

M c l c

M c l c	1
1 Gi i thi u v SQL Server 2005	5
1.1 Cài t SQL Server 2005 Express Edition	5
1.1.1 Các yêu c u cho h th ng 32bit	5
1.1.2 Các b c cài t SQL Server 2005 Express Edition	7
1.2 M t s thao tác c b n trên SQL Server 2005 Express Edition.	16
1.2.1 T o m t CSDL m i	16
1.2.2 T o b ng m i	17
1.2.3 Xóa b ng, xóa CSDL	19
1.2.4 M m t query editor vi t câu l nh SQL	19
2 Structured Query Language (SQL)	20
2.1 SQL là ngôn ng c a c s d li u quan h	20
2.2 Vai trò c a SQL	20
2.3 Gi i thi u s l c v Transact SQL (T-SQL)	21
2.3.1 Ngôn ng nh ngh a d li u (Data Definition Language – DDL)	22
2.3.2 Ngôn ng i u khi n d li u (Data control language – DCL)	22
2.3.3 Ngôn ng thao tác d li u (Data manipulation language – DML).....	23
2.3.4 Cú pháp c a T-SQL	24
2.3.5 Các ki u d li u	25
2.3.6 Bi n (Variables)	26
2.3.7 Hàm (Function)	27
2.3.8 Các toán t (Operators)	27
2.3.9 Các thành ph n i u khi n (Control of flow)	28
2.3.10 Chú thích (Comment)	28
2.3.11 Giá tr NULL	28
3 Ngôn ng thao tác d li u – DML	29
3.1 Câu l nh SELECT	29
3.1.1 Danh sách ch n trong câu l nh SELECT	30
3.1.2 M nh FROM	34
3.1.3 M nh WHERE - i u ki n truy v n d li u	34
3.1.4 Phép h p (UNION)	38
3.1.5 Phép n i	41
3.1.6 Các lo i phép n i	43

3.1.7	Phép nối theo chuẩn SQL-92.....	45
3.1.8	Mệnh GROUP BY.....	47
3.1.9	Truy vấn con (Subquery).....	50
3.2	Thêm, cập nhật và xóa dữ liệu.....	51
3.2.1	Thêm dữ liệu.....	52
3.2.2	Cập nhật dữ liệu.....	53
3.2.3	Xóa dữ liệu.....	54
4	Ngôn ngữ định nghĩa dữ liệu – DDL.....	56
4.1	Tạo bảng.....	56
4.2	Các loại ràng buộc.....	58
4.2.1	Ràng buộc CHECK.....	58
4.2.2	Ràng buộc PRIMARY KEY.....	59
4.2.3	Ràng buộc FOREIGN KEY.....	60
4.3	Sửa định nghĩa bảng.....	61
4.4	Xóa bảng.....	63
4.5	Khung nhìn - VIEW.....	63
4.6	Thêm, cập nhật, xóa dữ liệu trong VIEW.....	65
4.7	Thay đổi định nghĩa khung nhìn.....	65
4.8	Xóa khung nhìn.....	66
5	Thủ tục lưu trữ, hàm và trigger.....	67
5.1	Thủ tục lưu trữ (Stored procedure).....	67
5.1.1	Tạo thủ tục lưu trữ.....	68
5.1.2	Lưu trữ thủ tục.....	69
5.1.3	Bị khóa trong thủ tục lưu trữ.....	69
5.1.4	Giá trị trả về trong thủ tục lưu trữ.....	70
5.1.5	Tham số và giá trị mặc định.....	71
5.1.6	Sửa thủ tục.....	72
5.1.7	Xóa thủ tục.....	72
5.2	Hàm do người dùng định nghĩa (User Defined Function-UDF).....	72
5.2.1	Hàm vô hướng - Scalar UDF.....	73
5.2.2	Hàm nội tuyến - Inline UDF.....	74
5.2.3	Hàm bao gồm nhiều câu lệnh bên trong – Multi statement UDF.....	75
5.2.4	Thay đổi hàm.....	76
5.2.5	Xóa hàm.....	77
5.3	Trigger.....	77

5.3.1	Các c i m c a trigger	77
5.3.2	Các tr ã ng h p s d ã ng trigger.....	77
5.3.3	Kh ã n ã ng sau c a trigger	78
5.3.4	nh ã ng a trigger.....	78
5.3.5	Kích ho t trigger d ã trẽn s thay ã d ã li u trẽn c t.....	82
5.3.6	S d ã ng trigger và Giao tác (TRANSACTION)	83
5.4	DDL TRIGGER	84
5.5	Enable/ Disable TRIGGER.....	85
6	Sao l u và ph c h i d ã li u (Backup and Restore)	87
6.1	Các lý do ph i th c hi ã n Backup	87
6.2	Các lo i Backup	87
6.2.1	Full backup và Differential backup	87
6.2.2	Transaction log backup.....	88
6.3	Các thao tác th c hi ã n quá trình Backup và Restore trong SQL Server 2005 Express Edition.....	89
6.3.1	Sao l u (Backup)	89
6.3.2	Ph c h i (Restore)	91
7	Các hàm quan tr ã ng trong T-SQL	94
7.1	Các hàm làm vi c v i ki u d ã li u s	94
7.1.1	Hàm ISNUMERIC.....	94
7.1.2	Hàm ROUND	94
7.2	Các hàm làm vi c v i ki u d ã li u chu i.....	95
7.2.1	Hàm LEFT	95
7.2.2	Hàm RIGHT	95
7.2.3	Hàm SUBSTRING	95
7.2.4	Hàm LEN.....	96
7.2.5	Hàm REPLACE.....	96
7.2.6	Hàm STUFF.....	96
7.2.7	Hàm LOWER/UPPER.....	97
7.2.8	Hàm LTRIM/RTRIM	97
7.3	Các hàm làm vi c v i ki u d ã li u Ngày tháng/ Th i gian	97
7.3.1	Hàm GETDATE	97
7.3.2	Hàm DAY/ MONTH/ YEAR	97
7.3.3	Hàm DATEPART.....	98
7.3.4	Hàm DATENAME.....	99

7.4	Hàm CAST và CONVERTER.....	99
8	Kết nối vào SQL Server 2005 từ các ngôn ngữ lập trình xây dựng các ứng dụng liên quan đến CSDL	101
8.1	Cấu hình Microsoft SQL Server 2005	101
8.1.1	Cho phép tiếp nhận các kết nối từ xa trên thiết bị của SQL Server	102
8.1.2	Kích hoạt dịch vụ SQL Server Browser	102
8.1.3	Tắt các cổng trên Windows Firewall	103
8.2	Kết nối vào SQL Server trong các ngôn ngữ lập trình.....	104
8.2.1	C# và VB.NET.....	104
8.2.2	VB 6.....	106
	Tài liệu tham khảo	108

1 Gi i thi u v SQL Server 2005

SQL Server 2005 là m t h th ng qu n lý c s d li u (Relational Database Management System (RDBMS)) s d ng Transact-SQL trao i d li u gi a Client computer và SQL Server computer. M t RDBMS bao g m databases, database engine và các ng d ng dùng qu n lý d li u và các b ph n khác nhau trong RDBMS.

SQL Server 2005 c t i u có th ch y trên môi tr ng c s d li u r t l n (Very Large Database Environment) lên n Tera-Byte và có th ph c v cùng lúc cho hàng ngàn user. SQL Server 2005 có th k t h p " n ý" v i các server khác nh Microsoft Internet Information Server (IIS), E-Commerce Server, Proxy Server....

Các phiên b n c a SQL Server 2005:

Enterprise: H tr không gi i h n s l ng CPU và kích th c Database. H tr không gi i h n RAM (nh ng tùy thu c vào kích th c RAM t i a mà H H h tr) và các h th ng 64bit.

Standard: T ng t nh b n Enterprise nh ng ch h tr 4 CPU. Ngoài ra phiên b n này c ng không c trang b m t s tính n ng cao c p khác.

Workgroup: T ng t b n Standard nh ng ch h tr 2 CPU và t i a 3GB RAM

Express: B n mi n phí, h tr t i a 1CPU, 1GB RAM và kích th c Database gi i h n trong 4GB.

Chi ti t có th tham kh o t i a ch :

<http://www.microsoft.com/sql/prodinfo/features/compare-features.msp>

1.1 Cài t SQL Server 2005 Express Edition

1.1.1 Các yêu c u cho h th ng 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	<ul style="list-style-type: none">Windows XP with Service Pack 2 hoặc cao hơnMicrosoft Windows 2000 Professional SP4

Express Edition System Requirements		
32-bit		
	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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.msp>

Download và cài đặt Microsoft .NET Framework 2.0: 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.

Gồm 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 t SQL Server 2005 Express Edition: Microsoft SQL Server 2005 Express Edition là phiên b n mi n phí, d s d ng và “nh ” c a Microsoft SQL Server 2005. Microsoft SQL Server 2005 Express Edition c tích h p trong Visual Studio 2005 t o ra s d dàng trong vi c phát tri n các ng d ng h ng CSDL. SQL Server 2005 Express Edition c t do s d ng trong các ng d ng th ng m i và d dàng c p nh t lên các phiên b n cao h n khi c n thi t.

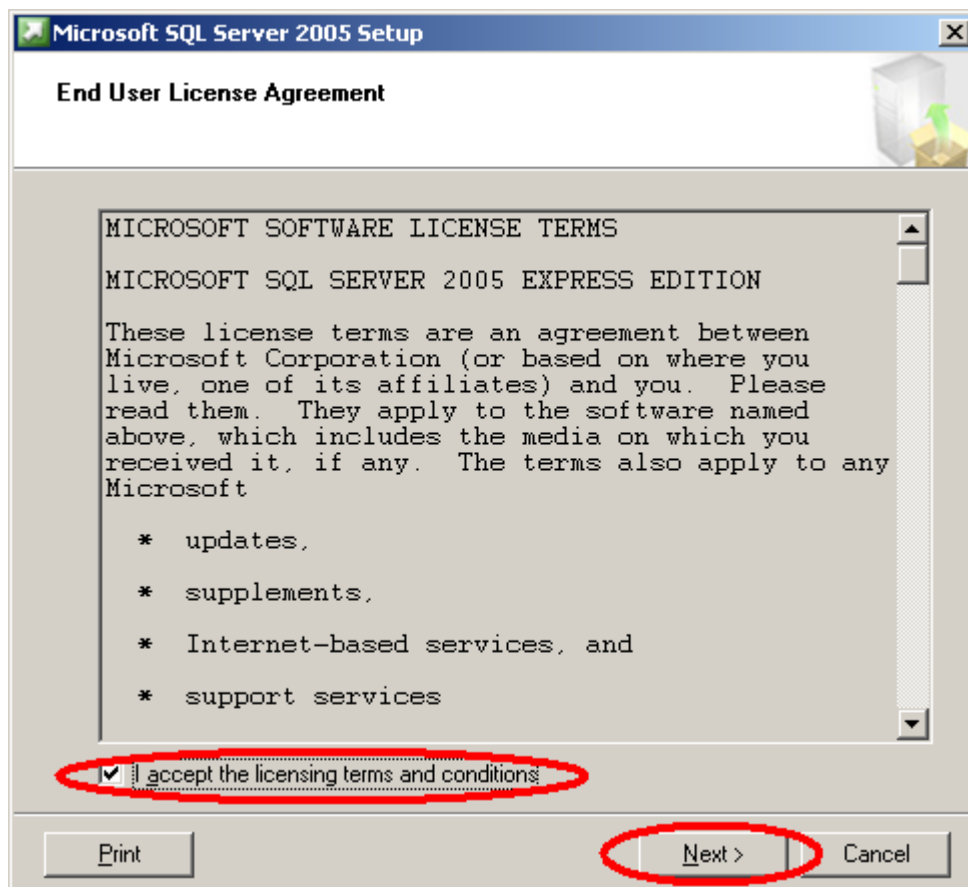
Cài t SQL Server Management Studio Express: SQL Server Management Studio Express cung c p giao di n ng i dùng d dàng t ng tác v i các thành ph n c a Microsoft SQL Server 2005 Express Edition. Tr c khi cài t SQL Server Management Studio Express, MSXML 6.0 ph i c cài t

Download t i a ch :

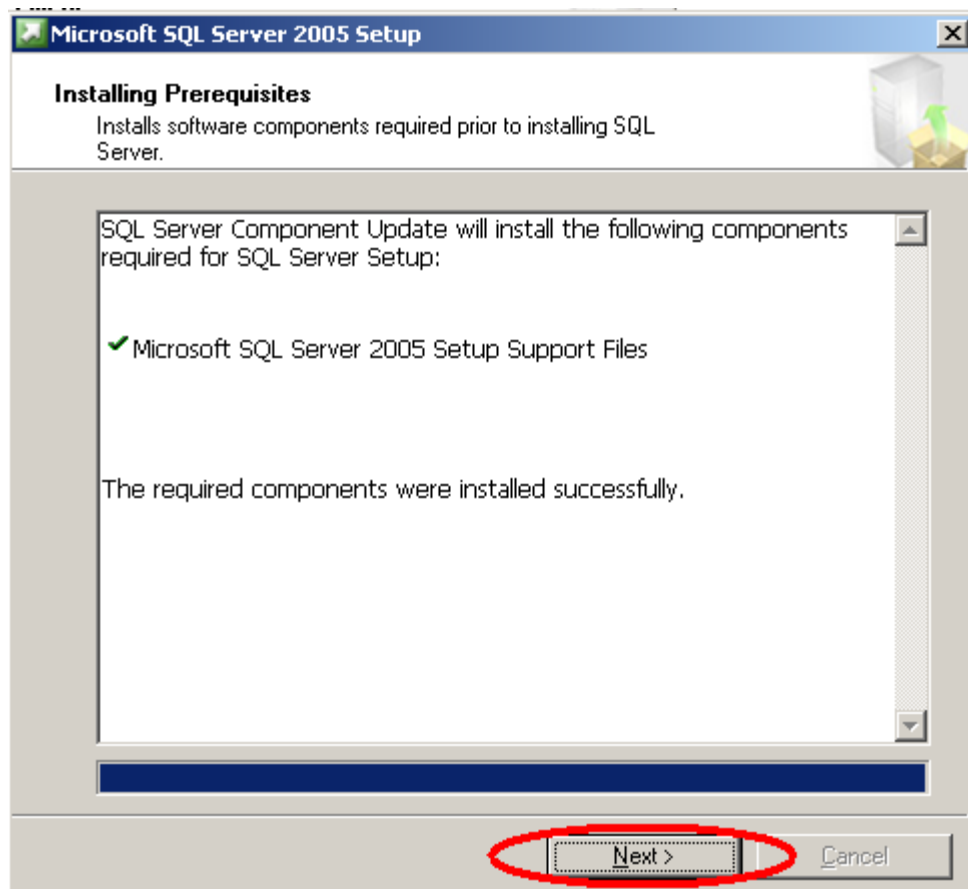
<http://www.microsoft.com/express/sql/download/default.aspx>

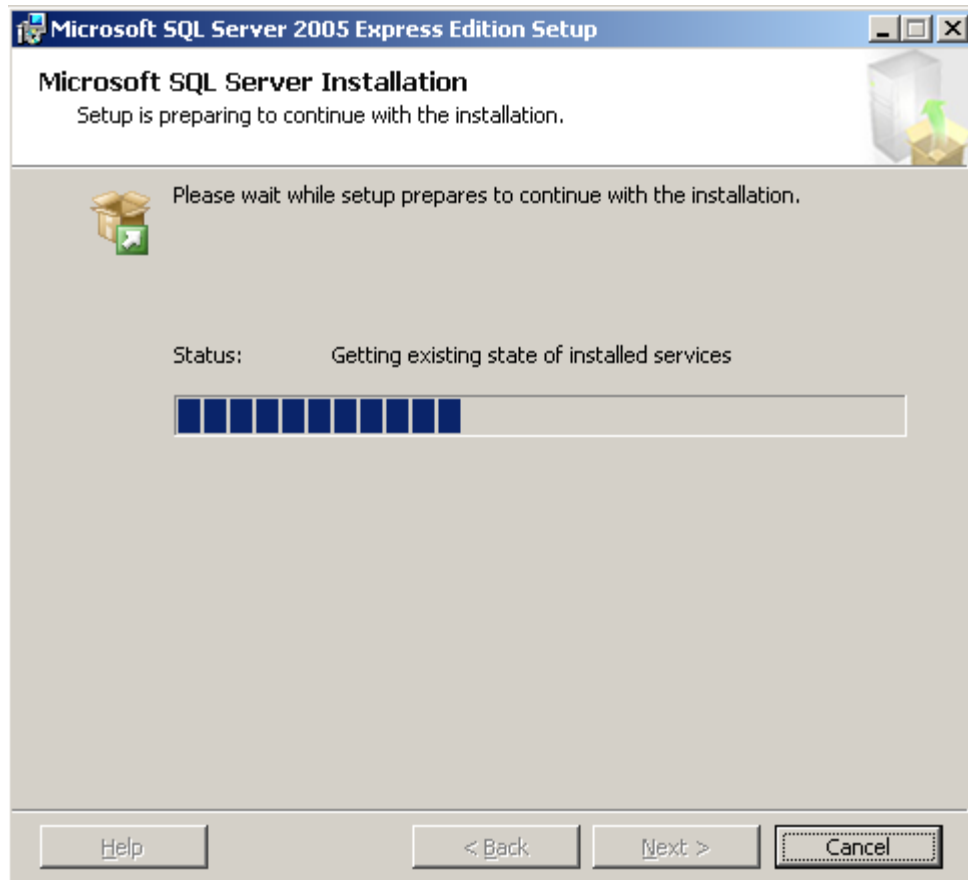
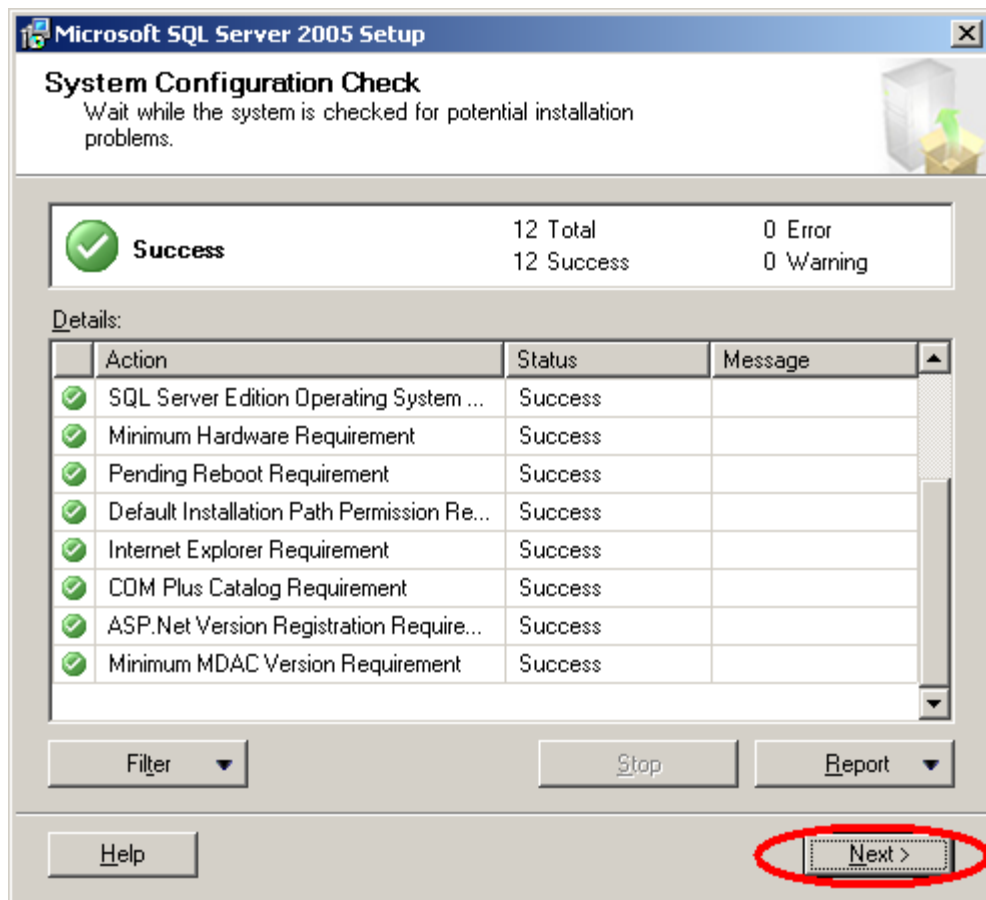
1.1.2 Các b c cài t SQL Server 2005 Express Edition

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



Click Next:





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:

Company:

☐ Hide advanced configuration options

Microsoft SQL Server 2005 Express Edition Setup

Feature Selection

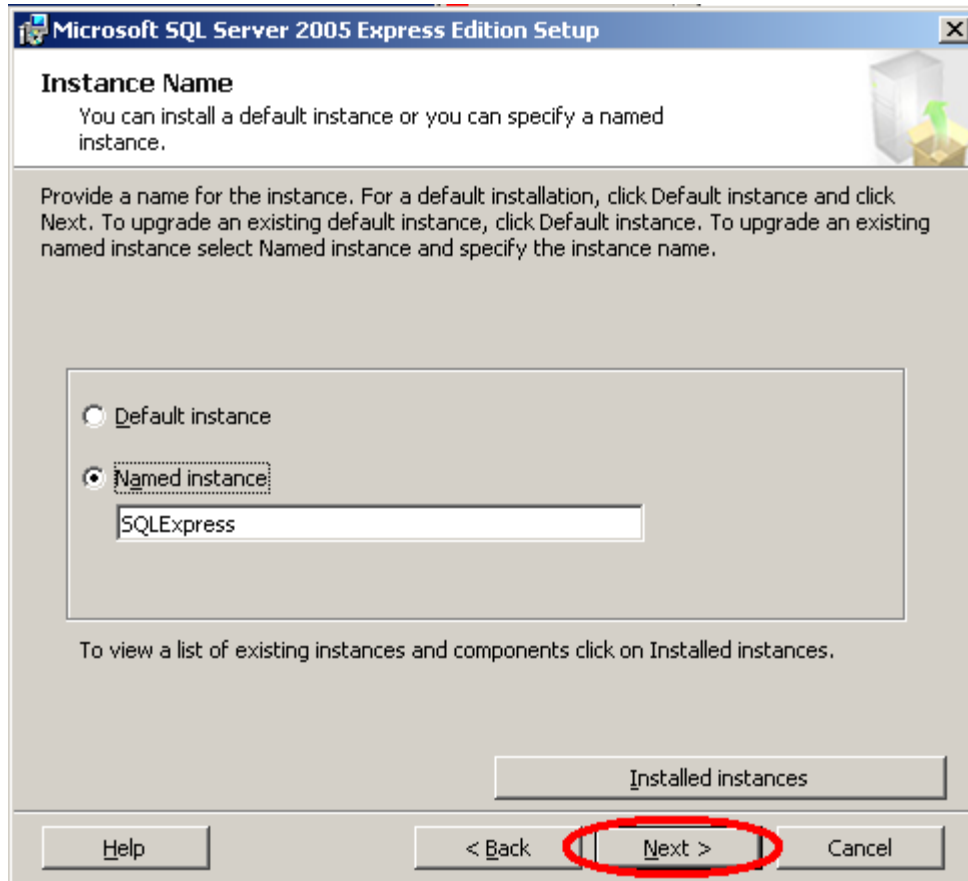
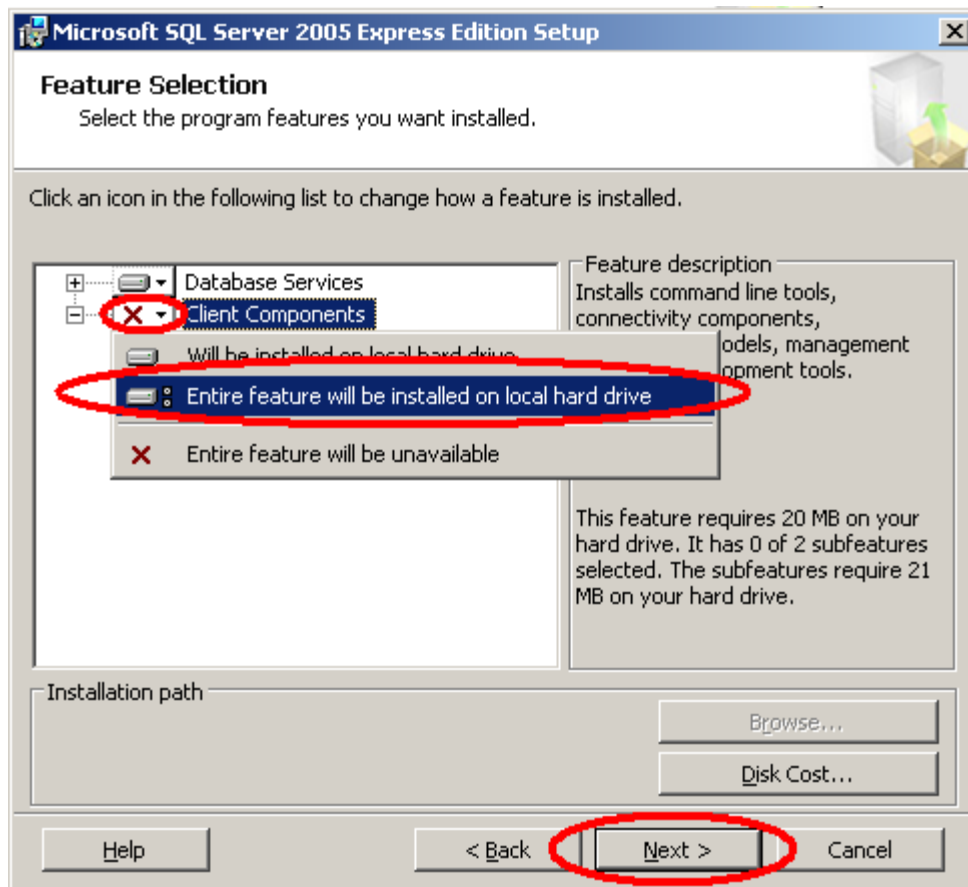
Select the program features you want installed.

Click an icon in the following list to change how a feature is installed.

Feature description
Database Services Installs the SQL Server Database Engine for managing relational data and replication.
Will be installed on local hard drive
Entire feature will be installed on local hard drive
Entire feature will be unavailable

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\



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 login password below:

Enter password:

Confirm password:

Help < Back **Next >** Cancel

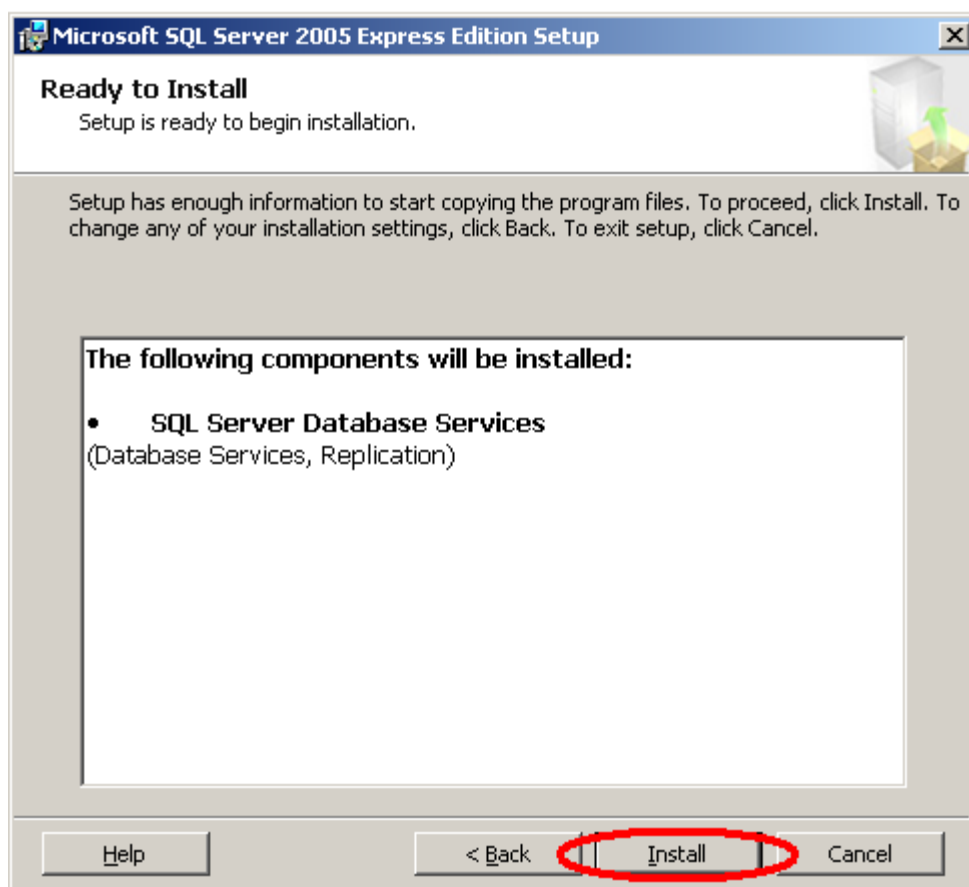
L u ý: SQL Server 2005 có hai ki u authentication (ki m tra ng i dùng).

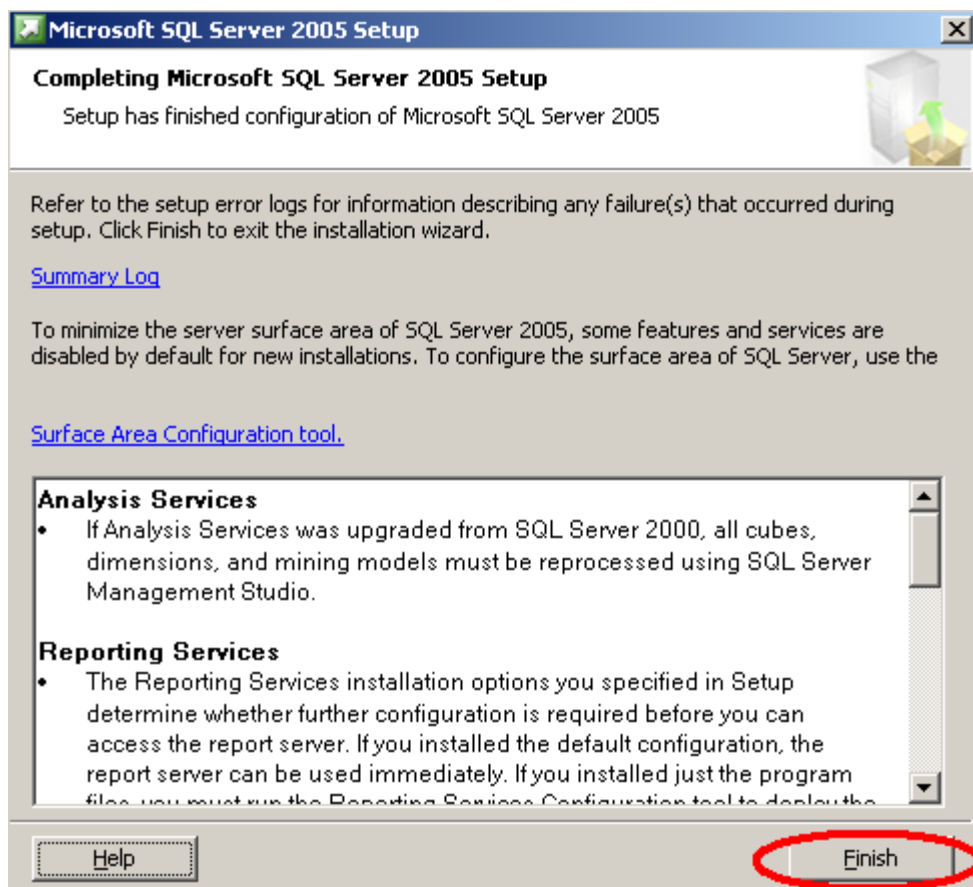
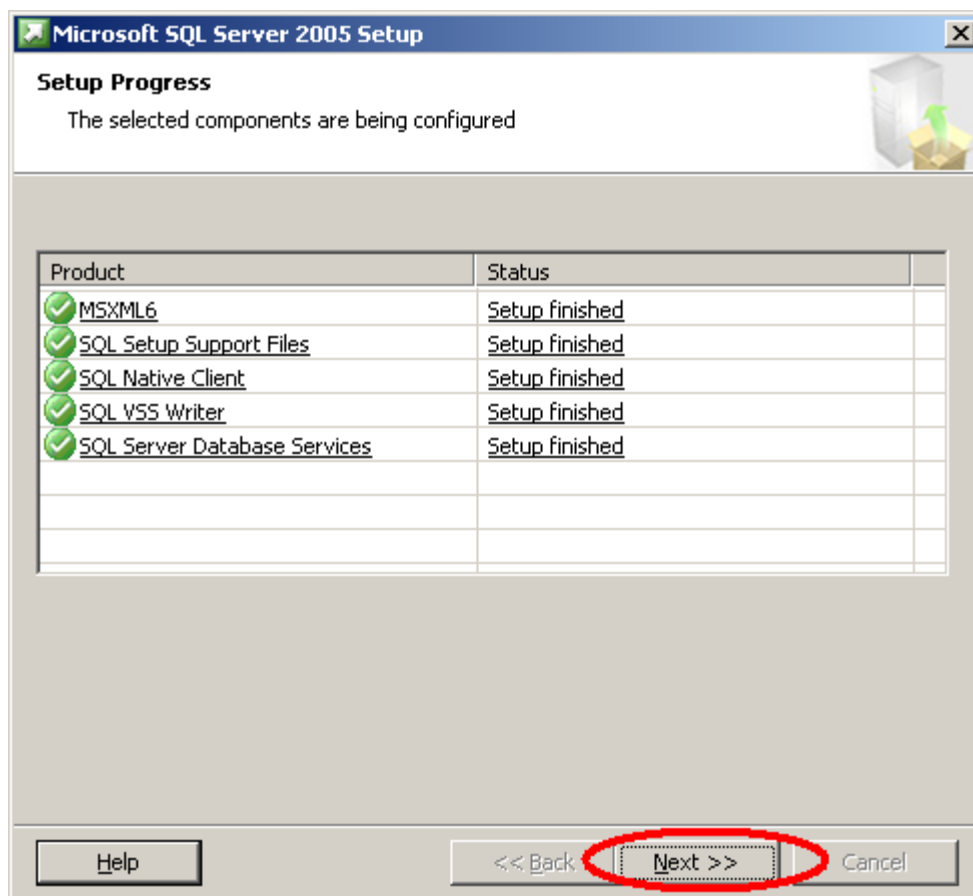
Windows authentication mode: Vì c ki m tra ng i dùng c a SQL Server 2005 s ph thu c vào vì c ki m tra ng i dùng c a Windows. Khi ng i dùng có quy n ng nh p vào Windows, ng i dùng ó s có quy n ng nh p vào SQL Server. Ki u ki m tra ng i dùng này th ng c s d ng khi ng d ng khai thác d li u và SQL Server c cài trên cùng m t máy tính.

SQL Server authentication mode: Vì c ki m tra ng i dùng c a SQL Server 2005 s không ph thu c vào vì c ki m tra ng i dùng c a Windows. Khi ng i dùng có quy n ng nh p vào Windows, ng i dùng ó ch a ch c s có quy n ng nh p vào SQL Server. ng nh p vào SQL Server, ng i dùng này ph i có m t b username và password do SQL Server qu n lý. Ki u ki m tra ng i dùng này th ng c s d ng khi ng d ng khai thác d li u và SQL Server không c cài trên cùng m t máy tính.

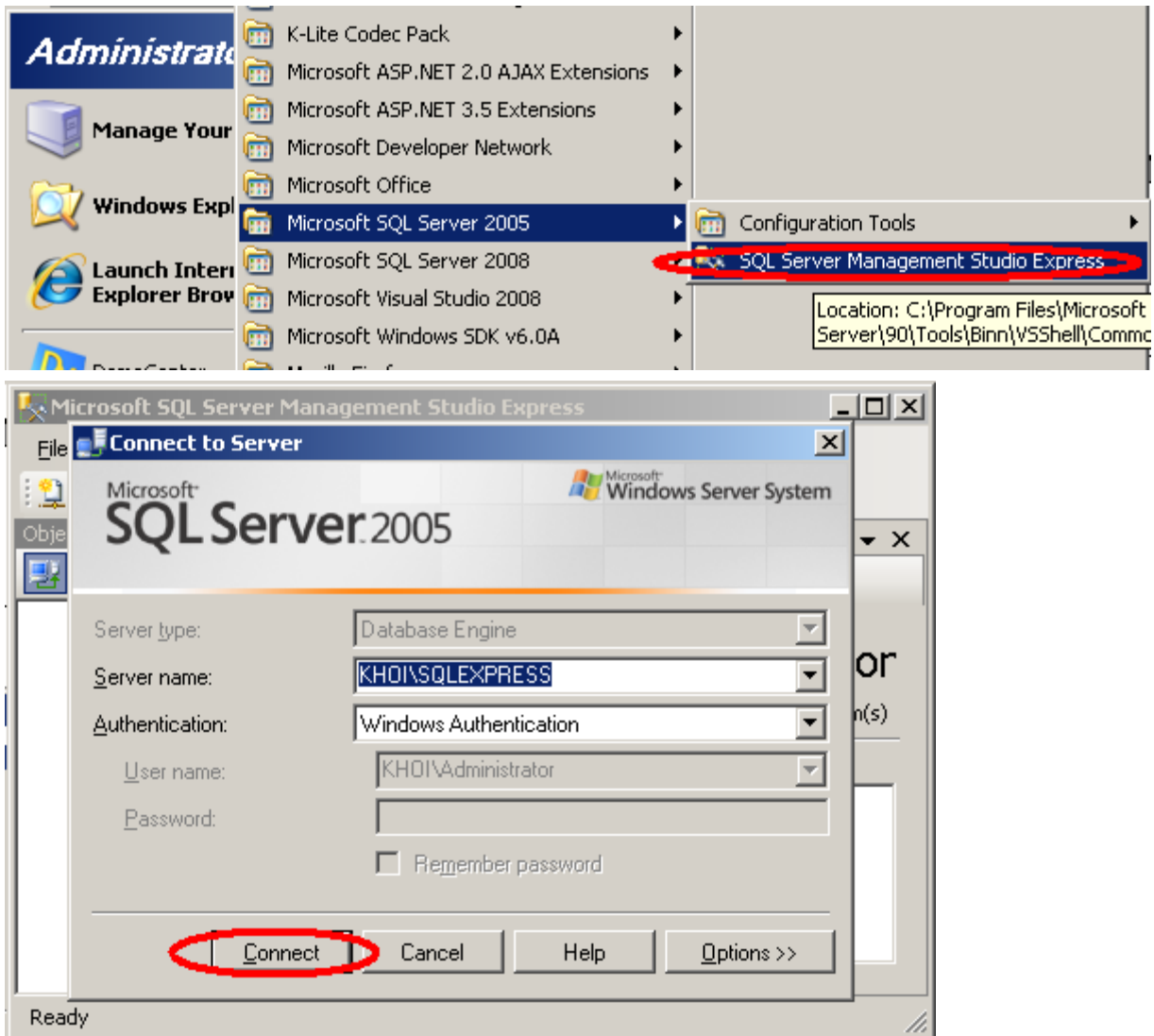
Khi ch n Mixed mode, SQL Server có th dùng b t k ki u ki m tra ng i dùng nào khi c n thi t. ây là m t thi t l p th c s r t h u ích khi xây d ng các ng d ng CSDL. Ngoài ra, ta c ng ph i ánh password vào hai ô bên d i có th ng nh p vào SQL Server khi ta xây d ng m t ng d ng truy xu t vào CSDL máy này khi ta ang máy khác.

Click Next ba l n:





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



Khi nhập có thể chọn Windows Authentication hoặc SQL Server Authentication. Nếu chọn SQL Server Authentication thì phải nhập password. Password này có thể tìm thấy 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. Khi click vào Start->Run nhập services.msc->Enter.

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

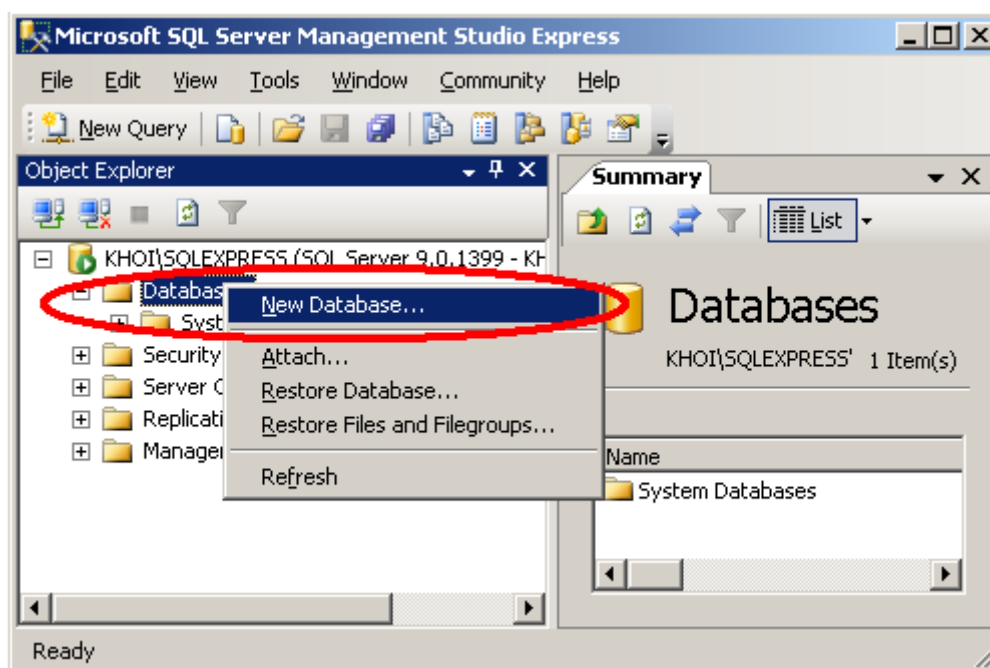
Giao diện sau khi nhập thành công



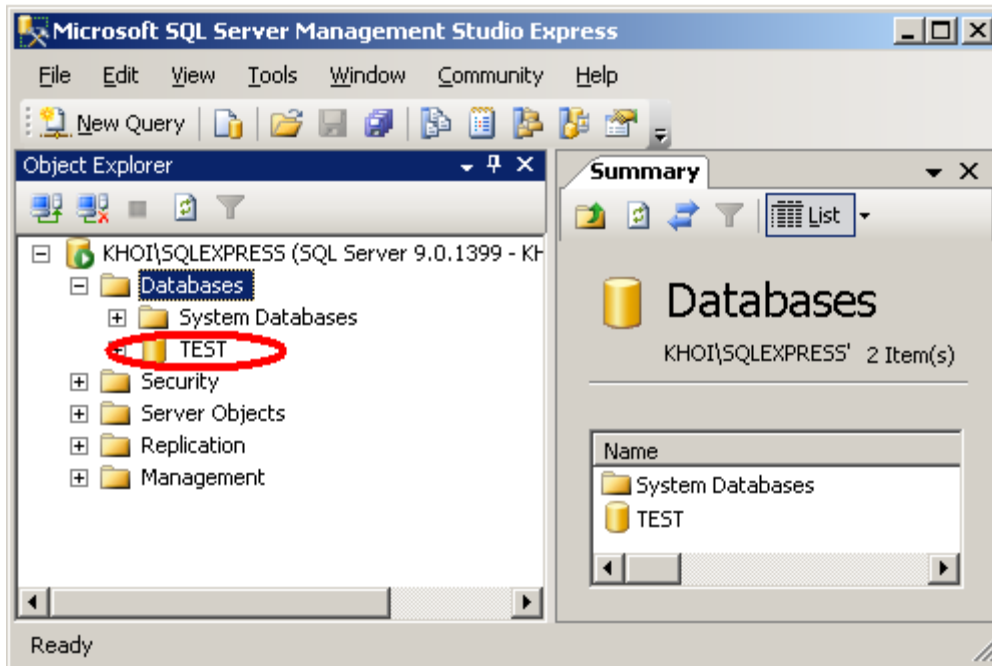
1.2 M t s thao tác c b n trên SQL Server 2005 Express Edition.

Microsoft SQL Server Management Studio cung c p m t giao di n thân thi n giúp cho ng i dùng th c hi n các thao tác m t cách d d dàng. M t s các thao tác c b n bao g m: t o CSDL m i, xóa CSDL, t o b ng, xóa b ng...C ng c n l u ý r ng các thao tác th c hi n thông qua giao di n thì u có th c th c hi n c b ng các câu l nh SQL.

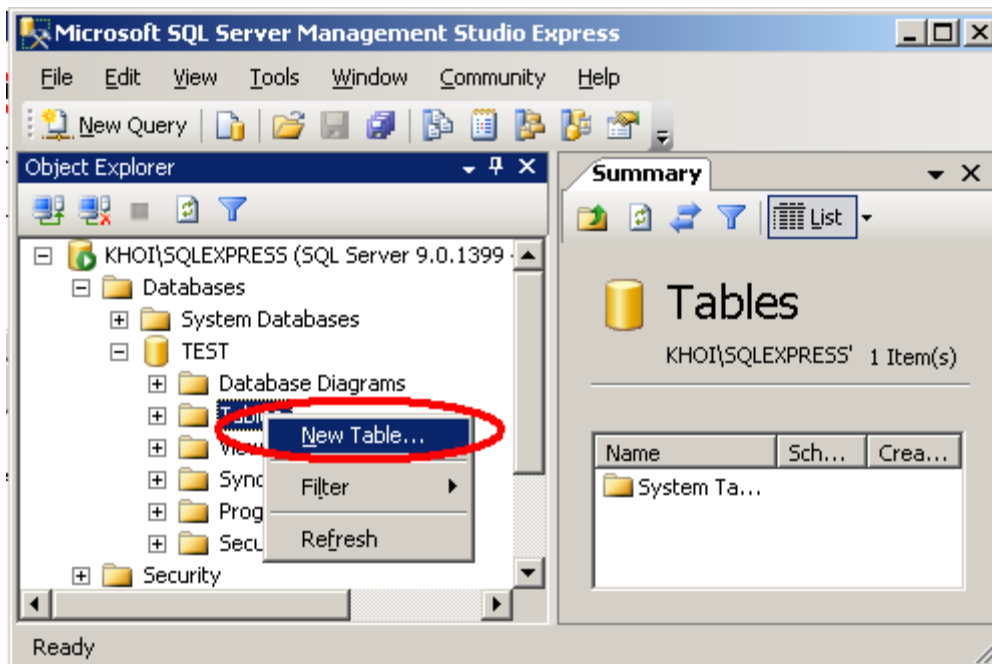
1.2.1 T o m t CSDL m i

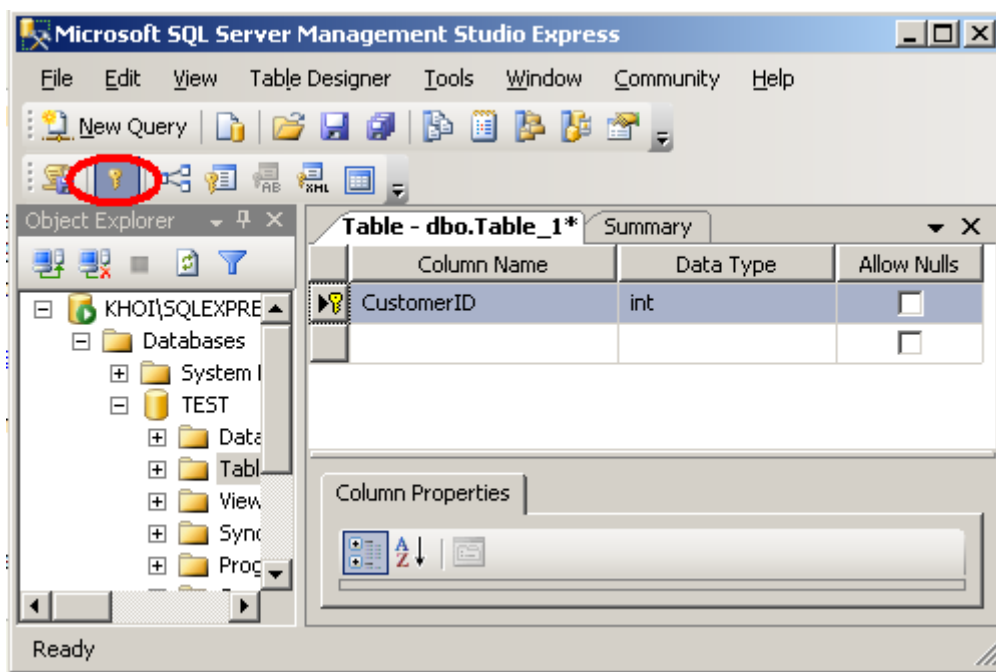


t tên Database trong Textbox Database Name, click OK.

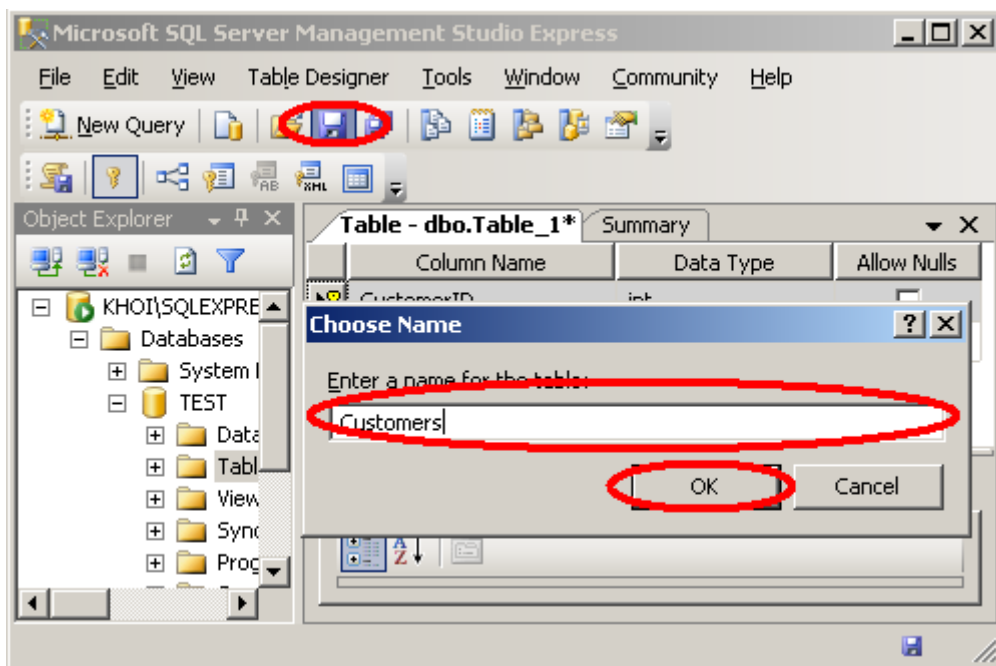


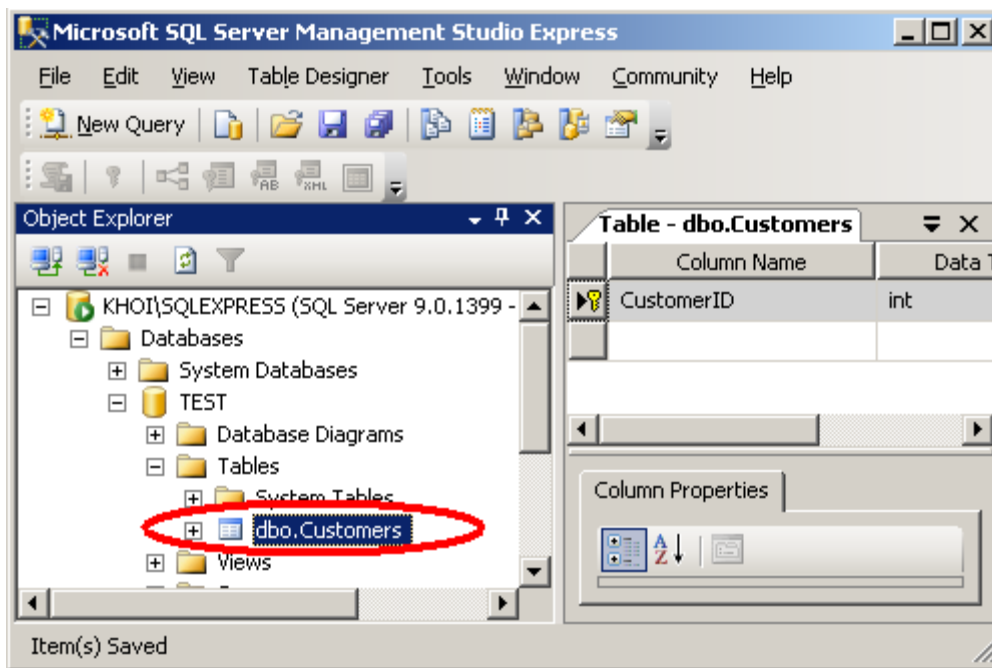
1.2.2 T o b ng m i





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

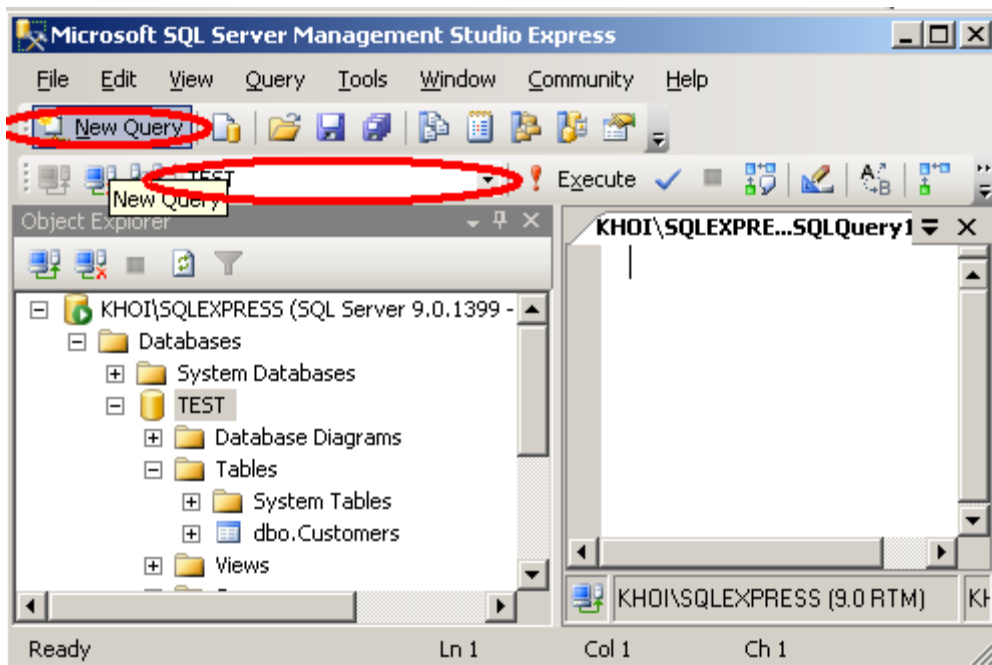




1.2.3 Xóa bảng, xóa CSDL

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 có 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 màn hình query editor viết câu lệnh SQL



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

2 Structured Query Language (SQL)

2.1 SQL là ngôn ngữ các sđl quan hệ

SQL, viết tắt của Structured Query Language (ngôn ngữ có cấu trúc), là công cụ sđl thống nhất, quản lý và truy xuất dữ liệu chủ yếu trong các sđl quan hệ. SQL là một hệ thống ngôn ngữ bao gồm tập các câu lệnh sđl thống nhất các sđl quan hệ.

Khả năng của SQL vượt xa so với một công cụ truy xuất dữ liệu, mà điều này là một ích 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 sđl khi cần thực hiện các chức năng mà một hệ quản trị sđl quan hệ cung cấp cho người dùng bao gồm:

nhập và xuất dữ liệu: SQL cung cấp khả năng nhập và xuất các sđl, các cấu trúc dữ liệu và các dữ liệu quan hệ liên quan giữa các thành phần dữ liệu.

Truy xuất và thao tác dữ liệu: Với SQL, người dùng có thể dàng thể hiện các thao tác truy xuất, bổ sung, cập nhật và xóa dữ liệu trong các sđl quan hệ.

kiểm soát truy cập: SQL có thể được sđl để phát và kiểm soát các thao tác của người sđl trên dữ liệu, đảm bảo an toàn cho các sđl quan hệ.

mô tả toàn vẹn dữ liệu: SQL nhập và các ràng buộc toàn vẹn trong các sđl quan hệ để mô tả tính hợp lệ và chính xác của dữ liệu thực hiện các thao tác cập nhật cũng như các liên hệ.

Như vậy, có thể nói rằng SQL là một ngôn ngữ hoàn thiện được sđl trong các hệ thống sđl quan hệ và là một thành phần không thể thiếu trong các hệ quản trị sđl quan hệ. 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 thống nhất các sđl quan hệ.

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 thể hiện trên các sđl quan hệ mà không cần phải chỉ ra cách thức thể hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ điển hình và dễ sđl.

2.2 Vai trò của SQL

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

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

SQL là ngôn ngữ hướng tính tác: Người sử dụng có thể dàng thông qua các trình biên dịch để các yêu cầu để đưa các câu lệnh SQL lên 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 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 cơ sở dữ liệu, định nghĩa các cấu trúc lưu trữ dữ liệu, lưu khi 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 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, hệ thống các máy chủ Web cũng như các máy chủ trên Internet sử dụng SQL với vai trò là ngôn ngữ tác vụ dữ liệu trong các cơ sở dữ liệu.

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

SQL là ngôn ngữ sử dụng cho các công cụ 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 giao tiếp giữa các hệ quản trị cơ sở dữ liệu.

2.3 Giới thiệu 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 định nghĩa và cùng mục đích sử dụng song mỗi thành phần hệ quản trị cơ sở dữ liệu có thể có một số thay đổi nào đó. Vì vậy đôi khi để nắm bắt cú pháp chi tiết của các câu lệnh có thể sẽ khác nhau trong các hệ quản trị 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 ngôn ngữ 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  
)
```

Chạy câu lệnh SQL trên, mở 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. Có thể là mở 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 khỏi CSDL, nghĩa là toàn bộ ngôn ngữ 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 kiểm soát 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 hiện 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 cấp hay từ chối từ trước đó.

```
revoke select
on nhanvien
to public
```

Sau khi thực hiện lệnh này, các quyền cấp 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)

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

```
Select
Insert
Update
Delete
```

Ví dụ : Câu lệnh sau sẽ liệt kê 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 liệt kê tất cả các cột của bảng Nhanvien. Toán tử like và ký tự % đi kèm 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 Thị Thu t')

Câu lệnh sau sẽ cập nhật lại 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 cơ sở dữ liệu nằm trong các truy vấn SQL và được xem là đối tượng trong cơ sở dữ liệu quan hệ là tên bảng và tên cột.

Có hai loại Identifiers mà loại thông thường (Regular Identifier) và một loại gọi là Delimited Identifier, loại này còn có dấu "" hay dấu [] để ngăn cách. Loại Delimited sẽ dùng để tránh các từ khóa của SQL Server (reserved keyword) hay các từ có khoảng trống.

Ví dụ :

*Select **

From "My table"

Where [sum] = 10

Trong các cơ sở dữ liệu của nhiều ứng 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 sử dụng (tức là bảng do ta tạo ra). Thông thường, trong các hệ quản trị cơ sở dữ liệu này cho phép nhập tên 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 định một bảng do một ứng dụng khác sử dụng (hiển nhiên là phải có phép) thì tên của bảng phải có dấu chấm sau tên của ứng dụng và phân cách với tên ứng dụng bằng dấu chấm:

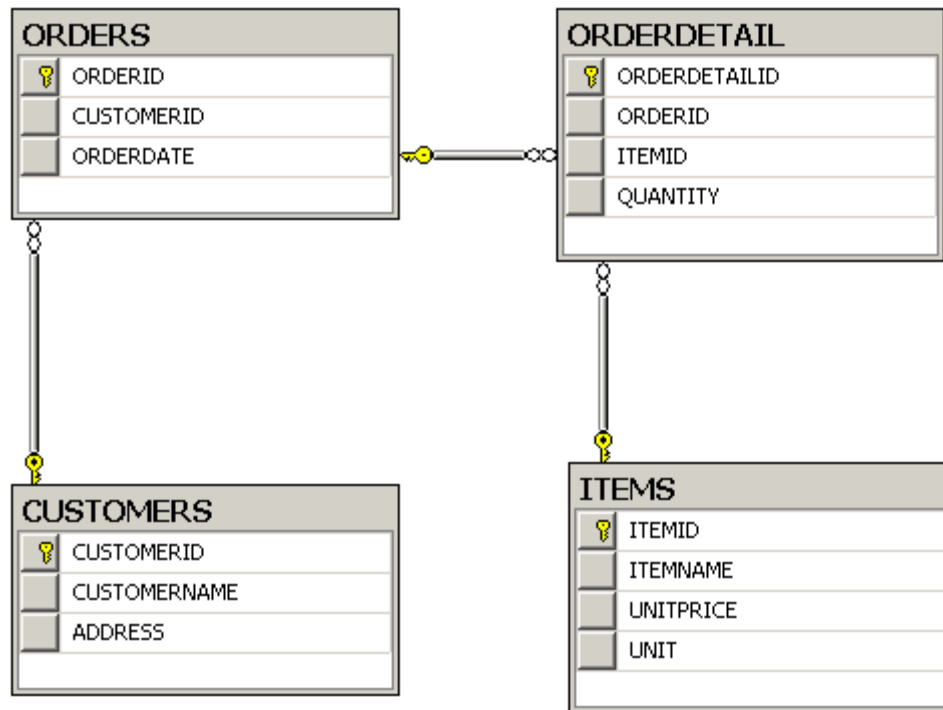
tên_ứng_dụng.tên_bảng

Một số đối tượng cơ sở dữ liệu khác (như khung nhìn, thủ tục, hàm), vì cơ sở dữ liệu tên của nó không thể viết tắt được.

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

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

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



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 thường gặp trong SQL.

Char(n)	Kiểu chuỗi có độ dài cố định
Nchar(n)	Kiểu chuỗi có độ dài cố định hỗ trợ UNICODE
Varchar(n)	Kiểu chuỗi có độ dài chính xác
Nvarchar(n)	Kiểu chuỗi có 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 chính xác cụ thể.
Decimal	Tổng thể 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
Bit	Kiểu bit (có giá trị 0 hoặc 1)
Datetime	Kiểu ngày giờ (chính xác đến phần trăm giây)
Smalldatetime	Kiểu ngày giờ (chính xác đến phút)
Binary	Dữ liệu nhị phân với độ dài cụ thể (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: Một số trong bảng sẽ chứa dữ liệu thu được duy nhất từ một kiểu dữ liệu trong SQL Server. Có thể chứa dữ liệu thu được từ kiểu nào sẽ quyết định lúc nhập vào.

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 có chức năng tương tự như trong các ngôn ngữ lập trình khác. Nó là cách khai báo trước cho dữ liệu trước khi sử dụng. Biến có thể bắt đầu bằng @ (Đặt đầu cho biến toàn cục - global variable - thì có hai đầu @@)

Ví dụ : Ví dụ dưới đây khai báo biến có tên @numberOfCustomers thông qua từ khóa declare. Biến này lưu số khách hàng mà chúng ta 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 các xây dựng sẵn trong SQL Server 2000/5 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 có cấu trúc như một table. Ví dụ như hàm OPENQUERY sẽ trả về một recordset và có thể truy vấn trực tiếp một table trong câu lệnh Select.

Aggregate Functions : Loại này làm việc trên một số giá trị và trả về một giá trị duy nhất 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 duy nhất và trả về một giá trị duy nhất. Trong loại này lại chia làm nhiều loại như các hàm về toán học, về thời gian, xử lý chuỗi... Ví dụ như hàm MONTH('2002-09-30') sẽ trả về tháng 9.

Các hàm User-Defined (có cấu trúc 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 duy nhất bằng câu lệnh RETURNS.

Table Functions : Loại này trả về một table

2.3.8 Các toán tử (Operators)

Trong SQL Server các biểu thức (expression) có thể xuất hiện nhiều toán tử. Ưu tiên của toán tử sẽ quy định thứ tự thực hiện của các phép tính. Thứ tự thực hiện như sau:

Bảng dưới đây mô tả các toán tử trong SQL Server 2005 Express Edition và mức 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, WHILE...

2.3.10 Chú thích (Comment)

T-SQL dùng ký hiệu -- chú thích cho một dòng và ký hiệu /*...*/ chú thích cho một nhóm dòng

Ví dụ :

/ Minh họa chú thích*

Chú thích cho một dòng 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 hiện tượng trong thế giới thực, do đó các giá trị dữ liệu tồn tại trong cơ sở dữ liệu có thể không xác định. Một giá trị không xác định 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 giá trị đó có tồn tại hay không.

Thời điểm hiện tại mà nó có giá trị chưa có nhúng rì có thể xảy ra.

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

Nhưng giá trị không xác định biểu diễn trong cơ sở dữ liệu quan hệ bởi các giá trị NULL. Đây là giá trị bất định và không nên nhầm lẫn với chuỗi rỗng (ví dụ dữ liệu chuỗi rỗng) hay giá trị không (ví dụ giá trị số). Giá trị NULL đóng một vai trò quan trọng trong các cơ sở dữ liệu và ảnh hưởng đến các quy trình cơ sở dữ liệu quan hệ hiện nay như truy vấn giá trị này.

3 Ngôn ngữ thao tác dữ liệu – DML

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

SELECT: Sẵn sàng truy xuất dữ liệu từ một hoặc nhiều bảng.

INSERT: Thêm dữ liệu.

UPDATE: Cập nhật dữ liệu

DELETE: Xóa dữ liệu

Trong số các câu lệnh này, có thể nói SELECT là câu lệnh tương đối phức tạp và có sẵn để dùng nhiều trong cơ sở dữ liệu. Vì câu lệnh này, ta không chỉ thể hiện các yêu cầu truy xuất dữ liệu đơn thuần mà còn có thể thể hiện các yêu cầu về logic để dữ liệu phức tạp. Chính vì vậy, phần cuối của chương này sẽ tập trung vào việc hiểu nhiều về câu lệnh SELECT. Các câu lệnh INSERT, UPDATE và DELETE sẽ bàn luận về chúng sau.

3.1 Câu lệnh SELECT

Câu lệnh SELECT có sẵn 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 thể 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 truy xuất dữ liệu). Ngoài ra, câu lệnh này còn cung cấp khả năng thể hiện các thao tác truy vấn và logic để 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]

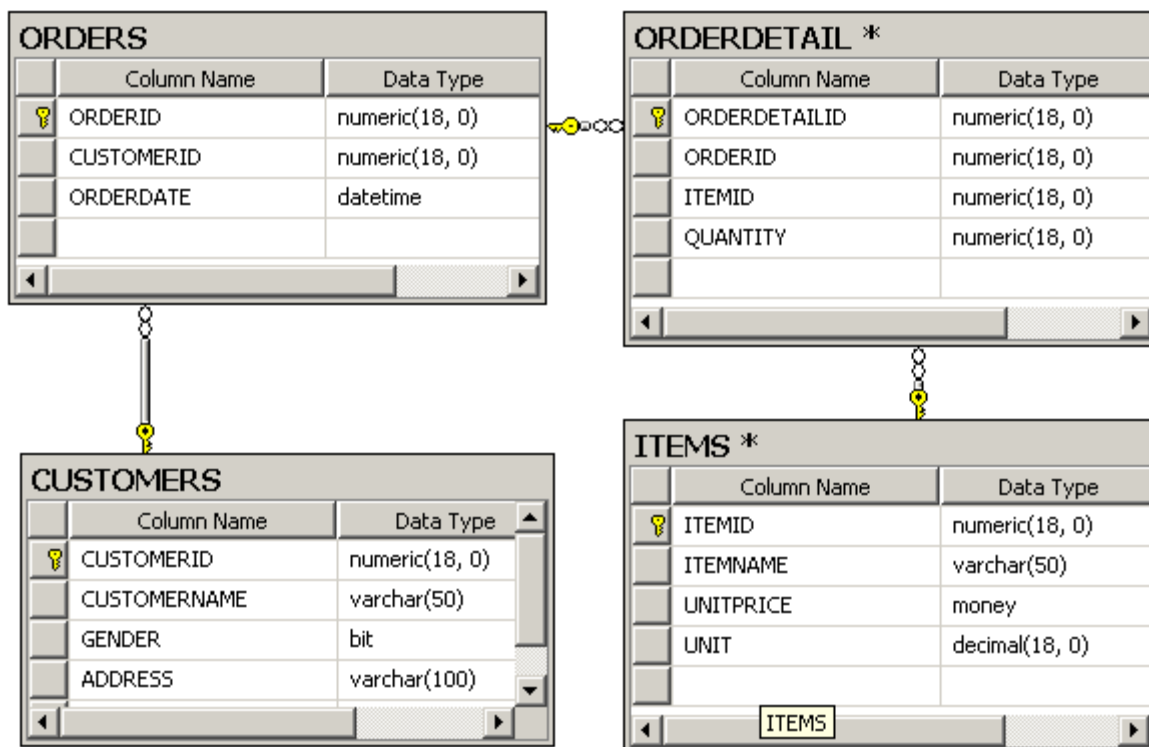
[ORDER BY cột_sắp_xếp]

[COMPUTE danh_sách_hàm_aggregate [BY danh_sách_cột]]

Điều cần lưu ý đầu tiên về câu lệnh này là các thành phần trong câu lệnh SELECT đều có sẵn để phải tuân theo đúng thứ tự trong cú pháp. Nếu không, câu lệnh sẽ có xem là không hợp lệ.

Câu lệnh SELECT sử dụng tác động lên các bảng dữ liệu và kết quả của câu lệnh cũng hiển thị dạng bảng, tức là môt 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 sử dụng để chọn các trường, các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường, các biểu thức cần chọn ngay sau từ khóa SELECT và phân cách nhau bởi dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

Ch n t t c các c t: Nh ấ n ấ nói trong ch ớ ng 1, chúng ta dùng d u * trong câu l nh Select hàm ý ch n h t t t c các c t. Trong tr ờ ng h p này, các c t c hi n th trong k t qu truy v n s tuân theo th t mà chúng ấ c t o ra khi b ng c nh ng a.

Ví d :

*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

Ch n m t s c t c th : Trong tr ờ ng h p c n ch nh c th các c t c n hi n th trong k t qu truy v n, ta ch nh danh sách các tên c t trong danh sách ch n. Th t c a các c t trong k t qu truy v n tuân theo th t c a các tr ờ ng trong danh sách ch n.

Ví d :

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

L u ý: N u truy v n c th c hi n trên nhi u b ng/khung nhìn và trong các b ng/khung nhìn có các tr ờ ng trùng tên thì tên c a nh ng tr ờ ng này n u xu t hi n trong danh sách ch n ph i c vi t đ i đ ng:

tên_b ng.tên_tr ờ ng

Thay i tiêu các c t:

Trong k t qu truy v n, tiêu c a các c t m c nh s là tên c a các tr ờ ng t ớ ng ng trong b ng. Tuy nhiên, các tiêu tr nên thân thi n h n, ta có th i tên các tiêu c a các c t. t tiêu cho m t c t nào ó, ta s đ ng cách vi t:

tiêu_ _c t = tên_tr ờ ng ho c

tên_tr ờ ng AS tiêu_ _c t ho c

tên_tr ờ ng tiêu_ _c t

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 cú trúc CASE...WHEN:

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

```
CASE bi_u_th_c  
WHEN bi_u_th_c_ki_m_tra THEN k_t_qu  
[ ... ]  
[ELSE k_t_qu _c_a_else]  
END  
hoặc:  
CASE  
WHEN i_u_ki_n THEN k_t_qu  
[ ... ]  
[ELSE k_t_qu _c_a_else]  
END
```

Ví dụ : Câu lệnh SQL dưới đây sẽ hiển thị giới tính của khách hàng tùy theo giá trị của cột u trong CSDL. Nếu giá trị trong CSDL là FALSE-> hiển thị giới tính Nữ, nếu giá trị là TRUE-> hiển thị giới tính Nam.

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

Câu lệnh trên có thể viết như sau:


```

select CUSTOMERNAME, ADDRESS,
case
    when GENDER = 1 then 'NAM'
    else '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, loại bỏ sự trùng lặp 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

Làm giảm số lượng ghi trong các dòng:

Từ khóa TOP 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 dụng 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 có sử dụng nhằm chỉ ra 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	45/28 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 có sử dụng nhằm xác định các điều kiện áp dụng cho truy xuất dữ liệu. Sau mệnh đề WHERE là mệnh đề logic và chọn ra những dòng dữ liệu nào thỏa mãn điều kiện được chỉ định trong kết quả truy vấn.

Ví dụ : Liệt 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 giá trị nằm trong khoảng (BETWEEN/ NOT BETWEEN)

T p h p

Ki m tra khuôn d ng d li u.

Các giá tr NULL

Các toán t so sánh

Toán t	Ý ngh a
=	B ng
>	L n h n
<	Nh h n
>=	L n h n ho c b ng
<=	Nh h n ho c b ng
<>	Khác
!>	Không l n h n
!<	Không nh h n

Ví d : Ví d d i ây l y tên, ngày sinh theo nh d ng dd/MM/yyyy và a ch c a nh ng khách hàng có tên Le Thi Hoa và tu i các khách hàng này l n h n 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

Ki m tra gi i h n c a d li u

ki m tra xem giá tr d li u n m trong (ngoài) m t kho ng nào ó, ta s d ng toán t BETWEEN/ NOT BETWEEN nh sau:

M nh	Ý ngh a
variable BETWEEN a AND b	a <= variable <=b
variable NOT BETWEEN a AND b	variable <a ho c variable > b

Ví d : ví d này t ng t ví d trên nh ng i u ki n là tu i n m trong kho ng t 20 n 30 tu i.

```
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)

Tính toán IN/ NOT IN sử dụng khi ta cần chọn ra những dữ kiện tìm kiếm dựa trên 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 bằng 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)

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

Ký tự đại diện	Ý nghĩa
%	Chuỗi ký tự bất kỳ không hoặc nhiều ký tự
_	Một ký tự bất kỳ
[]	Một ký tự nằm trong gì đó trong ngoặc vuông. Ví dụ :[a-f] hàm ý chỉ một trong các ký tự : a, b, c, d, e, f.
[^]	Một ký tự không nằm trong gì đó trong ngoặc vuông. 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 đầu 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 mệnh đề cho phép NULL sẽ nhận giá trị NULL trong các trường hợp sau:

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

Ngay khi nhập giá trị NULL vào cho cột đó.

Mệnh đề có dữ liệu là khi sử dụng giá trị NULL nếu giá trị khác thì sẽ gây tràn số.

Trong mệnh đề WHERE, kiểm tra giá trị của mệnh đề 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

Tóm tắt ngữ pháp câu lệnh SELECT

Câu lệnh SELECT ... INTO có tác dụng tạo mới bảng mới có cấu trúc và dữ liệu xác định từ kết quả truy vấn. Bảng mới có cấu trúc và dữ liệu giống với bảng gốc và sẽ được thêm vào danh sách hiện tại và sẽ được thêm vào danh sách kết quả truy vấn.

Ví dụ:

select CUSTOMERNAME,

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

into NEWCUSTOMERS

from Customers

Lưu ý: Nếu trong danh sách hiện tại có các biểu thức thì những biểu thức này phải được đặt trong ngoặc đơn.

S p x p k t qu truy v n

M c nh, các dòng d li u trong k t qu c a câu truy v n tuân theo th t c a chúng trong b ng d li u ho c c s p x p theo ch m c (n u trên b ng có ch m c). Trong tr ng h p mu n d li u c s p x p theo chi u t ng ho c gi m c a giá tr c a m t ho c nhi u tr ng, ta s d ng thêm m nh ORDER BY trong câu l nh SELECT; Sau ORDER BY là danh sách các c t c n s p x p (t i a là 16 c t). D li u c s p x p có th theo chi u t ng (ASC) ho c gi m (DESC), m c nh là s p x p theo chi u t ng. N u sau ORDER BY có nhi u c t thì vì c s p x p d li u s c u tiên theo th t t trái qua ph i.

Ví d : Ví d i ây s p x p thông tin các khách hàng theo th t tu i gi m d n.

```
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 th ch nh s th t c a c t c n c s p x p. Câu l nh ví d trên có th c vi t l i nh sau:

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

3.1.4 Phép h p (UNION)

Phép h p c s d ng trong tr ng h p ta c n g p k t qu c a hai hay nhi u truy v n thành m t t p k t qu duy nh t. SQL cung c p toán t UNION th c hi n phép h p. Cú pháp nh sau:

Câu_l nh_1

UNION [ALL] Câu_l nh_2

[UNION [ALL] Câu_l nh_3]

...

[UNION [ALL] Câu_l nh_n]

[ORDER BY c t_s p_x p]

[COMPUTE danh_sách_hàm_g p [BY danh_sách_c t]]

Trong đó

Câu_lệnh_i 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 ($i = 2, \dots, n$) 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 dữ liệu cho kết quả như sau:

a	b	c	d
a	1	1	1
a	2	1	1
a	3	1	2
b	4	3	2
b	1	1	1
c	1	1	1
c	2	3	7
d	1	1	1
e	1	1	1
f	1	2	3
g	6	6	6

UNION

f	g
a	3
a	5
a	1
b	8
c	7



A	B
a	1
a	2
a	3
a	5
b	1
b	4
b	8
c	1
c	2
c	7
d	1
e	1
f	1
g	6

select A,B from A

union

select F,G from B

Mặc nhiên, 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 liệt kê một dòng. Nếu muốn giữ lại các dòng này, ta phải sử dụng thêm từ khóa ALL trong truy vấn thành phần.

a	b	c	d
a	1	1	1
a	2	1	1
a	3	1	2
b	4	3	2
b	1	1	1
c	1	1	1
c	2	3	7
d	1	1	1
e	1	1	1
f	1	2	3
g	6	6	6

UNION

f	g
a	3
a	5
a	1
b	8
c	7

A	B
a	1
a	2
a	3
b	4
b	1
c	1
c	2
d	1
e	1
f	1
g	6
a	3
a	5
a	1
b	8
c	7

Khi sử dụng toán tử UNION thì cần phải hiểu phép hợp, ta cần chú ý các nguyên tắc sau:

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

Các cột trong cùng một truy vấn thành phần phải cùng kiểu dữ liệu.

Các cột trong cùng một truy vấn thành phần cần phải có cùng thứ tự. Nguyên nhân là do phép hợp so sánh các cột trong cùng một truy vấn.

Khi các kiểu dữ liệu khác nhau thì cần phải chuyển đổi sang cùng một kiểu dữ liệu (nếu có thể).

Tên cột trong kết quả của phép hợp là tên cột chính trong truy vấn đầu tiên.

Mệnh từ ORDER BY và COMPUTE dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị theo thứ tự nào đó trong câu lệnh UNION. Chúng không được sử dụng trong bất kỳ truy vấn thành phần nào.

Mệnh từ GROUP BY và HAVING chỉ có thể sử dụng trong một truy vấn thành phần. Chúng không được phép sử dụng tác động lên kết quả của phép hợp.

Phép toán UNION có thể 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 thể hiện một yêu cầu truy vấn dữ liệu hai hay nhiều bảng, ta phải sử dụng phép nối. Một câu lệnh sẽ 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: 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	8	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 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 thể hiện yêu cầu truy vấn, chúng ta phải thể 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 cách thể hiện các yêu cầu truy vấn dữ liệu liên quan nhiều bảng. Một câu lệnh sẽ thể 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 để chọn ra những điều kiện cần và kết hợp các dòng thỏa mãn điều kiện thành những dòng trong kết quả truy vấn.

thể hiện các phép nối, cần phải xác định 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 thực hiện phép nối giữa các bảng dữ liệu là gì

Trong các yếu tố trên, việc xác định chính xác điều kiện thực hiện phép nối giữa các bảng đóng vai trò quan trọng nhất. Trong tất cả các trường hợp, điều kiện của phép nối cần xác định nhúng vào mối quan hệ giữa các bảng cần phải truy xuất dữ liệu. Thông thường, đó là điều kiện bắt buộc nhau giữa khóa chính và khóa ngoài của hai bảng có mối quan hệ với nhau. Như vậy, có thể tóm tắt câu lệnh để thực hiện chính xác yêu cầu truy vấn dữ liệu đòi hỏi phải hiểu mối quan hệ cũng như ý nghĩa của chúng giữa các bảng dữ liệu.

Một câu lệnh để chọn cột với từ khóa SELECT. Các cột cần chọn tên sau từ khóa SELECT là các cột cần hiển thị trong kết quả truy vấn. Việc sử dụng tên các cột trong danh sách chọn có thể là:

Tên của mỗi 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 viết đầy đủ

tên_bảng.tên_cột

Dấu sao (*) cần 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 mỗi bảng nào đó, ta sử dụng cách viết:

tên_bảng.*

Mệnh FROM trong phép nối

Sau mệnh FROM của câu lệnh là danh sách tên các bảng (hay khung nhìn) tham gia vào truy vấn. Nếu ta sử dụng dấu * trong danh sách chọn thì tất cả các bảng liệt kê sau FROM sẽ nhúng tất cả các cột cần hiển thị trong kết quả truy vấn.

Mệnh WHERE trong phép nối

Khi hai hay nhiều bảng cần liên hệ nhau, ta phải chọn điều kiện thực hiện phép nối ngay sau mệnh WHERE. Điều kiện này có biểu diễn đầy đủ bằng 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 cần dùng để xác định điều kiện

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 (equi-join): Một phép nối bằng (equi-join) là một phép nối trong đó giá trị của các cột có sẵn trong hai bảng so sánh với nhau dựa trên tiêu chuẩn bằng và tất cả các cột trong các bảng tham gia nối đều có mặt trong kết quả.

Một dạng cụ thể của phép nối bằng có sẵn nữa là phép nối tự nhiên (natural-join). Trong phép nối tự nhiên, điều kiện nối giữa hai bảng chính là điều kiện bằng giữa khóa ngoài và khóa chính của hai bảng; Và trong danh sách chọn của câu lệnh chỉ ghi 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 vì thế nên:
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. Sẽ dễ dàng để loại bỏ bằng cách sử dụng phép kết nối tự nhiên và vì thế chỉ cần rõ các cột cần truy xuất.

Trong các câu lệnh này, ngoài điều kiện của phép nối có thể thêm trong mệnh đề WHERE còn có thể thêm các điều kiện tìm kiếm dữ liệu khác (điều kiện chọn). Thông thường, các điều kiện này có thể 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 kết nối

Phép kết nối là phép nối mà trong đó điều kiện nối có thể 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 con phải được đặt biệt danh.

Ví dụ : Giả sử có yêu cầu tìm ra các khách hàng có nhiều hơn một đơn 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 có gì thích thú sau: Liệt kê ra các mã khách hàng, mã hóa đơn và ngày đặt hàng từ bảng c1, o1 để so sánh với các mã khách hàng, mã hóa đơn và ngày đặt hàng từ bảng c2, o2. Nếu vì có so sánh hai trường hợp này thì 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 sẽ cho vào kết quả truy vấn.

Phép nối ngoài

Trong các phép nối đã học trên, chúng ta có giá trị trong các cột có thể không thỏa mãn điều kiện kết nối mà chỉ hiện ra trong kết quả truy vấn, và có gì là phép nối trong (inner join) Theo một nghĩa nào đó, phép nối này loại bỏ thông tin chỉ có trong bảng không thỏa mãn điều kiện nối. Tuy nhiên, đôi khi ta cần cần gì đó để thông tin này bằng cách cho phép bảng không thỏa mãn điều kiện nối có mặt trong kết quả của phép nối. Vì vậy, 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: $\ast=$): 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 thỏa mãn điều kiện của phép nối.

Phép nối ngoài phải (ký hiệu: $=\ast$): 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 thỏa mãn điều kiện của phép nối.

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

3.1.7 Phép nối theo chuẩn SQL-92

Chuẩn SQL2 (SQL-92) đã đưa ra một cách khác biểu diễn cho phép nối, trong cách biểu diễn này, điều kiện của phép nối không có chỗ đứng trong mệnh đề WHERE mà có chỗ đứng ngay trong mệnh đề FROM của câu lệnh. Cách sử dụng phép nối này cho phép ta biểu diễn phép nối có nghĩa điều kiện nối rõ ràng, cụ thể là trong trường hợp phép nối có thể hiển thị trên ba bảng trở lên.

Phép nối trong

Điều kiện thể hiện phép nối trong chỗ đứng 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 (FULL OUTER JOIN)

Công thức tổng hợp phép nối trong, điều kiện của phép nối ngoài có thể có chỗ đứng 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 : tìm ra các khách hàng có tồn 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 chỉ những dòng dữ liệu không thỏa mãn điều kiện nối ở bên trái trong phép nối thì phép nối ngoài yếu hiển thị trong kết quả truy vấn chỉ những dòng dữ liệu không thỏa mãn điều kiện nối ở cả hai bên tham gia vào 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

Thì hiển thị phép nối ngoài trái, nối ngoài phải và nối ngoài yếu 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 yếu:

```
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 cú pháp quan trọng của SQL2 là cho phép biểu diễn phép nối trên nhiều bảng dữ liệu một cách rõ ràng. Thuật toán hiển thị phép nối giữa các bảng được xác định theo ngữ pháp 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 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 khi cần thiết để hiển thị các yêu cầu truy vấn dữ liệu thông thường (chỉ ra, chọn, nối, ...) như đã đề cập trong các phần trước, câu lệnh SELECT còn cho phép thể hiện các thao tác truy vấn và tính toán thông kê trên dữ liệu.

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu, và trên mỗi nhóm dữ liệu thể 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 bộ hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được phép xuất hiện trong mệnh đề WHERE.

SQL cung cấp các hàm gộp dữ liệu như sau:

Hàm gộp

SUM([ALL| DISTINCT] biểu thức)

AVG([ALL| DISTINCT] biểu thức)

COUNT([ALL|DISTINCT] biểu thức)

Chức năng

Tính tổng các giá trị.

Tính trung bình của các giá trị

Đếm các giá trị trong biểu thức.

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

Nếu muốn hiển thị tên khách hàng phía trên cho từng nhóm 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 chức năng của câu lệnh SELECT có các hàm gộp và nhóm bí quyết không phải là hàm gộp thì nhóm bí quyết này phải có mặt trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Mệnh đề HAVING chức năng điều kiện trong hàm gộp

Mệnh đề HAVING sử dụng nhằm chức năng điều kiện về các giá trị tổng kết của sinh tổng các hàm gộp tương tự như cách thức mệnh đề WHERE thiếp lập các điều kiện cho câu lệnh SELECT. Mệnh đề HAVING thì không thể sử dụng ngay 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 thể 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 mua 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 v n con (Subquery)

Truy v n con là m t câu l nh SELECT c l ng vào bên trong m t câu l nh SELECT, INSERT, UPDATE, DELETE ho c bên trong m t truy v n con khác. Lo i truy v n này c s d ng bi u di n cho nh ng truy v n trong ó i u ki n truy v n đ li u c n ph i s d ng n k t qu c a m t truy v n khác.

Cú pháp c a truy v n con nh sau:

```
(SELECT [ALL | DISTINCT] danh_sách_ch_n
FROM danh_sách_b_ng
[WHERE i_u_ki_n]
[GROUP BY danh_sách_c_t]
[HAVING i_u_ki_n])
```

Khi s d ng truy v n con c n l u ý m t s quy t c sau:

M t truy v n con ph i c vi t trong c p d u ngo c. Trong h u h t các tr ng h p, m t truy v n con th ng ph i có k t qu là m t c t (t c là ch có duy nh t m t c t trong danh sách ch n).

M nh COMPUTE và ORDER BY không c phép s d ng trong truy v n con.

Các tên c t xu t hi n trong truy v n con có th là các c t c a các b ng trong truy v n ngoài.

M t truy v n con th ng c s d ng làm i u ki n trong m nh WHERE ho c HAVING c a m t truy v n khác.

N u truy v n con tr v úng m t giá tr , nó có th s d ng nh là m t thành ph n bên trong m t bi u th c (ch ng h n xu t hi n trong m t phép so sánh b ng)

Phép so sánh i v i v i k t qu truy v n con

K t qu c a truy v n con có th c s d ng th c hi n phép so sánh s h c v i m t bi u th c c a truy v n cha. Trong tr ng h p này, truy v n con c s d ng d i d ng:

```
WHERE bi_u_th_c phép_toán_s_h_c [ANY|ALL] (truy_v_n_con)
```

Trong ó phép toán s h c có th s d ng bao g m: =, <>, >, <, >=, <=; Và truy v n con ph i có k t qu bao g m úng m t c t.

Ví d : Câu truy v n sau ây tìm tên khách hàng có tu i l n nh t

```
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 trỏ chỉ ra giá trị, vì cơ sở dữ liệu phép so sánh chỉ trên số không hợp lệ. Trong trường hợp này, sau phép toán so sánh phải thêm từ ALL hoặc ANY. Từ ALL chỉ ra cơ sở dữ liệu khi cần so sánh giá trị của biểu thức với tất cả các giá trị trả về trong kết quả truy vấn con; ngược lại, phép so sánh với từ ANY có kết quả đúng khi chỉ cần một giá trị bất kỳ nào đó trong kết quả truy vấn con thỏa mãn điều kiện.

Ví dụ :

Toán tử IN/NOT IN

Khi cần thể 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

Từ EXISTS chỉ ra cơ sở dữ liệu kết quả của truy vấn con đúng:

```
WHERE [NOT] EXISTS (truy_vấn_con)
```

Từ EXISTS (tương đương NOT EXISTS) trả về giá trị True (tương đương False) nếu kết quả của truy vấn con có ít nhất một dòng (tương đương không có dòng nào). Điều khác biệt của cơ sở dữ liệu EXISTS với hai cách nêu trên là trong danh sách chỉ cần truy vấn con có thể có nhiều hơn hai câu.

Ví dụ :

Truy vấn con và mệnh đề HAVING

Một truy vấn con có thể sử dụng trong mệnh đề HAVING của một truy vấn khác. Trong trường hợp này, kết quả của truy vấn con sẽ được tổng hợp dựa trên điều kiện của các hàm gộp.

Ví dụ :

3.2 Thêm, cập nhật và xóa dữ liệu

Các câu lệnh thao tác dữ liệu trong SQL không chỉ sử dụng truy vấn dữ liệu mà còn thay đổi và cập nhật dữ liệu trong cơ sở dữ liệu. So với câu lệnh SELECT, vì cơ sở dữ liệu các câu lệnh bổ sung, cập nhật hay xóa dữ liệu nên ghi nhớ điều kiện của các hàm gộp này sẽ có phần 3 câu lệnh:

Lệnh INSERT

Lệnh UPDATE

Lệnh DELETE

3.2.1 Thêm dữ liệu

Dữ liệu trong các bảng có thể hiển thị dòng các dòng (bản ghi). Để bổ sung thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT. Dưới đây là các hình thức CSDL đã có trên SQL cùng với các cách để thực hiện thao tác thêm dữ liệu cho bảng:

Thêm từng dòng dữ liệu với câu lệnh INSERT. Đây là các sử dụng thông dụng 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 thì bắt buộc phải chỉ định giá trị các trường có bản ghi mới. Các chỉ định trong danh sách trị. Trong trường hợp này, thì các giá trị trong danh sách trị phải bằng với số lượng các trường có bản ghi cần bổ sung dữ liệu cũng phải tuân theo đúng thứ tự của các trường khi bảng được định nghĩa.

Ví dụ: Thêm thông tin mới 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 có tính là "YES" nên ta không cần thêm giá trị trường này mà SQL sẽ tự động tạo ra một giá trị cho trường này. Chỉ tính trường identity sẽ nói trong chương 4.

Trong trường hợp chỉ định 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 cần 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 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 là sử dụng bổ sung nhiều dòng dữ liệu vào một bảng, các dòng dữ liệu này có thể lấy từ một bảng khác thông qua câu lệnh SELECT. Cách này, các giá trị dữ liệu được bổ sung vào bảng không cần nhập thủ công 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 cần chèn trong bảng đích và phải có thứ tự thích hợp khi dữ liệu.

3.2.2 Cập nhật dữ liệu

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

```
UPDATE tên_bảng
SET tên_cột = bi_u_th_cột
[, ..., tên_cột_k = bi_u_th_cột_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 chèn các danh sách tên cột và biểu thức tính toán sau từ khóa SET. Mệnh đề WHERE trong câu lệnh UPDATE là sử dụng để chỉ định các dòng dữ liệu cụ thể mà câu lệnh (nếu không có mệnh đề, phạm vi tác động của câu lệnh sẽ 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

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

```

DELETE FROM tên_bảng
[FROM danh_sách_bảng]
[WHERE điều_kiện]

```

Trong câu lệnh này, tên của bảng cần xóa dữ liệu sẽ nằm sau DELETE FROM.

Mệnh WHERE trong câu lệnh sẽ sử dụng để chỉ ra điều kiện để xóa các dòng dữ liệu cần xóa. Nếu câu lệnh DELETE không có mệnh WHERE thì toàn bộ các dòng dữ liệu trong bảng sẽ bị xóa.

Ví dụ :

```

delete from Items
where itemid = 3

```

Xóa dữ liệu khi điều kiện liên quan đến nhiều bảng

Nếu điều kiện trong câu lệnh DELETE liên quan đến các bảng không phải là bảng cần xóa dữ liệu, ta phải sử dụng thêm mệnh FROM và sau đó là danh sách tên các bảng đó. Trong trường hợp này, trong mệnh WHERE ta chỉ thêm điều kiện liên quan đến 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 lồng vào trong mệnh WHERE trong câu lệnh DELETE để làm điều kiện cho câu lệnh tiếp theo là 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 nhả dữ liệu khỏi cơ sở dữ liệu mà còn xoá các dòng dữ liệu cụ thể trong bảng. Thay vì sử dụng câu lệnh DELETE trong trường hợp này, ta có thể sử dụng câu lệnh TRUNCATE có cú pháp như sau:

```
TRUNCATE TABLE tên_bảng
```

Ví dụ :

```
truncate table tmp1
```

4 Ngôn ngữ nhúng a d li u – DDL

Trong chương này sẽ trình bày nhóm các câu lệnh sử dụng ngôn ngữ nhúng a d li u và quản lý các đối tượng CSDL như bảng, khung nhìn, chỉ mục,... và đặc biệt là ngôn ngữ nhúng a d li u (DDL).

Với ngôn ngữ nhúng a d li u bao gồm các lệnh:

CREATE: nhúng a và tạo mới đối tượng CSDL.

ALTER: thay đổi nhúng a của đối tượng CSDL.

DROP: Xóa đối tượng CSDL đã có.

4.1 Tạo bảng

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

Bảng mới có thể 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 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, kiểu như phải là dữ liệu hay không; nếu có thì sử dụng 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 đặt 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 thể chấp nhận giá trị NULL hay không. Trong đó kiểu dữ liệu là thuộc tính bắt buộc.

Các_ràng_bu c: g m các ràng bu c v khuôn d ng d li u (ràng bu c CHECK) hay các ràng bu c v bào toàn d li u (PRIMARY KEY, FOREIGN KEY, UNIQUE)

Ví d : Ví d d i ây t o m t b ng 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
)

C t *customerid* có ki u d li u *int*, c ch nh thu c tính *identity(1,1)* ngh a là d li u c t này c thêm t ng b t u t l và m i l n có dòng m i thêm vào, giá tr c t này c t ng lên 1. C t này c ng c ch nh làm khóa chính c a b ng thông qua thu c tính *primary key*

Thu c tính NULL/ NOT NULL ch ra r ng c t ó có ch p nh n/ không ch p nh n giá tr NULL.

C t *gender* c ch nh giá tr m c nh là *true* ngh a là n u không ch nh giá tr cho c t này thì c t này có giá tr là *true*

Ví d :

Thêm dòng m i vào b ng customers v i giá tr truy n vào y cho các c t

insert into customers

values('Nguyen Van An', '22 Nguyen Thien Thuat', '5/5/1988', 'True')

Thêm dòng m i vào b ng customers s d ng giá tr m c nh

insert into customers (customername, address, birthday)

values('Nguyen Van An', '22 Nguyen Thien Thuat', '5/5/1988')

Thêm dòng m i vào b ng customers và không truy n giá tr cho các c t cho phép giá tr 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 00:00:00.000	1
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 sử dụng nhằm kiểm tra tính hợp lệ của dữ liệu. Mỗi khi có sự thay đổi dữ liệu trên bảng (INSERT, UPDATE), ràng buộc này sẽ sử dụng hàm 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_bu_c] 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)),
```

```
)
```

Thì hiển thị vì cố 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 có sẵn trong hầu hết các ngôn ngữ lập trình cơ sở dữ liệu. Khóa chính của một bảng là một hoặc nhiều cột mà giá trị của chúng là duy nhất trong bảng. Hay nói cách khác, giá trị của khóa chính sẽ giúp cho ta xác định được duy nhất một dòng (bản ghi) trong bảng dữ liệu. Một bảng chỉ có thể có duy nhất một khóa chính và bản thân khóa chính không chấp nhận giá trị NULL. Ràng buộc PRIMARY KEY là cần thiết cho việc tính toán và phân tích dữ liệu.

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 khóa chính của bảng chỉ bao gồm một cột và ràng buộc PRIMARY KEY được áp dụng cho cột đó, ta không cần thiết phải chỉ định danh sách cột sau từ khóa PRIMARY KEY. Tuy nhiên, nếu vì cần khai báo khóa chính cho nhiều cột (sử dụng khi số lượng các cột tham gia vào khóa là từ hai trở lên) thì bắt buộc phải chỉ định danh sách cột ngay sau từ khóa PRIMARY KEY và tên các cột sẽ phân cách nhau bằng 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: nhúng a 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 cách thay thế để thể hiện mối quan hệ giữa các dữ liệu 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 có thể có ràng buộc duy nhất. FOREIGN KEY có thể chứa giá trị NULL. Mặc dù mục đích chính của ràng buộc FOREIGN KEY là kiểm soát dữ liệu chứa trong bảng có FOREIGN KEY (table con) nhưng thực chất nó cũng kiểm soát luôn cả dữ liệu trong bảng chứa PRIMARY KEY (table cha). Ví dụ nếu ta xóa dữ liệu trong bảng cha thì dữ liệu trong bảng con trở nên "mất gốc" (orphan) vì không thể tham chiếu ngược về bảng cha. Do đó ràng buộc FOREIGN KEY sẽ đảm bảo điều này 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 sẽ nhúng a theo cú pháp dưới đây:

```

[CONSTRAINT tên_ràng_buộc] FOREIGN KEY [(danh_sách_cột)]
REFERENCES tên_bảng_tham_chi_u(danh_sách_cột_tham_chi_u)
[ON DELETE CASCADE / NO ACTION / SET NULL / SET DEFAULT]
[ON UPDATE CASCADE / NO ACTION / SET NULL / SET DEFAULT]

```

Vì thế nhúng a một ràng buộc FOREIGN KEY bao gồm các yếu tố sau:

Tên cột hoặc danh sách cột của bảng sẽ nhúng a tham gia vào khóa ngoài.

Tên của bảng sẽ tham chiếu về khóa ngoài và danh sách các cột sẽ tham chiếu đến trong bảng tham chiếu.

Cách thực hiện với các bảng ghi trong bảng con nhúng trong trường hợp các bảng ghi con tham chiếu trong bảng tham chiếu bị xóa (ON DELETE) hay cập nhật (ON UPDATE). SQL cho chúng ta 4 cách thực hiện

CASCADE: Trường xóa (cập nhật) của bảng ghi con tham chiếu bị xóa (cập nhật).

NO ACTION: (Mặc định) Nếu bảng ghi trong bảng tham chiếu đang cập nhật tham chiếu bị mất thì bảng ghi con tham chiếu trong bảng con nhúng sẽ không có phép xóa hoặc cập nhật (đối với các tham chiếu).

SET NULL: Cập nhật liên kết ngoài của bảng ghi thành giá trị NULL (nếu có cho phép nhận giá trị NULL).

SET DEFAULT: Cập nhật liên kết ngoài của bảng ghi nhận giá trị mặc định (nếu có 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 nhúng bảng

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

Bổ sung cột vào bảng.

Xoá m t c t kh i b ng.

Thay i nh ngh a c a m t c t trong b ng.

Xoá b ho c b sung các ràng bu c cho b ng

Cú pháp c a câu l nh ALTER TABLE nh sau:

ALTER TABLE tên_b ng

ADD nh_ngh a_c t /

ALTER COLUMN tên_c t ki u_d _li u [NULL / NOT NULL]

DROP COLUMN tên_c t /

ADD CONSTRAINT tên_ràng_bu c nh_ngh a_ràng_bu c

DROP CONSTRAINT tên_ràng_bu c

Ví d 1: Thêm m t c t m i vào b ng ORDERS

alter table orders

add description nvarchar(100) not null

Ví d 2: Thay i nh ngh a c t *description*

alter table orders

alter column description nvarchar(200) null

Ví d 3: Thêm ràng bu c CHECK vào c t *description*

alter table orders

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

Ví d 4: Xóa ràng bu c CHECK

alter table orders

drop chk_descriptionlength

Ví d 5: Xóa c t *description*

alter table orders

drop column description

Ví d 6: Thêm m t c t m i vào b ng *orders* và thêm ràng bu c cho c t này

alter table orders

add

description nvarchar(100) null,

constraint chk_descriptionlength CHECK (len(description) > 0)

N u b sung thêm m t c t vào b ng và trong b ng ã có ít nh t m t b n ghi thì c t m i c n b sung ph i cho phép ch p nh n giá tr NULL ho c ph i có giá tr m c nh.

Mu n xoá m t c t ang c ràng bu c b i m t ràng bu c ho c ang c tham chi u b i m t khoá ngoài, ta ph i xoá ràng bu c ho c khoá ngoài tr c sao cho trên c t không còn b t k m t ràng bu c và không còn c tham chi u b i b t k khoá ngoài nào.

Nếu bổ sung thêm ràng buộc cho một bảng đã có dữ liệu và ràng buộc bổ sung không có tồn tại thì các ràng buộc đã có trong bảng thì câu lệnh ALTER TABLE không thể hiện được.

4.4 Xóa bảng

Khi một bảng không còn cần thiết, ta có thể xóa nó ra khỏi cơ sở dữ liệu bằng câu lệnh DROP TABLE. Câu lệnh này cũng có thể xóa tất cả các 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 xóa 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 hiện được nếu bảng cần xóa đang có tham chiếu từ một ràng buộc FOREIGN KEY. Trong trường hợp này, ràng buộc FOREIGN KEY đang tham chiếu hoặc bảng đang tham chiếu từ bảng cần xóa phải được xóa trước.

Khi một bảng bị xóa, tất cả các ràng buộc, chỉ mục và trigger liên quan đến bảng cũng sẽ bị xóa theo. Do đó, nếu ta có một bảng thì cần phải liệt kê các ràng buộc này.

Ví dụ: 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 chỉ tạo ra khi sử dụng, khung nhìn là một công cụ thu thập CSDL.

Khung nhìn chỉ tạo ra từ câu lệnh truy vấn dữ liệu (lệnh SELECT), truy vấn tạm thời hoặc dữ liệu đã lưu.

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

Có thể thể 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 thực thể lưu trữ dữ liệu thực tế trong cơ sở dữ liệu. Thay vì thế, 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

Thì hiển thị câu truy vấn trên khung nhìn và tạo ra:

*select * from customerinfo*

CUSTOMERNAME	AGE	ADDRESS
Cao Văn Trung	49	33 Nguyễn Trung Trực
Trần Văn Phúc	38	99 Nguyễn Thị Minh Khai
Trần Việt Cường	28	45/2B Đa Tường
Nguyễn Văn Đại	53	76 Trần Phú
Lê Thị Hoa	26	56 Lê Hồng Phong
Nguyễn Thanh Thái	68	29A Phương Sài
Cao Văn Chung	NULL	12 Nguyễn Thiên Thuật
Nguyễn Văn An	32	14 Thông Nhất
Nguyễn Văn An	32	14 Thông Nhất

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 đặt tên các cột của khung nhìn cụ thể, chúng phải có cùng số lượng và vị trí của các 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 k t qu c a câu l nh SELECT có ít nh t m t c t c sinh ra b i m t bi u th c (t c là không ph i là m t tên c t trong b ng c s) và c t ó không c t tiêu .

T n t i hai c t trong k t qu c a câu l nh SELECT có cùng tiêu c t.

4.6 Thêm, c p nh t, xóa d li u trong VIEW

i v i m t s khung nhìn, ta có th ti n hành th c hi n các thao tác c p nh p, thêm và xóa d li u. Th c ch t, nh ng thao tác này s c chuy n thành nh ng thao tác trên các b ng c s và có tác ng n nh ng b ng c s .

V m t lý thuy t, có th th c hi n thao tác b sung, c p nh t và xóa, m t khung nhìn tr c tiên ph i tho mẫn các i u ki n sau ây:

Trong câu l nh SELECT nh ngh a khung nhìn không c s d ng t khoá DISTINCT, TOP, GROUP BY và UNION.

Các thành ph n xu t hi n trong danh sách ch n c a câu l nh SELECT ph i là các c t trong các b ng c s . Trong danh sách ch n không c ch a các bi u th c tính toán, các hàm g p.

Ngoài nh ng i u ki n trên, các thao tác thay i n d li u thông qua khung nhìn còn ph i m b o tho mẫn các ràng bu c trên các b ng c s , t c là v n m b o tính toàn v n d li u.

M c dù thông qua khung nhìn có th th c hi n c thao tác b sung và c p nh t d li u cho b ng c s nh ng ch h n ch i v i nh ng khung nhìn n gi n. i v i nh ng khung nhìn ph c t p thì th ng không th c hi n c; hay nói cách khác là d li u trong khung nhìn là ch c.

4.7 Thay i nh ngh a khung nhìn

Câu l nh ALTER VIEW dùng nh ngh a l i khung nhìn có c u trúc nh sau:

ALTER VIEW tên_khung_nhìn [(danh_sách_tên_c t)] AS

Câu_l nh_SELECT

Ví d : Ví d d i ây nh ngh a l i khung nhìn CUSTOMERINFO

alter view customerinfo

as

select CUSTOMERNAME, (year(getdate()) - year(birthday)) as AGE,

ADDRESS, GENDER

from customers

L u ý: l nh CREATE VIEW không làm thay i các quy n ã c c p phát cho ng i s d ng tr c ó.

4.8 Xóa khung nhìn

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

DROP VIEW tên_khung_nhìn

Ví dụ :

drop view customerinfo

Lưu ý: Nếu muốn xóa khung nhìn, toàn bộ quyền đã cấp phát cho người sử dụng trên khung nhìn cũng sẽ bị xóa. Do đó, nếu ta xóa khung nhìn thì phải tiến hành cấp phát lại quyền cho người sử dụng.

5 Th t c l u tr , hàm và trigger

5.1 Th t c l u tr (Stored procedure)

Th t c l u tr là m t i t ng trong CSDL, bao g m nhi u câu l nh T-SQL c t p h p l i v i nhau thành m t nhóm, và t t c các l nh này s c th c thi khi th t c l u tr c th c thi.

V i th t c l u tr , m t ph n nào ó kh n ng c a ngôn ng l p trình c a vào trong ngôn ng SQL. Th t c l u tr có th có các thành ph n sau:

Các c u trúc i u khi n (IF, WHILE, FOR) có th c s d ng trong th t c.

Bên trong th t c l u tr có th s d ng các bi n nh trong ngôn ng l p trình nh m l u gi các giá tr tính toán c, các giá tr c truy xu t c t c s d li u.

M t t p các câu l nh SQL c k t h p l i v i nhau thành m t kh i l nh bên trong m t th t c. M t th t c có th nh n các tham s truy n vào c ng nh có th tr v các giá tr thông qua các tham s (nh trong các ngôn ng l p trình). Khi m t th t c l u tr ã c nh ngh a, nó có th c g i thông qua tên th t c, nh n các tham s truy n vào, th c thi các câu l nh SQL bên trong th t c và có th tr v các giá tr sau khi th c hi n xong.

L i ích c a v i c s d ng th t c l u tr :

SQL Server ch biên d ch các th t c l u tr m t l n và s d ng l i k t qu biên d ch này trong các l n ti p theo tr khi ng i dùng có nh ng thi t l p khác. V i c s d ng l i k t qu biên d ch không làm nh h ng n hi u su t h th ng khi th t c l u tr c g i liên t c nhi u l n.

Th t c l u tr c phân tích, t i u khi t o ra nên v i c th c thi chúng nhanh h n nhi u so v i v i c ph i th c hi n m t t p r i r c các câu l nh SQL t ng ng theo cách thông th ng.

Th t c l u tr cho phép chúng ta th c hi n cùng m t yêu c u b ng m t câu l nh n gi n thay vì ph i s d ng nhi u dòng l nh SQL. i u này s làm gi m thi u s l u thông trên m ng.

Thay vì c p phát quy n tr c ti p cho ng i s d ng trên các câu l nh SQL và trên các i t ng c s d li u, ta có th c p phát quy n cho ng i s d ng thông qua các th t c l u tr , nh ó t ng kh n ng b o m t i v i h th ng.

Các th t c l u tr tr v k t qu theo 4 cách:

S d ng các tham s output

S d ng các l nh tr v giá tr , các l nh này luôn tr v giá tr s nguyên.

Tập các giá trị trả về của mệnh đề SELECT có trong kết quả truy vấn hoặc quá trình ghi nhận kết quả truy vấn khác trong một kết quả truy vấn.

Một biểu thức toán học có thể tham chiếu bên ngoài kết quả.

5.1.1 Tạo thủ tục

Thủ tục 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 mỗi khi thực thi.

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

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 hàng mới của khách hàng có mã khách hàng là 3

Nhập các chi tiết đơn hàng cho đơn hàng trên.

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

Trước tiên nhập đơn 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 hàng cho hóa đơn này. Giả sử rằng đơn hàng có mã là 4 và khách hàng đặt mua đơn 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 hàng mới, do đó người dùng sẽ phải viết lại câu lệnh tương tự nhau cho các khách hàng khác nhau. Một cách giải quyết vấn đề này là dùng thủ tục và dùng tham số nhận các thông tin thay thế.

```
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

```

Th c hi n th t c l u tr này nh sau:

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

5.1.2 L i g i th t c

Th t c l u tr c g i theo c u trúc

```
Tên_th _t c_l u _tr [danh_sách_tham_s ]
```

C n l u ý là danh sách tham s truy n vào trong l i g i ph i theo úng th t khai báo các tham s trong th t c l u tr .

N u th t c c g i t m t th t c khác, th c hi n bên trong m t trigger hay ph i h p v i câu l nh SELECT, c u trúc nh sau;

```
Exec Tên_th _t c_l u _tr [danh_sách_tham_s ]
```

5.1.3 Bi n trong th t c l u tr

Trong th t c l u tr có th có các bi n nh m l u các k t qu tính toán hay truy xu t t CSDL. Các bi n trong th t c c khai báo b ng t khóa DECLARE theo c u trúc nh sau:

```
DECLARE @tên_bi n ki u_d _li u
```

Ví d :

```

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 chỉ truy vấn cho thủ tục khi có lệnh insert thì thủ tục là biến, nhưng thay đổi giá trị của biến trong thủ tục sẽ không có gì lạ 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 gửi lại giá trị của biến sau khi kết thúc thủ tục, ta phải khai báo tham số của thủ tục theo cú pháp như sau:

```
@tên_tham_sử_kiểm_lại OUTPUT
```

Ví dụ trên có ví dụ 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 tự như vì cú pháp tham số OUTPUT, câu lệnh RETURN trả về giá trị cho kết quả của thủ tục stored procedure.

Ví dụ :

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

Thủ tục thử nghiệm kết quả như sau:

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

5.1.5 Tham số với giá trị mặc định

Các tham số có thể 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 giá trị cho tham số khi có lỗi xảy ra trong thủ tục.

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

@tên_tham_số_kiểm_duyệt_liệu = giá_trị_mặc_định

Ví dụ :

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

Thủ tục thử nghiệm kết quả 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

Thức thì thức và truy vấn 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 lại thức

Khi muốn thức cũ của ta có thể tiến hành chỉnh sửa lại thức có bằng câu lệnh ALTER PROCEDURE có cú pháp như sau:

```
ALTER PROCEDURE tên_thức [(danh_sách_tham_số)]
[WITH RECOMPILE/ENCRYPTION/RECOMPILE,ENCRYPTION]
AS
```

Các câu lệnh của thức

Câu lệnh này sử dụng cùng tên câu lệnh CREATE PROCEDURE. Vì vậy sửa lại thức cũ có không làm thay đổi các quy tắc phát triển trên thức cũ không tác động đến các thức khác hay trigger phụ thuộc vào thức này.

5.1.7 Xóa thức

Để xóa thức cũ có, ta sử dụng câu lệnh DROP PROCEDURE với cú pháp như sau:

```
DROP PROCEDURE tên_thức
```

Khi xóa thức, thức các quy tắc cho nó sẽ được gỡ bỏ trên thức có cùng tên thì bị xóa bỏ. Do đó, nếu muốn thức, ta phải tiến hành cập nhật lại các quy tắc trên thức cũ.

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 nôi 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: Các sử dụng trả về một duy nhất một giá trị dựa trên một các tham số truy vấn vào. Ví dụ: ta có thể tạo ra một UDF vô hướng như 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: c ng tr v k t qu là m t t p h p nh ng có th đ a trên nhi u câu l nh SQL.

5.2.1 Hàm vô h ng - Scalar UDF

Scalar UDF c t o ra b ng câu l nh CREATE FUNCTION có c u trúc nh sau;

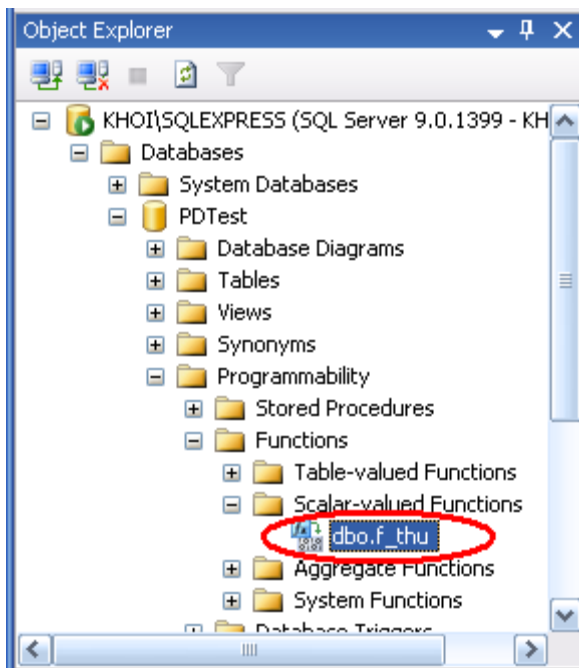
```
CREATE FUNCTION tên_hàm  
([danh_sách_tham_s ]) RETURNS (ki u_tr _v _c a_hàm)  
AS BEGIN  
các_câu_l nh_c a_hàm  
END
```

Ví d :

Câu l nh d i ây nh ngh a hàm tính ngày trong tu n (th trong tu n) c a m t giá tr ki u 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'ch nh t'  
when 2 then N'th hai'  
when 3 then N'th ba'  
when 4 then N'th t '  
when 5 then N'th n m'  
when 6 then N'th sáu'  
else N'th b y'  
end  
return (@st) /* tr tr v c a hàm */  
end
```

Sau khi ch y thành công, hàm tr thành m t i t ng trong CSDL và có th c truy xu t nh các hàm c xây d ng s n trong SQL Server 2005 Express Edition.



Ví dụ :

```
select CUSTOMERNAME, dbo.f_thu(BIRTHDAY)
from customers
```

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

5.2.2 Hàm n i tuyền - Inline UDF

Hàm n i tuyền c ãnh ngh ã b ãng l ãnh CREATE FUNCTION.

CREATE FUNCTION tên_hàm ([danh_sách_tham_s])

RETURNS TABLE

AS

RETURN (câu_l ãnh_select)

Cú pháp c ã hàm n i tuyền ph ãi tuân theo các qui t ãc sau:

Kì u tr ã v ã c ã hàm ph ãi c ãnh b ãi m ãnh RETURNS TABLE.

Trong ph n thân c a hàm ch có duy nh t m t câu l nh RETURN xác nh giá tr tr v c a hàm thông qua duy nh t m t câu l nh SELECT. Ngoài ra, không s d ng b t k câu l nh nào khác trong ph n thân c a hàm.

Ví d : Ví d d i ây l y ra các khách hàng tùy thu c vào giá tr mã khách hàng truy n vào cho tham s .

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

Vì c g i các hàm n i tuy n c ng t ng t nh vì c g i các hàm vô h ng.

Ví d :

```
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 g m nhi u câu l nh bên trong – Multi statement UDF

Hàm này c ng c nh ngh a b ng l nh CREATE FUNCTION

```
CREATE FUNCTION tên_hàm
([danh_sách_tham_s ])
RETURNS @bi n_b ng TABLE nh_ngh a_b ng
AS
BEGIN các_câu_l nh_trong_thân_hàm
RETURN
END
```

L u ý: sau t khóa RETURNS là m t bi n b ng c nh ngh a. Và sau t khóa RETURN cu i hàm không có tham s nào i kèm.

Ví d :

```
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

```

Vì c g i hàm multi statement UDF c ng t ng t các lo i 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 i hàm

Dùng l nh ALTER FUNCTION thay i nh ngh a hàm. C u trúc c a câu l nh ALTER FUNCTION t ng t nh CREATE FUNCTION

Ví d :

```

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

Trong trường hợp này, khi hàm bị xóa các quy định cho nó sẽ dùng trên hàm gốc để xóa. Do đó khi nhập lại hàm này, ta phải nhập lại quy định cho các nó dùng.

5.3 Trigger

Trigger là một dạng cơ chế để thực hiện một số hành 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 cơ chế của trigger

Trigger có thể thực hiện thông qua các sự kiện mà không thể hiển thị ngay.

Trigger sẽ được kích hoạt khi có sự thay đổi dữ liệu.

Khi trigger thực hiện theo các sự kiện Insert hoặc Delete thì dữ liệu khi thay đổi sẽ chuyển sang các bảng INSERTED và DELETED, là 2 bảng tạm thời chứa trong bộ nhớ, các bảng này chỉ tồn tại trong phạm vi các lệnh trong trigger. Các bảng này thường sẽ được khôi phục lại khi kết thúc phiên làm việc (roll back).

Trigger chia thành 2 loại INSTEAD OF và AFTER: INSTEAD OF là loại trigger mà hoạt động của sự kiện trigger sẽ bị bỏ qua và thay vào đó là các lệnh trong trigger sẽ thực hiện. AFTER trigger là loại trigger thông thường, khác với loại INSTEAD OF thì loại trigger này sẽ thực hiện các lệnh bên 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 biến pháp bổ sung toàn vẹn dữ liệu khác không bổ sung được. Các công việc này sẽ thực hiện kiểm tra tính toán và xử lý khi đưa dữ liệu vào CSDL, còn Trigger thực hiện kiểm tra tính toán và xử lý khi công việc đã thực hiện.

Khi CSDL chưa được chuẩn hóa (Normalization) thì có thể xảy ra dữ liệu thừa, chèn nhúng và trùng lặp trong CSDL thì yêu cầu xử lý là dữ liệu cần phải được chỉnh sửa trong mini. 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 thay đổi theo).

5.3.3 Khi nào sau cần trigger

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

Các thao tác trên dữ liệu (xóa, cập nhật và bổ sung) có thể có trigger phát hiện ra và thông báo cho người quản trị 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 những mối quan hệ phức tạp giữa các bảng trong cơ sở dữ liệu mà bản thân các ràng buộc không thể thể hiện được.

5.3.4 Định nghĩa trigger

Câu lệnh CREATE TRIGGER có sử dụng định nghĩa trigger và có cú 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 trên, chuẩn SQL định nghĩa hai bảng logic INSERTED và DELETED 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 thu được vào câu lệnh tác động lên bảng làm kích hoạt trigger; có thể trong các trường hợp sau:

Khi câu lệnh DELETE có thể thi trên bảng, các dòng dữ liệu bị xóa sẽ 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 về trigger bằng câu lệnh INSERT. Bảng DELETED trong trường hợp này không có dữ liệu.

Khi câu lệnh UPDATE có thể thi trên bảng, các dòng dữ liệu cũ sẽ sao chép vào bảng DELETED, còn trong bảng INSERTED sẽ là các dòng sau khi đã được cập nhật.

Họ t ng	B ng INSERTED	B ng DELETED
INSERT	d l u c insert	không có d l u
DELETE	không có d l u	d l u b xóa
UPDATE	d l u c c p nh t	d l u tr c khi c p nh t

Ví d 1: Ví d d i ây minh h a vi c trigger c kích ho t khi thêm d l u vào b ng CUSTOMERS

```

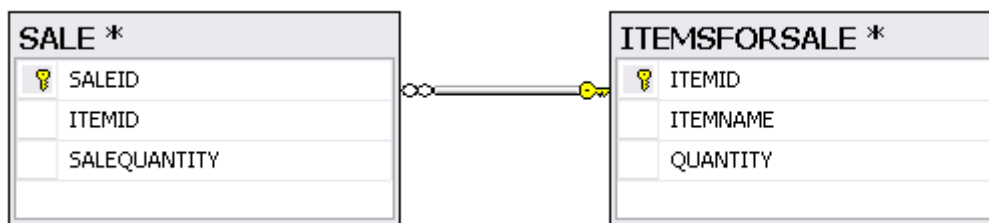
if exists (select name from sysobjects
where name = 't_CheckCustomerName' and type = 'TR')
drop trigger t_CheckCustomerName
go
create trigger t_CheckCustomerName
on customers
for insert
as
declare @lengthOfName int
select @lengthOfName = len(inserted.customername)
from inserted
if @lengthOfName <=1
print N'Tên không h   p l   '
rollback tran
go
Thêm m   t khách hàng m   i 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 d 2: Ví d d i ây minh h a trigger c kích ho t khi có s thay i mang tính ây chuy n gi a các b ng.

Gi s có CSDL nh sau:



Ví dụ dữ liệu trong 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 ITEMSFORSALE sẽ giảm xuống còn 90. Trigger đã cài đặt sẽ thực hiện công việc này.

```

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 cách minh họa trigger kích hoạt khi có sự thay đổi mang tính dây chuyền giữa các bảng trong trường hợp này dữ liệu thay đổi liên quan đến nhiều dòng.

Gi s ng i qu n lý mu n thay i s l ng bán m t hàng LAPTOP trong b ng SALE lên thêm 5 n v . Nh v y t k t qu ví d 2, ta th y c n ph i gi m s l ng LAPTOP trong b ng ITEMSFORSALE xu ng 10 n v . Tuy nhiên, trong th c t khi s l ng các dòng trong b ng SALE r t l n, khi ó ph i s d ng 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)

```

Th c hi n c p nh t cho b ng 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 d 4: Ví d này minh h a INSTEAD OF trigger. Trigger d i ây s không cho th c hi n thao tác xóa trên b ng 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 các bảng

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

Trên ví dụ 3 trong phần trước đây ta 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)
```

Trong ví dụ này trigger sẽ kích hoạt khi có sự thay đổi dữ liệu trong các bảng *salequantity* của bảng Sale. Nếu có sự thay đổi dữ liệu trên các bảng khác thì trigger sẽ không kích hoạt. Câu lệnh *delete* này 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 thực thi khi trigger kích hoạt.

5.3.6 S d ng trigger và Giao tác (TRANSACTION)

Khi m t trigger c kích ho t, SQL Server luôn t o ra m t giao tác theo dõi nh ng thay i do câu l nh kích ho t trigger ho c do b n thân trigger gây ra. S theo dõi này cho phép CSDL quay tr l i tr ng thái tr c ó.

Ví d : Ví d d i ây xây d ng trigger không cho phép nh p vào m t b n ghi trong b ng SALE khi s l ng hàng bán l n h n s l ng hàng th c t còn l i trong b ng 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's  l  ng nh  p vào l  n h  n s  l  ng hi  n có'
    rollback tran
```

Ti n hành thêm vào b ng SALE s l i u nh 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

c gì i thi u trong SQL Server 2005, khác v i DML trigger c kích ho t khi có s thay i d li u trên b ng, DDL trigger c thi t k áp ng l i các s ki n di n ra trên server hay trên CSDL. M t DDL trigger có th c kích ho t khi ng i dùng th c hi n các l nh CREATE TABLE hay DROP TABLE. c p server, DDL trigger có th c kích ho t khi có m t tài kho n m i c t o ra

DDL trigger c l u tr trong CSDL mà DDL trigger c g n vào. V i các Server DDL Trigger theo dõi các thay i c p Server, c l u tr trong CSDL master.

DDL trigger c t o ra c ng b ng câu l nh CREATE TRIGGER v i c u trúc nh sau:

```
CREATE TRIGGER tên_trigger
ON { ALL SERVER / DATABASE }
FOR { lo i_s _ki n } [ ,...n ]
AS { các_câu_l nh_SQL }
```

Trong ó:

ALL SERVER / DATABASE: quy nh trigger s kích ho t d a trên các s ki n di n ra trên Server hay các s ki n di n ra trên CSDL.

lo i_s _ki n: là m t s ki n n c p Server hay c p CSDL làm kích ho t DDL trigger nh : CREATE_TABLE, ALTER_TABLE, DROP_TABLE...

Ví d 1: Câu l nh d i ây xây d ng m t trigger c kích ho t khi x y ra các s ki n c p CSDL. Trigger này s ng n ch n các l nh DROP TABLE và ALTER TABLE.

```
create trigger t_safety
on database
for CREATE_TABLE, DROP_TABLE
as
print N'Ph i xóa trigger t_safety tr c khi ALTER hay DROP b ng'
rollback tran
```

Ti n hành xóa b ng ORDERDETAIL

```
drop table orderdetail
```



Ví dụ 2: Câu lệnh để hủy xây dựng một trigger để kích hoạt khi xảy ra các sự kiện trên Server. Trigger này sẽ ngăn chặn vì có tồn tại 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_DoNotAllowCreateNewLogin
ON ALL SERVER
FOR CREATE_LOGIN
AS
PRINT N'Phải DROP trigger t_DoNotAllowCreateNewLogin trước 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 vô hiệu hóa trong một số trường hợp:

Trigger gây ra lỗi trong quá trình xử lý CSDL

Quá trình nhập hay khôi phục dữ liệu không thể hoạt động trigger.

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

```
DISABLE TRIGGER tên_trigger
ON { tên_địa_chỉ / 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 d 2: Ví d này s khôi ph c l i trigger *t_DoNotAllowCreateNewLogin*
enable trigger t_DoNotAllowCreateNewLogin
on all server

Ti n hành t o m t account login m i:

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 l u và ph c h i d l i u (Backup and Restore)

Ch ng này s gi i thi u k thu t sao l u (backup) và khôi ph c (restore) d l i u, là k thu t th ng c s d ng b o m an toàn d l i u phòng tr ng h p CSDL có s c .

6.1 Các lý do ph i th c hi n Backup

Trong quá trình th c hi n qu n tr CSDL SQL Server thì m t s nguyên nhân sau ây b t bu c b n ph i xem xét n k thu t sao l u và khôi ph c d l i u:

Thi t b l u tr (CSDL n m trên các thi t b l u tr này) b h h ng.

Ng i dùng vô tình xóa d l i u.

Các hành ng vô tình hay c ý phá ho i CSDL.

6.2 Các lo i Backup

Microsoft SQL Server 2005 cung c p hai k thu t sao l u CSDL chính: full backup và differential backup.

6.2.1 Full backup và Differential backup

Full backup: sao l u m t b n y c a CSDL trên các ph ng ti n l u tr . Quá trình full backup có th ti n hành mà không c n offline CSDL, nh ng quá trình này l i chi m m t l ng l n tài nguyên h th ng và có th nh h ng nghiêm tr ng t i th i gian áp ng các yêu c u c a h th ng.

Differential backup: c xây d ng nh m làm gi m th i gian c n thi t th c hi n quá trình full backup. Differential backup ch sao l u nh ng thay i trên d l i u k t l n full backup g n nh t. Trong nh ng h th ng CSDL l n, quá trình differential backup s s d ng tài nguyên ít h n r t nhi u so v i quá trình full backup và có th không nh h ng n hi u su t c a h th ng.

Quá trình differential ch sao l u nh ng s thay i c a d l i u t l n full backup g n nh t, do ó khi có s c v i CSDL n u không có b n sao l u c a quá trình full backup thì b n sao l u c a quá trình differential backup s tr nên vô ngh a.

Ví d :

Công ty XYZ th c hi n full backup vào cu i ngày th 6 hàng tu n và th c hi n differential backup vào t i các ngày t th 2 t i th 5. N u CSDL có s c vào sáng th 4, qu n tr viên CSDL s ph c h i d l i u b ng b n sao l u c a quá trình full backup c a ngày th 6 tu n tr c và sau ó ph c h i các thay i c a d l i u b ng cách áp d ng b n sao l u c a quá trình differential backup vào ngày th 3.

6.2.2 Transaction log backup

Quá trình full backup và differential backup chỉ m nh u tài nguyên h th ng và nh h ng n hi u su t làm vi c h th ng nên th ng c th c hi n vào sau gi làm vi c. Tuy nhiên i u này có th d n n các m t mát d li u trong m t ngày làm vi c n u CSDL có s c tr c khi quá trình sao l u di n ra. Transaction log backup là m t gi i pháp nh m gi m thi u t i a l ng d li u có th m t khi có s c CSDL.

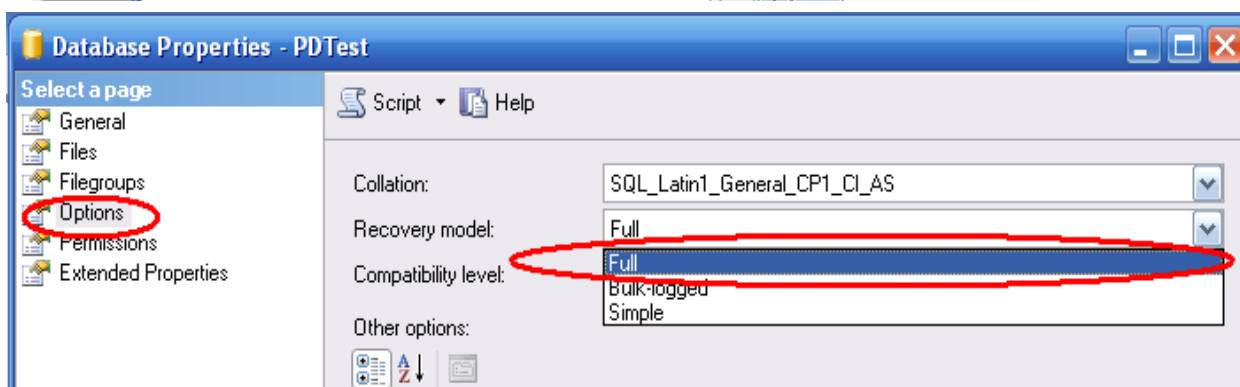
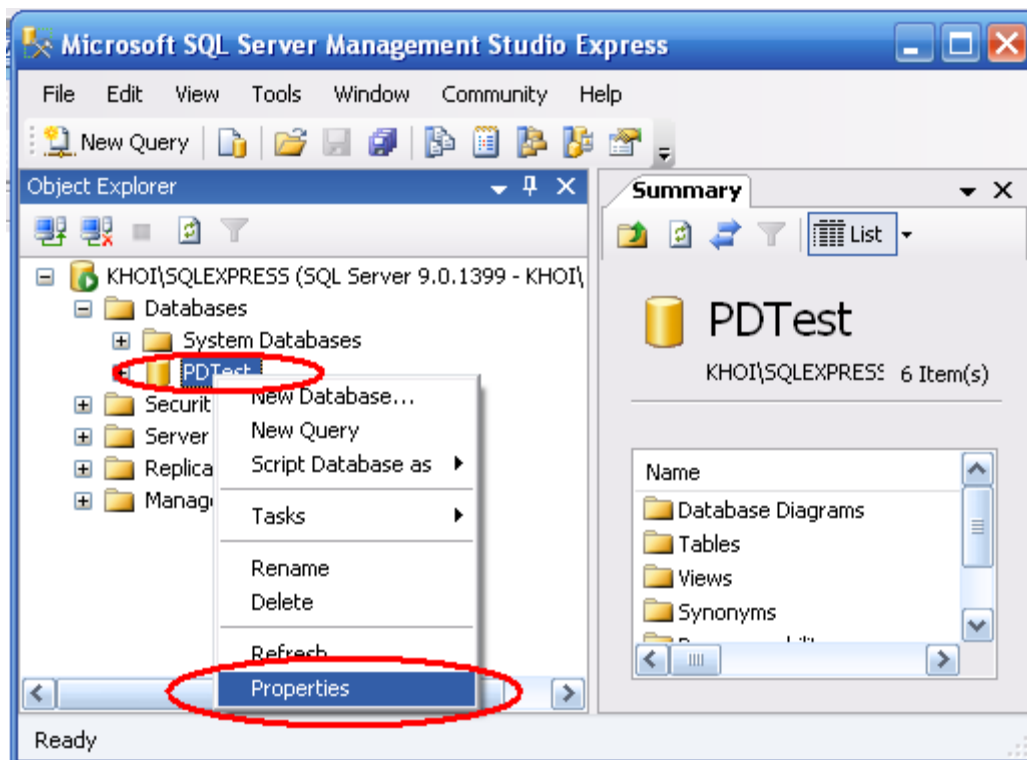
Trong quá trình ho t ng, SQL Server s d ng transaction log theo dõi t t c các thay i trên CSDL. Log b o m CSDL có th ph c h i sau nh ng s c t xu t và c ng m b o ng i dùng có th quay ng c các k t qu trong các giao tác CSDL. Các giao tác ch a hoàn thành c l u trong *log* tr c khi c l u v nh vi n trong CSDL.

Transaction log backup sao l u transaction log c a CSDL vào thi t b l u tr . M i khi transaction log c sao l u, SQL Server b i các transaction ã th c hi n thành công (committed transaction) và ghi các transaction vào ph ng ti n sao l u. Transaction log backup s d ng tài nguyên h th ng ít h n r t nhi u so v i full backup và differential backup, do ó có th s d ng transaction log backup b t k th i gian nào mà không s nh h ng n hi u su t h th ng.

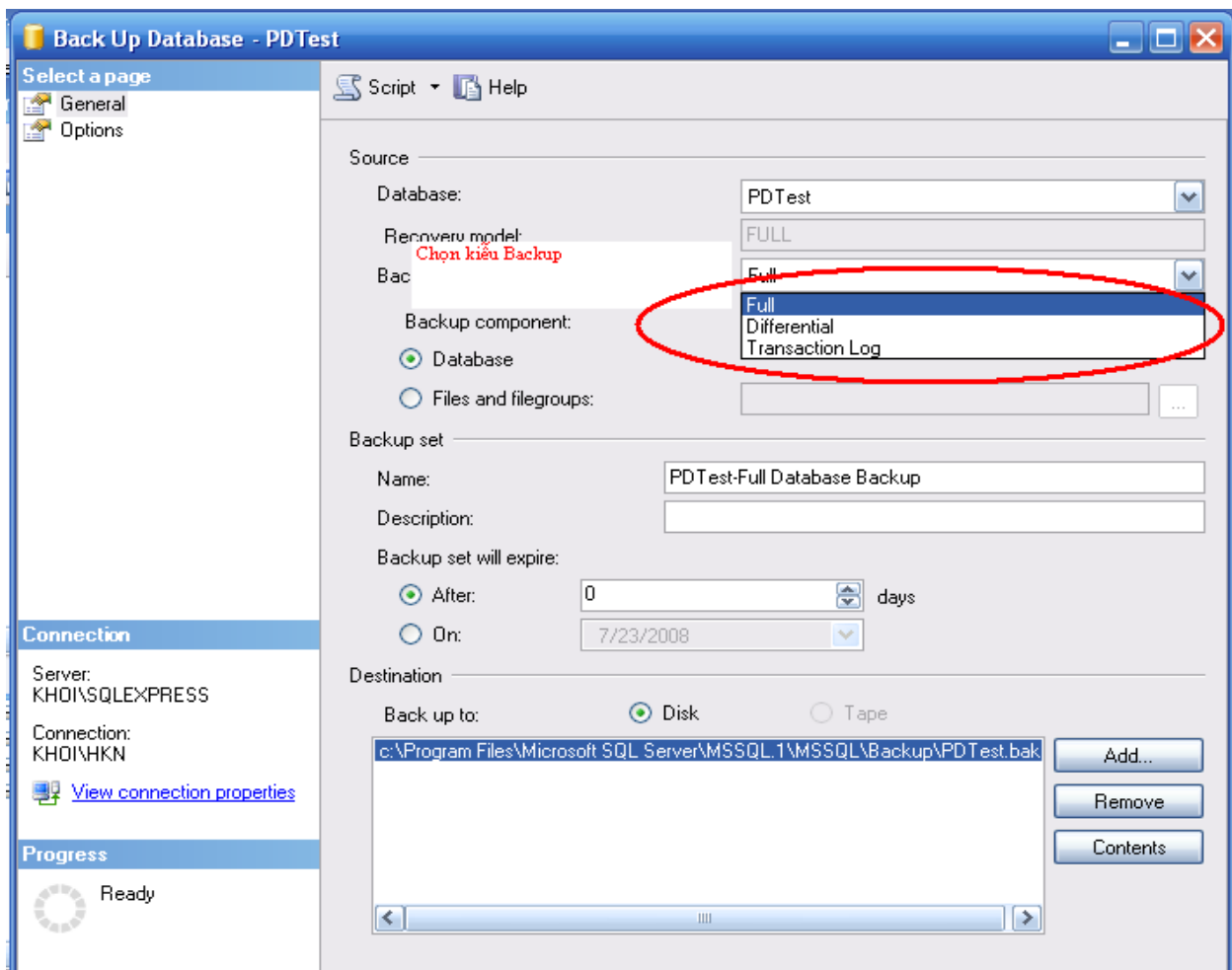
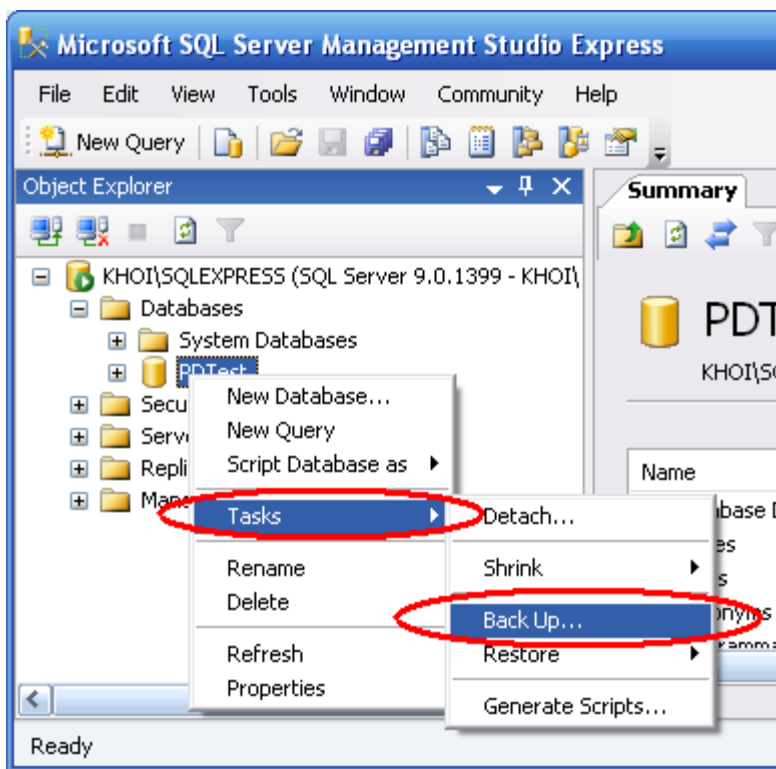
Tr l i v i ví d v công ty XYZ. Công ty này th c hi n full backup vào t i th 6 và differential backup vào t i th 2 t i th 5. Công ty th c hi n thêm quá trình transaction log backup m i gi m t l n. Gi s s c CSDL x y ra vào 9h:05 sáng th 4. Quá trình khôi ph c l i CSDL nhu sau: Dùng full backup và differential backup c a t i th 6 và t i th 3 ph c h i l i tr ng thái CSDL vào t i th 3. Tuy nhiên quá trình này v n còn m t d li u trong 2 gi (7 – 9h) sáng th 4. Ti p theo s d ng 2 b n sao l u transaction backup lúc 8h và 9h sáng khôi ph c CSDL v tr ng thái lúc 9h sáng th 4.

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

6.3.1 Sao l (Backup)

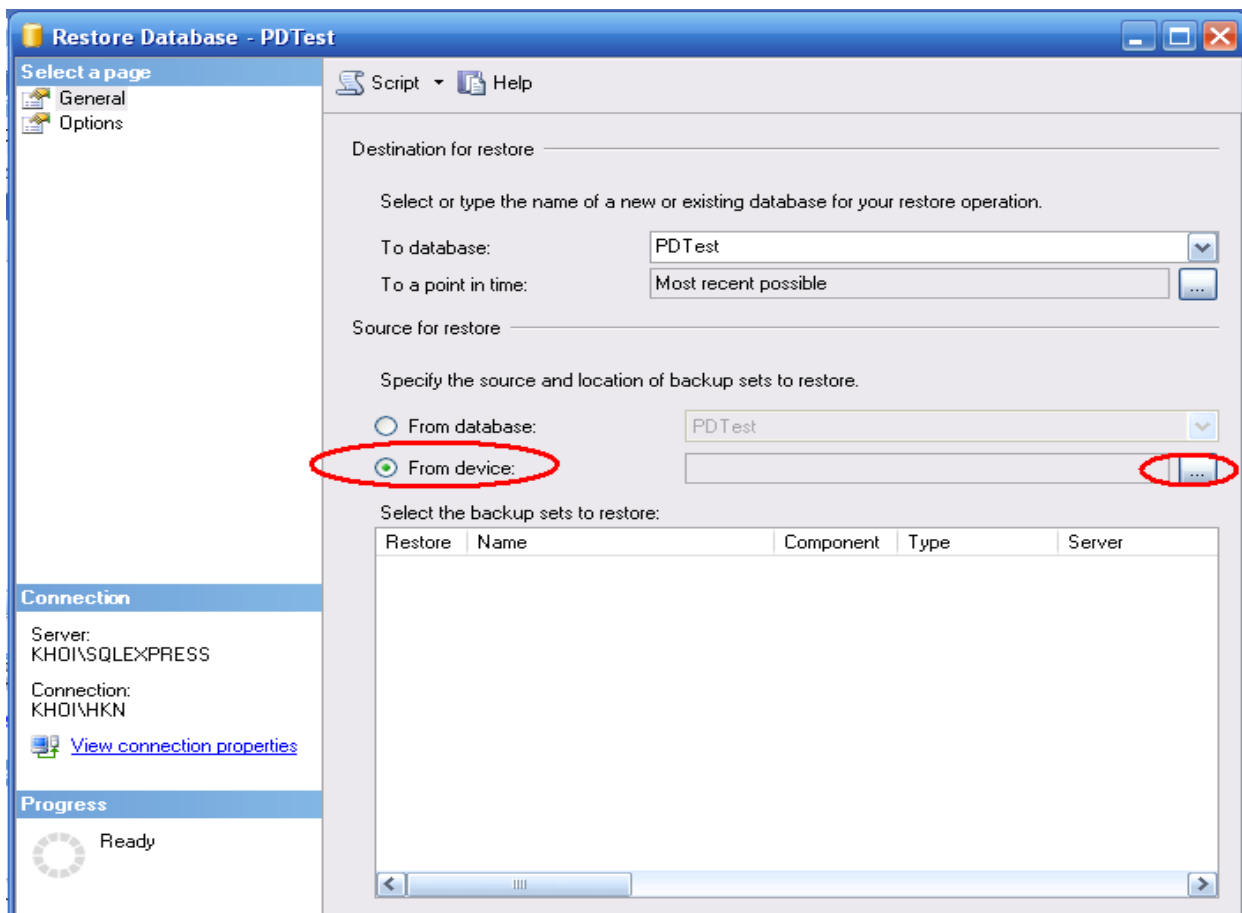
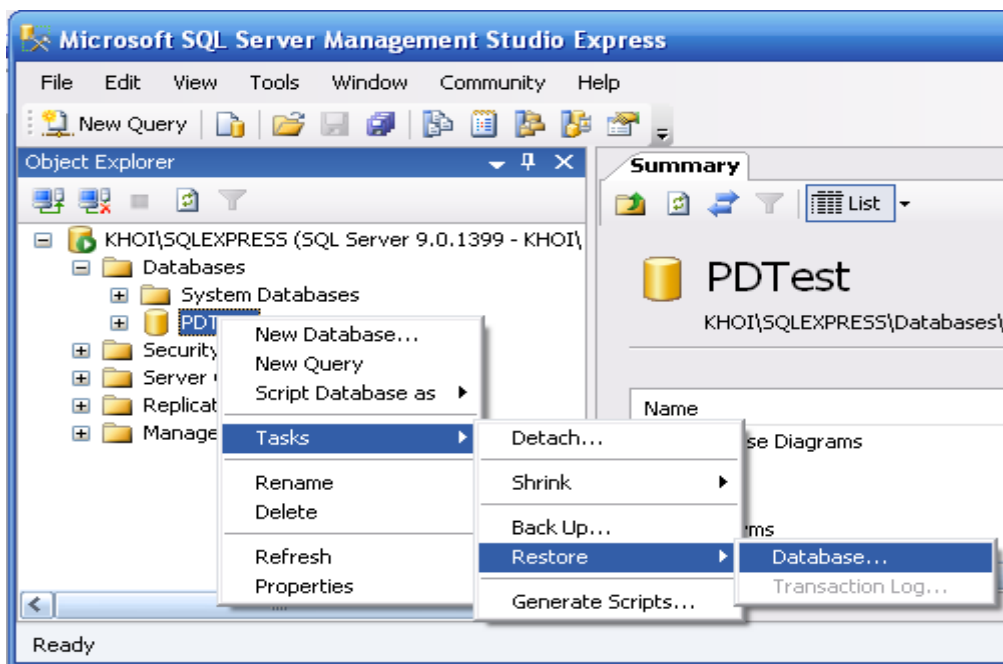


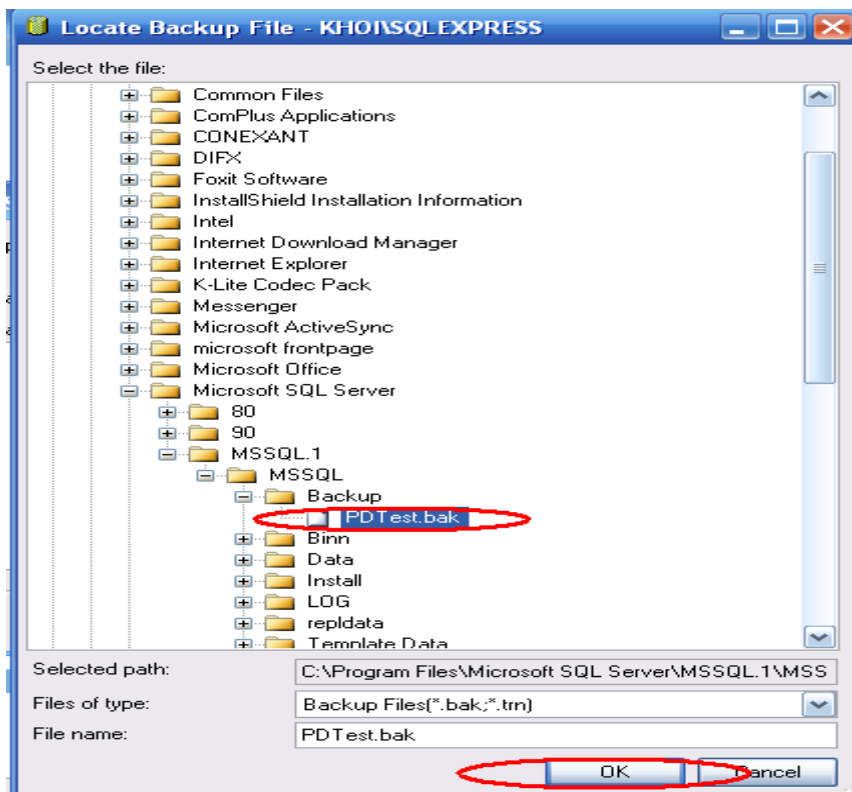
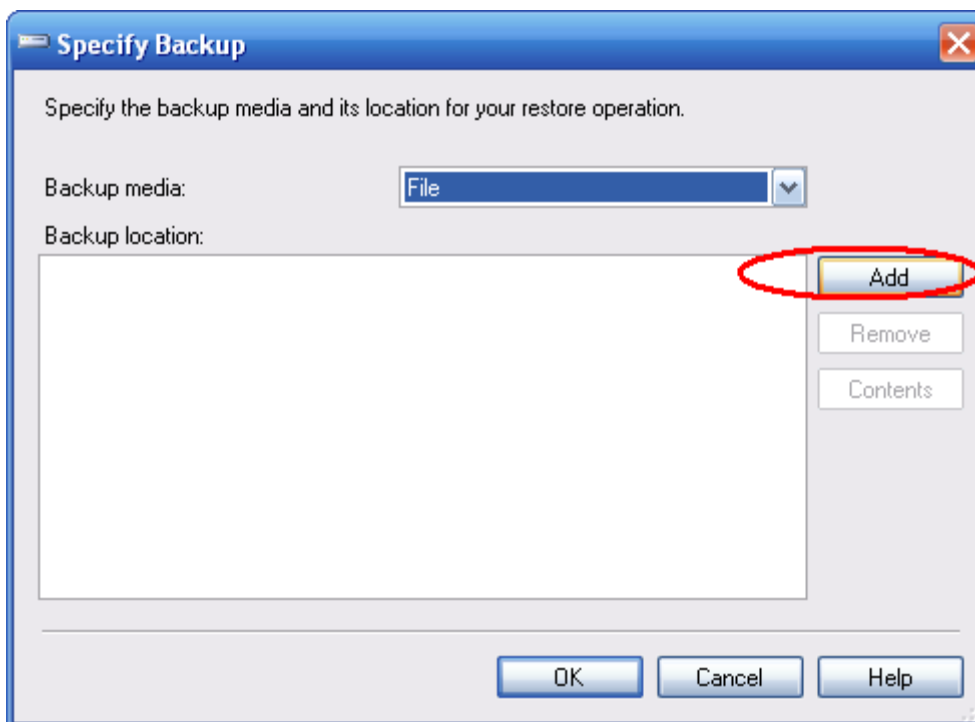
Click OK



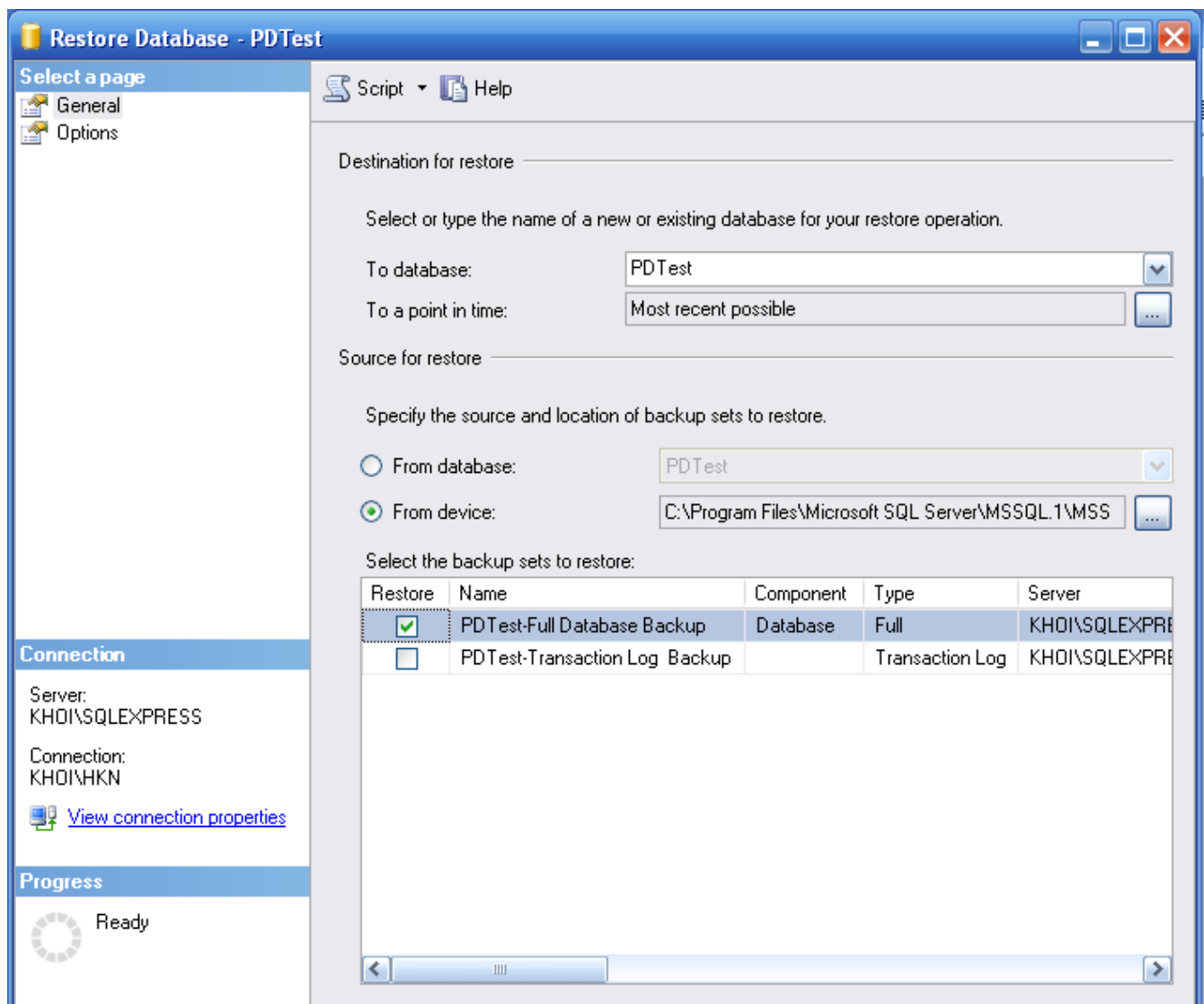
Click OK. Quá trình sao l u hoàn t t

6.3.2 Ph c h i (Restore)





Click OK hai l n



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 thể hiện các nhiệm vụ quan trọng khác nhau. Trong chương này sẽ trình bày một số các hàm thông dụng 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 hàm 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 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 tài chính nhất

Cú trúc hàm ROUND như sau:

ROUND (s_làm_tròn , _dài_làm_tròn)

Khi sử dụng hàm ROUND cần lưu ý:

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

Nếu *_dài_làm_tròn* là số âm và lớn hơn số chẵn phía trái của thập phân thì hàm ROUND trả về 0.

Ví dụ 1:

select ROUND(123.9994, 3), ROUND(123.9995, 3)

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

Ví d 2:

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

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

Ví d 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 vi c v i ki u d li u chu i

Các hàm quan tr ng bao g m LEFT, RIGHT, LEN, REPLACE, STUFF, SUBSTRING, LOWER, UPPER, LTRIM, and RTRIM.

7.2.1 Hàm LEFT

Hàm LEFT tr v m t chu i ký t có chi u dài c ch nh tính t bên trái c a chu i.

Ví d :

select left('Nha Trang', 5)

(No column name)
Nha T

7.2.2 Hàm RIGHT

Hàm RIGHT t ng t hàm LEFT nh ng tính t bên ph i c a chu i

Ví d :

select right('Nha Trang', 5)

(No column name)
Trang

7.2.3 Hàm SUBSTRING

Hàm STRING trích xu t m t chu i con t m t chu i cho tr c.

C u trúc hàm SUBSTRING nh sau:

SUBSTRING (chu i_ban_ u, v _trí_b t_ u, chi u_dài_chu i_con)

Ví d 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 chỉ số dài của chuỗi

Ví dụ :

Select len('Nha Trang')

(No column name)
9

7.2.5 Hàm REPLACE

Hàm REPLACE thay thế chuỗi bằng chuỗi khác

Ví dụ 1: Câu lệnh dưới đây thay thế “Nha” trong chuỗi Nha Trang bằng “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 ký tự xác định các ký tự trong chuỗi bằng chuỗi khác bắt đầu từ vị trí cho trước.

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 ký tự thường. Hàm UPPER chuyển các 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 trống bên trái của chuỗi, hàm RTRIM cắt các khoảng trống bên phải của 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 giá trị thời gian datetime.

Hàm MONTH trả về tháng của một giá trị thời gian 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.customerid = 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 ngày 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.

Cú 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 riêng biệt cho các yêu cầu đó. Bảng dưới đây mô tả các yêu cầu 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

Tổng hàm DATEPART như hàm DATENAME trả về một chuỗi ký tự

Ví dụ :

select datename(yyyy, orderdate) as YEAROFORDERDATE,
datename(qq, orderdate) as QUARTEROFORDERDATE,
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 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 thú vị khi dùng CONVERTER là khi chuyển đổi, hàm này còn cho phép người dùng sử dụng số để định nghĩa giá trị kết quả theo ý muốn.

Cú trúc hàm CAST và CONVERTER như sau:

CAST (biểu thức/giá trị AS kiểu dữ liệu [đài kiểu dữ liệu])

CONVERT (kiểu dữ liệu [đài kiểu dữ liệu], biểu thức/giá trị [, kiểu dữ liệu])

Nm 2 ch s	Nm 4 ch s	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 ho c 109	mon dd yyyy hh:mi:ss:mmmAM (PM)
10	110	mm-dd-yy
11	111	yy/mm/dd
12	112	yymmdd
	13 ho c 113	dd mon yyyy hh:mm:ss:mmm(24h)
14	114	hh:mi:ss:mmm(24h)

Ví d :

```
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 th s d ng k t h p v i nhau cho k t qua nh mong mu n.

Ví d :

```
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
```

8 K t n i vào SQL Server 2005 t các ngôn ng l p trình xây d ng các ng d ng liên quan n CSDL

M c tiêu cu i cùng c a vi c h c h qu n tr CSDL Microsoft SQL Server 2005 là ng i h c bi t dùng h qu n tr này trong vi c xây d ng các CSDL cho các ng d ng qu n lý trong th c t . ng th i s d ng các công c trong Microsoft SQL Server 2005 qu n tr CSDL c a mình.

8.1 C u hình Microsoft SQL Server 2005

Nh ã nói trong ch ng 1, Microsoft SQL Server 2005 s d ng hai ki u nh danh ng i dùng: Windows Authentication và SQL Server Authentication.

Windows Authentication: thích h p trong vi c xây d ng các ng d ng qu n lý trên máy n (ngh a là SQL Server và ng d ng qu n lý cùng trên m t máy). Nh ng ng d ng này th ng có CSDL khá nh và t c t ng tr ng c a CSDL không cao.

SQL Server Authentication: thích h p trong vi c xây d ng các ng d ng qu n lý có CSDL l n, nhi u ng i dùng cùng lúc. Trong các ng d ng này, CSDL c t trên Database Server, ng d ng trên các máy tr m s th c hi n các k t n i vào Database Server này và th c hi n các thao tác trên CSDL. Các k t n i này c g i là các k t n i t xa (remote connection).

Tuy nhiên, khi k t n i m t máy tính n m t th hi n (instance) c a Microsoft SQL Server 2005 t o m t k t n i t xa, b n có th nh n m t thông báo l i nh sau:

Microsoft SQL Native Client: An error has occurred while establishing a connection to the server. When connecting to SQL Server 2005, this failure may be caused by the fact that under the default settings SQL Server does not allow remote connections.

L i này x y ra vì Microsoft SQL Server ch a c c u hình ch p nh n các k t n i t xa.

M c nh, phiên b n SQL Server 2005 Express Edition và phiên b n SQL Server 2005 Developer Edition không cho phép các k t n i t xa. c u hình SQL Server 2005 ch p nh n các k t n i t xa c n th c hi n các b c sau:

Cho phép ti p nh n các k t n i t xa trên th hi n c a SQL Server mà các ng d ng máy tr m c n k t n i.

Kích ho t d ch v SQL Server Browser

C u hình t ng l a cho phép các d li u liên quan n SQL Server và d ch v SQL Server Browser c l u thông trên m ng.

8.1.1 Cho phép tiếp nhận các kết nối xa trên thính hiên của SQL Server

Click Start -> Programs -> Microsoft SQL Server 2005 -> Configuration Tools -> SQL Server Surface Area Configuration.

Trong trang SQL Server 2005 Surface Area Configuration, click Surface Area Configuration for Services and Connections.

Trong trang Surface Area Configuration for Services and Connections, nhấn nút Database Engine, click Remote Connections, click Local and remote connections, click chọn giao thức thích hợp (giao thức này có sẵn trên giao thức thiết lập sẵn trong môi trường làm việc), sau đó click Apply. Trong môi trường thiết lập, chúng ta thiết lập giao thức TCP/IP.

Lưu ý: Click OK khi nhận được thông báo:

Changes to Connection Settings will not take effect until you restart the Database Engine service.

Trong trang Surface Area Configuration for Services and Connections, nhấn nút Database Engine, click Service, click Stop, đợi khi dịch vụ MSSQLSERVER dừng lại, sau đó click Start khi dịch vụ MSSQLSERVER

.

Lưu ý: nếu chúng ta không cần sử dụng SQL Server 2005 thì ngay xuyên, ta có thể chỉ khi cần cài đặt dịch vụ MSSQLSERVER là manual (nghĩa là khi nào cần dùng thì sẽ kích hoạt) nhằm giảm bớt thời gian khởi động Windows và tiết kiệm tài nguyên hệ thống.

8.1.2 Kích hoạt dịch vụ SQL Server Browser

Nếu chúng ta thao tác trên SQL Server 2005 bằng vì các sử dụng tên thính hiên (instance name) và không chỉ định port cụ thể trong chuỗi kết nối (chúng ta sẽ nói về chuỗi kết nối cụ thể trong phần kết nối các ngôn ngữ lập trình với SQL Server 2005) thì chúng ta phải kích hoạt dịch vụ SQL Server Browser cho phép các kết nối xa. Ví dụ: SQL Server 2005 Express Edition có cài đặt với một thính hiên mặc định là tên_máy_tính\SQLEXPRESS. Chúng ta chỉ cần kích hoạt dịch vụ SQL Server Browser một lần bắt đầu chúng ta có bao nhiêu

th hi n c a SQL Server 2005 ang c s d ng (running). kích ho t d ch v SQL Server Browser, c n làm theo các b c sau:

Click Start -> Programs -> Microsoft SQL Server 2005 -> Configuration Tools, click SQL Server Surface Area Configuration.

Trong trang SQL Server 2005 Surface Area Configuration, click Surface Area Configuration for Services and Connections.

Trong trang Surface Area Configuration for Services and Connections, click SQL Server Browser, click Automatic for Startup type, click Apply.

L u ý: t ng t nh d ch v MSSQLSERVER, n u ch n startup type là Automatic thì d ch v SQL Server Browser s c kh i ng khi Windows kh i ng

Ki m tra tình tr ng d ch v trong Service status. Sau ó click Start n u d ch v này ang b Stop.

L u ý: Th c hi n các b c trên làm t ng các nguy c cho h th ng vì h th ng s hi n th thông tin c a các th hi n c a SQL Server ng ch y trên h th ng. Các nguy c này có th c gì m thì u b ng cách không kích ho t d ch v SQL Server Browser và k t n i vào th hi n c a SQL Server thông qua m t port. Chi ti t có th tham kh o Microsoft Book Online theo các ch sau:

SQL Server Browser Service

Connecting to the SQL Server Database Engine

Client Network Configuration

8.1.3 T o các ngo i l trên Windows Firewall

Các b c d i ây áp d ng cho Windows Firewall trong Windows XP Service Pack 2 (SP2) và trong Windows Server 2003.

N u t ng l a c s d ng trên máy tính cài t SQL Server 2005, các k t n i t bên ngoài s b ch n tr khi SQL Server 2005 và SQL Serve Browser có th liên l c qua t ng l a. Chúng ta ph i t o ra các ngo i l cho m i th hi n c a SQL Server 2005 (mu n th hi n nào ti p nh n k t n i t xa thì chúng ta ph i t o ngo i l cho th hi n ó) và m t ngo i l cho d ch v SQL Server Browser.

Vì c t o ngo i l theo các b c sau:

Click Start -> Programs -> Microsoft SQL Server 2005 -> Configuration Tools -> SQL Server Configuration Manager.

Trong trang SQL Server Configuration Manager, click SQL Server Services, right-click tên th hi n, và click Properties.

Trong trang SQL Server Browser Properties, click tab Advanced, xác nh instanceID trong danh sách thu c tính, và click OK.

L u ý: Chúng ta có th có nhi u th hi n SQL Server, do ó chúng ta ph i xác nh úng instanceID nào chúng tao mu n t o ngo i l .

m Windows Firewall, click Start, click Run, ánh firewall.cpl, và click OK.

T o ngo i l cho SQL Server 2005 trong Windows Firewall

Trong Windows Firewall, click tab Exceptions, sau ó click Add Program.

Trong c a s Add a Program window, click Browse.

C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn\sqlservr.exe, click Open, và click OK.

L u ý: ng d n có th khác nhau tùy thu c vào th m c cài t c a SQL Server 2005. MSSQL.1 là n i l u tr cho instanceID chúng ta thu c trong b c trên.

L p l i các b c trên cho m i th hi n c a SQL Server 2005 c n thi t l p ngo i l .

T o ngo i l cho d ch v SQL Server Browser trong Windows Firewall

Trong Windows Firewall, click tab Exceptions, và click Add Program.

Trong c a s Add a Program, click Browse.

C:\Program Files\Microsoft SQL Server\90\Shared\sqlbrowser.exe, click Open, và click OK.

L u ý: ng d n có th thay i tùy thu c vào th m c cài t c a SQL Server 2005.

8.2 K t n i vào SQL Server trong các ngôn ng l p trình

8.2.1 C# và VB.NET

Tùy thu c vào vi c s d ng .NET Data Provider (t p các i t ng ph c v vi c trao i d li u) nào trong .NET Framework và vi c s d ng hình th c Authentication nào truy xu t d li u trong SQL Server 2005, chúng ta s có các cách khác nhau k t n i n SQL Server 2005. Trong ph n trình bày này s s d ng System.Data.SqlClient là m t Data Provider ph bi n k t n i v i SQL Server 2005.

Trong C#:

public void ConnectToSql ()


```

{
    System.Data.SqlClient.SqlConnection conn =
        new System.Data.SqlClient.SqlConnection ();

    // TODO Xây dựng chuỗi kết nối

    conn.ConnectionString =
        "integrated security=true;data source=tên_SQLSERVER;" +
        "persist security info=False;initial catalog=tên_CSDL";
    try
    {
        conn.Open();
        // Xây dựng code thực hiện tác vụ i CSDL này
    }
    catch (Exception ex)
    {
        MessageBox.Show("Failed to connect to data source");
    }
    finally
    {
        conn.Close();
    }
}

```

Trong VB.NET

```

Public Sub ConnectToSql()
    Dim conn As New SqlConnection
    ' TODO Xây dựng chuỗi kết nối

    conn.ConnectionString = & _
        "integrated security=true;data source=tên_SQL Server;" & _
        "persist security info=False;initial catalog=tên_CSDL"
    Try
        conn.Open()
        ' Xây dựng code thực hiện tác vụ i CSDL này
    Catch ex As Exception

```

```

        MessageBox.Show("Failed to connect to data source")
    Finally
        conn.Close()
    End Try
End Sub

```

Trong hai ví dụ trên chúng ta xây dựng hàm kết nối vào SQL Server 2005 mà thành phần quản trị nhúng là chuỗi kết nối vào CSDL:

Trong C#:

```

"integrated security=true;data source=tên_SQLSERVER;" +
    "persist security info=False;initial catalog=tên_CSDL";

```

Trong VB.NET:

```

"integrated security=true;data source=tên_SQL Server;" & _
    "persist security info=False;initial catalog=tên_CSDL"

```

Trong đó:

Integrated security = true: sử dụng Windows Authentication

data source: chuỗi tên hiển thị của SQL Server 2005 mà chúng ta muốn kết nối.

persist security info: Thiết lập mặc định cho thuộc tính persist security info là false. Thiết lập sang giá trị true sẽ cho phép các dữ liệu nhạy cảm bao gồm UserID và password có thể truy xuất khi kết nối.

initial catalog: Tên CSDL mà chúng ta muốn thực tác.

Ví dụ chuỗi kết nối sử dụng Windows Authentication:

```

"integrated security=true;data source=.\SQLEXPRESS" +
    "persist security info=False;initial catalog=myDB";

```

Ví dụ chuỗi kết nối sử dụng SQL Server Authentication:

```

"persist security info=False;User ID = *****; password = ***** " +
    "initial catalog=myDB; data source=.\SQLEXPRESS ";

```

8.2.2 VB 6

Ví dụ để minh họa việc xây dựng ứng dụng CSDL bằng VB6. Giả sử chúng ta có CSDL tên là *Test*, SQL Server là *.\SQLEXPRESS*, User là *sa*, Password là *1234*.

```

Private Sub Command1_Click()
    Dim connectionString As String
    Dim commandString As String
    Dim sqlConnection As ADODB.Connection

```

Dim rs As Recordset

connectionString="PROVIDER=SQLOLEDB; DATA SOURCE=.\SQLEXPRESS;"
connectionString = "UID=sa; PWD=1234;DATABASE=Test"
commandString = "select count() as count from Users where UserName = "*
commandString = commandString & Text1.Text & "' and Password = "
commandString = commandString & Text2.Text & "'"

Set sqlConnection = New ADODB.Connection
sqlConnection.Open (connectionString)
Set rs = New Recordset
rs.Open commandString, sqlConnection

If (rs("count") = 1) Then
 MsgBox "Login successfully"
End If
End Sub

Tài liệu tham khảo

1. Giáo trình 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.