Customers

where birthday is null

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS
9	Cao Van Trung	NULL	1	12 Nguyen Thien Thuat

Tạo mới bảng dữ liệu từ cau lệnh SELECT

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

Ví dụ:

select CUSTOMERNAME,

convert(varchar,BIRTHDAY, 103) as BIRTHDAY, ADDRESS

into NEWCUSTOMERS

from Customers

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

Sắp xếp kết quả truy vấn

Mặc định, các dòng dữ liệu trong kết quả của câu truy vấn tuân theo thứ tự của chúng trong bảng dữ liệu hoặc được sắp xếp theo chỉ mục (nếu trên bảng có chỉ mục). Trong trường hợp muốn dữ liệu được sắp xếp theo chiều tăng hoặc giảm của giá trị của một hoặc nhiều trường, ta sử dụng thêm mệnh đề ORDER BY trong câu lệnh SELECT; Sau ORDER BY là danh sách các cột cần sắp xếp (tối đa là 16 cột). Dữ liệu được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng. Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo thứ tự từ trái qua phải.

Ví dụ: Ví dụ đưới đây sắp xếp thông tin các khách hàng theo thứ tự tuổi giảm dần.

select CUSTOMERNAME, year(getdate()) - year(BIRTHDAY) as AGE, ADDRESS
from Customers
order by AGE DESC

CUSTOMERNAME	AGE	ADDRESS
Nguyen Thanh Thai	68	29A Phuong Sai
Nguyen Van Dai	53	76 Tran Phu
Cao Van Trung	49	33 Nguyen Trung Truc
Tran Van Phuc	38	99 Nguyen Thi Minh Khai
Tran Viet Cuong	28	45/2B Da Tuong
Le Thi Hoa	26	56 Le Hong Phong
Cao Van Trung	NULL	12 Nguyen Thien Thuat

Ta có thể chỉ định số thứ tự của cột cấn được sắp xếp. Câu lệnh ở ví dụ trên có thể được viết lại như sau:

select CUSTOMERNAME, year(getdate()) - year(BIRTHDAY) as AGE, ADDRESS
from Customers
order by 2 DESC

3.1.4 Phép hợp (UNION)

Phép hợp được sử dụng trong trường hợp ta cần gộp kết quả của hai hay nhiều truy vấn thành một tập kết quả duy nhất. SQL cung cấp toán tử UNION để thực hiện phép hợp. Cú pháp như sau:

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

Trong đó

Câu_lệnh_1 có dạng

SELECT danh_sách_cột

[INTO tên_bảng_mới]

[FROM danh_sách_bảng|khung_nhìn]

[WHERE điều_kiện]

[GROUP BY danh_sách_cột]

[HAVING điều_kiện]

và Câu_lệnh_i (i = 2,..,n) có dạng

SELECT danh_sách_cột

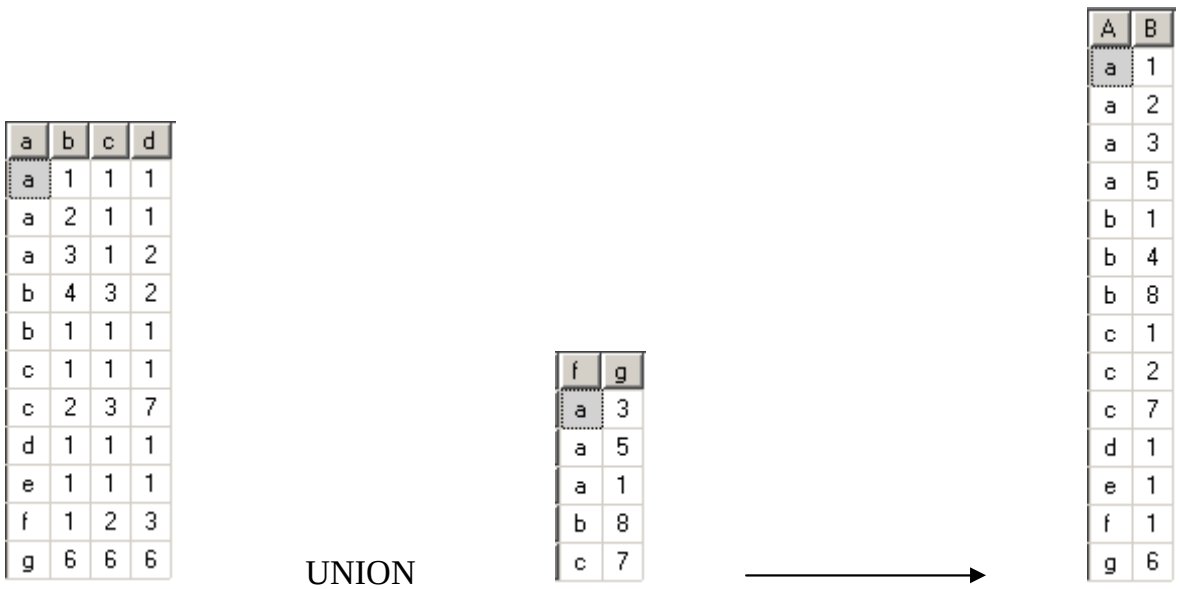
[FROM danh_sách_bảng|khung_nhìn]

[WHERE điều_kiện]

[GROUP BY danh_sách_cột]

[HAVING điều_kiện]

Ví dụ: Phép hợp giữa hai bảng dưới đây cho kết quả như sau:

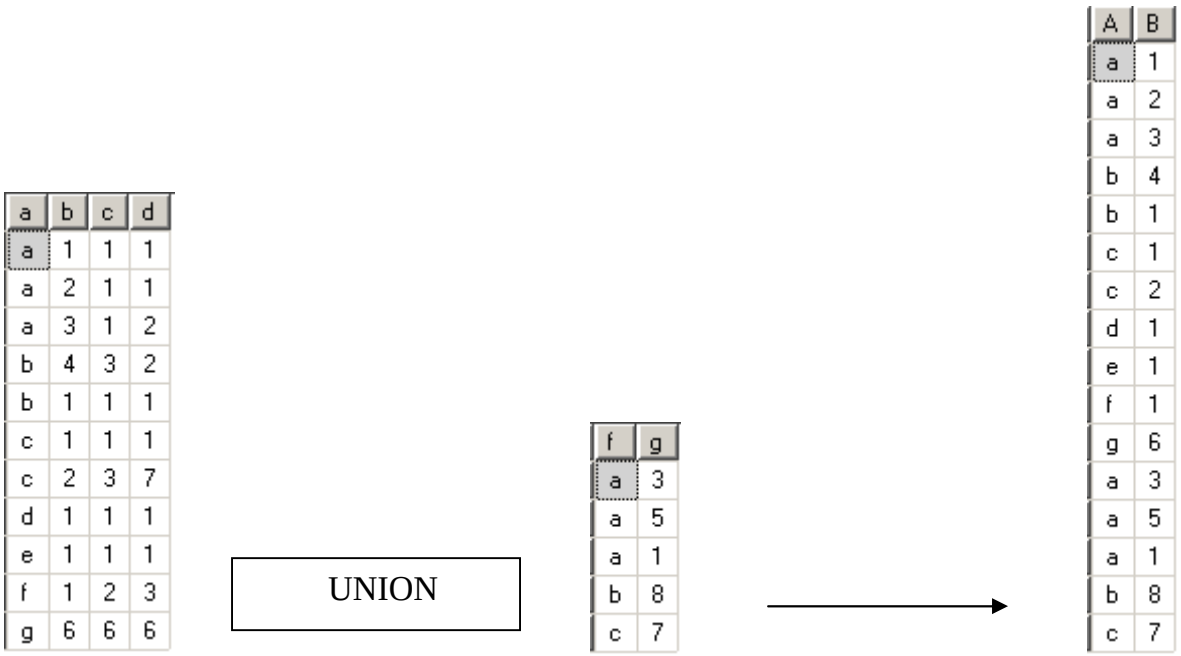


select A,B from A

union

select F,G from B

Mặc định, nếu trong các truy vấn thành phần của phép hợp xuất hiện những dòng dữ liệu giống nhau thì trong kết quả truy vấn chỉ giữ lại một dòng. Nếu muốn giữ lại các dòng này, ta phải sử dụng thêm từ khoá ALL trong truy vấn thành phần.



Khi sử dụng toán tử UNION để thực hiện phép hợp, ta cần chú ý các nguy ên tắc sau:

Danh sách cột trong các truy vấn thành phần phải có cùng số lượng.

Các cột tương ứng trong tất cả các bảng, hoặc tập con bất kỳ các cột được sử dụng trong bản thân mỗi truy vấn thành phần phải cùng kiểu dữ liệu.

Các cột tương ứng trong bản thân từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau. Nguyên nhân là do phép hợp so sánh các cột từng cột một theo thứ tự được cho trong mỗi truy vấn.

Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể được).

Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

Mệnh đề ORDER BY và COMPUTE dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị thống kê chỉ được sử dụng ở cuối câu lệnh UNION. Chúng không được sử dụng ở trong bất kỳ truy vấn thành phần nào.

Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần. Chúng không được phép sử dụng để tác động lên kết quả chung của phép hợp.

Phép toán UNION có thể được sử dụng bên trong câu lệnh INSERT.

Phép toán UNION không được sử dụng trong câu lệnh CREATE VIEW.

3.1.5 Phép nối

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

Ví dụ: Để tìm ra khách hàng có mã là 3 đã đặt hàng trong những ngày nào thì câu truy vấn như sau:

```
select c.CUSTOMERNAME, o.ORDERDATE
from customers c, orders o
where c.customerid = o.customerid
 and c.customerid = 3
```

CUSTOMERNAME	ORDERDATE
Tran Viet Cuong	2008-01-01 00:00:00.000

Giải thích câu truy vấn:

Bảng Customers

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS
6	Cao Van Trung	1959-06-12 00:00:00.000	1	33 Nguyen Trung Truc
2	Tran Van Phuc	1970-03-02 00:00:00.000	1	99 Nguyen Thi Minh Khai
3	Tran Viet Cuong	1980-01-01 00:00:00.000	1	45/2B Da Tuong
4	Nguyen Van Dai	1955-03-04 00:00:00.000	1	76 Tran Phu
7	Le Thi Hoa	1982-04-05 00:00:00.000	0	56 Le Hong Phong
8	Nguyen Thanh Thai	1940-03-01 00:00:00.000	0	29A Phuong Sai
9	Cao Van Trung	NULL	1	12 Nguyen Thien Thuat

Bảng Order

ORDERID	CUSTOMERID	ORDERDATE
1	6	2007-12-06 00:00:00.000
2	3	2008-01-01 00:00:00.000

Trước tiên vào bảng Customers tìm ra dòng có có mã khách hàng là 3.

Tìm kiếm trong bảng Orders các dòng có giá trị trường CUSTOMERID là 3 và cho các dòng này vào kết quả truy vấn.

Như vậy để thực hiện yêu cầu truy vấn, chúng ta phải thực hiện phép kết nối giữa hai bảng Customers và Orders với điều kiện kết nối là CUSTOMERID của bảng CUSTOMERS bằng với CUSTOMERID của bảng ORDERS.

Phép nối là cơ sở để thực hiện các yêu cầu truy vấn dữ liệu liên quan đến nhiều bảng. Một câu lệnh nối thực hiện lấy các dòng dữ liệu trong các bảng tham gia truy vấn, so sánh giá trị của các dòng này trên một hoặc nhiều cột được chỉ định trong điều kiện nối và kết hợp các dòng thoả mãn điều kiện thành những dòng trong kết quả truy vấn.

Để thực hiện được một phép nối, cần phải xác định được những yếu tố sau:

Những cột nào cần hiển thị trong kết quả truy vấn

Những bảng nào có tham gia vào truy vấn.

Điều kiện để thực hiện phép nối giữa các bảng dữ liệu là gì

Trong các yếu tố kể trên, việc xác định chính xác điều kiện để thực hiện phép nối giữa các bảng đóng vai trò quan trọng nhất. Trong đa số các trường hợp, điều kiện của phép nối được xác định nhờ vào mối quan hệ giữa các bảng cần phải truy xuất dữ liệu. Thông thường, đó là điều kiện bằng nhau giữa khoá chính và khoá ngoài của hai bảng có mối quan hệ với nhau. Như vậy, để có thể đưa ra một câu lệnh nối thực hiện chính xác yêu cầu truy vấn dữ liệu đòi hỏi phải hiểu được mối quan hệ cũng như ý nghĩa của chúng giữa các bảng dữ liệu.

Một câu lệnh nối cũng được bắt đầu với từ khóa SELECT. Các cột được chỉ định tên sau từ khoá SELECT là các cột được hiển thị trong kết quả truy vấn. Việc sử dụng tên các cột trong danh sách chọn có thể là:

Tên của một số cột nào đó trong các bảng có tham gia vào truy vấn. Nếu tên cột trong các bảng trùng tên nhau thì tên cột phải được viết dưới dạng

tên_bảng.tên_cột

Dấu sao (*) được sử dụng trong danh sách chọn khi cần hiển thị tất cả các cột của các bảng tham gia truy vấn.

Trong trường hợp cần hiển thị tất cả các cột của một bảng nào đó, ta sử dụng cách viết:

tên_bảng.*

Mệnh đề FROM trong phép nối

Sau mệnh đề FROM của câu lệnh nối là danh sách tên các bảng (hay khung nhìn) tham gia vào truy vấn. Nếu ta sử dụng dấu * trong danh sách chọn thì thứ tự của các bảng liệt kê sau FROM sẽ ảnh hưởng đến thứ tự các cột được hiển thị trong kết quả truy vấn.

Mệnh đề WHERE trong phép nối

Khi hai hay nhiều bảng được nối với nhau, ta phải chỉ định điều kiện để thực hiện phép nối ngay sau mệnh đề WHERE. Điều kiện nối được biểu diễn dưới dạng biểu thức logic so sánh giá trị dữ liệu giữa các cột của các bảng tham gia truy vấn.

Các toán tử so sánh dưới đây được sử dụng để xác định điều kiện nối

Phép toán	Ý nghĩa
=	Bằng
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn

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

3.1.6 Các loại phép nối

Phép nối bằng: Một phép nối bằng (equi-join) là một phép nối trong đó giá trị của các cột được sử dụng để nối được so sánh với nhau dựa trên tiêu chuẩn bằng và tất cả các cột trong các bảng tham gia nối đều được đưa ra trong kết quả.

Một dạng đặc biệt của phép nối bằng được sử dụng nhiều là phép nối tự nhiên (natural-join). Trong phép nối tự nhiên, điều kiện nối giữa hai bảng chính là điều kiện bằng giữa khoá ngoài và khoá chính của hai bảng; Và trong danh sách chọn của câu lệnh chỉ giữ lại một cột trong hai cột tham gia vào điều kiện của phép nối.

Ví dụ phép kết nối bằng:

*select **

from Customers c, Orders o

where c.customerid = o.customerid

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS	ORDERID	CUSTOMERID	ORDERDATE
6	Cao Van Trung	1959-06-12 00:00:00.000	1	33 Nguyen Trung Truc	1	6	2007-12-06 00:00:00.000
3	Tran Viet Cuong	1980-01-01 00:00:00.000	1	45/2B Da Tuong	2	3	2008-01-01 00:00:00.000

Ví dụ phép kết nối tự nhiên:

select c.CUSTOMERID, c.CUSTOMERNAME,

c.BIRTHDAY, c.GENDER, c.ADDRESS, o.ORDERDATE

from Customers c, Orders o

where c.customerid = o.customerid

hoặc viết gọn:

select c., o.ORDERDATE*

from Customers c, Orders o

where c.customerid = o.customerid

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS	ORDERDATE
6	Cao Van Trung	1959-06-12 00:00:00.000	1	33 Nguyen Trung Truc	2007-12-06 00:00:00.000
3	Tran Viet Cuong	1980-01-01 00:00:00.000	1	45/2B Da Tuong	2008-01-01 00:00:00.000

Trong phép kết nối bằng, trường CUSTOMERID xuất hiện hai lần. Sự dư thừa được loại bỏ bằng cách sử dụng phép kết nối tự nhiên và việc chỉ định rõ các cột cột cần truy xuất.

Trong các câu lệnh nối, ngoài điều kiện của phép nối được chỉ định trong mệnh đề WHERE còn có thể chỉ định các điều kiện tìm kiếm dữ liệu khác (điều kiện chọn). Thông thường, các điều kiện này được kết hợp với điều kiện nối thông qua toán tử AND.

Ví dụ:

```
select c.*, o.ORDERDATE
from Customers c, Orders o
where c.customerid = o.customerid
  and c.customerid = 3
```

Phép tự nối

Phép tự nối là phép nối mà trong đó điều kiện nối được chỉ định liên quan đến các cột của cùng một bảng. Trong trường hợp này, sẽ có sự xuất hiện tên của cùng một bảng nhiều lần trong mệnh đề FROM và do đó các bảng cần phải được đặt bí danh.

Ví dụ: Giả sử có yêu cầu tìm ra các khách hàng có nhiều hơn một đơn đặt hàng trong cùng ngày

```
select c1.CUSTOMERID, c1.CUSTOMERNAME
from customers c1, customers c2, orders o1, orders o2
where c1.customerid =  o1.customerid
and c2.customerid = o2.customerid
and c1.customerid = c2.customerid
and o1.orderdate = o2.orderdate
and o1.orderid <> o2.orderid
```

Câu truy vấn được giải thích như sau: Lần lượt lấy ra các mã khách hàng, mã hóa đơn và ngày đặt hàng từ bảng c1, o1 đem so sánh lần lượt với các mã khách hàng, mã hóa đơn và ngày đặt hàng từ bảng c2, o2. Nếu việc so sánh hai tập hợp này thỏa điều kiện sau đây: mã khách hàng trùng nhau, ngày đặt hàng trùng nhau và có mã hóa đơn khác nhau thì thông tin khách hàng này được cho vào kết qua truy vấn.

Phép nối ngoài

Trong các phép nối đã đề cập ở trên, chỉ những dòng có giá trị trong các cột được chỉ định thoả mãn điều kiện kết nối mới được hiển thị trong kết quả truy vấn, và được gọi là phép nối trong (inner join) Theo một nghĩa nào đó, những phép nối này loại bỏ thông tin chứa trong những dòng không thoả mãn điều kiện nối. Tuy nhiên, đôi khi ta cũng cần giữ lại những thông tin này bằng cách cho phép những dòng không thoả mãn điều kiện nối có mặt trong kết quả của phép nối. Để làm điều này, ta có thể sử dụng phép nối ngoài.

SQL cung cấp các loại phép nối ngoài sau đây:

Phép nối ngoài trái (ký hiệu: *=): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên trái trong điều kiện nối cho dù những dòng này không thoả mãn điều kiện của phép nối

Phép nối ngoài phải (ký hiệu: =*): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên phải trong điều kiện nối cho dù những dòng này không thoả điều kiện của phép nối.

Tuy nhiên trong SQL Server 2005 Express Edition không hỗ trợ trực tiếp các phép nối *= và =*. Mặt khác trong các phiên bản SQL Server sắp tới các phép nối này sẽ hoàn toàn không được hỗ trợ. Do đó Microsoft khuyến cáo người sử dụng dùng các phép nối LEFT JOIN, RIGHT JOIN. Các phép nối này sẽ được nói rõ trong phần dưới đây.

3.1.7 Phép nối theo chuẩn SQL-92

Chuẩn SQL2 (SQL-92) đưa ra một cách khác để biểu diễn cho phép nối, trong cách biểu diễn này, điều kiện của phép nối không được chỉ định trong mệnh đề WHERE mà được chỉ định ngay trong mệnh đề FROM của câu lệnh. Cách sử dụng phép nối này cho phép ta biểu diễn phép nối cũng như điều kiện nối được rõ ràng, đặc biệt là trong trường hợp phép nối được thực hiện trên ba bảng trở lên.

Phép nối trong

Điều kiện để thực hiện phép nối trong được chỉ định trong mệnh đề FROM theo cú pháp như sau:

tên_bảng_1 [INNER] JOIN tên_bảng_2 ON điều_kiện_nối

Ví dụ:

Phép nối ngoài

SQL2 cung cấp các phép nối ngoài sau đây:

Phép nối ngoài trái (LEFT OUTER JOIN)

Phép nối ngoài phải (RIGHT OUTER JOIN)

Phép nối ngoài đầy đủ (FULL OUTER JOIN)

Cũng tương tự như phép nối trong, điều kiện của phép nối ngoài cũng được chỉ định ngay trong mệnh đề FROM theo cú pháp:

tên_bảng_1 LEFT|RIGHT|FULL [OUTER] JOIN tên_bảng_2

ON điều_kiện_nối

Ví dụ: Để tìm ra các khách hàng có đặt hàng thay vì sử dụng câu truy vấn sau:

*select **

customers c, orders o

where c.customerid = o.orderid

Ta có thể sử dụng câu truy vấn sau:

*select **

from customers c inner join orders o

on c.customerid = o.customerid

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS	ORDERID	CUSTOMERID	ORDERDATE
6	Cao Van Trung	1959-06-12 00:00:00.000	1	33 Nguyen Trung Truc	1	6	2007-12-06 00:00:00.000
3	Tran Viet Cuong	1980-01-01 00:00:00.000	1	45/2B Da Tuong	2	3	2008-01-01 00:00:00.000

Nếu phép nối ngoài trái hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của bảng bên trái trong phép nối thì phép nối ngoài đầy đủ hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của cả hai bảng tham gia v ào phép nối.

Ví dụ: Giả sử có CSDL như sau:

FACULTYID	FACULTYNAME
1	IT
2	General Sciences
3	Faculty Of Aquaculture
4	Economics

CLASSNAME	FACULTYID
46TH	1
47CB	2
48N	10

Thực hiện phép nối ngoài trái, nối ngoài phải và nối ngoài đầy đủ cho kết quả như sau:

Phép nối ngoài trái:

*select **

from faculty f left join class c

on f.facultyid = c.facultyid

FACULTYID	FACULTYNAME	CLASSNAME	FACULTYID
1	IT	46TH	1
2	General Sciences	47CB	2
3	Faculty Of Aquaculture	NULL	NULL
4	Economics	NULL	NULL

Phép nối ngoài phải:

*select **

from faculty f right join class c

on f.facultyid = c.facultyid

FACULTYID	FACULTYNAME	CLASSNAME	FACULTYID
1	IT	46TH	1
2	General Sciences	47CB	2
NULL	NULL	48N	10

Phép nối ngoài đầy đủ:

*select **

from faculty f full join class c

on f.facultyid = c.facultyid

FACULTYID	FACULTYNAME	CLASSNAME	FACULTYID
1	IT	46TH	1
2	General Sciences	47CB	2
3	Faculty Of Aquaculture	NULL	NULL
4	Economics	NULL	NULL
NULL	NULL	48N	10

Một đặc điểm nổi bật của SQL2 là cho phép biểu diễn phép nối trên nhiều bảng dữ liệu một cách rõ ràng. Thứ tự thực hiện phép nối giữa các bảng được xác định theo nghĩa kết quả của phép nối này được sử dụng trong một phép nối khác.

Ví dụ: Liệt kê tên các mặt hàng có trong đơn đạt hàng có mã là 1.

select i.ITEMNAME, o.ORDERDATE

from (orders o inner join orderdetail od on o.orderid = od.orderid)

inner join items i on od.itemid = i.itemid

where o.orderid = 1

ITEMNAME	ORDERDATE
Nuoc Yen dong hop	2007-12-06 00:00:00.000
PPC	2007-12-06 00:00:00.000

3.1.8 Mệnh đề GROUP BY

Ngoài khả năng thực hiện các yêu cầu truy vấn dữ liệu thông thường (chiếu, chọn, nối,…) như đã đề cập như ở các phần trước, câu lệnh SELECT còn cho phép thực hiện các thao tác truy vấn và tính toán thống kê trên dữ liệu.

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

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

SQL cung cấp các hàm gộp dưới đây:

Hàm gộp	Chức năng
SUM([ALL| DISTINCT] biểu_thức)	Tính tổng các giá trị.
AVG([ALL| DISTINCT] biểu_thức)	Tính trung bình của các giá trị
COUNT([ALL|DISTINCT] biểu_thức)	Đếm số các giá trị trong biểu thức.

COUNT(*)	Đếm số các dòng được chọn.
MAX(biểu_thức)	Tính giá trị lớn nhất
MIN(biểu_thức)	Tính giá trị nhỏ nhất

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

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

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

Mặc định, các hàm gộp thực hiện tính toán thống kê trên toàn bộ dữ liệu. Trong trường hợp cần loại bỏ bớt các giá trị trùng nhau (chỉ giữ lại một giá trị), ta chỉ định thêm từ khoá DISTINCT ở trước biểu thức là đối số của hàm.

Thống kê trên toàn bộ dữ liệu

Khi cần tính toán giá trị thống kê trên toàn bộ dữ liệu, ta sử dụng các hàm gộp trong danh sách chọn của câu lệnh SELECT. Trong trường hợp này, trong danh sách chọn không được sử dụng bất kỳ một tên cột hay biểu thức nào ngoài các hàm gộp.

Ví dụ: Tính tuổi trung bình, tuổi nhỏ nhất và lớn nhất của các khách hàng

```
select  min(year(getdate())-year(BIRTHDAY)) as MINAGE,
        max(year(getdate())-year(BIRTHDAY)) as MAXAGE,
        avg(year(getdate())-year(BIRTHDAY)) as AVGAGE
from customers
```

MINAGE	MAXAGE	AVGAGE
26	68	43

Thống kê trên nhóm

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

Ví dụ: Câu truy vấn sau cho biết số tổng số tiền khách hàng phải trả cho tất cả các lần đặt hàng

```
select c.CUSTOMERID, c.CUSTOMERNAME,
convert(varchar(20),cast(SUM(i.UNITPRICE*od.QUANTITY) as money),1) as SUMTOTAL
from    customers c inner join orders o on o.customerid = c.customerid
        inner join orderdetail od on o.orderid = od.orderid
        inner join items i on i.itemid = od.itemid
group by c.customerid, c.customername
```

CUSTOMERID	CUSTOMERNAME	SUMTOTAL
3	Tran Viet Cuong	60,600,000.00
6	Cao Van Trung	100,600,000.00

Nếu muốn hiện số tiền khách hàng phải trả cho từng đơn đặt hàng, chỉ cần thêm trường ORDERID vào mệnh đề group by.

select c.CUSTOMERID, c.CUSTOMERNAME,

*convert(varchar(20),cast(SUM(i.UNITPRICE*od.QUANTITY)as money),1) as SUMTOTAL*

from customers c inner join orders o on o.customerid = c.customerid

inner join orderdetail od on o.orderid = od.orderid

inner join items i on i.itemid = od.itemid

group by c.customerid, c.customername, o.orderid

CUSTOMERID	CUSTOMERNAME	SUMTOTAL
6	Cao Van Trung	100,600,000.00
3	Tran Viet Cuong	10,200,000.00
3	Tran Viet Cuong	50,400,000.00

Lưu ý: Trong trường hợp danh sách chọn của câu lệnh SELECT có cả các hàm gộp và những biểu thức không phải là hàm gộp thì những biểu thức này phải có mặt đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Mệnh đề HAVING chỉ định điều kiện trong hàm gộp

Mệnh đề HAVING được sử dụng nhằm chỉ định điều kiện đối với các giá trị thống kê được sản sinh từ các hàm gộp tương tự như cách thức mệnh đề WHERE thiết lập các điều kiện cho câu lệnh SELECT. Mệnh đề HAVING thường không thực sự có nghĩa nếu như không sử dụng kết hợp với mệnh đề GROUP BY. Một điểm khác biệt giữa HAVING và WHERE là trong điều kiện của WHERE không được có các hàm gộp trong khi HAVING lại cho phép sử dụng các hàm gộp trong điều kiện của mình.

Ví dụ: Tìm ra các khách hàng có tổng số tiền phải thanh toán cho tất cả các lần đặt hàng lớn hơn 100 triệu.

select c.CUSTOMERID, c.CUSTOMERNAME,

*convert(varchar(20),cast(SUM(i.UNITPRICE*od.QUANTITY)as money),1) as SUMTOTAL*

from customers c inner join orders o on o.customerid = c.customerid

inner join orderdetail od on o.orderid = od.orderid

inner join items i on i.itemid = od.itemid

group by c.customerid, c.customername

*having sum(i.UNITPRICE*od.QUANTITY) > 100000000*

CUSTOMERID	CUSTOMERNAME	SUMTOTAL
6	Cao Van Trung	100,600,000.00

3.1.9 Truy vấn con (Subquery)

Truy vấn con là một câu lệnh SELECT được lồng vào bên trong một câu lệnh

SELECT, INSERT, UPDATE, DELETE hoặc bên trong một truy vấn con khác. Loại truy vấn này được sử dụng để biểu diễn cho những truy vấn trong đó điều kiện truy vấn dữ liệu cần phải sử dụng đến kết quả của một truy vấn khác.

Cú pháp của truy vấn con như sau:

(SELECT [ALL | DISTINCT] danh_sách_chọn

FROM danh_sách_bảng

[WHERE điều_kiện]

[GROUP BY danh_sách_cột]

[HAVING điều_kiện])

Khi sử dụng truy vấn con cần lưu ý một số quy tắc sau:

Một truy vấn con phải được viết trong cặp dấu ngoặc. Trong hầu hết các trường hợp, một truy vấn con thường phải có kết quả là một cột (tức là chỉ có duy nhất một cột trong danh sách chọn).

Mệnh đề COMPUTE và ORDER BY không được phép sử dụng trong truy vấn con.

Các tên cột xuất hiện trong truy vấn con có thể là các cột của các bảng trong truy vấn ngoài.

Một truy vấn con thường được sử dụng làm điều kiện trong mệnh đề WHERE hoặc HAVING của một truy vấn khác.

Nếu truy vấn con trả về đúng một giá trị, nó có thể sử dụng như là một thành phần bên trong một biểu thức (chẳng hạn xuất hiện trong một phép so sánh bằng)

Phép so sánh đối với với kết quả truy vấn con

Kết quả của truy vấn con có thể được sử dụng đề thực hiện phép so sánh số học với một biểu thức của truy vấn cha. Trong trường hợp này, truy vấn con được sử dụng dưới dạng:

WHERE biểu_thức phép_toán_số_học [ANY|ALL] (truy_vấn_con)

Trong đó phép toán số học có thể sử dụng bao gồm: =, <>, >, <, >=, <=; V à truy vấn con phải có kết quả bao gồm đúng một cột.

Ví dụ: Câu truy vấn sau đây tìm tên khách hàng có tuổi lớn nhất

select c.CUSTOMERNAME, c.ADDRESS

from customers c

where year(getdate()) - year(BIRTHDAY) =

(select max(year(getdate()) - year(BIRTHDAY))

from customers)

CUSTOMERNAME	ADDRESS
Nguyen Thanh Thai	29A Phuong Sai

Nếu truy vấn con trả về nhiều hơn một giá trị, việc sử dụng phép so sánh như trên sẽ không hợp lệ. Trong trường hợp này, sau phép toán so sánh phải sử dụng thêm lượng từ ALL hoặc ANY. Lượng từ ALL được sử dụng khi cần so sánh giá trị của biểu thức với tất cả các giá trị trả về trong kết quả của truy vấn con; ngược lai, phép so sánh với lượng từ ANY có kết quả đúng khi chỉ cần một giá trị bất kỳ nào đó trong kết quả của truy vấn con thoả mãn điều kiện

Ví dụ:

Toán tử IN/NOT IN

Khi cần thực hiện phép kiểm tra giá trị của một biểu thức có xuất hiện (không xuất hiện) trong tập các giá trị của truy vấn con hay không, ta có thể sử dụng toán tử IN (NOT IN) như sau:

WHERE biểu_thức [NOT] IN (truy_vấn_con)

Ví dụ:

Truy vấn con với EXISTS

Lượng từ EXISTS được sử dụng kết hợp với truy vấn con dưới dạng:

WHERE [NOT] EXISTS (truy_vấn_con)

Lượng từ EXISTS (tương ứng NOT EXISTS) trả về giá trị True (tương ứng False) nếu kết quả của truy vấn con có ít nhất một dòng (tương ứng không có dòng nào). Điều khác biệt của việc sử dụng EXISTS với hai cách đã nêu ở trên là trong danh sách chọn của truy vấn con có thể có nhiều hơn hai cột.

Ví dụ:

Truy vấn con và mệnh đề HAVING

Một truy vấn con có thể được sử dụng trong mệnh đề HAVING của một truy vấn khác. Trong trường hợp này, kết quả của truy vấn con được sử dụng để tạo nên điều kiện đối với các hàm gộp.

Ví dụ:

3.2 Thêm, cập nhật và xóa dữ liệu

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

Lệnh INSERT

Lệnh UPDATE

Lệnh DELETE

3.2.1 Thêm dữ liệu

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

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

Thêm nhiều dòng dữ liệu bằng cách truy xuất dữ liệu từ các bảng dữ liệu khác.

Thêm từng dòng dữ liệu

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

INSERT INTO tên_bảng[(danh_sách_cột)] VALUES(danh_sách_trị)

Trong câu lệnh INSERT, danh sách cột ngay sau tên bảng không cần thiết phải chỉ định nếu giá trị các trường của bản ghi mới được chỉ định đầy đủ trong danh sách trị. Trong trường hợp này, thứ tự các giá trị trong danh sách trị phải bằng với số lượng các trường của bảng cần bổ sung dữ liệu cũng như phải tuân theo đúng thứ tự của các trường như khi bảng được định nghĩa

Ví dụ: Thêm thông tin một khách hàng mới vào bảng Customer

insert into customers (customername, birthday, gender, address)

values('Nguyen Van An', '4/2/1976', 'True', '14 Thong Nhat')

hoặc

insert into customers

values('Nguyen Van An', '4/2/1976', 'True', '14 Thong Nhat')

Lưu ý: Trường CUSTOMERID được thiết lập identity là “YES” nên ta không cần thêm giá trị trường này mà SQL sẽ tự động tạo ra một giá trị cho trường này. Chi tiết về identity sẽ nói trong chương 4.

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

Thêm một tập các dòng dữ liệu vào bảng

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

Cú pháp câu lệnh INSERT có dạng như sau:

INSERT INTO tên_bảng[(danh_sách_cột)] câu_lệnh_SELECT

Ví dụ:

insert into Customers_Backup

*select * from Customers*

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

3.2.2 Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng. Câu lệnh này có cú pháp như sau:

UPDATE tên_bảng

SET tên_cột = biểu_thức

[, ..., tên_cột_k = biểu_thức_k]

[FROM danh_sách_bảng]

[WHERE điều_kiện]

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

Ví dụ:

update customers

set customername = 'Cao Van Chung'

where customerid = 9

Trong câu lệnh UPDATE có thể sử dụng CASE…WHEN.

Ví dụ:

*select **

into tmp1

from customers

update tmp1

set address = case when customerid < 2 then 'Nguyen Trung Truc'

else 'Nguyen Thi Minh Khai'

end

3.2.3 Xóa dữ liệu

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

DELETE FROM tên_bảng

[FROM danh_sách_bảng]

[WHERE điều_kiện]

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

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

Ví dụ:

delete from Items

where itemid = 3

Xoá dữ liệu khi điều kiện liên quan đến nhiều bảng

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

Ví dụ:

delete

from orderdetail

from items

where items.itemid = orderdetail.itemid

and items.itemname = 'LAPTOP'

Sử dụng truy vấn con trong câu lệnh DELETE

Một câu lệnh SELECT có thể được lồng vào trong mệnh đề WHERE trong câu lệnh DELETE để làm điều kiện cho câu lệnh tương tự như câu lệnh UPDATE.

Ví dụ:

delete

from orderdetail

from items

where items.itemid = (select i.itemid

from items i inner join orderdetail od

on i.itemid = od.itemid

WHERE itemname = 'LAPTOP')

Xoá toàn bộ dữ liệu trong bảng

Câu lệnh DELETE không chỉ định điều kiện đối với các d òng dữ liệu cần xoá trong mệnh đề WHERE sẽ xoá to àn bộ dữ liệu trong bảng. Thay vì sử dụng câu lệnh DELETE trong tr ường hợp này, ta có thể sử dụng câu lệnh TRUNCATE có cú pháp nh ư sau:

TRUNCATE TABLE tên_b ảng

Ví dụ:

truncate table tmp1

4 Ngôn ngữ định nghĩa dữ liệu – DDL

Trong chương này sẽ đề cập đến nhóm các câu lệnh được sử dụng để định nghĩa và quản lý các đối tượng CSDL như bảng, khung nhìn, chỉ mục,... và được gọi là ngôn ngữ định nghĩa dữ liệu (DDL).

Về cơ bản, ngôn ngữ định nghĩa dữ liệu bao gồm các lệnh:

CREATE: định nghĩa và tạo mới đối tượng CSDL.

ALTER: thay đổi định nghĩa của đối tượng CSDL.

DROP: Xoá đối tượng CSDL đã có.

4.1 Tạo bảng

Câu lệnh CREATE TABLE được sử dụng để định nghĩa một bảng dữ liệu mới trong CSDL. Khi định nghĩa một bảng dữ liệu mới, ta cần phải xác định được các yêu cầu sau đây:

Bảng mới được tạo ra sử dụng với mục đích gì và có vai trò như thế nào trong cơ sở dữ liệu.

Cấu trúc của bảng bao gồm những trường (cột) nào, mỗi một trường có ý nghĩa như thế nào trong việc biểu diễn dữ liệu, kiểu dữ liệu của mỗi trường là gì và trường đó có cho phép nhận giá trị NULL hay không.

Những trường nào sẽ tham gia vào khóa chính của bảng. Bảng có quan hệ với những bảng khác hay không và nếu có thì quan hệ như thế nào.

Trên các trường của bảng có tồn tại những ràng buộc về khuôn dạng, điều kiện hợp lệ của dữ liệu hay không; nếu có thì sử dụng ở đâu và như thế nào.

Câu lệnh CREATE TABLE có cú pháp như sau

CREATE TABLE tên_bảng

(

tên_cột thuộc_tính_cột các_ràng_buộc

[,...

,tên_cột_n thuộc_tính_cột_n các_ràng_buộc_cột_n]

[,các_ràng_buộc_trên_bảng]

)

Tên_bảng: tuân theo quy tắc định danh, không vượt quá 128 ký tự

Tên_cột: các cột trong bảng, mỗi bảng có ít nhất một cột.

Thuộc_tính_cột: bao gồm kiểu dữ liệu của cột, giá trị mặc định của cột, cột có được thiết lập thuộc tính *identity*, cột có chấp nhận giá trị NULL hay không. Trong đó kiểu dữ liệu là thuộc tính bắt buộc.

Các_ràng_buộc: gồm các ràng buộc về khuôn dạng dữ liệu (ràng buộc CHECK) hay các ràng buộc về bào toàn dữ liệu (PRIMARY KEY, FOREIGN KEY, UNIQUE)

Ví dụ: Ví dụ dưới đây tạo một bảng có tên CUSTOMERS

```
create table customers
(
  customerid int identity (1,1) primary key,
  customername nvarchar(50) not null,
  address nvarchar(100 ) null ,
  birthday datetime null,
  gender bit default('true') not null
)
```

Cột *customerid* có kiểu dữ liệu *int*, được chỉ định thuộc tính *identity(1,1)* nghĩa là dữ liệu cột này được thêm tự động bắt đầu từ 1 và mỗi lần có dòng mới thêm vào, giá trị cột này được tăng lên 1. Cột này cũng được chỉ định làm khóa chính của bảng thông qua thuộc tính *primary key*

Thuộc tính NULL/ NOT NULL chỉ ra rằng cột đó có chấp nhận/ không chấp nhận giá trị NULL.

Cột *gender* được chỉ định giá trị mặc định là *true* nghĩa là nếu không chỉ định giá trị cho cột này thì cột này có giá trị là *true*

Ví dụ:

Thêm dòng mới vào bảng customers với giá trị truyền vào đầy đủ cho các cột

```
insert into customers
values('Nguyen Van An', '22 Nguyen Thien Thuat', '5/5/1988', 'True')
```

Thêm dòng mới vào bảng customers sử dụng giá trị mặc định

```
insert into customers (customername, address, birthday)
values('Nguyen Van An', '22 Nguyen Thien Thuat', '5/5/1988')
```

Thêm dòng mới vào bảng customers và không truyền giá trị cho các cột cho phép giá trị NULL

```
insert into customers (customername )
values('Nguyen Van An')
```

customerid	customername	address	birthday	gender
1	Nguyen Van An	22 Nguyen Thien Thuat	1988-05-05 00	Click to select the whc
2	Nguyen Van An	22 Nguyen Thien Thuat	1988-05-05 00:00:00.000	1
3	Nguyen Van An	NULL	NULL	1

4.2 Các loại ràng buộc

4.2.1 Ràng buộc CHECK

Ràng buộc CHECK được sử dụng nhằm chỉ định điều kiện hợp lệ đối với dữ liệu. Mỗi khi có sự thay đổi dữ liệu trên bảng (INSERT, UPDATE), những ràng buộc này sẽ được sử dụng nhằm kiểm tra xem dữ liệu mới có hợp lệ hay không.

Ràng buộc CHECK được khai báo theo cú pháp như sau:

[CONSTRAINT tên_ràng_buộc] CHECK (điều_kiện)

Ví dụ:

```
create table students
(
  studentid int identity(1,1) primary key,
  studentname nvarchar(50) not null,
  address nvarchar(100) not null,
  score1 tinyint not null
  constraint chk_score1 CHECK (score1 >= 0 and score1 <= 10),
  score2 tinyint not null
  constraint chk_score2 CHECK (score2 between 0 and 10),
  score3 tinyint not null
  constraint chk_score3 CHECK (score3 in (1,2,3,4,5,6,7,8,9,10)),
)
```

Thực hiện việc thêm một dòng có dữ liệu không thỏa điều kiện

```
insert into students
values('Nguyen Van Dung', '12 Tran Quang Khai', 10, 10, -2)
```

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chk_score3".
The conflict occurred in database "abc", table "dbo.students", column 'score3'.
The statement has been terminated.
```

Có thể gộp chung các ràng buộc CHECK lại trong một ràng buộc duy nhất như sau

```
create table students
(
  studentid int identity(1,1) primary key,
  studentname nvarchar(50) not null,
  address nvarchar(100) not null,
  score1 tinyint not null ,
  score2 tinyint not null,
```

```
  score3 tinyint not null,
  constraint chk_score  CHECK(
  (score1>= 0 and score1 <=10)
  and (score2 between 0 and 10)
  and (score3 in (1,2,3,4,5,6,7,8,9,10)))
)
```

4.2.2 Ràng buộc PRIMARY KEY

Ràng buộc PRIMARY KEY được sử dụng để định nghĩa khoá chính của bảng. Khoá chính của một bảng là một hoặc một tập nhiều cột mà giá trị của chúng là duy nhất trong bảng. Hay nói cách khác, giá trị của khoá chính sẽ giúp cho ta xác định được duy nhất một dòng (bản ghi) trong bảng dữ liệu. Mỗi một bảng chỉ có thể có duy nhất một khoá chính và bản thân khoá chính không chấp nhận giá trị NULL. Ràng buộc PRIMARY KEY là cơ sở cho việc đảm bảo tính toàn vẹn thực thể cũng như toàn vẹn tham chiếu.

Để khai báo một ràng buộc PRIMARY KEY, ta sử dụng cú pháp như sau:

[CONSTRAINT tên_ràng_buộc] PRIMARY KEY [(danh_sách_cột)]

Nếu khoá chính của bảng chỉ bao gồm đúng một cột và ràng buộc PRIMARY KEY được chỉ định ở mức cột, ta không cần thiết phải chỉ định danh sách cột sau từ khoá PRIMARY KEY. Tuy nhiên, nếu việc khai báo khoá chính được tiến hành ở mức bảng (sử dụng khi số lượng các cột tham gia vào khoá là từ hai trở lên) thì bắt buộc phải chỉ định danh sách cột ngay sau từ khóa PRIMARY KEY và tên các cột được phân cách nhau bởi dấu phẩy.

Ví dụ 1: Định nghĩa một bảng chỉ có một khóa chính

```
create table customers
(
  customerid int identity(1,2)
  constraint chk_primarykey primary key,
  customername nvarchar(50) not null,
  address nvarchar(100) not null,
  gender bit not null
)
```

Hoặc là

```
create table customers
(
  customerid int identity(1,2)  primary key,
  customername nvarchar(50) not null,
```

```
  address nvarchar(100) not null,
  gender bit not null
)
```

Ví dụ 2: Định nghĩa bảng có hai khóa chính:

```
create table orderdetail
(
  customerid int,
  orderid int,
  itemid int not null,
  quantity decimal(8,2) not null,
  constraint chk_primarykey primary key (customerid, orderid)
)
```

4.2.3 Ràng buộc FOREIGN KEY

FOREIGN KEY là một cột hay một sự kết hợp của nhiều cột được sử dụng để áp đặt mối liên kết dữ liệu giữa hai table. FOREIGN KEY của một bảng sẽ giữ giá trị của PRIMARY KEY của một bảng khác và chúng ta có thể tạo ra nhiều FOREIGN KEY trong một table.

FOREIGN KEY có thể tham chiếu vào PRIMARY KEY hay cột có ràng buộc duy nhất. FOREIGN KEY có thể chứa giá trị NULL. Mặc dù mục đích chính của ràng buộc FOREIGN KEY là để kiểm soát dữ liệu chứa trong bảng có FOREIGN KEY (tức table con) nhưng thực chất nó cũng kiểm soát luôn cả dữ liệu trong bảng chứa PRIMARY KEY (tức table cha). Ví dụ nếu ta xóa dữ liệu trong bảng cha thì dữ liệu trong bảng con trở nên "mồ côi" (orphan) vì không thể tham chiếu ngược về bảng cha. Do đó ràng buộc FOREIGN KEY sẽ đảm bảo điều đó không xảy ra. Nếu bạn muốn xóa dữ liệu trong bảng cha thì trước hết bạn phải xóa hay vô hiệu hóa ràng buộc FOREIGN KEY trong bảng con trước.

Ràng buộc FOREIGN KEY được định nghĩa theo cú pháp dưới đây:

```
[CONSTRAINT tên_ràng_buộc] FOREIGN KEY [(danh_sách_cột)]
REFERENCES tên_bảng_tham_chiếu(danh_sách_cột_tham_chiếu)
[ON DELETE CASCADE | NO ACTION | SET NULL | SET DEFAULT]
[ON UPDATE CASCADE | NO ACTION | SET NULL | SET DEFAULT]
```

Việc định nghĩa một ràng buộc FOREIGN KEY bao gồm các yếu tố sau:

Tên cột hoặc danh sách cột của bảng được định nghĩa tham gia vào khoá ngoài.

Tên của bảng được tham chiếu bởi khoá ngoài và danh sách các cột được tham chiếu đến trong bảng tham chiếu.

Cách thức xử lý đối với các bản ghi trong bảng được định nghĩa trong trường hợp các bản ghi được tham chiếu trong bảng tham chiếu bị xoá (ON DELETE) hay cập nhật (ON UPDATE). SQL chuẩn đưa ra 4 cách xử lý

CASCADE: Tự động xoá (cập nhật) nếu bản ghi được tham chiếu bị xoá (cập nhật).

NO ACTION: (Mặc định) Nếu bản ghi trong bảng tham chiếu đang được tham chiếu bởi một bản ghi bất kỳ trong bảng được định nghĩa thì bàn ghi đó không được phép xoá hoặc cập nhật (đối với cột được tham chiếu).

SET NULL: Cập nhật lại khoá ngoài của bản ghi thành giá trị NULL (nếu cột cho phép nhận giá trị NULL).

SET DEFAULT: Cập nhật lại khoá ngoài của bản ghi nhận giá trị mặc định (nếu cột có qui định giá trị mặc định).

Ví dụ:

```
drop table orderdetail
create table orderdetail
(
  orderid int
  constraint fk_orderdetail_orders foreign key references orders(orderid)
  on delete cascade
  on update cascade,
  customerid int
  constraint fk_orderdetail_customer foreign key references customers(customerid)
  on delete cascade
  on update cascade,
  itemid int
  constraint fk_orderdetail_items foreign key references items(itemid)
  on delete cascade
  on update cascade,
  quantity decimal(18,2) not null,
)
```

4.3 Sửa đổi định nghĩa bảng

Một bảng sau khi đã được định nghĩa bằng câu lệnh CREATE TABLE có thể được sửa đổi thông qua câu lệnh ALTER TABLE. Câu lệnh này cho phép thực hiện được các thao tác sau:

Bổ sung một cột vào bảng.

Xoá một cột khỏi bảng.

Thay đổi định nghĩa của một cột trong bảng.

Xoá bỏ hoặc bổ sung các ràng buộc cho bảng

Cú pháp của câu lệnh ALTER TABLE như sau:

ALTER TABLE tên_bảng

ADD định_nghĩa_cột |

ALTER COLUMN tên_cột kiểu_dữ_liệu [NULL | NOT NULL]

DROP COLUMN tên_cột |

ADD CONSTRAINT tên_ràng_buộc định_nghĩa_ràng_buộc

DROP CONSTRAINT tên_ràng_buộc

Ví dụ 1: Thêm một cột mới vào bảng ORDERS

alter table orders

add description nvarchar(100) not null

Ví dụ 2: Thay đổi định nghĩa cột *desciption*

alter table orders

alter column description nvarchar(200) null

Ví dụ 3: Thêm ràng buộc CHECK vào cột decription

alter table orders

add constraint chk_descriptionlength CHECK (len(description) > 10)

Ví dụ 4: Xóa ràng buộc CHECK

alter table orders

drop chk_descriptionlength

Ví dụ 5: Xóa cột description

alter table orders

drop column description

Ví dụ 6: Thêm một cột mới vào bảng orders và thêm ràng buộc cho cột này

alter table orders

add

description nvarchar(100) null,

constraint chk_descriptionlength CHECK (len(description) > 0)

Nếu bổ sung thêm một cột vào bảng và trong bảng đã có ít nhất một bản ghi thì cột mới cần bổ sung phải cho phép chấp nhận giá trị NULL hoặc phải có giá trị mặc định.

Muốn xoá một cột đang được ràng buộc bởi một ràng buộc hoặc đang được tham chiếu bởi một khoá ngoài, ta phải xoá ràng buộc hoặc khoá ngoài trước sao cho trên cột không còn bất kỳ một ràng buộc và không còn được tham chiếu bởi bất kỳ khoá ngoài nào.

Nếu bổ sung thêm ràng buộc cho một bảng đã có dữ liệu và ràng buộc cần bổ sung không được thoả mãn bởi các bản ghi đã có trong bảng thì câu lệnh ALTER TABLE không thực hiện được.

4.4 Xóa bảng

Khi một bảng không còn cần thiết , ta có thể xoá nó ra khỏi cơ sở dữ liệu bằng câu lệnh DROP TABLE. Câu lệnh này cũng đồng thời xoá tất cả những ràng buộc, chỉ mục, trigger liên quan đến bảng đó.

Câu lệnh có cú pháp như sau:

DROP TABLE tên_bảng

Trong các hệ quản trị cơ sở dữ liệu, khi đã xoá một bảng bằng lệnh DROP TABLE, ta không thể khôi phục lại bảng cũng như dữ liệu của nó. Do đó, cần phải cẩn thận khi sử dụng câu lệnh này.

Câu lệnh DROP TABLE không thể thực hiện được nếu bảng cần xoá đang được tham chiếu bởi một ràng buộc FOREIGN KEY. Trong trường hợp này, ràng buộc FOREIGN KEY đang tham chiếu hoặc bảng đang tham chiếu đến bảng cần xoá phải được xoá trước.

Khi một bảng bị xoá, tất cả các ràng buộc, chỉ mục và trigger liên quan đến bảng cũng đồng thời bị xóa theo. Do đó, nếu ta tạo lại bảng thì cũng phải tạo lại các đối tượng này.

Ví dụ: Để xóa bảng ORDERS trước tiên ta phải xóa ràng buộc FOREIGN KEY từ bảng ORDERDETAIL

alter table orderdetail

drop constraint fk_orderdetail_orders

Sau đó xóa bảng ORDERS

drop table orders

4.5 Khung nhìn - VIEW

Khung nhìn là một bảng tạm thời, có cấu trúc như một bảng, khung nhìn không lưu trữ dữ liệu mà nó được tạo ra khi sử dụng, khung nhìn là đối tượng thuộc CSDL.

Khung nhìn được tạo ra từ câu lệnh truy vấn dữ liệu (lệnh SELECT), truy vấn từ một hoặc nhiều bảng dữ liệu.

Khung nhìn được sử dụng khai thác dữ liệu như một bảng dữ liệu, chia sẻ nhiều người dùng, an toàn trong khai thác, không ảnh hưởng dữ liệu gốc.

Có thể thực hiện truy vấn dữ liệu trên cấu trúc của khung nhìn.

Như vậy, một khung nhìn trông giống như một bảng với một tên khung nhìn và là một tập bao gồm các dòng và các cột. Điểm khác biệt giữa khung nhìn và bảng là khung nhìn không được xem là một cấu trúc lưu trữ dữ liệu tồn tại trong cơ sở dữ liệu. Thực chất dữ liệu quan sát được trong khung nhìn được lấy từ các bảng thông qua câu lệnh truy vấn dữ liệu.

Câu lệnh CREATE VIEW được sử dụng để tạo ra khung nhìn và có cú pháp như sau:

CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)] AS

câu_lệnh_SELECT

Ví dụ:

```
create view CUSTOMERINFO
as
select CUSTOMERNAME, (year(getdate()) - year(BIRTHDAY)) as AGE, ADDRESS
from customers
```

Thực hiện câu truy vấn trên khung nhìn vừa tạo ra:

```
select * from customerinfo
```

CUSTOMERNAME	AGE	ADDRESS
Cao Van Trung	49	33 Nguyen Trung Truc
Tran Van Phuc	38	99 Nguyen Thi Minh Khai
Tran Viet Cuong	28	45/2B Da Tuong
Nguyen Van Dai	53	76 Tran Phu
Le Thi Hoa	26	56 Le Hong Phong
Nguyen Thanh Thai	68	29A Phuong Sai
Cao Van Chung	NULL	12 Nguyen Thien Thuat
NGuyen Van An	32	14 Thong Nhat
NGuyen Van An	32	14 Thong Nhat

Nếu trong câu lệnh CREATE VIEW, ta không chỉ định danh sách các tên cột cho khung nhìn, tên các cột trong khung nhìn sẽ chính là tiêu đề các cột trong kết quả của câu lệnh SELECT. Trong trường hợp tên các cột của khung nhìn đươc chỉ định, chúng phải có cùng số lượng với số lượng cột trong kết quả của câu truy vấn.

Ví dụ:

```
create view CUSTOMERINFO (CUSTOMERNAME, AGE, ADDRESS)
as
select CUSTOMERNAME,  year(getdate()) - year(BIRTHDAY), ADDRESS
from customers
```

Lưu ý:

Phải đặt tên cho các cột của khung nhìn trong các trường hợp sau đây:

Trong kết quả của câu lệnh SELECT có ít nhất một cột được sinh ra bởi một biểu thức (tức là không phải là một tên cột trong bảng cơ sở) và cột đó không được đặt tiêu đề.

Tồn tại hai cột trong kết quả của câu lệnh SELECT có cùng tiêu đề cột.

4.6 Thêm, cập nhật, xóa dữ liệu trong VIEW

Đối với một số khung nhìn, ta có thể tiến hành thực hiện các thao tác cập nhập, thêm và xoá dữ liệu. Thực chất, những thao tác này sẽ được chuyển thành những thao tác trên các bảng cơ sở và có tác động đến những bảng cơ sở.

Về mặt lý thuyết, để có thể thực hiện thao tác bổ sung, cập nhật và xoá, một khung nhìn trước tiên phải thoả mãn các điều kiện sau đây:

Trong câu lệnh SELECT định nghĩa khung nhìn không được sử dụng từ khoá DISTINCT, TOP, GROUP BY và UNION.

Các thành phần xuất hiện trong danh sách chọn của câu lệnh SELECT phải là các cột trong các bảng cơ sở. Trong danh sách chọn không được chứa các biểu thức tính toán, các hàm gộp.

Ngoài những điều kiện trên, các thao tác thay đổi đến dữ liệu thông qua khung nhìn còn phải đảm bảo thoả mãn các ràng buộc trên các bảng cơ sở, tức là vẫn đảm bảo tính toàn vẹn dữ liệu.

Mặc dù thông qua khung nhìn có thể thực hiện được thao tác bổ sung và cập nhật dữ liệu cho bảng cơ sở nhưng chỉ hạn chế đối với những khung nhìn đơn giản. Đối với những khung nhìn phức tạp thì thường không thực hiện được; hay nói cách khác là dữ liệu trong khung nhìn là chỉ đọc.

4.7 Thay đổi định nghĩa khung nhìn

Câu lệnh ALTER VIEW dùng để định nghĩa lại khung nhìn có cấu trúc như sau:

ALTER VIEW tên_khung_nhìn [(danh_sách_tên_cột)] AS

Câu_lệnh_SELECT

Ví dụ: Ví dụ dưới đây định nghĩa lại khung nhìn CUSTOMERINFO

```
alter view customerinfo
as
select CUSTOMERNAME, (year(getdate())  - year(birthday)) as AGE,
ADDRESS, GENDER
from customers
```

Lưu ý: lệnh CREATE VIEW không làm thay đổi các quyền đã được cấp phát cho người sử dụng trước đó.

4.8 Xóa khung nhìn

Câu lệnh DROP VIEW dùng để xóa khung nhìn có cấu trúc như sau:

DROP VIEW tên_khung_nhìn

Ví dụ:

drop view customerinfo

Lưu ý: Nếu một khung nhìn bị xoá, toàn bộ những quyền đã cấp phát cho người sử dụng trên khung nhìn cũng đồng thời bị xoá. Do đó, nếu ta tạo lại khung nhìn thì phải tiến hành cấp phát lại quyền cho người sử dụng.

5 Thủ tục lưu trữ, hàm và trigger

5.1 Thủ tục lưu trữ (Stored procedure)

Thủ tục lưu trữ là một đối tượng trong CSDL, bao gồm nhiều câu lệnh T-SQL được tập hợp lại với nhau thành một nhóm, và tất cả các lệnh này sẽ được thực thi khi thủ tục lưu trữ được thực thi.

Với thủ tục lưu trữ, một phần nào đó khả năng của ngôn ngữ lập trình được đưa vào trong ngôn ngữ SQL. Thủ tục lưu trữ có thể có các thành phần sau:

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

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

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

Lợi ích của việc sử dụng thủ tục lưu trữ:

SQL Server chỉ biên dịch các thủ tục lưu trữ một lần và sử dụng lại kết quả biên dịch này trong các lần tiếp theo trừ khi người dùng có những thiết lập khác. Việc sử dụng lại kết quả biên dịch không làm ảnh hưởng đến hiệu suất hệ thống khi thủ tục lưu trữ được gọi liên tục nhiều lần.

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

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

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

Các thủ tục lưu trữ trả về kết quả theo 4 cách:

Sử dụng các tham số output

Sử dụng các lệnh trả về giá trị, các lệnh này luôn trả về giá trị số nguyên.

Tập các giá trị trả về của mỗi câu lệnh SELECT có trong thủ tục lưu trữ hoặc của quá trình gọi một thủ tục lưu trữ khác trong một thủ tục lưu trữ.

Một biến con trỏ toàn cục có thể tham chiếu từ bên ngoài thủ tục.

5.1.1 Tạo thủ tục lưu trữ

Thủ tục lưu trữ được tạo thông qua câu lệnh CREATE PROCEDURE.

CREATE PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]

[WITH RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION]

AS

Các_câu_lệnh_của_thủ_tục

Trong đó:

WITH RECOMPILE: yêu cầu SQL Server biên dịch lại thủ tục lưu trữ mỗi khi được gọi.

WITH ENCRYPTION: yêu cầu SQL Server mã hóa thủ tục lưu trữ.

Các_câu_lệnh_của_thủ_tục: Các lệnh T-SQL. Các lệnh này có thể nằm trong cặp BEGIN…END hoặc không.

Ví dụ: Giả sử cần thực hiện các công việc theo thứ tự như sau:

Nhập một đơn đặt hàng mới của khách hàng có mã khách hàng là 3

Nhập các chi tiết đơn đặt hàng cho đơn đặt hàng trên.

Để thực hiện các công việc trên chúng ta cần các câu lệnh như sau:

Trước tiên nhập đơn đặt hàng cho khách hàng có mã khách hàng là 3

insert into orders

values(3, '7/22/2008')

Tiếp theo thêm các chi tiết đơn đặt hàng cho hóa đơn này. Giả sử rằng đơn đặt hàng có mã là 4 và khách hàng đặt một mặt hàng có mã là 1.

insert into orderdetail

values(4, 1, 10)

Cách viết như trên có hạn chế là: trong quá trình làm việc sẽ có rất nhiều đơn đặt hàng mới, do đó người dùng sẽ phải viết đi viết lại những câu lệnh tương tự nhau cho các khách hàng khác nhau. Một cách giải quyết vấn đề này là dùng thủ tục lưu trữ và dùng tham số để nhận các thông tin thay đổi.

create procedure sp_InsertOrderAndOrderDetail

@customerid int,

@orderdate datetime,

@orderid int,

@itemid int,

@quantity decimal,

as

begin

insert into orders

values(@customerid, @orderdate)

insert into orderdetail

values(@orderid, @itemid, @quantity)

end

Thực hiện thủ tực lưu trữ này như sau:

sp_InsertOrderAndOrderDetail '3', '22/7/2008', '4', '1', '10')

5.1.2 Lời gọi thủ tục

Thủ tục lưu trữ được gọi theo cấu trúc

Tên_thủ_tục_lưu _trữ [danh_sách_tham_số]

Cần lưu ý là danh sách tham số truyền vào trong lời gọi phải theo đúng thứ tự khai báo các tham số trong thủ tục lưu trữ.

Nếu thủ tục được gọi từ một thủ tục khác, thực hiện bên trong một trigger hay phối hợp với câu lệnh SELECT, cấu trúc như sau;

Exec *Tên_thủ_tục_lưu _trữ [danh_sách_tham_số]*

5.1.3 Biến trong thủ tục lưu trữ

Trong thủ tục lưu trũ có thể có các biến nhằm lưu các kết quả tính toán hay truy xuất từ CSDL. Các biến trong thủ tục được khai báo bằng từ khóa DECLARE theo cấu trúc như sau:

DECLARE @tên_biến kiểu_dữ_liệu

Ví dụ:

create procedure sp_SelectCustomerWithMaxAge

as

begin

declare @maxAge int

select @maxAge = max(year(getdate())-year(BIRTHDAY))

from customers

select CUSTOMERNAME, BIRTHDAY

from customers

where year(getdate())-year(BIRTHDAY)=@maxAge

end

5.1.4 Giá trị trả về trong thủ tục lưu trữ

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

Ví dụ: Có thủ tục lưu trữ như sau

create procedure sp_TestOutput
@a int,
@b int,
@c int
as
select @c = @a + @b

Thực thi thủ tục:

Declare @tong int
set @tong = 0
sp_TestOutput 100, 200, @tong
select @tong

Kết quả là 0.

Sử dụng tham số OUTPUT

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

@tên_tham_số kiểu_dữ_liệu OUTPUT

Ví dụ trên được viết lại như sau:

create procedure sp_TestOutput
@a int,
@b int,
@c int output
as
select @c = @a + @b

Thực thi thủ tục:

Declare @tong int
set @tong = 0
sp_TestOutput 100, 100, @tong output
select @tong

Kết quả là 200.

Sử dụng lệnh RETURN

Tương nhự như việc sử dụng tham số OUTPUT, câu lệnh RETURN trả về giá trị cho đối tượng thực thi stored procedure.

Ví dụ:

```
create procedure sp_TestReturn
as
begin
  declare @out int
  select @out = count(*)
  from customers
  return @out
end
```

Thực thi thủ tục lưu trữ

```
declare @a int
exec  @a = sp_TestReturn
select @a
```

5.1.5 Tham số với giá trị mặc định

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

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

@tên_tham_số kiểu_dữ_liệu = giá_trị_mặc_định

Ví dụ:

```
create procedure sp_TestDefault
@customerid int = 3
as
begin
  select * from customers
  where customerid = @customerid
end
```

Thực thi thủ tục lưu trữ theo giá trị mặc định của tham số.

```
sp_TestDefault
```

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS
3	Tran Viet Cuong	1980-01-01 00:00:00.000	1	45/2B Da Tuong

Thực thi thủ tục và truyền giá trị cho tham số:

sp_TestDefault 4

CUSTOMERID	CUSTOMERNAME	BIRTHDAY	GENDER	ADDRESS
4	Nguyen Van Dai	1955-03-04 00:00:00.000	1	76 Tran Phu

5.1.6 Sửa đổi thủ tục

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

ALTER PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]

[WITH RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION]

AS

Các_câu_lệnh_của_thủ_tục

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

5.1.7 Xóa thủ tục

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

DROP PROCEDURE tên_thủ_tục

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

5.2 Hàm do người dùng định nghĩa (User Defined Function-UDF)

Hàm do người dùng định nghĩa được chia làm 3 loại: (1) scalar (hàm vô hướng), (2) inline table-valued (hàm nội tuyến, giá trị trả về dạng bảng), (3) multi-statement table-valued (hàm bao gồm nhiều câu lệnh SQL bên trong, trả về giá trị dạng bảng)

Scalar UDF: được sử dụng để trả về một duy nhất một giá trị dựa trên một các tham số truyền vào. Ví dụ: ta có thể tạo ra một UDF vô hướng nhận Customerid là tham số và trả về CustomerName.

Inline table-valued: trả về một bảng dựa trên một câu lệnh SQL duy nhất định nghĩa các dòng và các cột trả về.

Multi-statement table-value: cũng trả về kết quả là một tập hợp nhưng có thể dựa trên nhiều câu lệnh SQL.

5.2.1 Hàm vô hướng - Scalar UDF

Scarlar UDF được tạo ra bằng câu lệnh CREATE FUNCTION có cấu trúc nh ư sau;

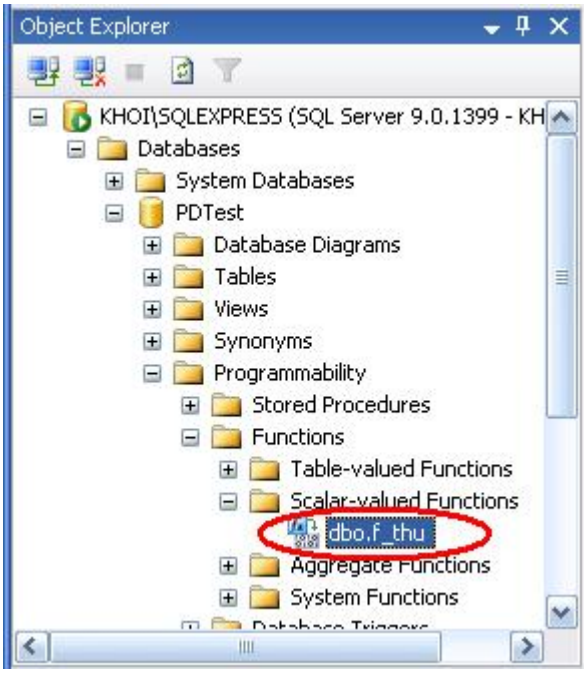
CREATE FUNCTION tên_hàm
([danh_sách_tham_số]) RETURNS (kiểu_trả_về_của_hàm)
AS BEGIN
các_câu_lệnh_của_hàm
END

Ví dụ:

Câu lệnh dưới đây định nghĩa hàm tính ngày trong tuần (thứ trong tuần) của một giá trị kiểu ngày

```
create function f_ thu(@ngay datetime)
returns nvarchar(10)
as
begin
  declare @st nvarchar(10)
  select @st=case datepart(dw,@ngay)
  when 1 then N'chủ nhật'
  when 2 then N'thứ hai'
  when 3 then N 'thứ ba'
  when 4 then N 'thứ tư'
  when 5 then N 'thứ năm'
  when 6 then N 'thứ sáu'
   else N 'thứ bảy'
end
return (@st)    /* trị trả về của hàm */
end
```

Sau khi chạy thành công, hàm trở thành một đối tượng trong CSDL và có thể được truy xuất như các hàm được xây dựng sẵn trong SQL Server 2005 Express Edition.



Ví dụ:

select CUSTOMERNAME, dbo.f_thu(BIRTHDAY)

from customers

CUSTOMERNAME	(No column name)
Cao Van Trung	thû sáu
Tran Van Phuc	thû hai
Tran Viet Cuong	thû ba
Nguyen Van Dai	thû sáu
Le Thi Hoa	thû hai
Nguyen Thanh Thai	thû sáu
Cao Van Chung	thû bảy
NGuyen Van An	thû sáu
NGuyen Van An	thû sáu

5.2.2 Hàm nội tuyến - Inline UDF

Hàm nội tuyến được định nghĩa bằng lệnh CREATE FUNCTION.

CREATE FUNCTION tên_hàm ([danh_sách_tham_s ố])

RETURNS TABLE

AS

RETURN (câu_lệnh_select)

Cú pháp của hàm nội tuyến phải tuân theo các qui tắc sau:

Kiểu trả về của hàm phải được chỉ định bởi mệnh đề RETURNS TABLE.

Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

Ví dụ: Ví dụ dưới đây lấy ra các khách hàng tùy thuộc vào giá trị mã khách hàng truyền vào cho tham số.

```
create function f_SelectCustomer
(@customerid int)
returns table
as
return (select * from customers
        where customerid > @customerid)
```

Việc gọi các hàm nội tuyến cũng tương tự như việc gọi các hàm vô hướng.

Ví dụ:

```
select tmp.CUSTOMERNAME, o.ORDERDATE
from orders o inner join dbo.f_SelectCustomer(3) as tmp
on o.customerid = tmp.customerid
```

CUSTOMERNAME	ORDERDATE
Cao Van Trung	2007-12-06 00:00:00.000

5.2.3 Hàm bao gồm nhiều câu lệnh bên trong – Multi statement UDF

Hàm này cũng được định nghĩa bằng lệnh CREATE FUNCTION

```
CREATE FUNCTION tên_hàm
([danh_sách_tham_số])
RETURNS @biến_bảng TABLE định_nghĩa_bảng
AS
BEGIN các_câu_lệnh_trong_thân_hàm
 RETURN
END
```

Lưu ý: sau từ khóa RETURNS là một biến bảng được định nghĩa. Và sau từ khóa RETURN ở cuối hàm không có tham số nào đi kèm.

Ví dụ:

```
create function f_SelectCustomer (@customerid int)
returns @myCustomers table
(
  customerid int,
```

```
  customername nvarchar(50),
  orderdate datetime
)
as
begin
  if @customerid = 0
        insert into @myCustomers
        select c.customerid, c.customername, o.orderdate
        from customers c inner join orders o on o.customerid = c.customerid
  else
        insert into @myCustomers
        select c.customerid, c.customername, o.orderdate
        from customers c inner join orders o on c.customerid = o.customerid
        where c.customerid = @customerid
  return
end
```

Việc gọi hàm multi statement UDF cũng tương tự các loại hàm khác

*select * from f_SelectCustomer(0)*

customerid	customername	orderdate
6	Cao Van Trung	2007-12-06 00:00:00.000
3	Tran Viet Cuong	2008-01-01 00:00:00.000
3	Tran Viet Cuong	2008-05-01 00:00:00.000

*select * from f_SelectCustomer(3)*

customerid	customername	orderdate
3	Tran Viet Cuong	2008-01-01 00:00:00.000
3	Tran Viet Cuong	2008-05-01 00:00:00.000

5.2.4 Thay đổi hàm

Dùng lệnh ALTER FUNCTION để thay đổi định nghĩa hàm. Cấu trúc của câu lệnh ALTER FUNCTION tương tự như CREATE FUNCTION

Ví dụ:

```
alter  function f_SelectCustomer
(@customerid int)
returns table
as
return (select * from customers
```

where customerid > @customerid)

5.2.5 Xóa hàm

Dùng lệnh DROP FUNCTION để xóa hàm. Cấu trúc lệnh DROP FUNCTION như sau

DROP FUNCTION tên_hàm

Ví dụ:

drop function f_thu

Tương tự như thủ tục lưu trữ, khi hàm bị xóa các quyền cấp cho người dùng trên hàm đó cũng bị xóa. Do đó khi định nghĩa lại hàm này, ta phải cấp lại quyền cho các người dùng.

5.3 Trigger

Trigger là một dạng đặc biệt của thủ tục lưu trữ, được thực thi một cách tự động khi có sự thay đổi dữ liệu (do tác động của câu lệnh INSERT, UPDATE, DELETE) trên một bảng nào đó.

5.3.1 Các đặc điểm của trigger

Trigger chỉ thực thi tự động thông qua các sự kiện mà không thực hiện bằng tay.

Trigger sử dụng được với khung nhìn.

Khi trigger thực thi theo các sự kiện Insert hoặc Delete thì dữ liệu khi thay đổi sẽ được chuyển sang các bảng INSERTED và DELETED, là 2 bảng tạm thời chỉ chứa trong bộ nhớ, các bảng này chỉ được sử dụng với các lệnh trong trigger. Các bảng này thường được sử dụng để khôi phục lại phần dữ liệu đã thay đổi (roll back).

Trigger chia thành 2 loại INSTEAD OF và AFTER: INSTEAD OF là loại trigger mà hoạt động của sự kiện gọi trigger sẽ bị bỏ qua và thay vào đó là các lệnh trong trigger được thực hiện. AFTER trigger là loại ngầm định, khác với loại INSTEAD OF thì loại trigger này sẽ thực hiện các lệnh bênh trong sau khi đã thực hiện xong sự kiện kích hoạt trigger.

5.3.2 Các trường hợp sử dụng trigger

Sử dụng Trigger khi các biện pháp bảo đảm toàn vẹn dữ liệu khác không bảo đảm được. Các công cụ này sẽ thực hiện kiểm tra tính toán vẹn trước khi đưa dữ liệu vào CSDL, còn Trigger thực hiện kiểm tra tính toàn vẹn khi công việc đã thực hiện

Khi CSDL chưa được chuẩn hóa (Normalization) thì có thể xảy ra dữ liệu thừa, chứa ở nhiều vị trí trong CSDL thì yêu cầu đặt ra là dữ liệu cần cập nhật thống nhất trong mọi nơi. Trong trường hợp này ta phải sử dụng Trigger.

Khi xảy ra thay đổi dây chuyền dữ liệu giữa các bảng với nhau (khi dữ liệu bảng này thay đổi thì dữ liệu trong bảng khác cũng được thay đổi theo).

5.3.3 Khả năng sau của trigger

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

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

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

5.3.4 Định nghĩa trigger

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

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

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

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

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

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

Hoạt động	Bảng INSERTED	Bảng DELETED
INSERT	dữ liệu được insert	không có dữ liệu
DELETE	không có dữ liệu	dữ liệu bị xóa
UPDATE	dữ liệu được cập nhật	dữ liệu trước khi cập nhật

Ví dụ 1: Ví dụ dưới đây minh họa việc trigger được kích hoạt khi thêm dữ liệu vào bảng CUSTOMERS

```
if exists (select name from sysobjects
where name = 't_CheckCustomerName' and type = 'TR')
drop trigger t_CheckCustomerName
go
create trigger t_CheckCustomerName
on customers
for insert
as
  declare @lengthOfName int
  select @lengthOfName = len(inserted.customername)
  from inserted
  if @lengthOfName <=1
  print N'Tên không hợp lệ'
  rollback tran
go
```

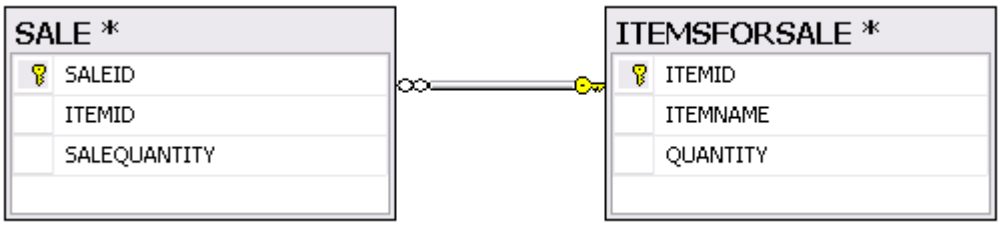
Thêm một khách hàng mới có tên là A

```
insert into customers
values('A', '5/5/1978', 'True', '35 Hung Vuong')
```

```
Tên không hợp lệ
Msg 3609, Level 16, State 1, Line 2
The transaction ended in the trigger. The batch has been aborted.
```

Ví dụ 2: Ví dụ dưới đây minh họa trigger được kích hoạt khi có sự thay đổi mang tính dây chuyền giữa các bảng.

Giả sử có CSDL như sau:



Với dữ liệu trong từng bảng là:

ITEMID	ITEMNAME	QUANTITY
1	LAPTOP	100.00
2	PPC	2000.00
3	IPOD	10.00

SALEID	ITEM...	SALEQUANTITY
1	1	10.00
2	2	10.00

Giả sử có một khách hàng mua 10 đơn vị mặt hàng LAPTOP. Khi đó số lượng LAPTOP trong bảng ITEMFORSALE sẽ giảm xuống còn 90. Trigger dưới đây sẽ thực hiện công việc đó.

```
if exists (select name from sysobjects
where name = 't_DecreaseQuantityOfItemForSale')
  drop trigger t_DecreaseQuantityOfItemForSale
go
create trigger t_DecreaseQuantityOfItemForSale
on SALE
for insert
as
  update ITEMSFORSALE
  set itemsforsale.quantity = itemsforsale.quantity  - inserted.salequantity
  from itemsforsale  inner join inserted
  on itemsforsale.itemid = inserted.itemid
go
```

Thực hiện thêm dòng vào bảng SALE

```
insert into sale
values( 1, 10)
```

ITEMID	ITEMNAME	QUANTITY
1	LAPTOP	90.00
2	PPC	100.00
3	IPOD	20.00

SALEID	ITEM...	SALEQUANTITY
1	1	10.00
2	2	10.00
9	1	10.00

Ví dụ 3: Ví dụ này minh họa cũng minh họa trigger được kích hoạt khi có sự thay đổi mang tính dây chuyền giữa các bảng nhưng trong trường hợp này dữ liệu thay đổi liên quan đến nhiều dòng.

Giả sử người quản lý muốn thay đổi số lượng bán mặt hàng LAPTOP trong bãng SALE lên thêm 5 đơn vị. Như vậy từ kết quả ví dụ 2, ta thấy cần phải giảm số lượng LAPTOP trong bảng ITEMSFORSALE xuống 10 đơn vị. Tuy nhiên, trong thực tế khi số lượng các dòng trong bảng SALE rất lớn, khi đó phải sử dụng trigger:

```
if exists (select name from sysobjects
where name = 't_DecreaseSumQuantityOfItemForSale')
  drop trigger t_DecreaseSumQuantityOfItemForSale
go
create trigger t_DecreaseSumQuantityOfItemForSale
on SALE
for update
as
  if update(salequantity)
        update ITEMSFORSALE
        set itemsforsale.quantity = itemsforsale.quantity  -
        (select sum(inserted.salequantity  -  deleted.salequantity)
        from deleted  inner join inserted
        on deleted.saleid = inserted.saleid
        where inserted.itemid = itemsforsale.itemid)
        where itemsforsale.itemid in (select inserted.itemid
                                                    from inserted)
```

Thực hiện cập nhật cho bảng SALE:

```
update sale
set salequantity = salequantity + 10
where itemid = 1
```

SALEID	ITEM...	SALEQUANTITY
1	1	20.00
2	2	10.00
9	1	20.00

ITEMID	ITEMNAME	QUANTITY
1	LAPTOP	70.00
2	PPC	100.00
3	IPOD	20.00

Ví dụ 4: Ví dụ này minh họa INSTEAD OF trigger. Trigger dưới đây sẽ không cho thực hiện thao tác xóa trên bảng CUSTOMERS.

```
create trigger t_RollbackDelete
on customers
after delete
as
```

```
rollback tran
```

5.3.5 Kích hoạt trigger dựa trên sự thay đổi dữ liệu trên cột

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

Trở lại ví dụ 3 trong phần định nghĩa trigger:

```
if exists (select name from sysobjects
where name = 't_DecreaseSumQuantityOfItemForSale')
  drop trigger t_DecreaseSumQuantityOfItemForSale
go
create trigger t_DecreaseSumQuantityOfItemForSale
on SALE
for update
as
  if update(salequantity)
        update ITEMSFORSALE
        set itemsforsale.quantity = itemsforsale.quantity -
        (select sum(inserted.salequantity - deleted.salequantity)
        from deleted inner join inserted
        on deleted.saleid = inserted.saleid
        where inserted.itemid = itemsforsale.itemid)
        where itemsforsale.itemid in (select inserted.itemi d
                                              from inserted)
```

Trong ví dụ này trigger sẽ được kích hoạt khi có sự thay đổi dữ liệu trong cột *salequantity* của bảng Sale. Nếu có sự thay đổi dữ liệu trên các cột khác thì trigger sẽ không được kích hoạt. Câu lệnh dưới đây không làm cho trigger kích hoạt.

```
update sale
set itemid = 3
where itemid = 2
```

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

5.3.6 Sử dụng trigger và Giao tác (TRANSACTION)

Khi một trigger được kích hoạt, SQL Server luôn tạo ra một giao tác theo dõi những thay đổi do câu lệnh kích hoạt trigger hoặc do bản thân trigger gây ra. Sự theo d õi này cho phép CSDL quay trở lại trạng thái trước đó.

Ví dụ: Ví dụ dưới đây xây dựng trigger không cho phép nhập v ào một bản ghi trong bảng SALE khi số lượng hàng bán lớn hơn số lượng hàng thực tế còn lại trong bảng ITEMSFORSALE

```
if exists (select name from sysobjects
where name = 't_CheckQuantity' and type = 'TR')
drop trigger t_CheckQuantity
go

create trigger t_CheckQuantity
on sale
for insert
as
  declare @insertedQuantity decimal(18,2)
  declare @currentQuantity decimal(18,2)
  declare @itemid int

  select @itemid = itemid from inserted

  select @insertedQuantity = salequantity from inserted

  select @currentQuantity = quantity
  from itemsforsale
  where itemid = @itemid

  if(@currentquantity < @insertedquantity)
        print N'số lượng nhập vào lớn hơn số lượng hiện có'
        rollback tran
```

Tiến hành thêm vào bảng SALE số liệu như sau:

```
insert into sale
values(2, 1000)
```

```
(1 row(s) affected)
số lượng nhập vào lớn hơn số lượng hiện có
Msg 3609, Level 16, State 1, Line 2
The transaction ended in the trigger. The batch has been aborted.
```

5.4 DDL TRIGGER

Được giới thiệu trong SQL Server 2005, khác với DML trigger được kích hoạt khi có sự thay đổi dữ liệu trên bảng, DDL trigger được thiết kế để đáp ứng lại các sự kiện diễn ra trên server hay trên CSDL. Một DDL trigger có thể được kích hoạt khi người dùng thực hiện các lệnh CREATE TABLE hay DROP TABLE. Ở cấp độ server, DDL trigger có thể được kích hoạt khi có một tài khoản mới được tạo ra

DDL trigger được lưu trữ trong CSDL mà DDL trigger được gắn vào. Với các Server DDL Trigger theo dõi các thay đổi ở cấp độ Server, được lưu trữ trong CSDL master.

DDL trigger được tạo ra cũng bằng câu lệnh CREATE TRIGGER với cấu trúc như sau:

CREATE TRIGGER tên_trigger

ON { ALL SERVER | DATABASE }

FOR { loại_sự_kiện } [,...n]

AS { các_câu_lệnh_SQL}

Trong đó:

ALL SERVER | DATABASE: quy định trigger sẽ kích hoạt dựa trên các sự kiện diễn ra trên Server hay các sự kiện diễn ra trên CSDL.

loại_sự_kiện: là một sự kiện đơn ở cấp độ Server hay cấp độ CSDL làm kích hoạt DDL trigger như: CREATE_TABLE, ALTER_TABLE, DROP_TABLE…

Ví dụ 1: Câu lệnh dưới đây xây dựng một trigger được kích hoạt khi xảy ra các sự kiện ở cấp độ CSDL. Trigger này sẽ ngăn chặn các lệnh DROP TABLE và ALTER TABLE.

```
create trigger t_safety
on database
for CREATE_TABLE, DROP_TABLE
as
  print N'Phải xóa trigger t_safety trước khi ALTER hay DROP bảng'
  rollback tran
```

Tiến hành xóa bảng ORDERDETAIL

```
drop table orderdetail
```

```
Phải xóa trigger t_safety trước khi ALTER hay DROP bảng
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

Ví dụ 2: Câu lệnh dưới đây xây dựng một trigger được kích hoạt khi xảy ra các sự kiện ở cấp độ Server. Trigger này sẽ ngăn chặn việc tạo ra một account login mới

```
IF EXISTS (SELECT * FROM sys.server_triggers
   WHERE name = 't_DoNotAllowCreateNewLogin')
DROP TRIGGER t_DoNotAllowCreateNewLogin
ON ALL SERVER
GO
CREATE TRIGGER t_DoNotAllowCreat eNewLogin
ON ALL SERVER
FOR CREATE_LOGIN
AS
   PRINT N'Phải DROP trigger t_DoNotAllowCreateNewLogin tr ước khi tạo account'
   rollback
GO
```

Tiến hành tạo một account login mới:

```
create login test with password = '123456'
```

```
Phải DROP trigger t_DoNotAllowCreateNewLogin trước khi tạo account
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

5.5 Enable/ Disable TRIGGER

Trigger cần bị vô hiệu hóa trong một số trường hợp:

Trigger gây ra lỗi trong quá trình xử lý CSDL

Quá trình nhập hay khôi phục những dữ liệu không thỏa trigger.

Vô hiệu hóa trigger bằng lệnh DISABLE TRIGGER có cấu trúc như sau:

```
DISABLE TRIGGER  tên_trigger
ON { tên_đối_tượng | DATABASE | SERVER }
```

Ví dụ 1: Ví dụ này sẽ vô hiệu hóa trigger *t_DoNotAllowCreateNewLogin*

```
disable trigger t_DoNotAllowCreateNewLogin
on all server
```

Tiến hành tạo một account login mới:

```
create login newLogin with password = '12345'
```

```
Command(s) completed successfully.
```

Ví dụ 2: Ví dụ này sẽ khôi phục lại trigger *t_ DoNotAllowCreateNewLogin*

enable trigger t_DoNotAllowCreateNewLogin

on all server

Tiến hành tạo một account login mới:

create login newLogin1 with password = '12345'

```
Phải DROP trigger t_DoNotAllowCreateNewLogin trước khi tạo account
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

6 Sao lưu và phục hồi dữ liệu (Backup and Restore)

Chương này sẽ giới thiệu kỹ thuật sao lưu (backup) và khôi phục (restore) dữ liệu, là kỹ thuật thường được sử dụng bảo đảm an toàn dữ liệu phòng trường hợp CSDL có sự cố.

6.1 Các lý do phải thực hiện Backup

Trong quá trình thực hiện quản trị CSDL SQL Server thì một số nguyên nhân sau đây bắt buộc bạn phải xem xét đến kỹ thuật sao lưu và khôi phục dữ liệu:

Thiết bị lưu trữ (CSDL nằm trên các thiết bị lưu trữ này) bị hư hỏng.

Người dùng vô tình xóa dữ liệu.

Các hành động vô tình hay cố ý phá hoại CSDL.

6.2 Các loại Backup

Microsoft SQL Server 2005 cung cấp hai kỹ thuật sao lưu CSDL chính: full backup và differential backup.

6.2.1 Full backup và Differential backup

Full backup: sao lưu một bản đầy đủ của CSDL trên các phương tiện lưu trữ. Quá trình full backup có thể tiến hành mà không cần offline CSDL, nhưng quá trình này lại chiếm một lượng lớn tài nguyên hệ thống và có thể ảnh hưởng nghiêm trọng tới thời gian đáp ứng các yêu cầu của hệ thống.

Differential backup: được xây dựng nhằm làm giảm thời gian cần thiết để thực hiện quá trình full backup. Differential backup chỉ sao lưu những thay đổi trên dữ liệu kể từ lần full backup gần nhất. Trong những hệ thống CSDL lớn, quá trình differential backup sẽ sử dụng tài nguyên ít hơn rất nhiều so với quá trình full backup và có thể không ảnh hưởng đến hiệu suất của hệ thống.

Quá trình differential chỉ sao lưu những sự thay đổi của dữ liệu từ lần full backup gần nhất, do đó khi có sự cố với CSDL nếu không có bản sao lưu của quá trình full backup thì bản sao lưu của quá trình differential backup sẽ trở nên vô nghĩa.

Ví dụ:

Công ty XYZ thực hiện full backup vào cuối ngày thứ 6 hàng tuần và thực hiện differential backup vào tối các ngày từ thứ 2 tới thứ 5. Nếu CSDL có sự cố vào sáng thứ 4, quản trị viên CSDL sẽ phục hồi dữ liệu bằng bản sao lưu của quá trình full backup của ngày thứ 6 tuần trước và sau đó phục hồi các thay đổi của dữ liệu bằng cách áp dụng bản sao lưu của quá trình differential backup vào ngày thứ 3.

6.2.2 Transaction log backup

Quá trình full backup và differential backup chiếm nhiều tài nguyên hệ thống và ảnh hưởng đến hiệu suất làm việc hệ thống nên thường được thực hiện vào sau giờ làm việc. Tuy nhiên điều này có thể dẫn đến các mất mát dữ liệu trong một ngày làm việc nếu CSDL có sự cố trước khi quá trình sao lưu diễn ra. Transaction log backup là một giải pháp nhằm giảm thiểu tối đa lượng dữ liệu có thể mất khi có sự cố CSDL.

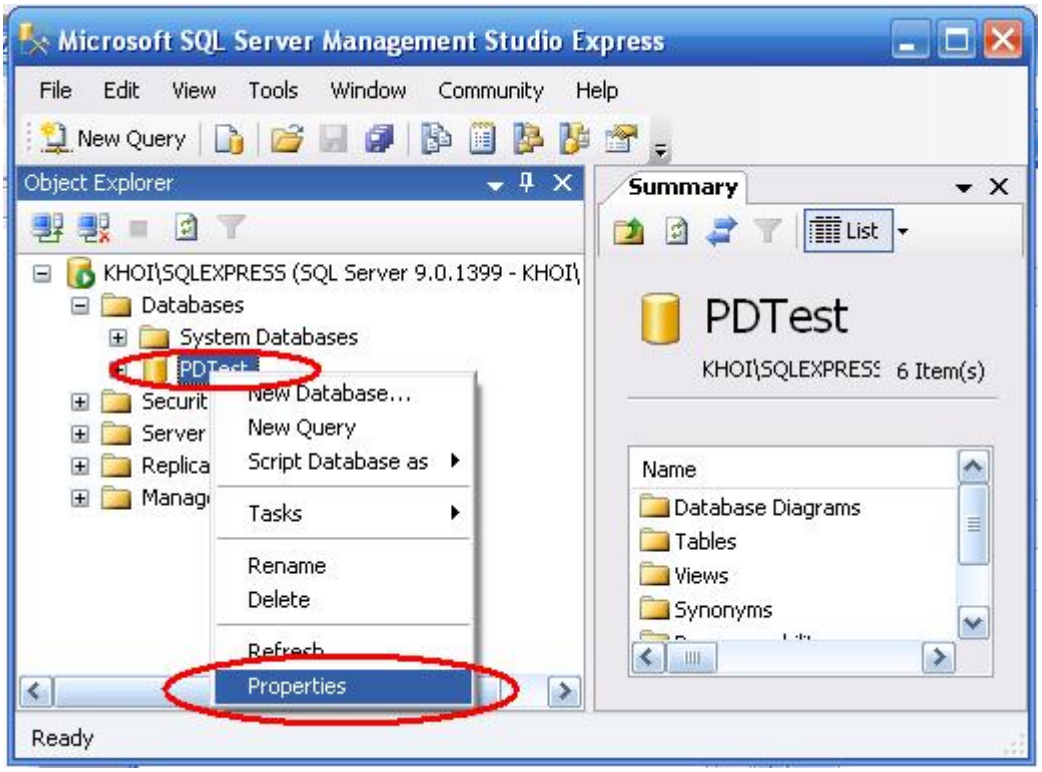
Trong quá trình hoạt động, SQL Server sử dụng transaction log để theo dõi tất cả các thay đổi trên CSDL. *Log* bảo đảm CSDL có thể phục hồi sau những sự cố đột xuất và cũng đảm bảo người dùng có thể quay ngược các kết quả trong các giao tác CSDL. Các giao tác chưa hoàn thành được lưu trong *log* trước khi được lưu vĩnh viễn trong CSDL.

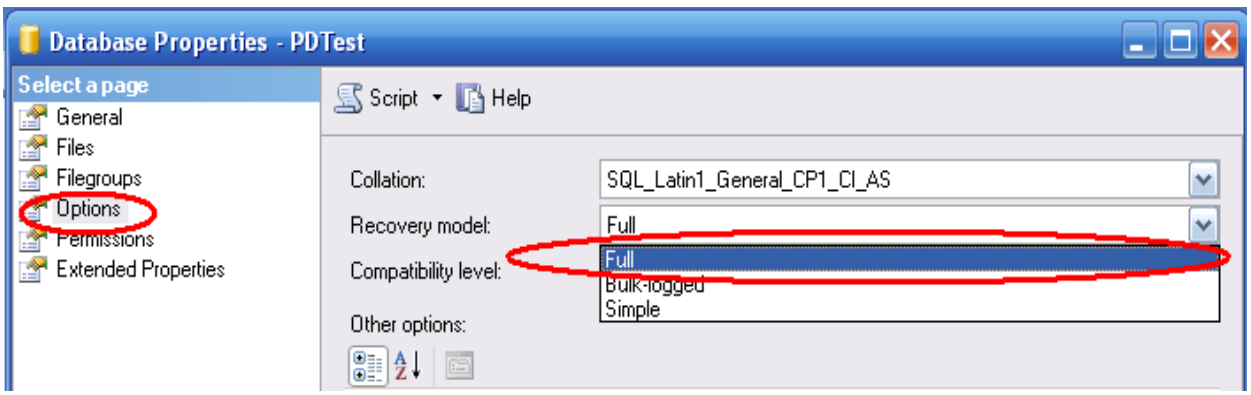
Transaction log backup sao lưu transaction log của CSDL vào thiết bị lưu trữ. Mỗi khi transaction log được sao lưu, SQL Server bỏ đi các transaction đã thực hiện thành công (committed tracsaction) và ghi các transaction vào phương tiện sao lưu. Transaction log backup sử dụng tài nguyên hệ thống ít hơn rất nhiều so với full backup và differential backup, do đó có thể sử dụng transaction log backup bất kỳ thời gian nào mà không sợ ảnh hưởng đến hiệu suất hệ thống.

Trở lại với ví dụ về công ty XYZ. Công ty này thực hiện full backup vào tối thứ 6 và differential backup vào tối từ thứ 2 tới thứ 5. Công ty thực hiện thêm quá trình transaction log backup mỗi giờ một lần. Giả sử sự cố CSDL xảy ra vào 9h:05 sáng thứ 4. Quá trình khôi phục lại CSDL nhu sau: Dùng full backup và differential backup của tối thứ 6 và tối thứ 3 để phục hồi lại trạng thái CSDL vào tối thứ 3. Tuy nhiên quá trình này vẫn còn để mất dữ liệu trong 2 giờ (7 – 9h) sáng thứ 4. Tiếp theo sử dụng 2 bản sao lưu transaction backup lúc 8h và 9h sáng để khôi phục CSDL về trạng thái lúc 9h sáng thứ 4.

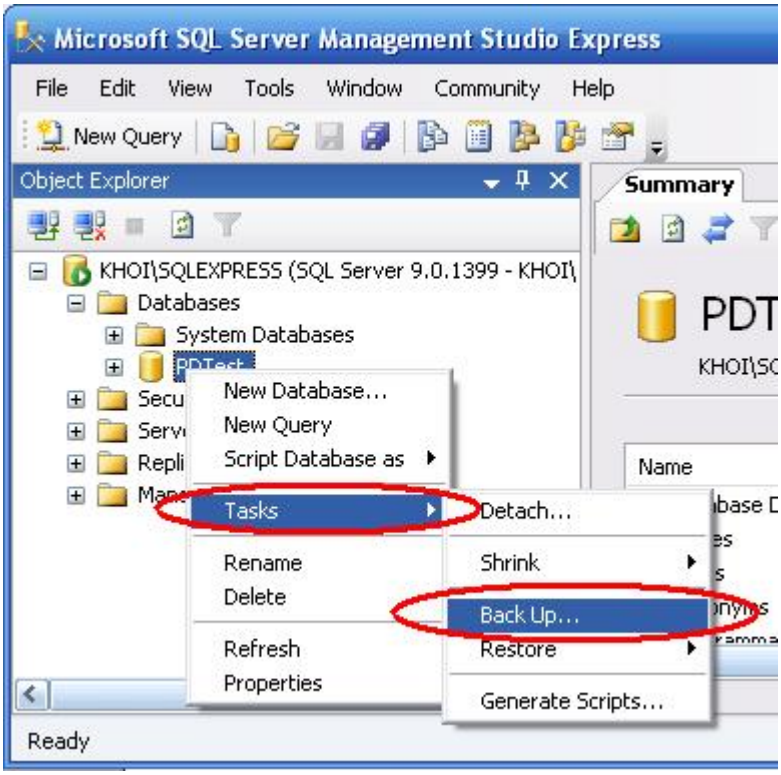
6.3 Các thao tác thực hiện quá trình Backup và Restore trong SQL Server 2005 Express Edition

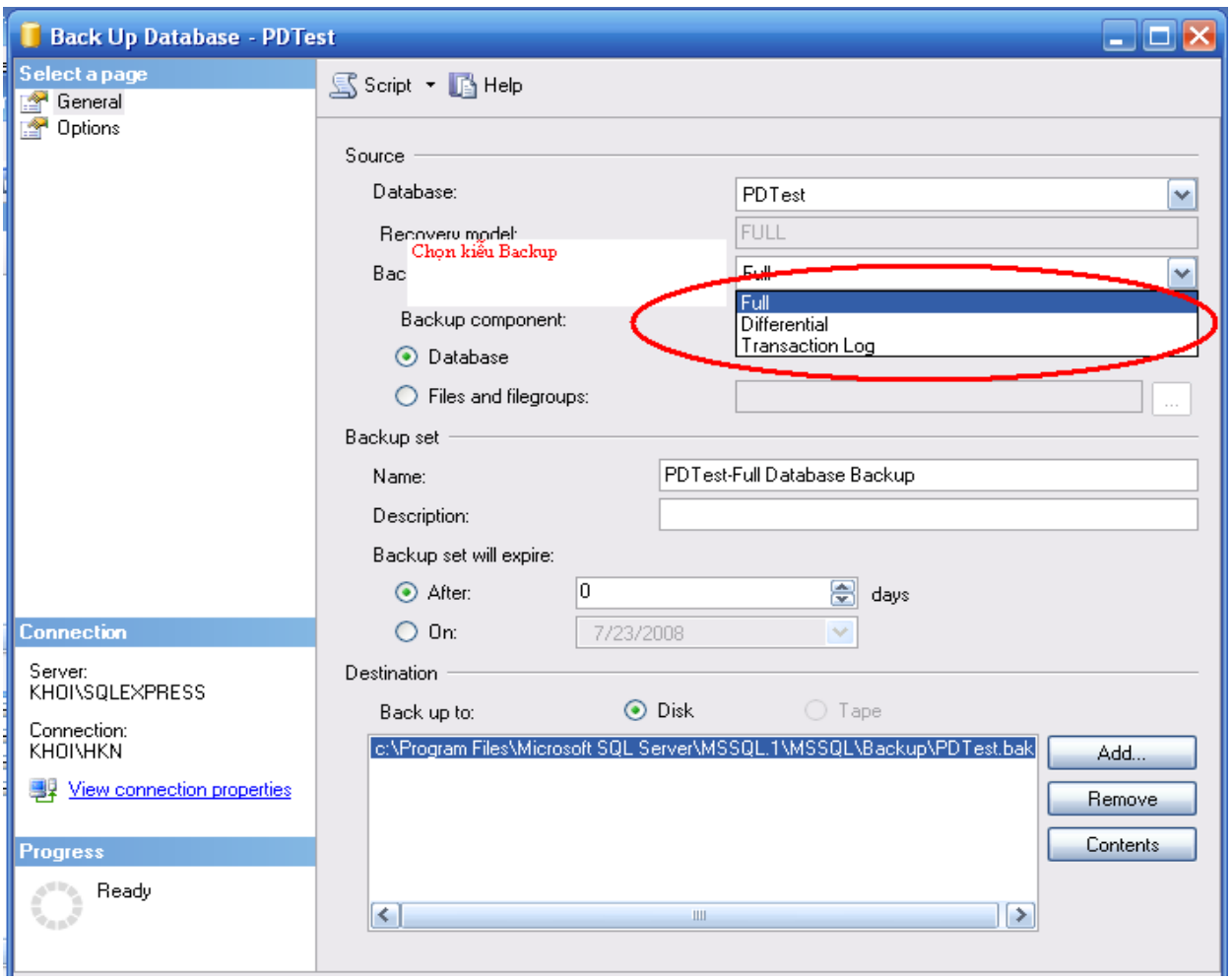
6.3.1 Sao lưu (Backup)





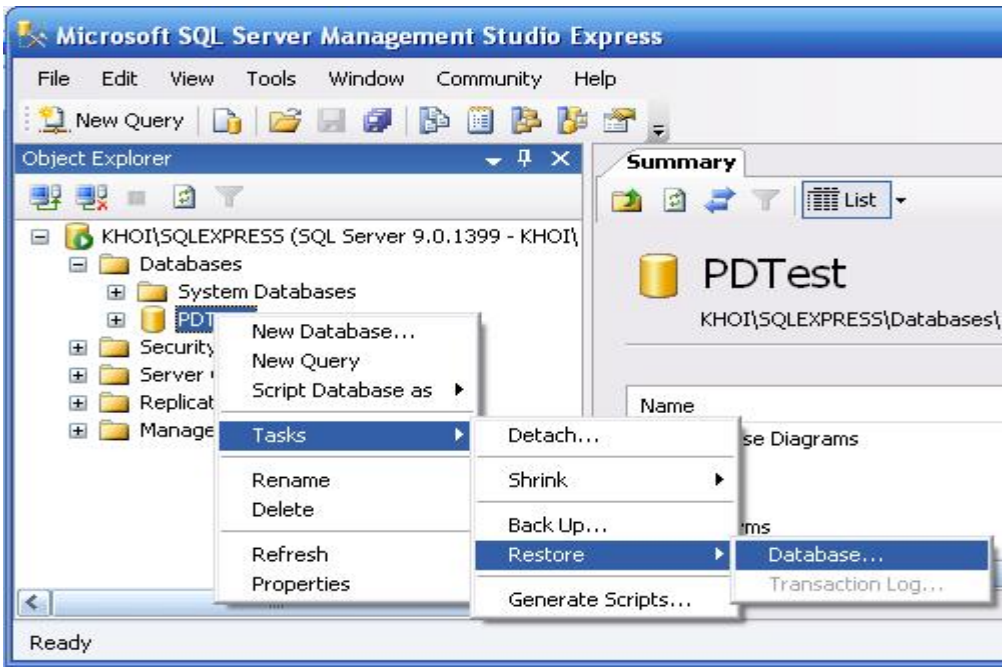
Click OK

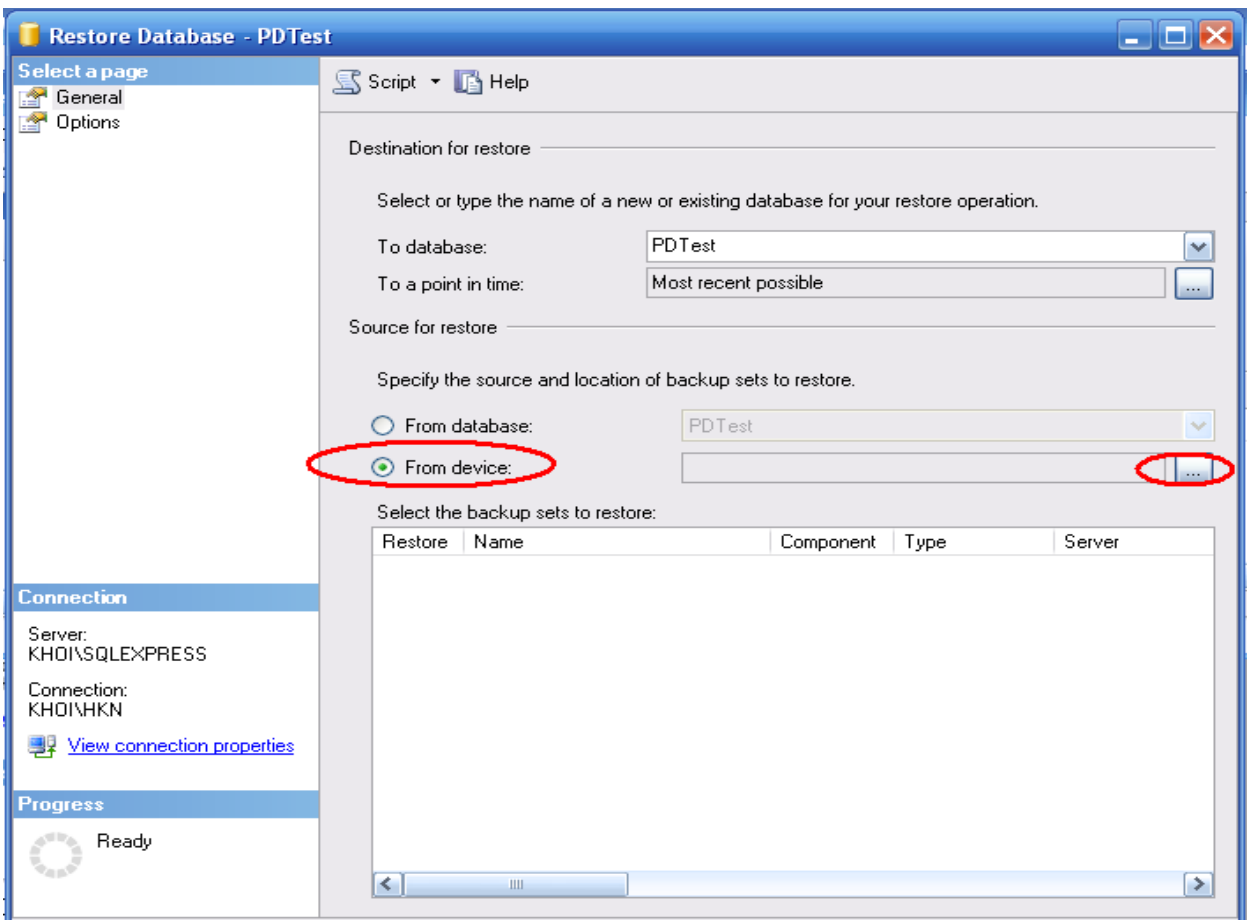

Microsoft SQL Server Management Studio Express
File Edit View Tools Window Community Help
New Query
Object Explorer
KHOI\SQLEXPRESS (SQL Server 9.0.1399 - KHOI\
Databases
System Databases
PDTest
New Database...
New Query
Script Database as
Tasks
Rename
Delete
Refresh
Properties
Detach...
Shrink
Back Up...
Restore
Generate Scripts...
Summary
Name
Ready

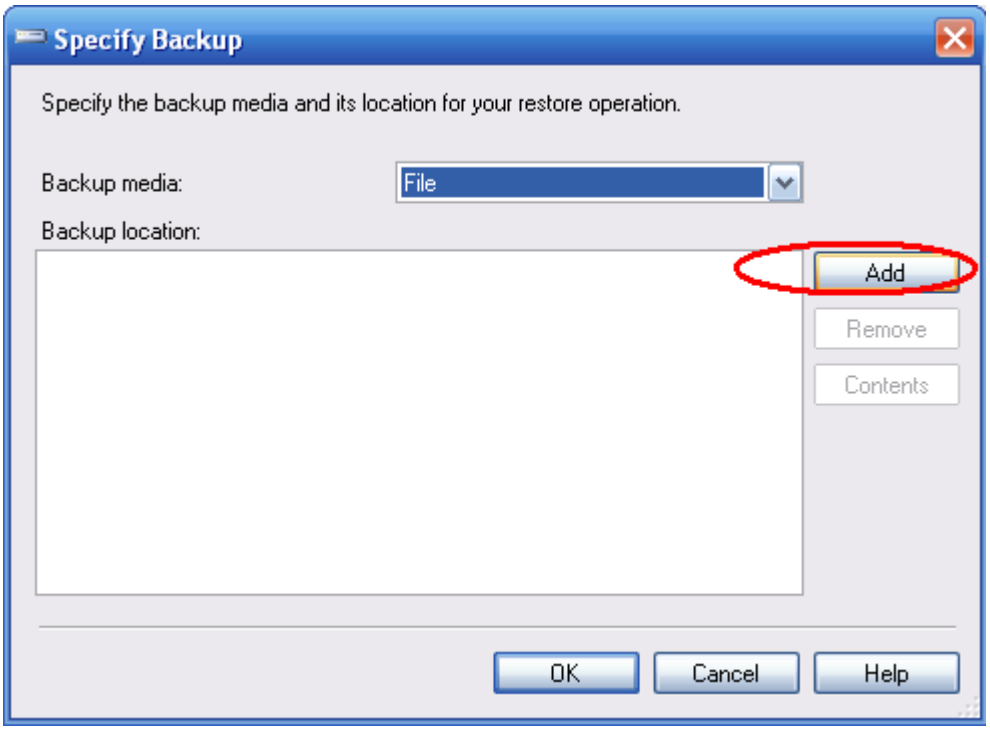

Back Up Database - PDTest
Select a page
General
Options
Script
Help
Source
Database:
PDTest
FULL
Chọn kiểu Backup
Full
Differential
Transaction Log
Backup component:
Database
Files and filegroups:
Backup set
Name:
PDTest-Full Database Backup
Description:
Backup set will expire:
After:
0
days
On:
7/23/2008
Destination
Back up to:
Disk
Tape
c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup\PDTest.bak
Add...
Remove
Contents
Connection
Server:
KHOI\SQLEXPRESS
Connection:
KHOI\HKN
View connection properties
Progress
Ready

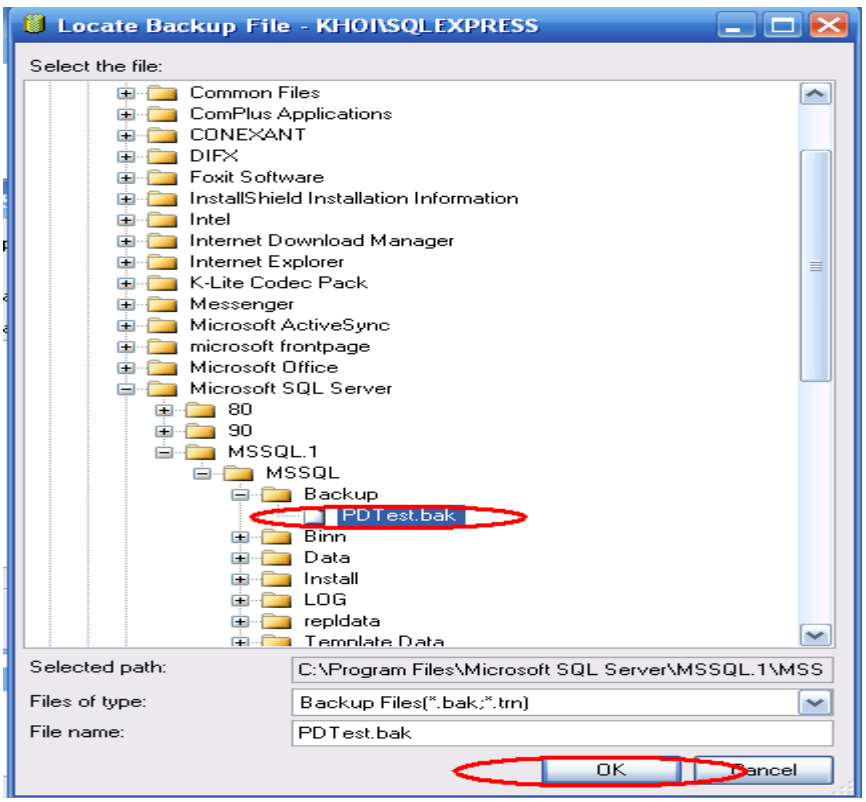
Click OK. Quá trình sao lưu hoàn tất

6.3.2 Phục hồi (Restore)

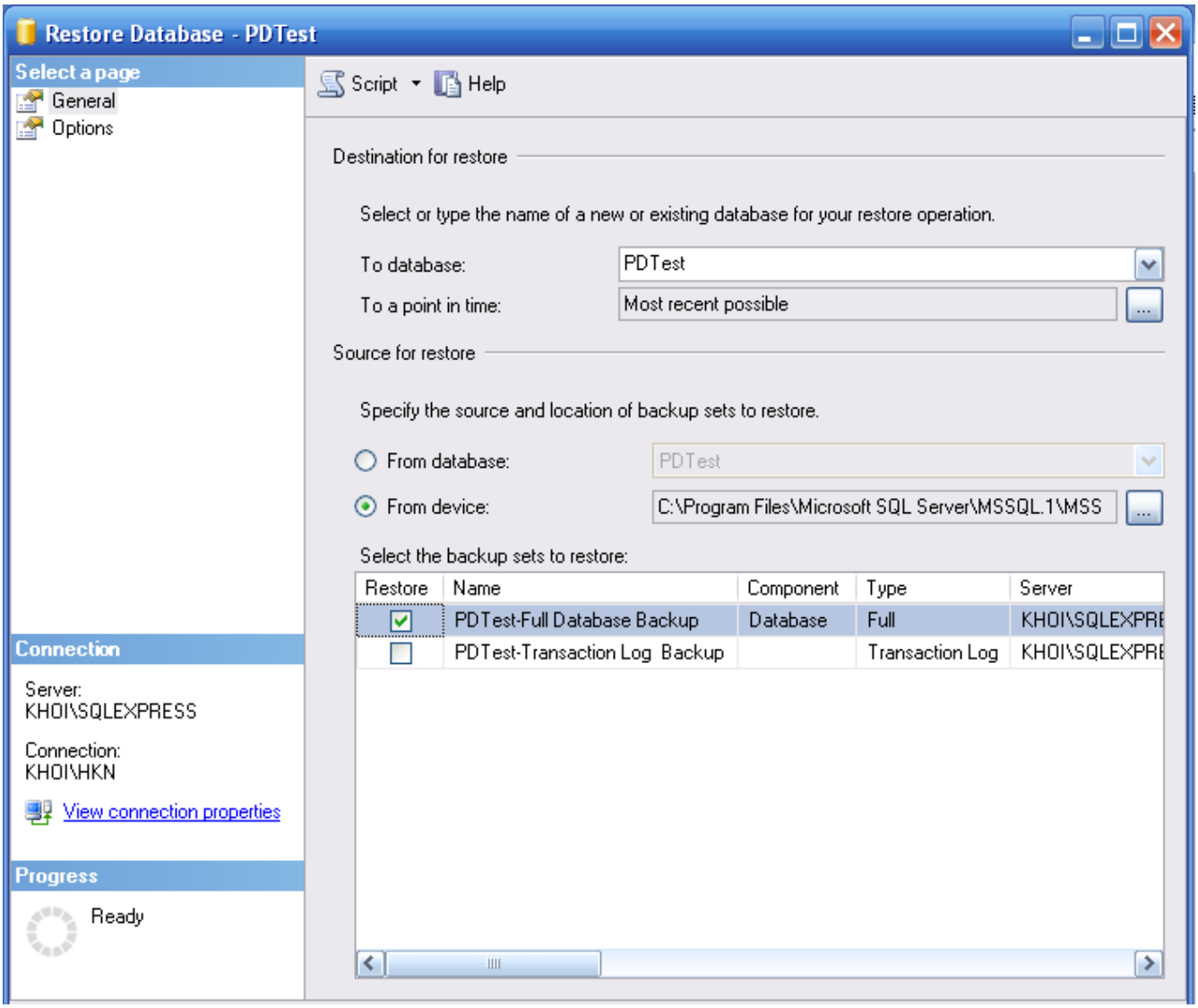








Click OK hai lần



Click OK. Quá trình phục hồi hoàn tất

7 Các hàm quan trọng trong T-SQL

Ngôn ngữ T-SQL có nhiều hàm có thể tham gia vào câu lệnh T-SQL. Những hàm này thực hiện các nhiệm vụ quan trọng khác nhau. Trong chương này sẽ trình bày một số các hàm thông dụng để làm việc với các kiểu dữ liệu số, chuỗi, ngày/thời gian và giá trị NULL trong SQL Server 2005.

7.1 Các hàm làm việc với kiểu dữ liệu số

Các hàm quan trọng làm việc với kiểu dữ liệu số là hàm ISNUMERIC và ham ROUND

7.1.1 Hàm ISNUMERIC

Hàm isNumeric kiểm tra một giá trị có phải thuộc kiểu dữ liệu số hay không.

Ví dụ: Câu lệnh dưới đây trả về tên khách hàng, và một cột có tên NUMERIC. Cột này sẽ mang giá trị 0 nếu địa chỉ khách hàng không phải là số và ngược lại

select CUSTOMERNAME, isnumeric(ADDRESS) as ISNUMERIC
from customers

CUSTOMERNAME	ADDRESS
Cao Van Trung	0
Tran Van Phuc	0
Tran Viet Cuong	0
Nguyen Van Dai	0
Le Thi Hoa	0
Nguyen Thanh Thai	0
Cao Van Chung	0
NGuyen Van An	0
NGuyen Van An	0

7.1.2 Hàm ROUND

Hàm ROUND trả về một giá trị số, đã được làm tròn theo một độ đài chỉ định

Cấu trúc hàm ROUND như sau:

ROUND (số_làm_tròn , độ_dài_làm_tròn)

Khi sử dụng hàm ROUND cần lưu ý:

số_làm_tròn phải có kiểu dữ liệu số (numeric data type) như int, float, decimal… trừ kiểu dữ liệu dạng nhị phân. Cho dù *số_làm_tròn* thuộc kiểu dữ liệu gì, kết quả hàm ROUND luôn trả về kiểu số nguyên.

Nếu *độ_dài_làm_tròn* là số âm và lớn hơn số chữ số phía trước dấu thập phân thì hàm ROUND trả về 0.

Ví dụ 1:

select ROUND(123.9994, 3), ROUND(123.99 95, 3)

(No column name)	(No column name)
123.9990	124.0000

Ví dụ 2:

select ROUND(123.4545, 2),ROUND(123.45, -2)

(No column name)	(No column name)
123.4500	100.00

Ví dụ 3:

SELECT ROUND(150.75, 0), ROUND(150.75, 0, 1)

(No column name)	(No column name)
151.00	150.00

7.2 Các hàm làm việc với kiểu dữ liệu chuỗi

Các hàm quan trọng bao gồm LEFT, RIGHT, LEN, REPLACE, STUFF, SUBSTRING, LOWER, UPPER, LTRIM, and RTRIM.

7.2.1 Hàm LEFT

Hàm LEFT trả về một chuỗi ký tự có chiều dài được chỉ định tính từ bên trái của chuỗi.

Ví dụ:

select left('Nha Trang', 5)

(No column name)
Nha T

7.2.2 Hàm RIGHT

Hàm RIGHT tương tự hàm LEFT nhưng tính từ bên phải của chuỗi

Ví dụ:

select right('Nha Trang', 5)

(No column name)
Trang

7.2.3 Hàm SUBSTRING

Hàm STRING trích xuất một chuỗi con từ một chuỗi cho trước.

Cấu trúc hàm SUBSTRING như sau:

SUBSTRING (chuỗi_ban_đầu, vị_trí_bắt_đầu, chiều_dài_chuỗi_con)

Ví dụ 1:

select substring ('Nha Trang', 2, 5)

(No column name)
ha Tr

Ví dụ 2:

Select substring(‘Nha Trang’, -2, 5)

(No column name)
Nh

7.2.4 Hàm LEN

Hàm LEN trả về chiều dài một chuỗi

Ví dụ:

Select len(‘Nha Trang’)

(No column name)
9

7.2.5 Hàm REPLACE

Hàm REPLACE thay thế một chuỗi bởi một chuỗi khác

Ví dụ 1: Câu lệnh dưới đây thay thế chữ “Nha” trong chuỗi Nha Trang bằng chữ “nha”

Select replace(‘Nha Trang’, ‘Nha ’, ‘nha)

(No column name)
nha Trang

Ví dụ 2:

select replace(ADDRESS, 'Minh', 'Ninh')

from customers

(No column name)
33 Nguyen Trung Truc
99 Nguyen Thi Ninh Khai
45/2B Da Tuong
76 Tran Phu
56 Le Hong Phong
29A Phuong Sai
12 Nguyen Thien Thuat
14 Thong Nhat
14 Thong Nhat

7.2.6 Hàm STUFF

Hàm STUFF thay thế một số lượng xác định các ký tự trong một chuỗi bằng một chuỗi khác bắt đầu từ một vị trí được chỉ định.

Ví dụ:

*select stuff('Nha Trang', 2, 3, '***')*

(No column name)
N***Trang

7.2.7 Hàm LOWER/UPPER

Hàm LOWER chuyển các ký tự hoa trong chuỗi thành các kí tự thường. Hàm UPPER chuyển các chuỗi ký tự thường trong chuỗi thành các ký tự hoa.

Ví dụ:

select lower('Nha Trang'), upper('Nha Trang')

(No column name)	(No column name)
nha trang	NHA TRANG

7.2.8 Hàm LTRIM/RTRIM

Hàm LTRIM cắt các khoảng trắng bên trái của chuỗi, hàm RTRIM cắt khoảng trắng bên phải chuỗi.

Ví dụ:

declare @llen int
declare @rlen int
declare @len int
select @llen = len(ltrim(' Nha Trang')),
@rlen = len(rtrim('Nha Trang ')),
@len = len('Nha Trang')
select @llen, @rlen, @len

(No column name)	(No column name)	(No column name)
9	9	9

7.3 Các hàm làm việc với kiểu dữ liệu Ngày tháng/ Thời gian

7.3.1 Hàm GETDATE

Hàm GETDATE trả về ngày giờ lúc thực hiện câu truy vấn.

Ví dụ:

select getdate()

7.3.2 Hàm DAY/ MONTH/ YEAR

Hàm DAY trả về ngày của một một giá trị thuộc kiểu datetime.

Hàm MONTH trả về tháng của một giá trị thuộc kiểu datetime

Hàm YEAR trả về năm của một giá trị thuộc kiểu datetime.

Ví dụ:

select day(orderdate) as DAYOFORDER,

month(orderdate) as MONTHOFORDER,

year(orderdate) as YEAROFORDER

from orders o inner join customers c on c.cus tomerid = o.customerid

where c.customerid = 3

DAYOFORDER	MONTHOFORDER	YEAROFORDER
1	1	2008
1	5	2008

7.3.3 Hàm DATEPART

Trong quá trình làm việc với các CSDL, đôi lúc ta muốn biết xem một ng ày nào đó thuộc quý mấy trong năm, hay thuộc tuần thứ mấy trong tháng . Hàm DATEPART giúp giải quyết các yêu cầu trên một cách dễ dàng.

Cấu trúc hàm DATEPART như sau:

DATEPART (yêu_cầu_trích_xuất, giá_trị_trích_xuất)

giá_trị_trích_xuất là một giá trị thuộc kiểu datetime.

yêu_cầu_trích_xuất: ngày, tháng, năm, quý,….

Khi có một yêu cầu trích xuất nào đó, chúng ta sẽ có các chữ viết tắt tương ứng với các yêu cầu đó. Bảng dưới đây mô tả các yêu chữ viết tắt và các yêu cầu trích xuất tương ứng.

Ý nghĩa	Chữ viết tắt
Năm	yy, yyyy
Quý	qq,q
Tháng	mm,m
Số ngày đã qua trong năm	dy,y
Ngày	dd,d
Tuần	wk,ww
Số ngày đã qua trong tuần	dw
Giờ	hh
Phút	mi,n
Giây	ss,s

Ví dụ:

select datepart(yyyy, orderdate)as YEAROFORDERDATE,

datepart(qq, orderdate)as QUARTEROFORDERDATE,

datepart(m, orderdate) as MONTHOFORDERDATE,
datepart(wk, orderdate) as WEEKOFORDERDATE,
datepart(d, orderdate) as DATEOFORDERDATE,
datepart(dy, Orderdate), datepart(dw, orderdate)
from orders

YEAROFORDERDATE	QUARTEROFORDERDATE	MONTHOFORDERDATE	WEEKOFORDERDATE	DATEOFORDERDATE	(No column name)	(No column name)
2007	4	12	49	6	340	5
2008	1	1	1	1	1	3
2008	2	5	18	1	122	5

7.3.4 Hàm DATENAME

Tương tự hàm DATEPART nhưng hàm DATENAME trả về một chuỗi ký tự
Ví dụ:
select datename(yyyy, orderdate)as YEAROFORDERDATE,
datename(qq, orderdate)as QUARTEROFORDER DATE,
datename(m, orderdate) as MONTHOFORDERDATE,
datename(wk, orderdate) as WEEKOFORDERDATE,
datename(d, orderdate) as DATEOFORDERDATE,
datename(dy, Orderdate), datename(dw, orderdate)
from orders

YEAROFORDERDATE	QUARTEROFORDERDATE	MONTHOFORDERDATE	WEEKOFORDERDATE	DATEOFORDERDATE	(No column name)	(No column name)
2007	4	December	49	6	340	Thursday
2008	1	January	1	1	1	Tuesday
2008	2	May	18	1	122	Thursday

7.4 Hàm CAST và CONVERTER

Chuyển đổi một giá trị thuộc kiểu dữ liệu này sang một kiểu dữ liệu khác. Hàm CAST và CONVERTER cung cấp cùng một chức năng. Một điểm thuận lợi khi dùng CONVERTER là khi chuyển đổi, hàm này cũng cho phép người dùng sẽ định dạng lại giá tri kết quả theo ý muốn.

Cấu trúc hàm CAST và CONVERTER như sau:
CAST (biểu_thức/giá_ trị AS kiểu_dữ liệu [độ_dài_kiểu_dữ_liệu])
CONVERT (kiểu_dữ liệu [độ_dài_kiểu_dữ_liệu] , biểu_thức/giá_ trị [,kiểu_định_dạng])

Năm 2 chữ số	Năm 4 chữ số	Output
	0 hoặc 100	mon dd yyyy hh:mi AM (PM)
1	101	mm/dd/yy
2	102	yy.mm.dd

3	103	dd/mm/yy
4	104	dd.mm.yy
5	105	dd-mm-yy
6	106	dd mon yy
7	107	Mon dd, yy
8	108	hh:mm:ss
	9 hoặc 109	mon dd yyyy hh:mi:ss:mmmAM (PM)
10	110	mm-dd-yy
11	111	yy/mm/dd
12	112	yymmdd
	13 hoặc 113	dd mon yyyy hh:mm:ss:mmm(24h)
14	114	hh:mi:ss:mmm(24h)

Ví1 dụ:

select CUSTOMERNAME,

convert (varchar, BIRTHDAY, 103) as BIRTHDAY, ADDRESS

from Customers

where Customername = 'Le Thi Hoa'

and year(getdate()) - year(BIRTHDAY) > 20

Hàm CONVERT và hàm CAST có thể sử dụng kết hợp với nhau để cho kết qua như mong muốn.

Ví dụ:

select c.CUSTOMERID, c.CUSTOMERNAME,

*convert(varchar(20),cast(SUM(i.UNITPRICE*od.QUANTITY) as money),1) as SUMTOTAL*

from customers c inner join orders o on o.customerid = c.customerid

inner join orderdetail od on o.orderid = od.orderid

inner join items i on i.itemid = od.itemid

group by c.customerid, c.customername

Tài liệu tham khảo

1. Giáo trình hệ quản trị cơ sở dữ liệu SQL Server, Khoa CNTT, Đại học Huế.
2. SQL Server 2005, T-SQL Recipes: Problem, Solution, Approach – Appress Publisher.
3. Sams Teach yourself Microsoft SQL Server 2005 Express in 24 hours.