

Chương 1

Giới Thiệu SQL Server 2008

Kết thúc chương này các bạn có thể :

- Trình bày được các khái niệm cơ bản SQL Server 2008
- Mô tả được các thành phần và kiến trúc SQL Server 2008.
- Thực hành được cách tạo cơ sở dữ liệu, tạo bảng, tạo kết nối giữa các bảng...
- Thực hành được sao lưu và phục hồi cơ sở dữ liệu.

1.1 Tổng quan về SQL Server 2008

Trong một thế giới dữ liệu ngày nay, dữ liệu và các hệ thống quản lý dữ liệu đó cần phải luôn luôn được bảo đảm và ở trạng thái có sẵn. SQL Server 2008 cho phép các nhà phát triển giảm được sự phức tạp của cơ sở hạ tầng trong khi đó vẫn bảo đảm cung cấp một nền tảng dữ liệu doanh nghiệp có khả năng bảo mật, khả năng mở rộng và quản lý tốt hơn, cùng với thời gian chết của ứng dụng giảm.

Những điểm mới của SQL server 2008:

- **Nền tảng cho các nhiệm vụ then chốt** - SQL Server 2008 cho phép các tổ chức có thể chạy hầu hết các ứng dụng phức tạp của họ trên một nền tảng an toàn, tin cậy và có khả năng mở rộng. Bên cạnh đó còn giảm được sự phức tạp trong việc quản lý cơ sở hạ tầng dữ liệu. SQL Server 2008 cung cấp một nền tảng tin cậy và an toàn bằng cách bảo đảm những thông tin có giá trị trong các ứng dụng đang tồn tại và nâng cao khả năng sẵn có của dữ liệu. SQL Server 2008 giới thiệu một cơ chế quản lý cách tân dựa trên chính sách, cơ chế này cho phép các chính sách có thể được định nghĩa quản trị tự động cho các thực thể máy chủ trên một hoặc nhiều máy chủ.Thêm vào đó, SQL Server 2008 cho phép thi hành truy vấn dự báo với một nền tảng tối ưu.
- **Sự phát triển động** - SQL Server 2008 cùng với .NET Framework đã giảm được sự phức tạp trong việc phát triển các ứng dụng mới. ADO.NET Entity Framework cho

phép các chuyên gia phát triển phần mềm có thể nâng cao năng suất bằng làm việc với các thực thể dữ liệu logic đáp ứng được các yêu cầu của doanh nghiệp thay vì lập trình trực tiếp với các bảng và cột. Các mở rộng của ngôn ngữ truy vấn tích hợp (LINQ) mới trong .NET Framework đã cách mạng hóa cách các chuyên gia phát triển truy vấn dữ liệu bằng việc mở rộng Visual C#® và Visual Basic® .NET để hỗ trợ cú pháp truy vấn giống SQL vốn đã có. Hỗ trợ cho các hệ thống kết nối cho phép chuyên gia phát triển xây dựng các ứng dụng cho phép người dùng mang dữ liệu cùng với ứng dụng này vào các thiết bị và sau đó đồng bộ dữ liệu của chúng với máy chủ trung tâm.

- **Dữ liệu quan hệ mở rộng** - SQL Server 2008 cho phép các chuyên gia phát triển khai thác triệt để và quản lý bất kỳ kiểu dữ liệu nào từ các kiểu dữ liệu truyền thống đến dữ liệu không gian địa lý mới.
- **Thông tin trong toàn bộ doanh nghiệp** - SQL Server 2008 cung cấp một cơ sở hạ tầng có thể mở rộng, cho phép quản lý các báo cáo, phân tích với bất kỳ kích thước và sự phức tạp nào, bên cạnh đó nó cho phép người dùng dễ dàng hơn trong việc truy cập thông tin thông qua sự tích hợp sâu hơn với Microsoft Office. Điều này cho phép CNTT đưa được thông tin của doanh nghiệp rộng khắp trong tổ chức. SQL Server 2008 tạo những bước đi tuyệt vời trong việc lưu trữ dữ liệu, cho phép người dùng hợp nhất các trung tâm dữ liệu vào một nơi lưu trữ dữ liệu tập trung của toàn doanh nghiệp.



Hình 1.1. *Toàn cảnh nền tảng dữ liệu của Microsoft*

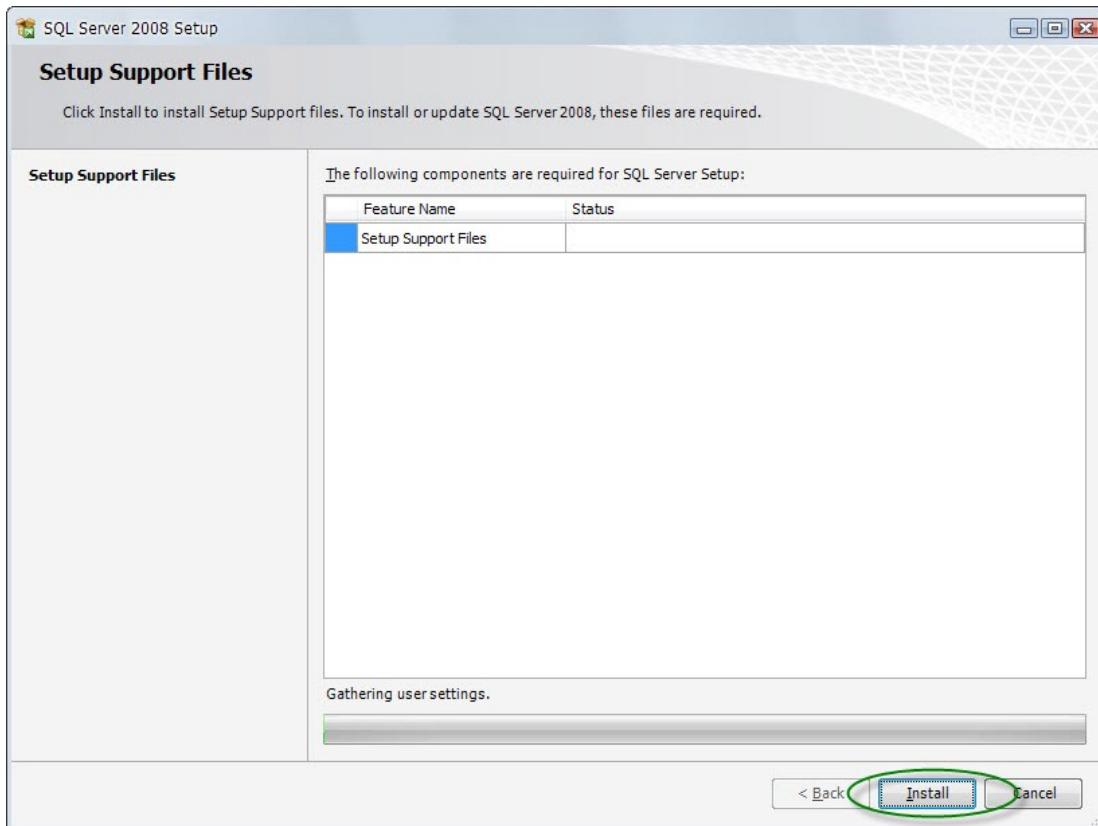
Các bước cài đặt SQL Server 2008

Bước 1: Chạy file setup.exe để cài đặt, chọn Installation -> New SQL Server stand-alone ...



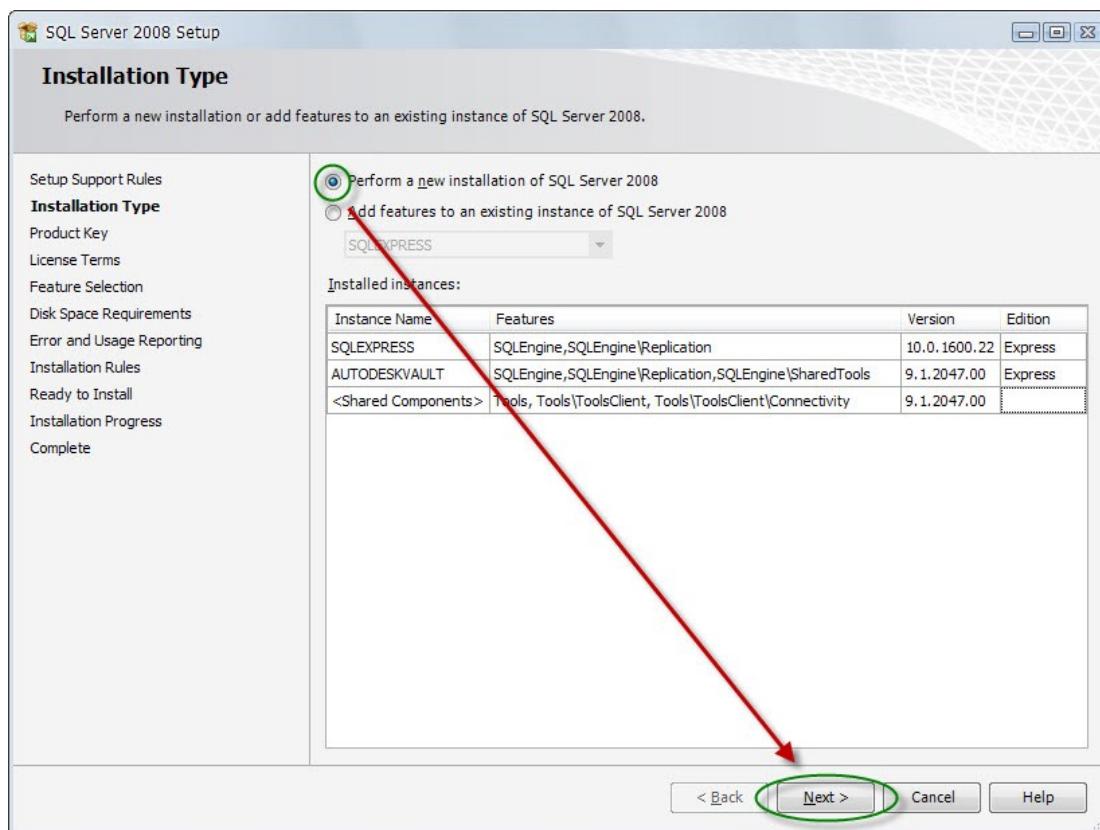
Hình 1.2. *Giao diện SQL Server Installation Center*

Bước 2: Chọn Ok -> Next .



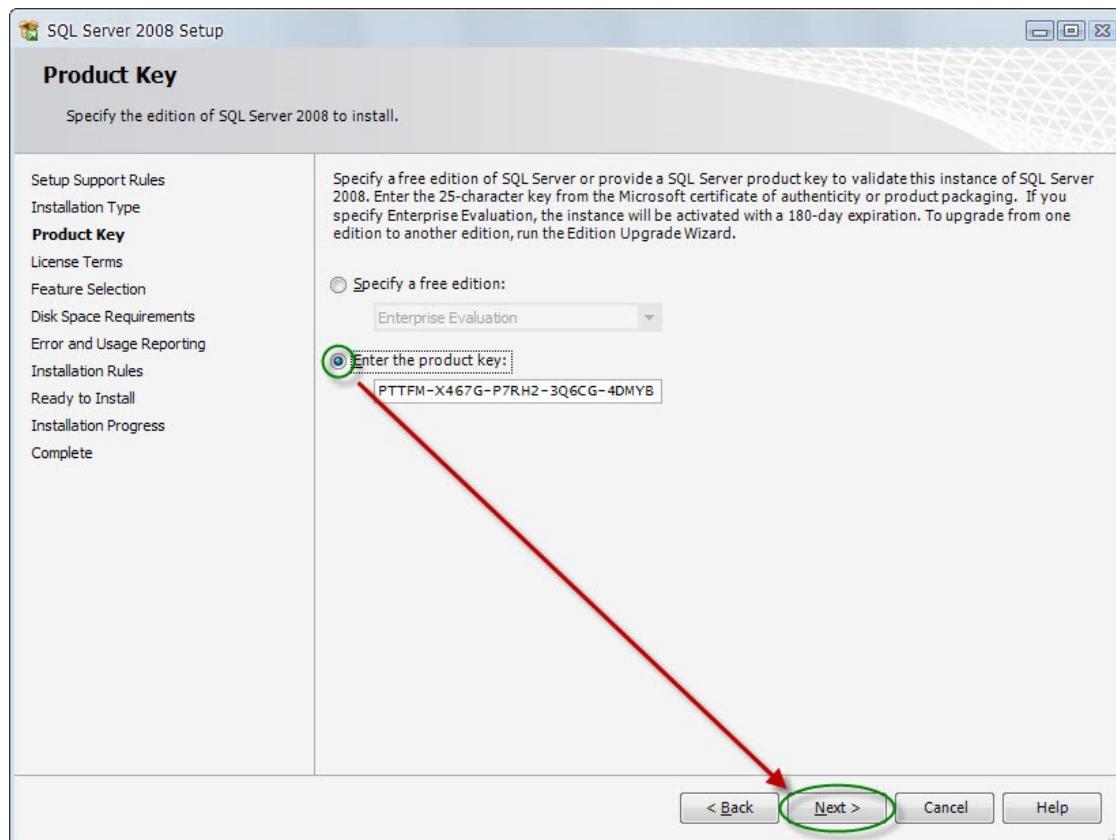
Hình 1.3. *Giao diện Setup Support Files*

Bước 3: chọn kiểu cài đặt mới



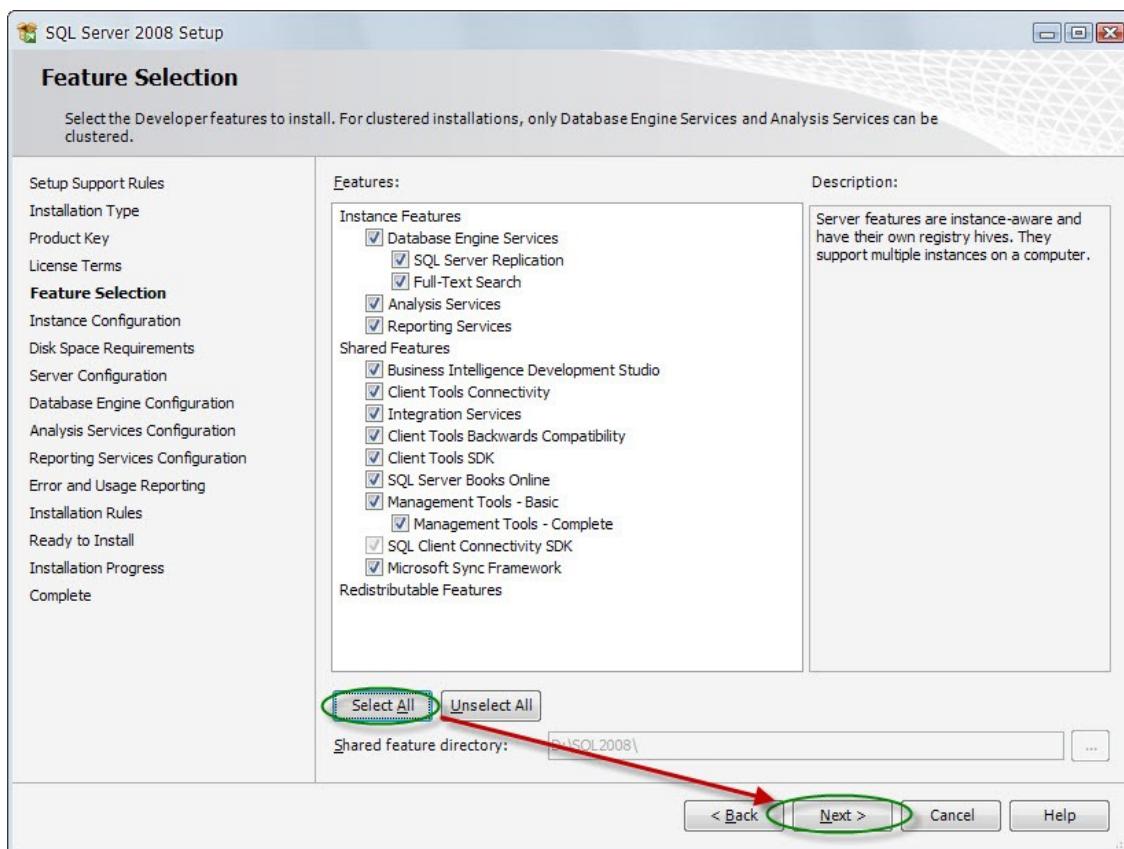
Hình 1.4. Giao diện Installation Type

Bước 4: Nhập product key



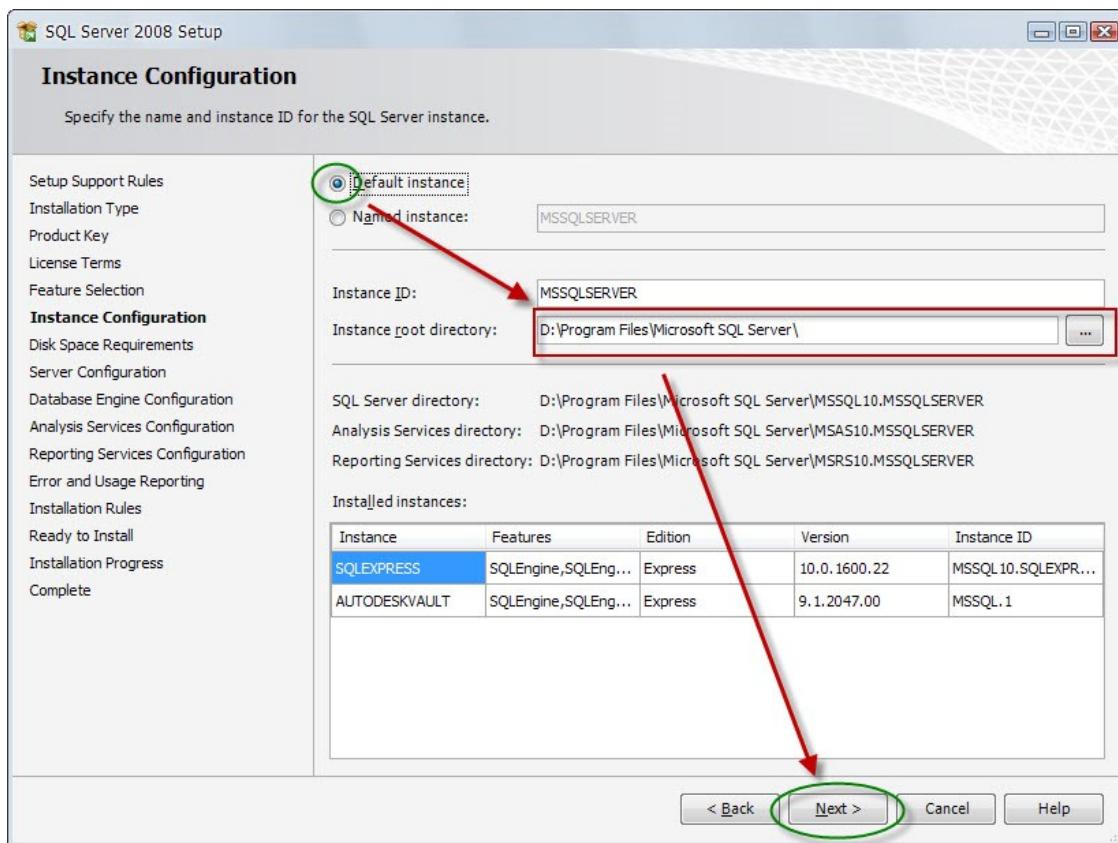
Hình 1.5. Giao diện Product Key

Bước 5: Sau khi đồng ý License Terms, chọn các thành phần cài đặt



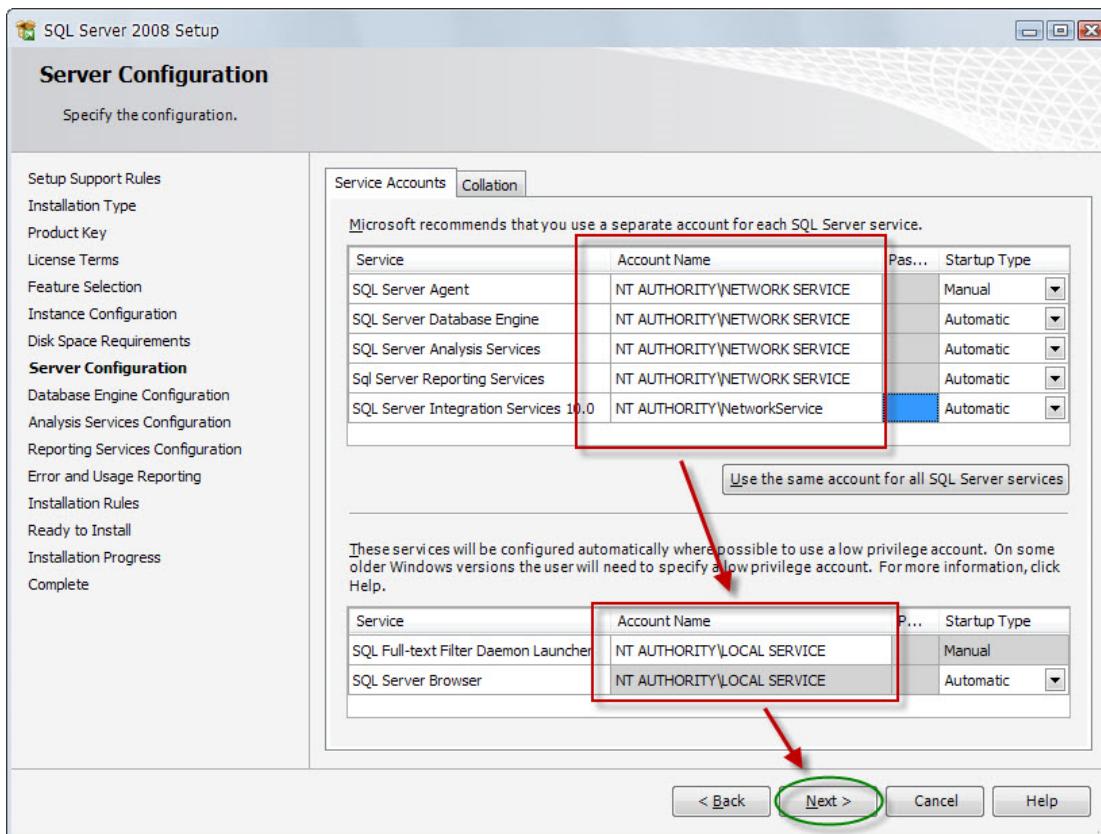
Hình 1.6. Giao diện Feature Selection

Bước 6: Thiết lập cài đặt chọn Default instance



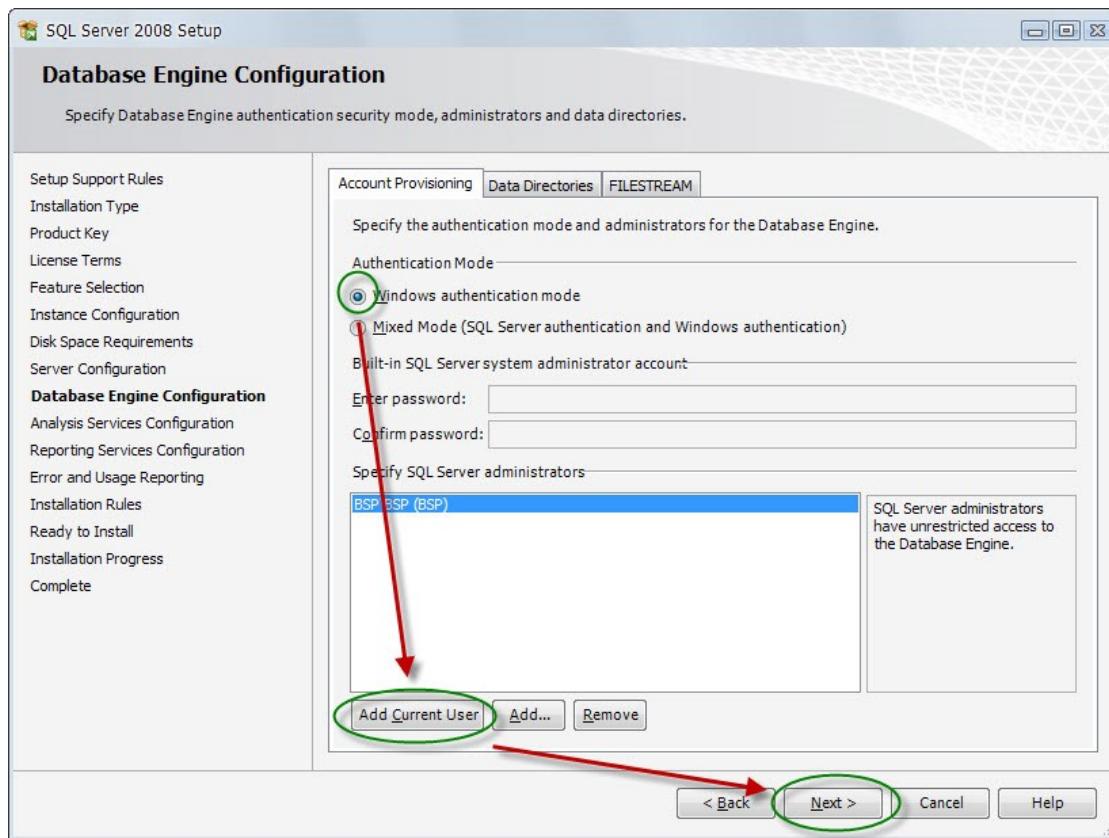
Hình 1.7. Giao diện Instance Configuration

Bước 7: Cấu hình server



Hình 1.8. Giao diện Server Configuration

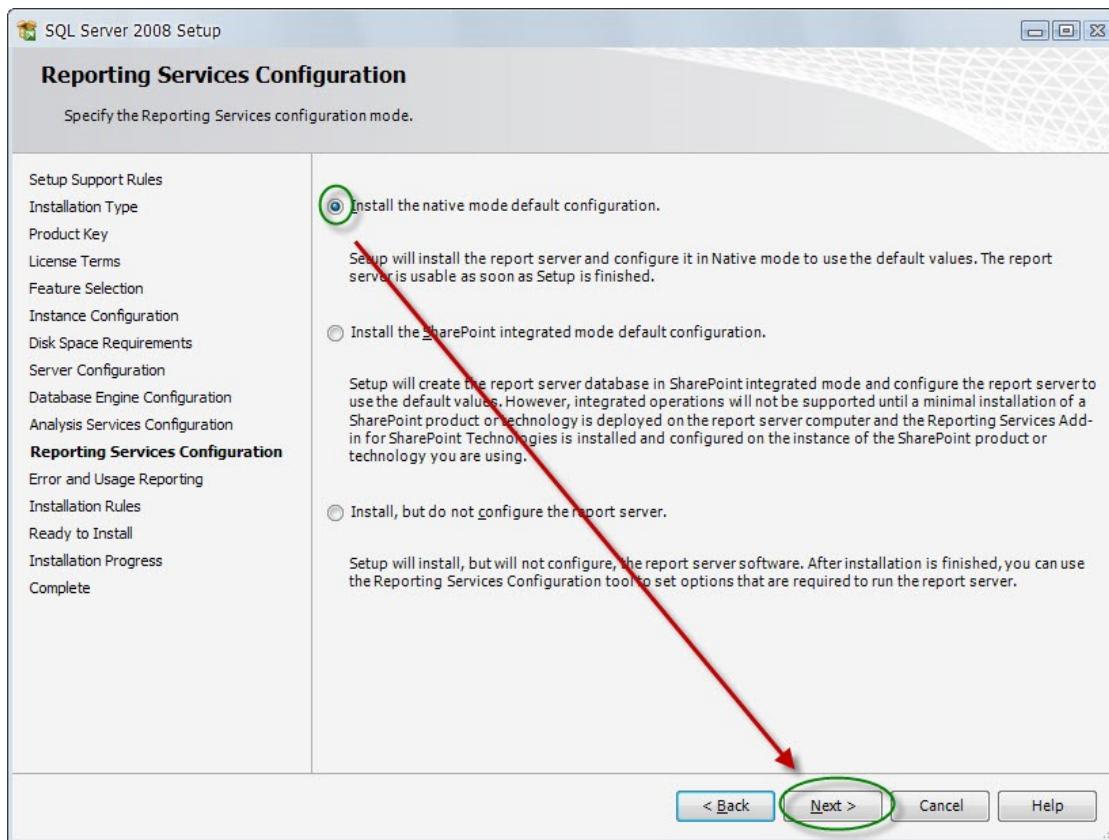
Bước 8: Cấu hình dữ liệu như sau chọn Window Authentication và Add current User



Hình 1.9. Giao diện Database Engine Configuration

Bước 9: Cấu hình analysis services Add Current User

Bước 10: Cấu hình report chọn option như hình nhấn Next, Next ... Cho đến khi hoàn tất



Hình 1.10. Giao diện Reporting Services Configuration

1.2 SQL Server Management Studio

Mở SQL Server Management Studio ta làm như sau: Vào start -> chọn program -> chọn Microsoft SQL Server 2008 -> chọn SQL Server Management Studio

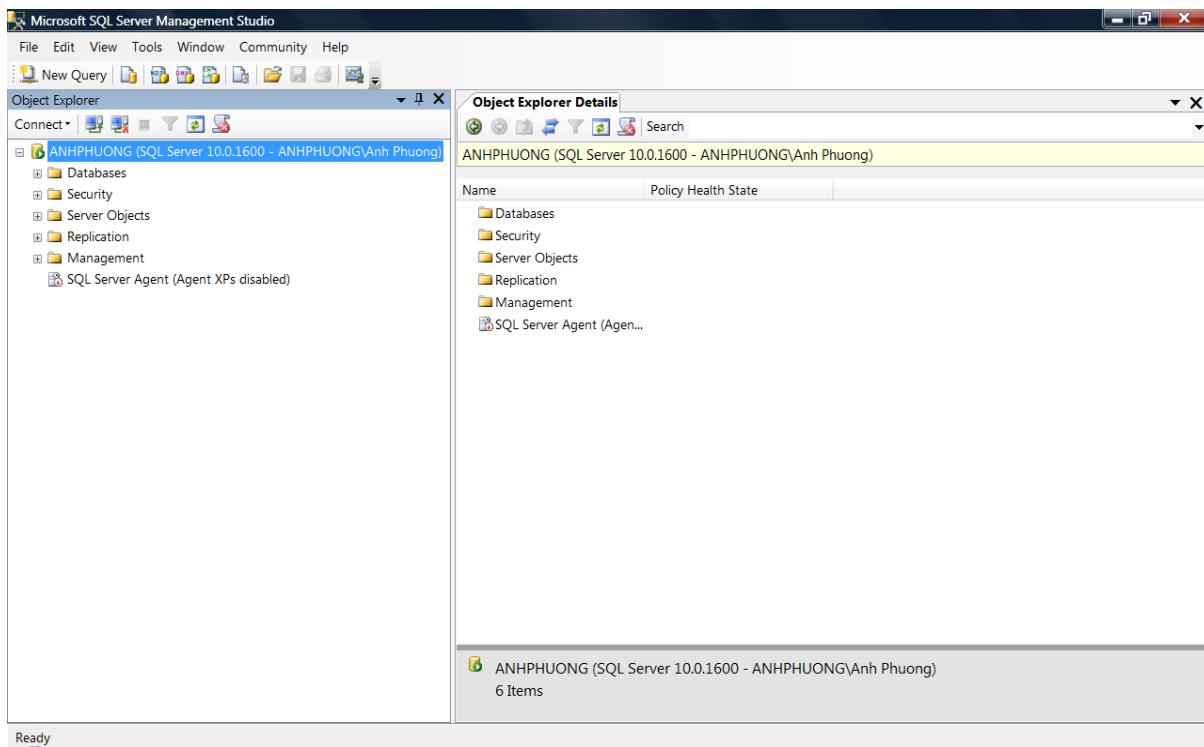


Hình 1.11. Kết nối vào SQL Server

Chú ý những thành phần trên hộp thoại sau:

- **Server Type:** Như ví dụ của cuốn sách này, cho phép server type là Database Engine. Các tùy chọn khác là kiểu dữ liệu khác nhau của servers nó sẽ hiển thị kết nối.
- **Server Name:** Hộp combo thứ 2 chứa 1 danh sách của SQL Server cài đặt mà chọn. Trong hộp thoại hình 12, bạn sẽ thấy tên của máy tính được cài đặt trên local. Nếu bạn mở hộp Server name bạn có thể tìm kiếm nhiều server local hoặc network connection bằng cách chọn <Browse for more...>.
- **Authentication:** Combobox cuối cùng xác định các loại hình kết nối bạn muốn sử dụng. Trong giáo trình này chúng ta kết nối đến SQL Server sử dụng Windows Authentication. Nếu bạn cài đặt SQL Server với chế độ hỗn hợp(mix mode), thì bạn có thể thay đổi chọn lựa SQL Server authentication, thì nó sẽ mở hai hộp thoại và cho phép nhập username và password.

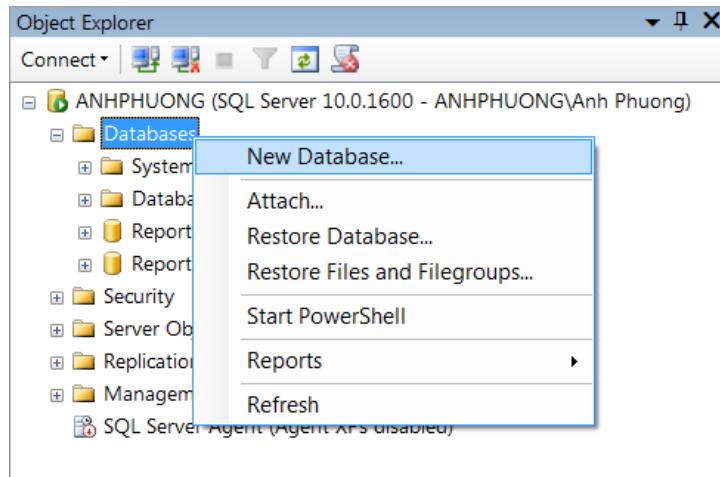
Sau khi nhấn nút Connect sẽ xuất hiện màn hình sau:



Hình 1.12. *SQL Server Management Studio*

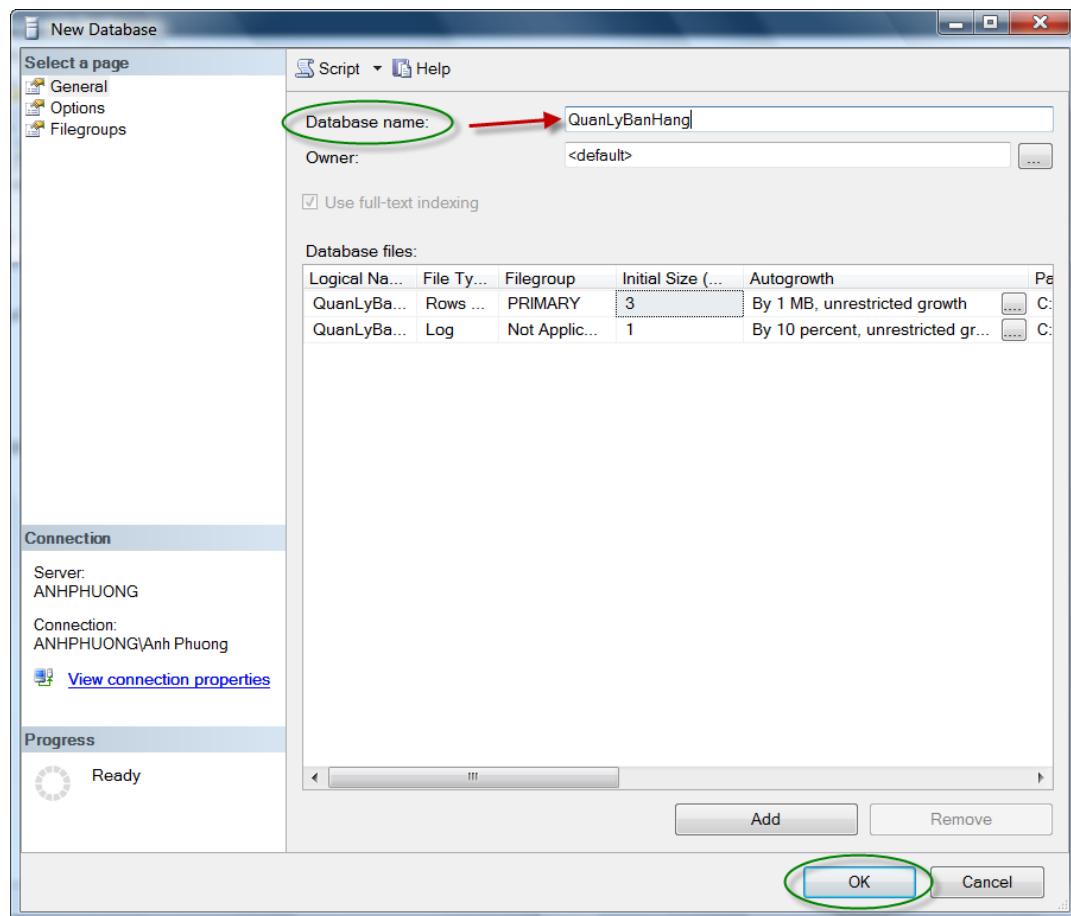
Tạo cơ sở dữ liệu (database)

Chọn database -> Click phải -> Chọn New Database...



Hình 1.13. *Hộp thoại Object Explorer*

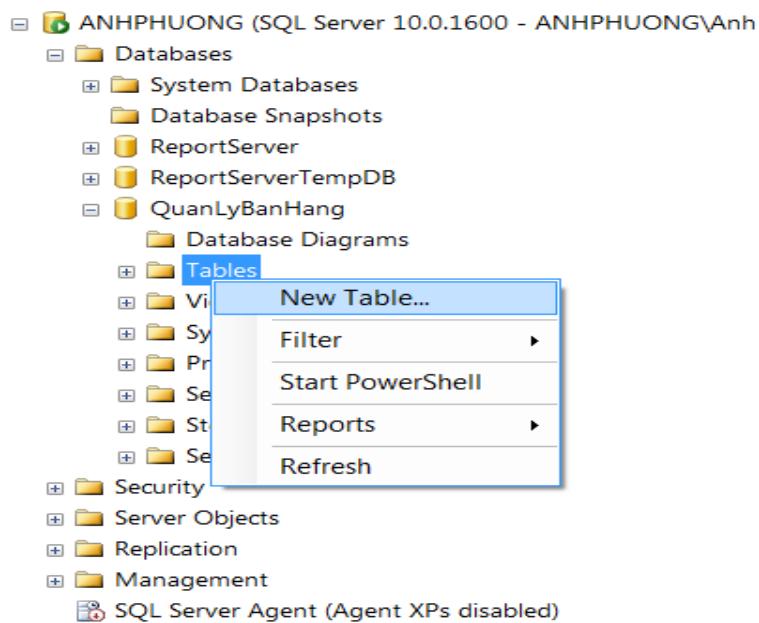
Trong hộp thoại New Database đặt tên cho database name -> Chọn OK



Hình 1.14. Giao diện New Database

✍ Tạo bảng (table)

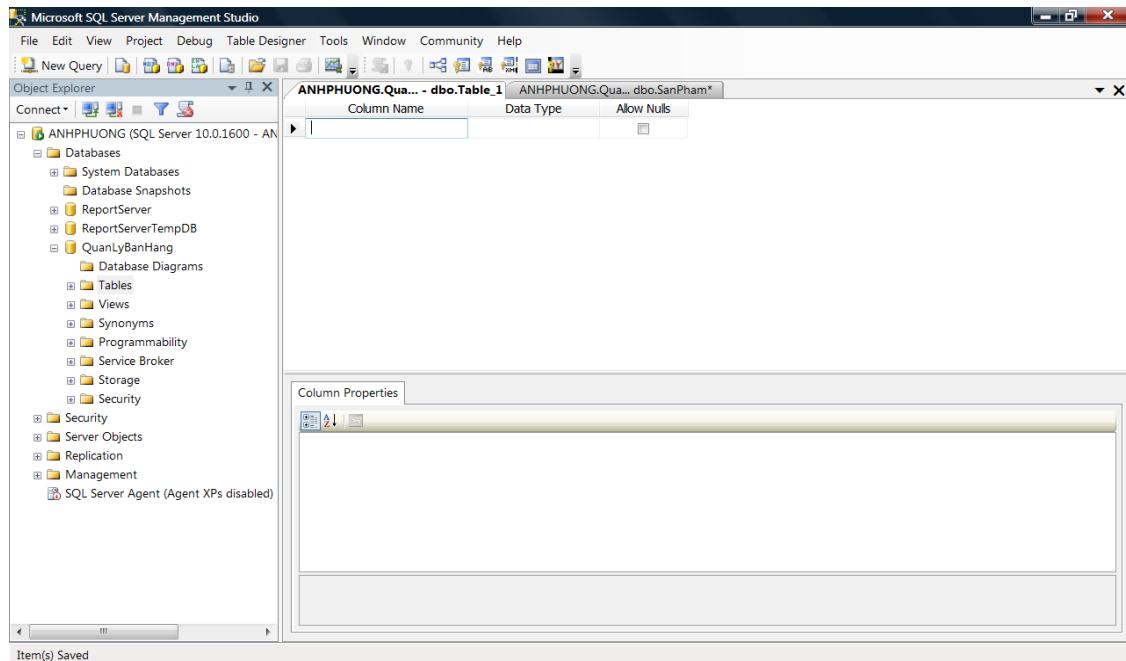
Vào database quản lý bán hàng chọn table. Sau đó click phải lên table -> Chọn New Table



Hình 1.15.

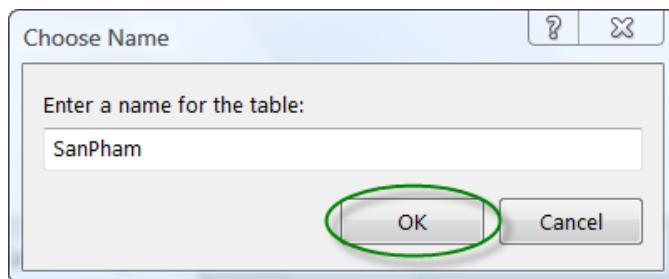
Khi chọn New Table sẽ xuất hiện bên phải màn hình bên dưới

Sau đó ta nhập Column Name, Data Type... Nhấn Enter để nhập cột kế tiếp.



Hình 1.16.

Lưu table trên thanh Standard toolbar -> chọn Save



Hình 1.17.

- ✍ Tạo quan hệ kết nối giữa các bảng (relationship)

1. **Tạo khóa chính (Primary key)** cho table trong SQL Server Management Studio, tạo cột và kiểu dữ liệu. Sau đó trên thanh toolbar, chọn nút Set Primary Key . Bạn cũng có thể click phải lên column chọn Set Primary Key.

	Column Name	Data Type	Allow Nulls
	MaSP	int	<input type="checkbox"/>
	TenSP	nvarchar(50)	<input checked="" type="checkbox"/>
	DonGia	float	<input checked="" type="checkbox"/>
	SoLuong	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Hình 1.18.

2. **Tạo khóa ngoại (Foreign key)** trong cửa sổ thiết kế table. Foreign được dùng để liên kết các table lại với nhau.

Cần lưu ý khi tạo foreign key là tên cột, kiểu dữ liệu giống tên cột của khóa chính mà table nó đại diện.

Ví dụ sau MaDMSP làm foreign key

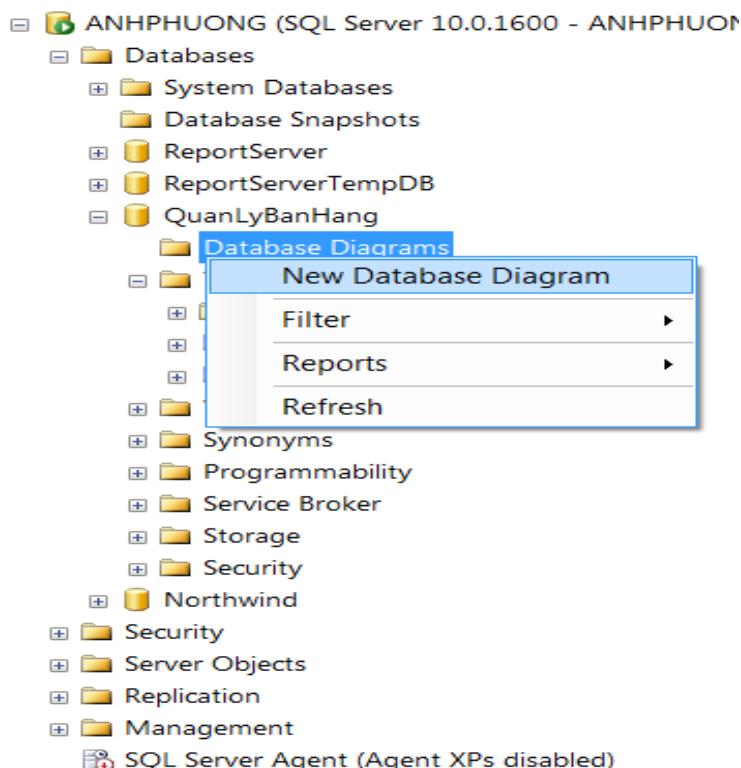
	Column Name	Data Type	Allow Nulls
MaSP	int	<input type="checkbox"/>	
TenSP	nvarchar(50)	<input checked="" type="checkbox"/>	
DonGia	float	<input checked="" type="checkbox"/>	
SoLuong	int	<input checked="" type="checkbox"/>	
MaDMSP	int	<input type="checkbox"/>	

Hình 1.19.

3. Tạo sơ đồ (Diagrams)

Diagrams là 1 cửa sổ hiển thị mối quan hệ giữa các table của 1 database. Tạo diagram ta thực hiện như sau:

- Trong cửa sổ Object Explorer chọn tên database cần tạo -> Click phải vào Database Diagrams -> Chọn New Database Diagram



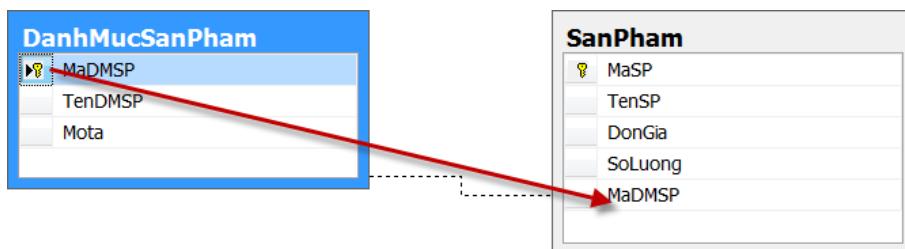
Hình 1.20.

- Sau khi chọn New Database Diagram sẽ xuất hiện hộp thoại để Add các table, sau khi add xong chọn Close.



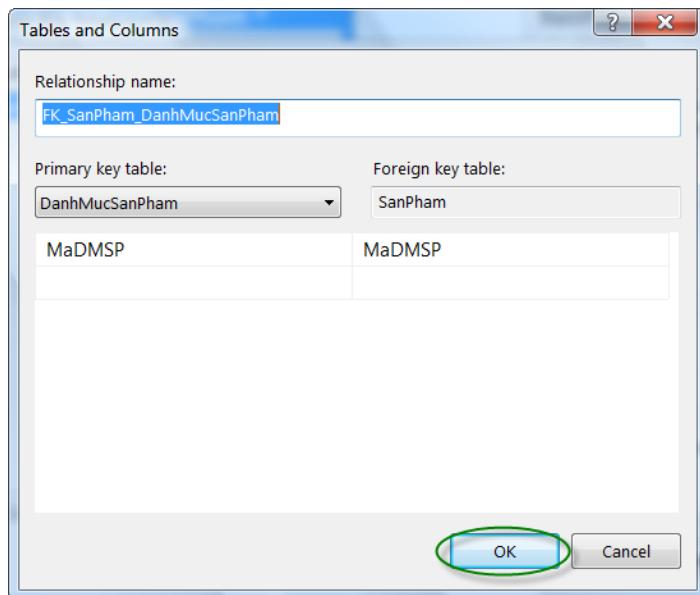
Hình 1.21.

Để thiết lập mối quan hệ giữa các table ta chọn cột dữ liệu của cột làm khóa chính trong bảng cha (parent table) và kéo nó đến khóa ngoại trong bảng con (child table)



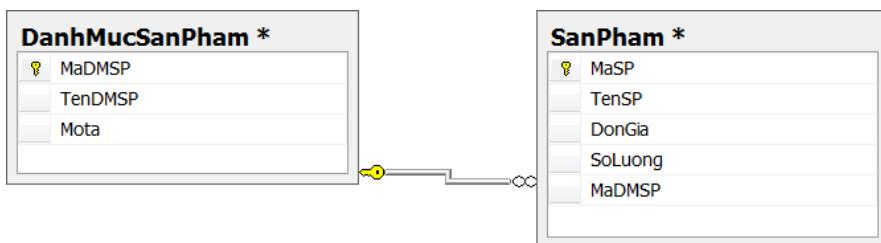
Hình 1.22.

Sau khi kéo mối quan hệ cho 2 table sẽ xuất hiện hộp thoại như hình 1.21.



Hình 1.23.

Khi ta chọn OK giữa 2 table sẽ xuất hiện một kết nối giữa 2 table



Hình 1.24.

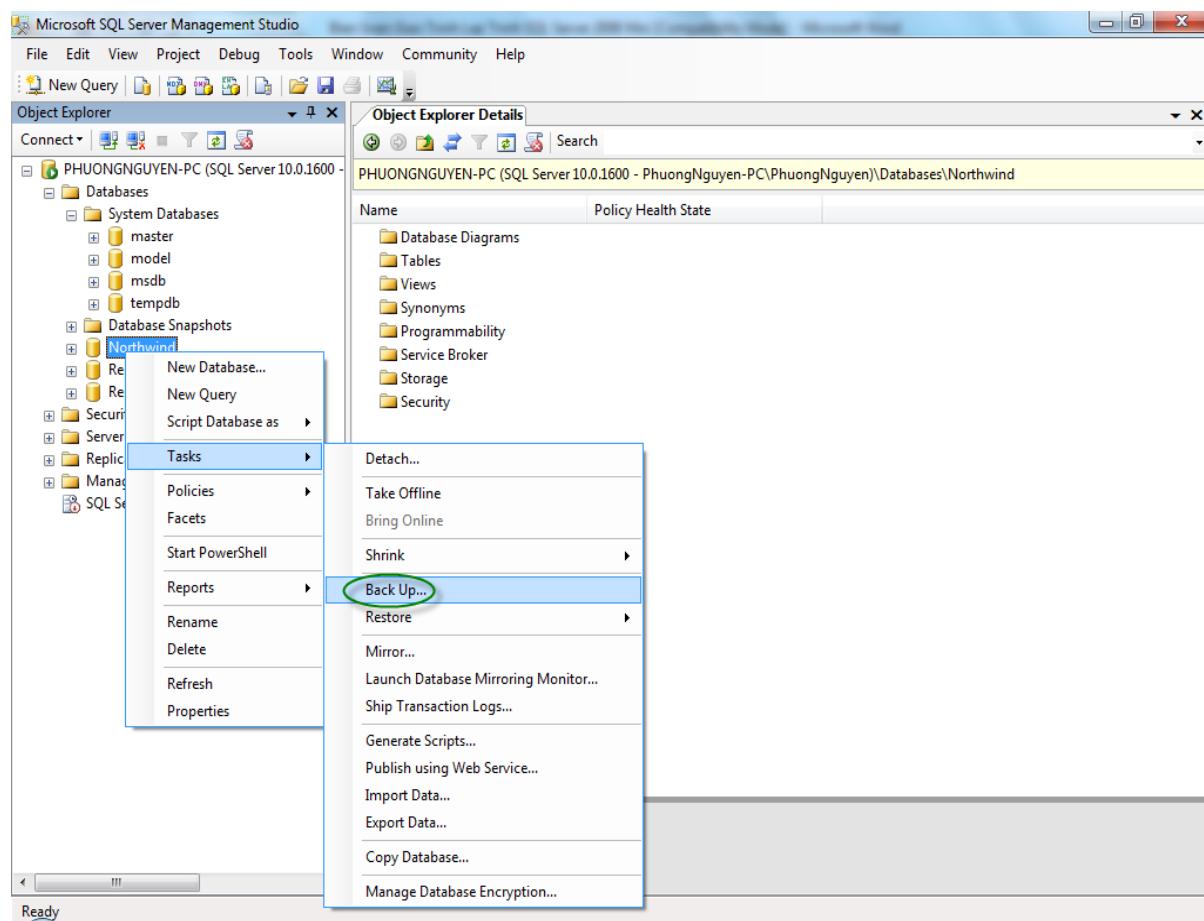
Trong cùng một cách, bạn có thể tạo mối quan hệ khác. Khi bạn đã hoàn tất, bạn có thể lưu và đóng diagram.

4. Back up và Restore dữ liệu

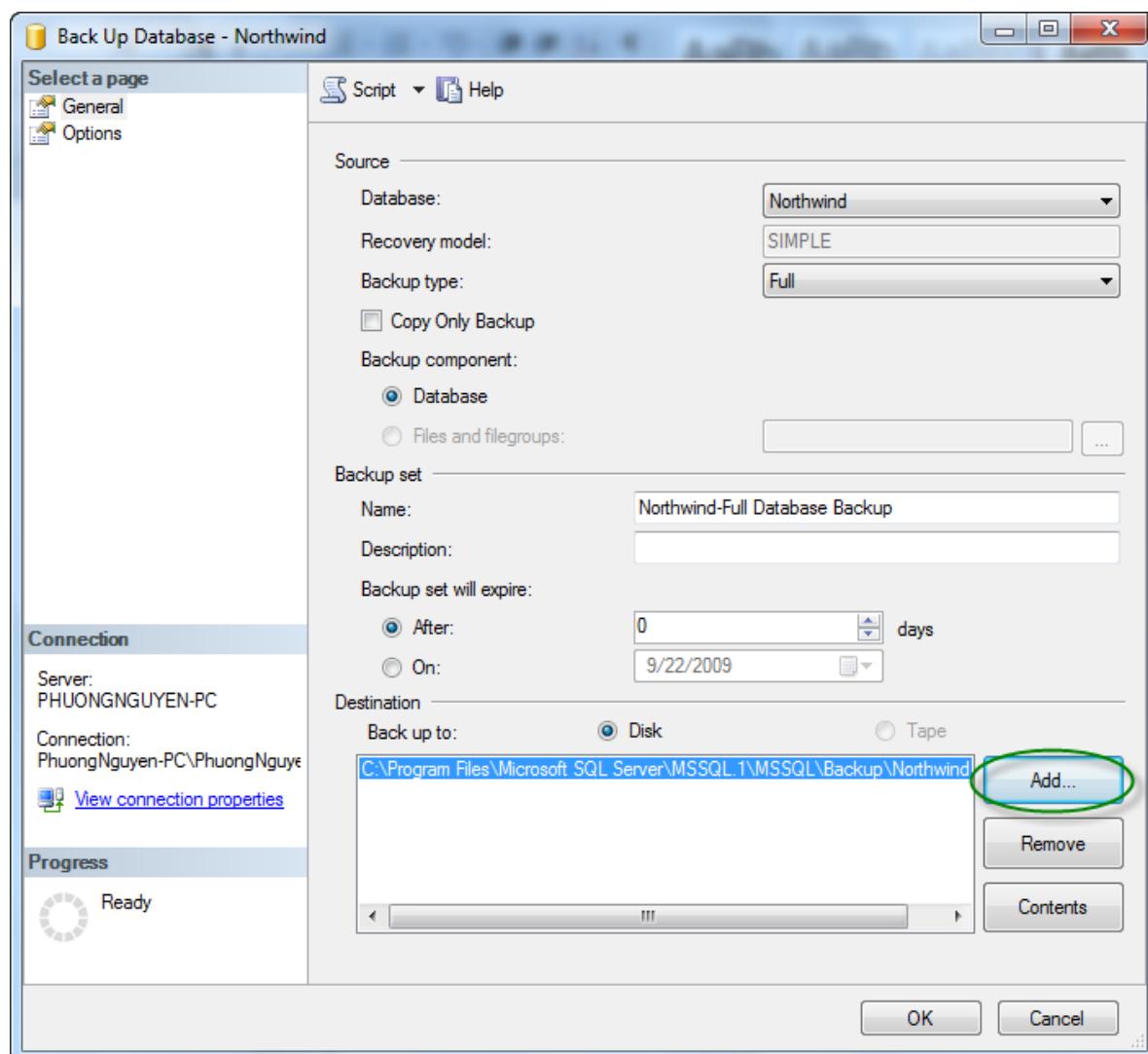
4.1. Back up

Click phải vào database cần back up -> Chọn Tasks -> Chọn Back up...

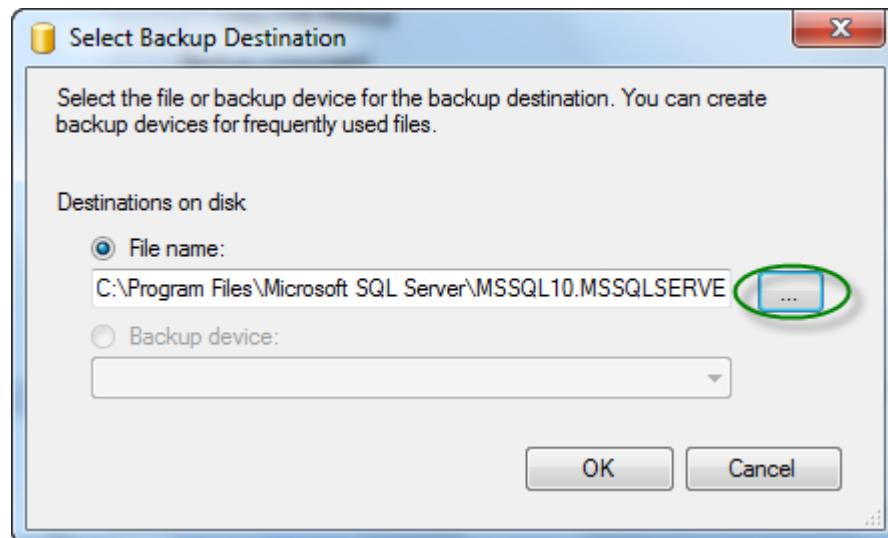
Thực hiện các thao tác theo thứ tự các hình bên dưới



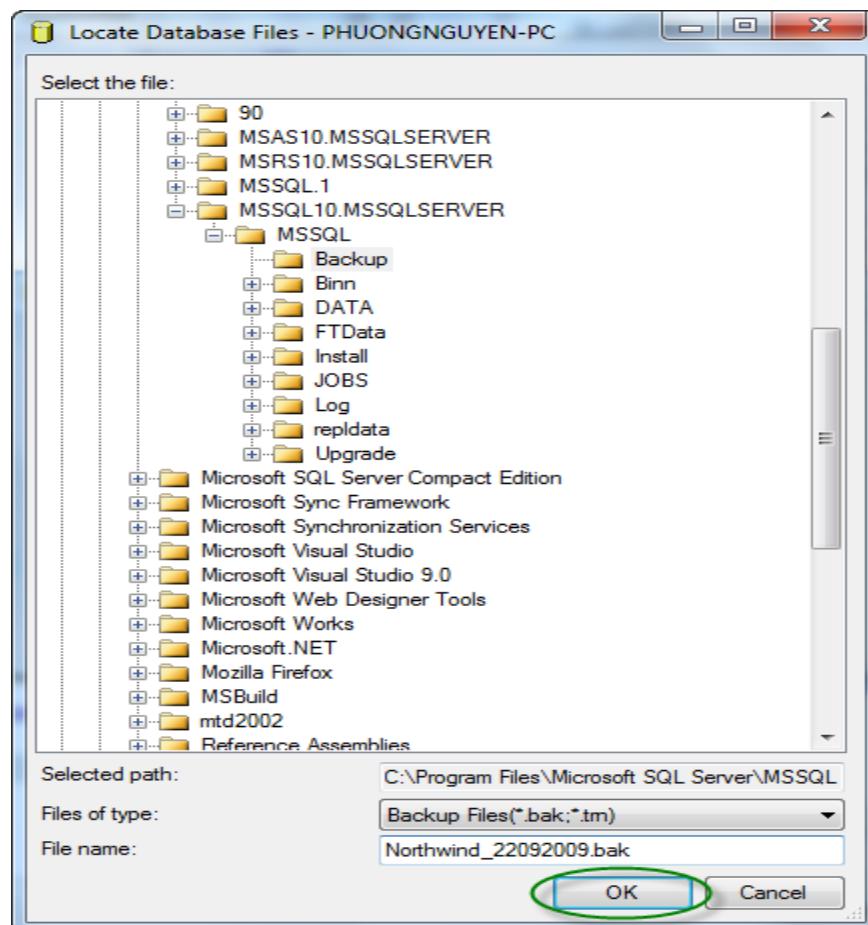
Hình 1.25.



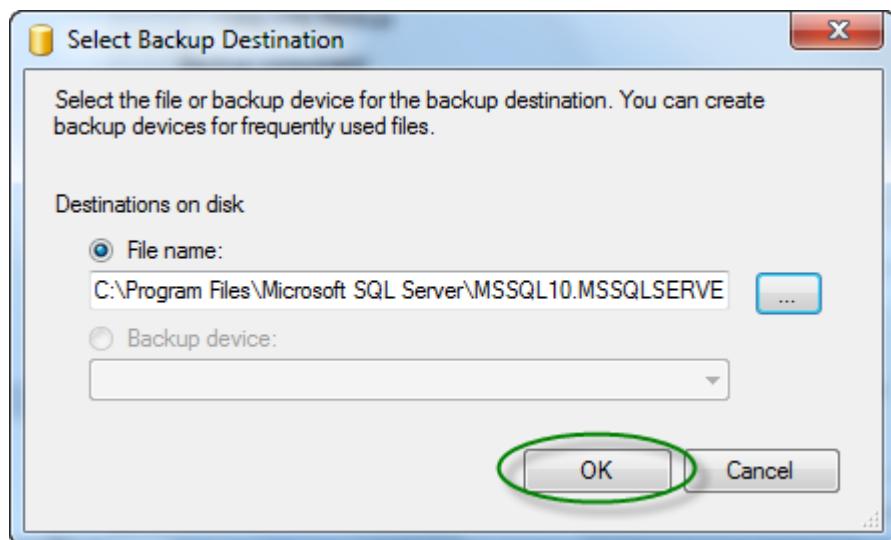
Hình 1.26.



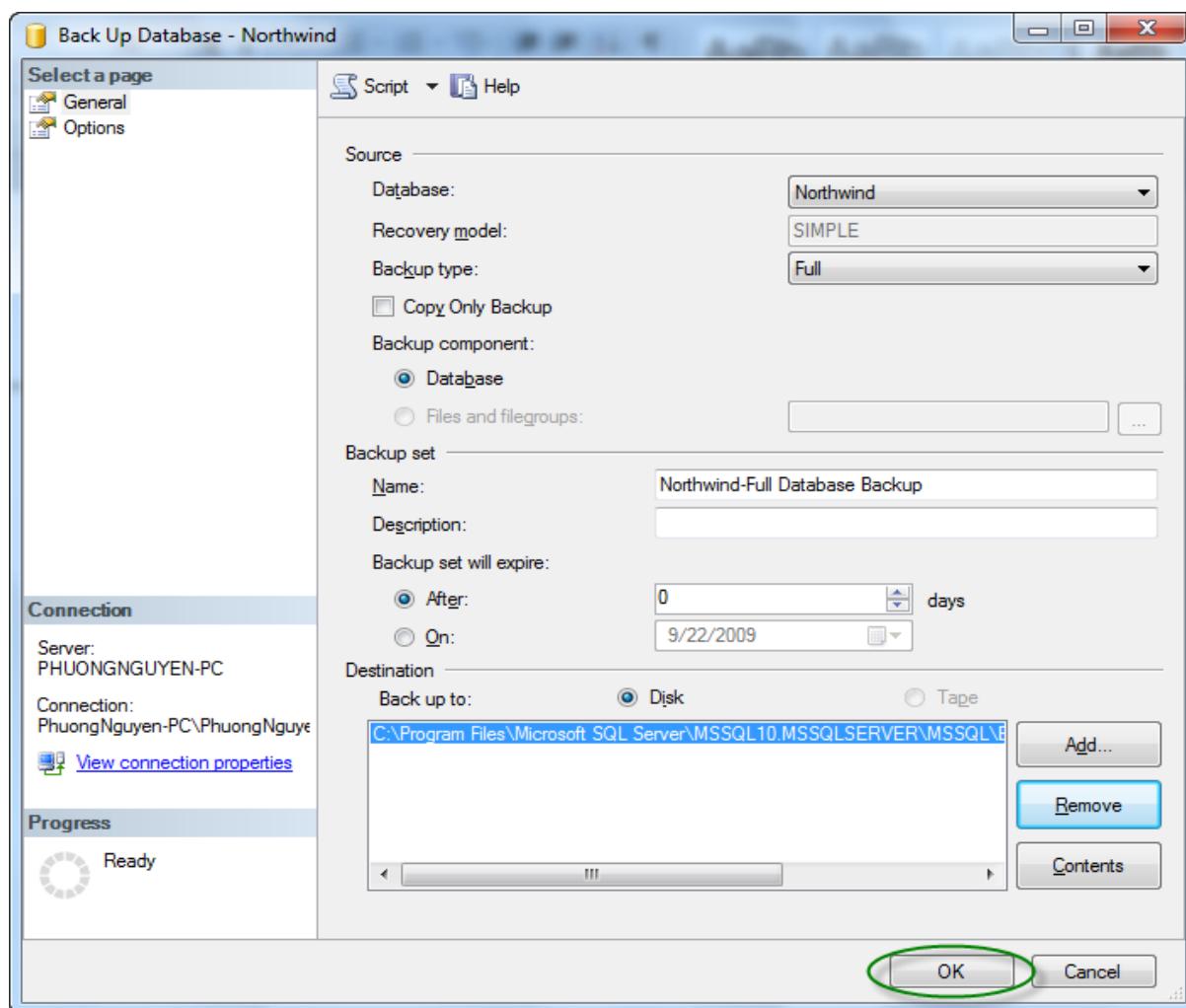
Hình 1.27.



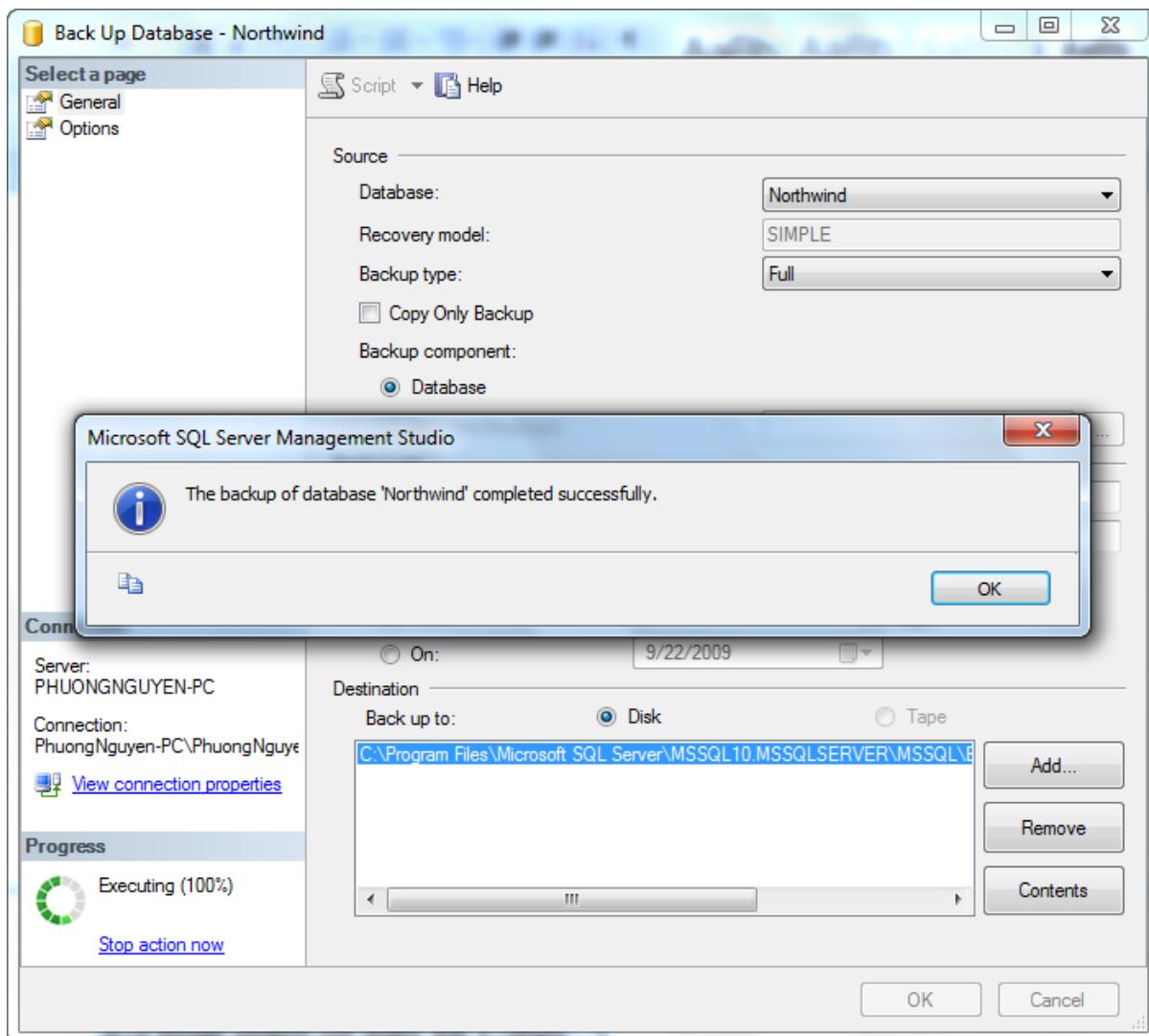
Hình 1.29.



Hình 1.30.



Hình 1.31.

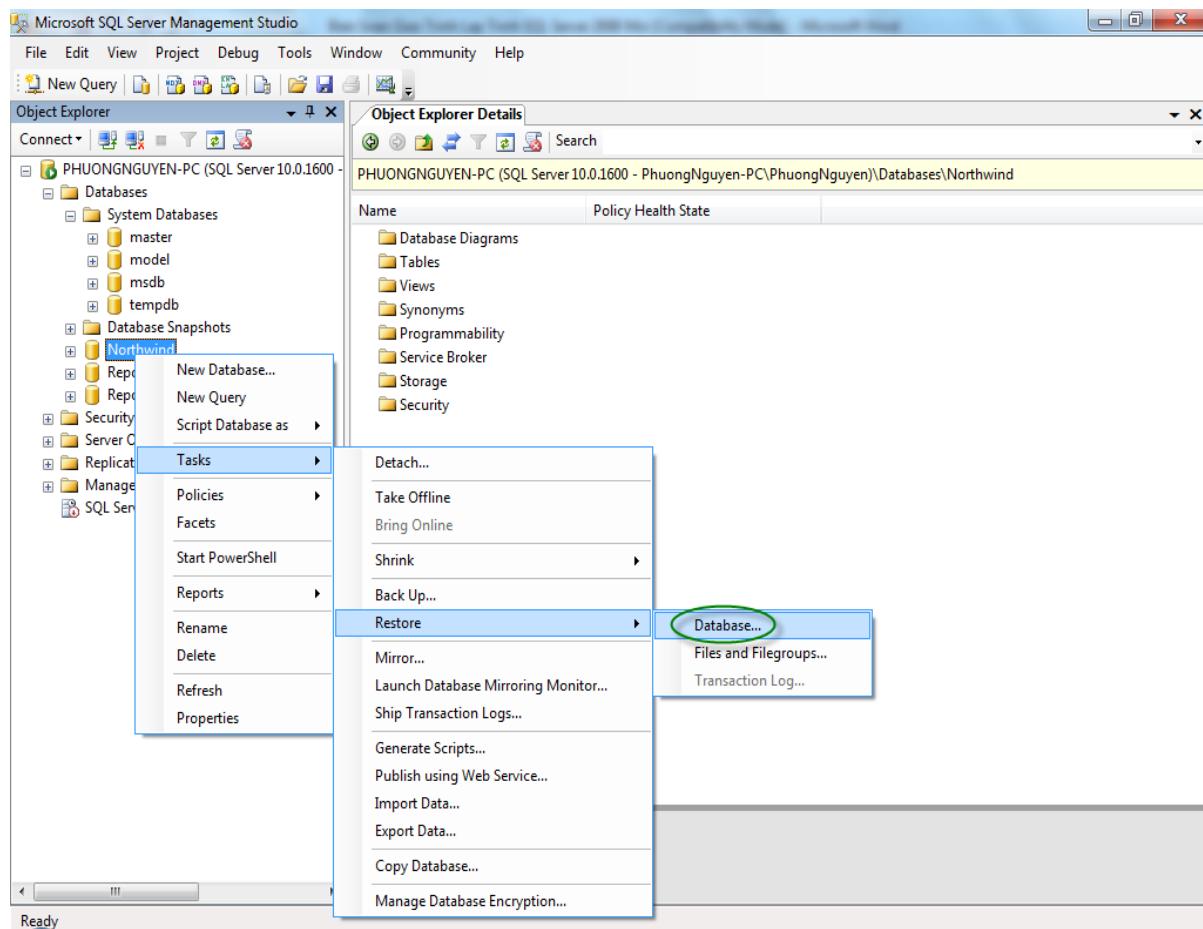


Hình 1.32.

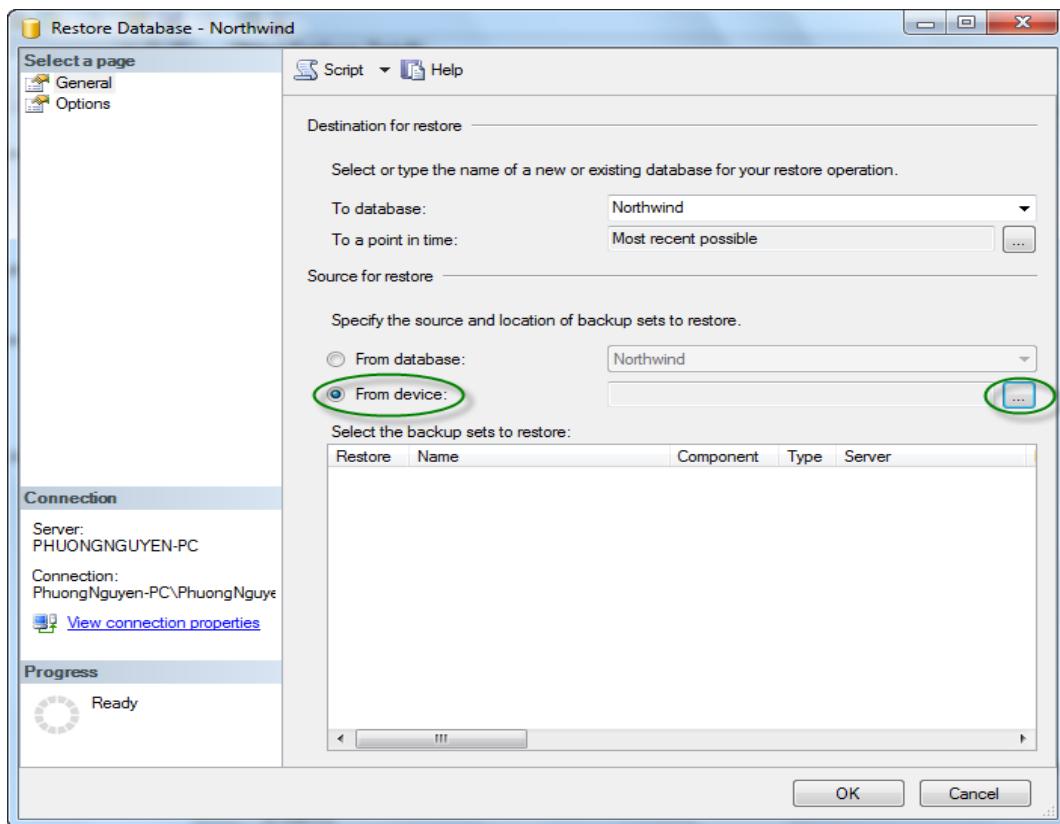
4.2. Restore

Click phải vào database cần Restore -> Chọn Tasks -> Chọn Restore -> Database...

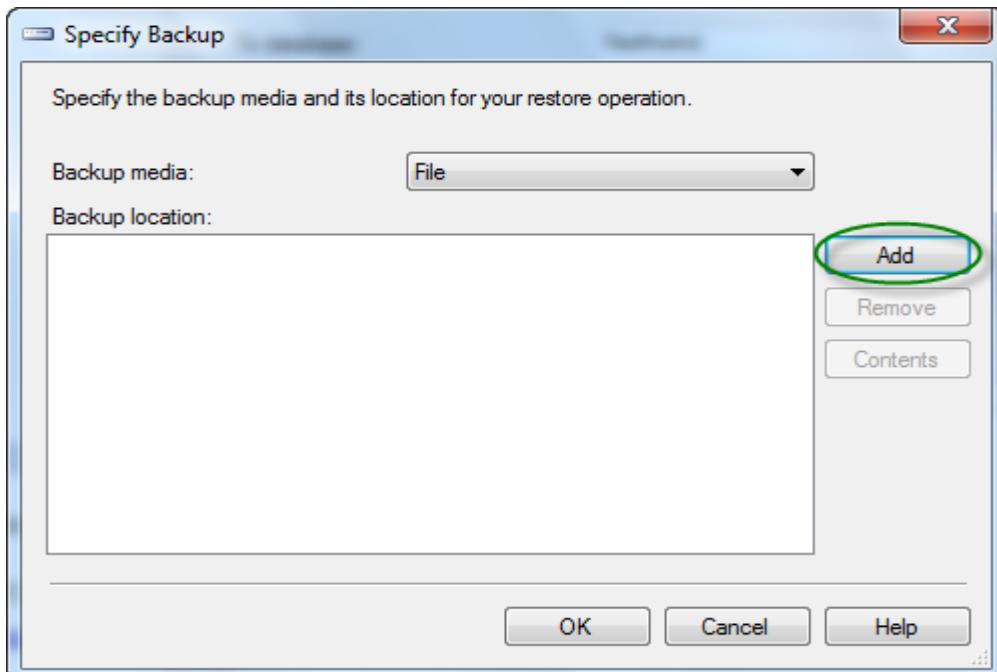
Thực hiện các thao tác theo thứ tự các hình bên dưới



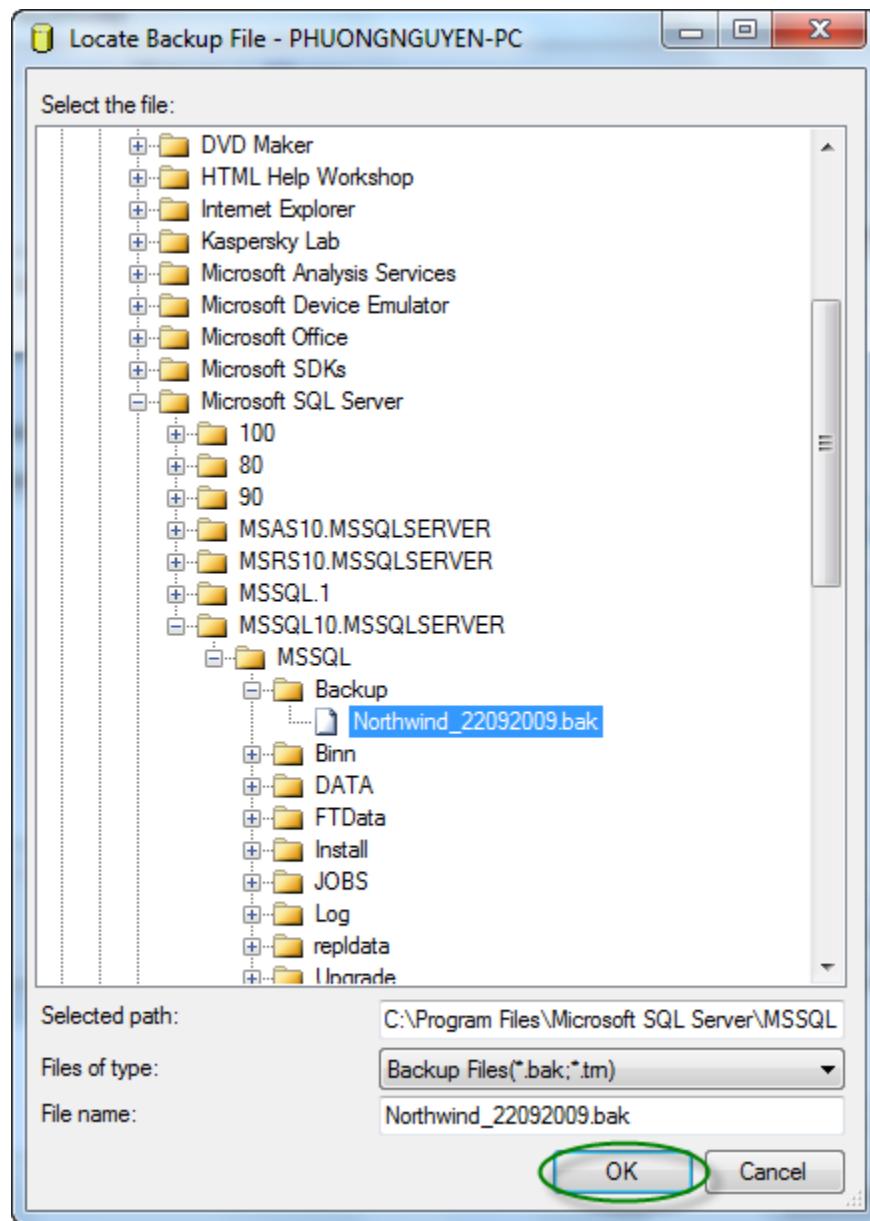
Hình 1.33.



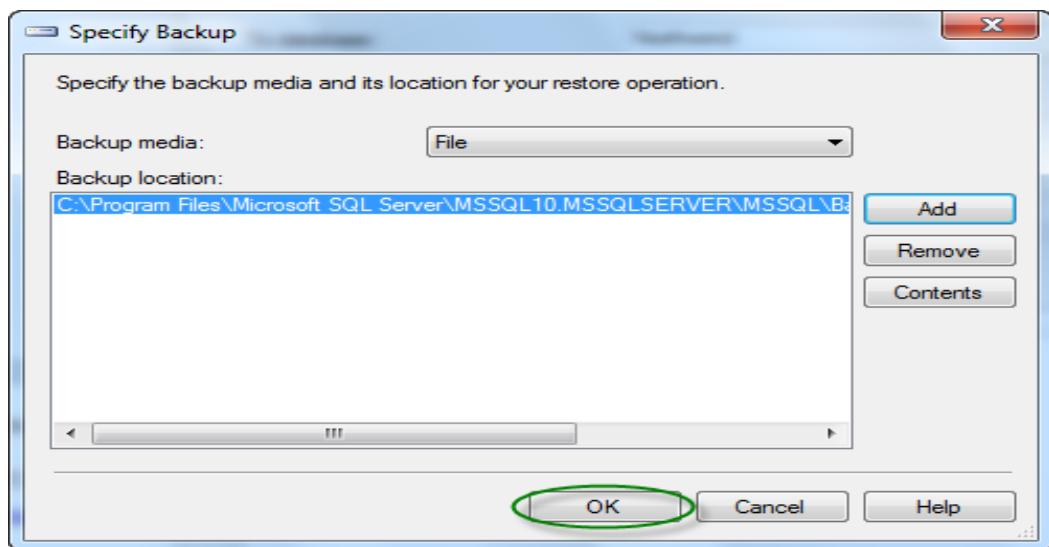
Hình 1.34.



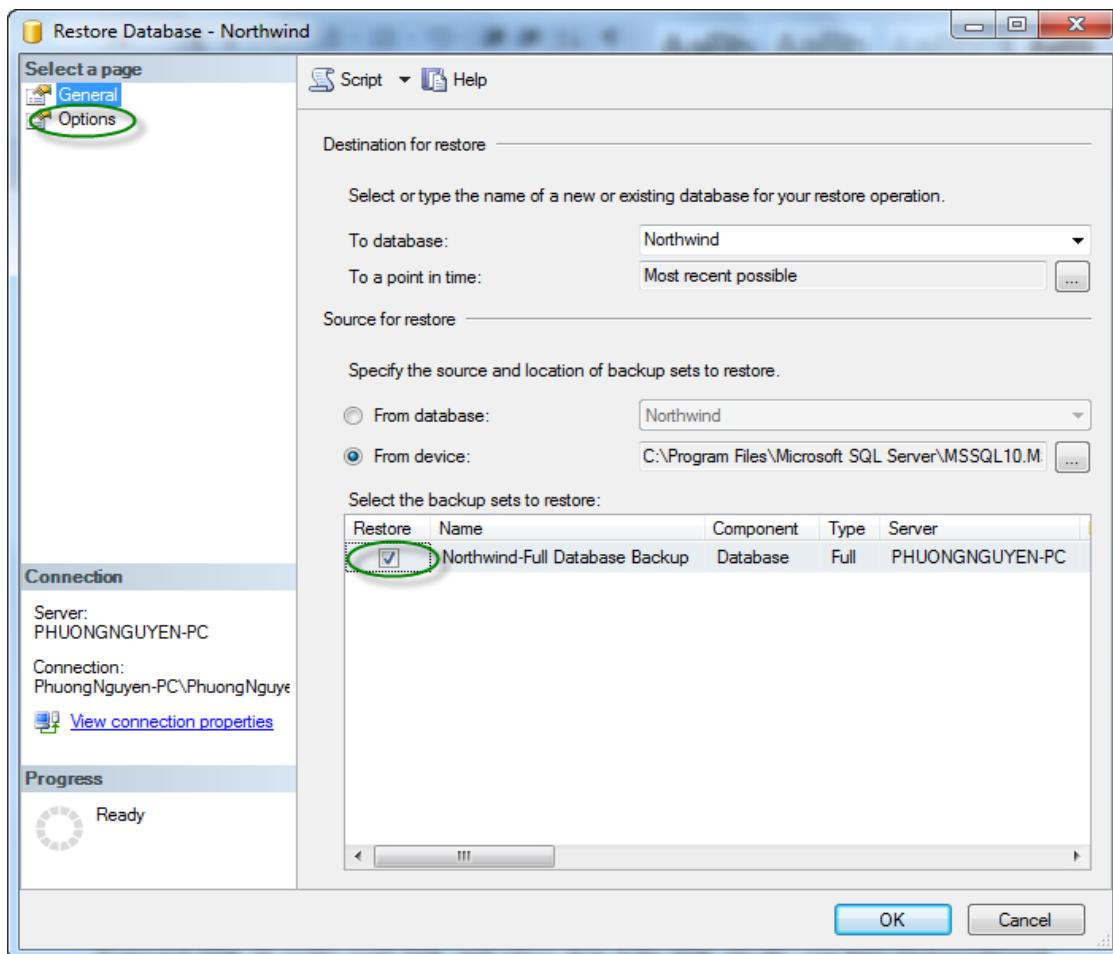
Hình 1.35



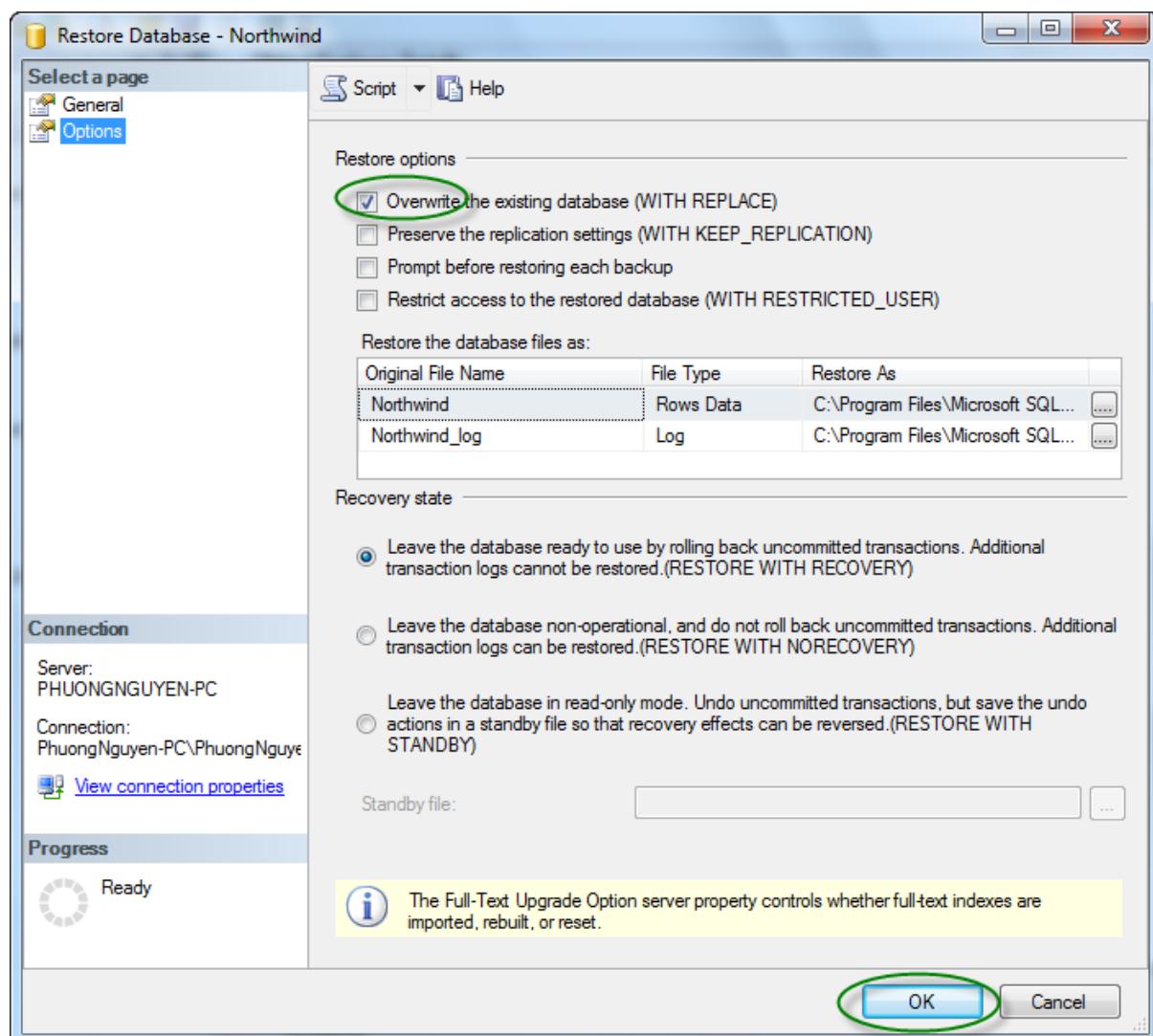
Hình 1.36



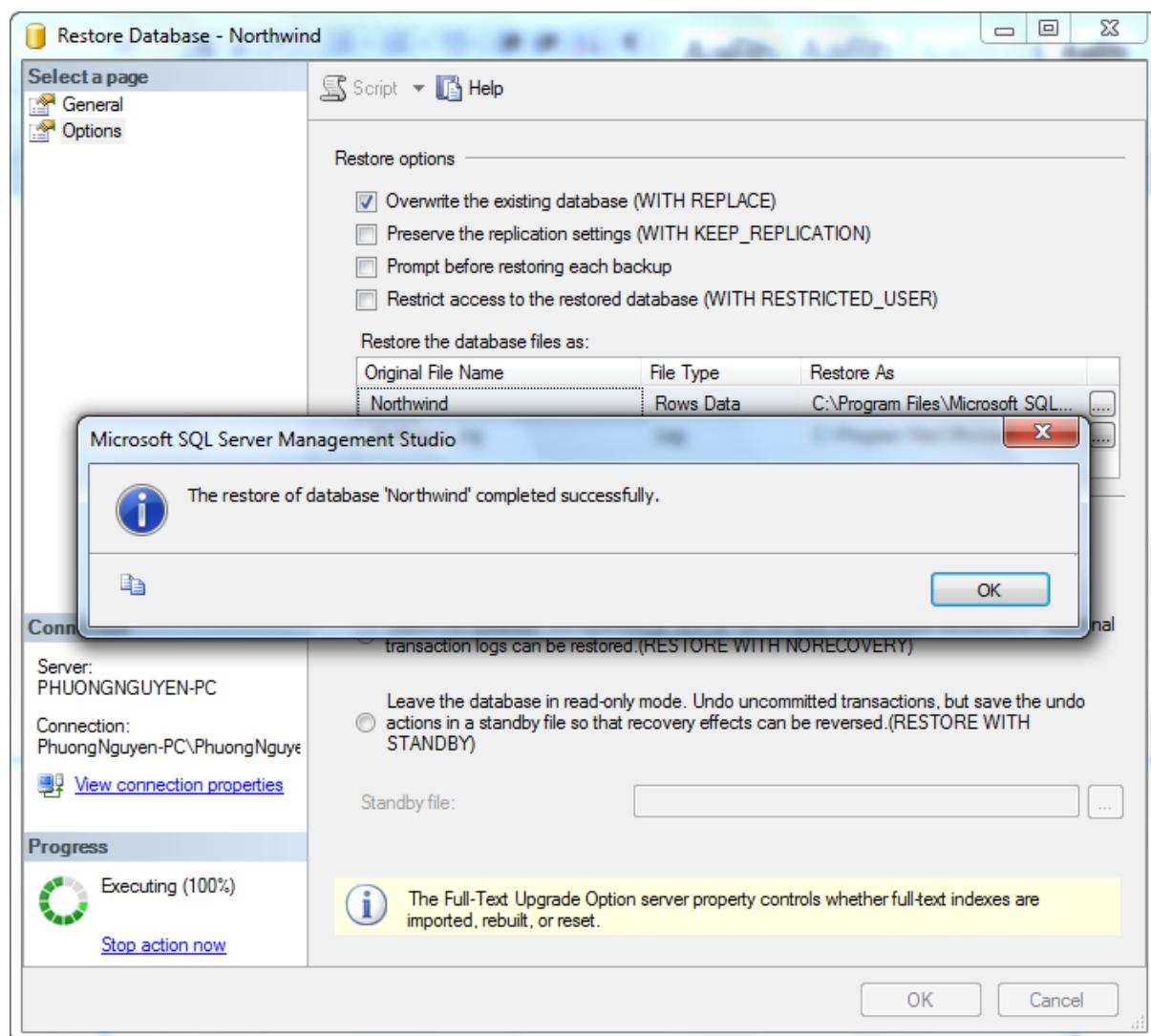
Hình 1.37



Hình 1.38



Hình 1.39



Hình 1.40

Chương 2

Truy Văn Dữ Liệu Cơ Bản

Kết thúc chương này các bạn có thể :

- Trình bày được các lệnh T-SQL : biến, if...else, case...when,...
- Thực hiện được thao tác truy vấn dữ liệu
- Trình bày và vận dụng được các mệnh đề trong truy vấn
- Trình bày và sử dụng được các hàm cơ bản của SQL Server

2.1 Khái niệm cơ bản về 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 PL-SQL ((Procedural Language/Structured Query Language) dùng trong Oracle.

➡ **Trong bài này chúng ta sẽ tìm hiểu sơ qua về T-SQL. Chúng được chia làm 3 nhóm:**

2.1.1. Data Definition Language (DDL):

Đây là những lệnh dùng để quản lý các thuộc tính của một database như định nghĩa các hàng hoặc cột của một table, hay vị trí data file của một database..., thường có dạng như sau :

- *Create object_Name*
- *Alter object_Name*
- *Drop object_Name*

Trong đó *object_Name* có thể là một table, view, stored procedure, indexes...

Ví dụ:

Lệnh **Create** sau sẽ tạo ra một table tên Importers với 3 cột
CompanyID,CompanyName,Contact

USE Northwind -- sử dụng cơ sở dữ liệu Northwind

CREATE TABLE Importers(

 CompanyID int NOT NULL,
 CompanyName varchar(40) NOT NULL,
 Contact varchar(40) NOT NULL
)

Lệnh **Alter** sau đây cho phép ta thay đổi định nghĩa của một table như thêm(hay bớt) một cột hay một Constraint...Trong ví dụ này ta sẽ thêm cột ContactTitle vào table Importers

USE Northwind

ALTER TABLE Importers
ADD ContactTitle varchar(20) NULL

Lệnh **Drop** sau đây sẽ hoàn toàn xóa table khỏi database **nghĩa là cả định nghĩa của table và data bên trong table đều biến mất** (khác với lệnh **Delete** chỉ xóa data nhưng table vẫn tồn tại).

USE Northwind
DROP TABLE Importers

2.1.2. Data Control Language (DCL):

Đây là những lệnh quản lý các quyền truy cập lên từng object (table, view, stored procedure...). Thường có dạng sau:

- **Grant**
- **Revoke**
- **Deny**

Ví dụ:

Lệnh sau sẽ cho phép user trong Public Role được quyền Select đối với table Customer trong database Northwind (**Role** là một khái niệm giống như Windows Group sẽ được bàn kỹ trong phần Security)

```
USE Northwind  
GRANT SELECT  
ON Customers  
TO PUBLIC
```

Lệnh sau sẽ từ chối quyền Select đối với table Customer trong database Northwind của các user trong Public Role

```
USE Northwind  
DENY SELECT  
ON Customers  
TO PUBLIC
```

Lệnh sau sẽ xóa bỏ tác dụng của các quyền được cho phép hay từ chối trước đó

```
USE Northwind  
REVOKE SELECT  
ON Customers  
TO PUBLIC
```

2.1.3. Data Manipulation Language (DML):

Đây là những lệnh phổ biến dùng để xử lý data như Select, Update, Insert, Delete

Ví dụ:

Select

```
USE Northwind  
SELECT CustomerID, CompanyName, ContactName  
FROM Customers  
WHERE (CustomerID = 'alfki' OR CustomerID = 'anatr')  
ORDER BY ContactName
```

Insert

```
USE Northwind  
INSERT INTO Territories VALUES (98101, 'Seattle', 2)
```

Update

```
USE Northwind  
UPDATE Territories  
SET TerritoryDescription = 'Downtown Seattle'  
WHERE TerritoryID = 98101
```

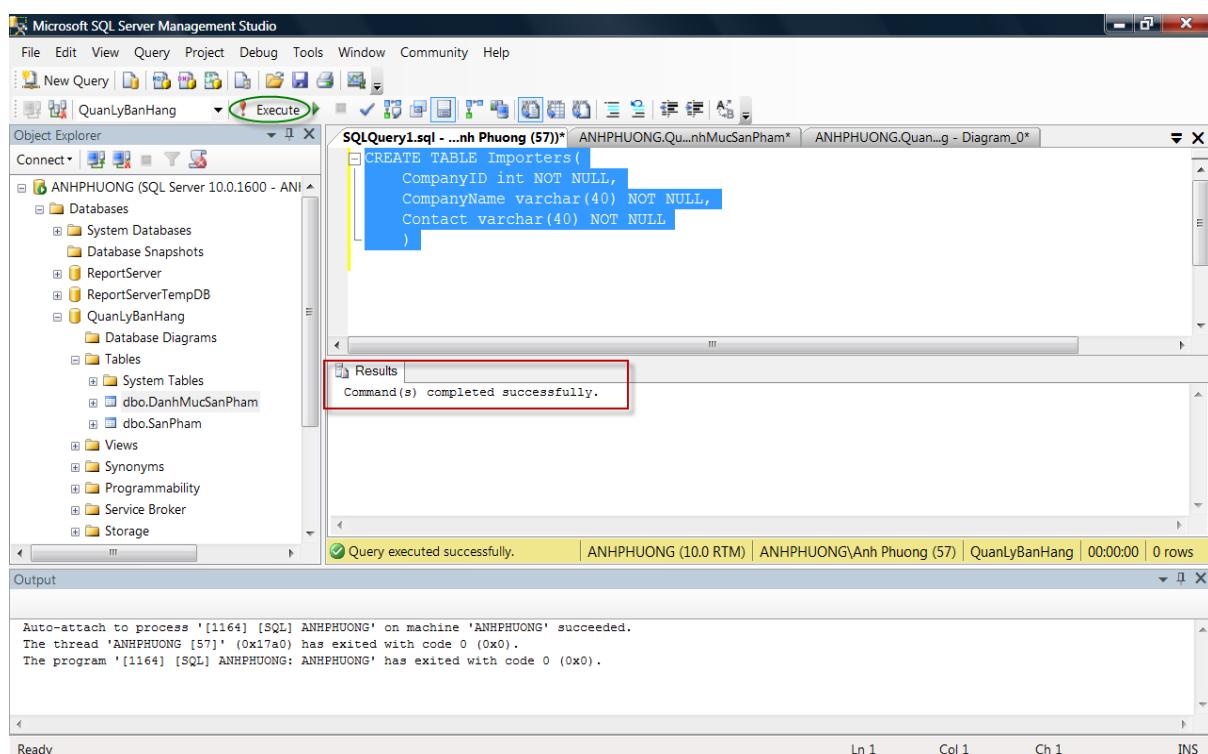
Delete

```
USE Northwind  
DELETE FROM Territories  
WHERE TerritoryID = 98101
```

Chú ý : trong lệnh Delete bạn có thể có chữ From hay không đều được.

Để kiểm tra các ví dụ trên ta làm như sau:

Trên thanh toolbar của màn hình SQL Server Management Studio -> Chọn New Query và gõ các câu lệnh như trên. Sau đây là 1 ví dụ tạo table bằng câu lệnh T-SQL.



Hình 2.1

Cú pháp của T-SQL

Phần này chúng ta sẽ bàn về các thành phần tạo nên cú pháp của T-SQL

Identifiers

Đây chính là tên của các database object. Nó dùng để xác định một object. (Chú ý khi nói đến Object trong SQL Server là chúng ta muốn đề cập đến table, view, stored procedure, index..... Vì hầu như mọi thứ trong SQL Server đều được thiết kế theo kiểu hướng đối tượng (object-oriented)). Trong ví dụ sau TableName, KeyName, Description là những identifiers

```
CREATE TABLE TableName  
(KeyName INT PRIMARY KEY, Description NVARCHAR(80))
```

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

Ví dụ:

```
SELECT * FROM [My Table]  
WHERE [Order] = 10
```

Trong ví dụ trên chữ Order trùng với keyword Order nên cần đặt trong dấu ngoặc vuông [].

Hàm (Functions)

Có 2 loại hàm một loại là built-in và một loại user-defined

Các hàm **Built-In** được chia làm 3 nhóm:

- **Rowset Functions** : Loại này thường trả về một object và được đối xử như một table. Ví dụ như hàm OPENQUERY sẽ trả về một recordset và có thể đứng vị trí của một table trong câu lệnh Select.
- **Aggregate Functions** : Loại này làm việc trên một số giá trị và trả về một giá trị đơn hay là các giá trị tổng. Ví dụ như hàm AVG sẽ trả về giá trị trung bình của một cột.

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

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

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

Chú Thích (Comments)

T-SQL dùng dấu `--` để đánh dấu phần chú thích cho câu lệnh đơn và dùng `/*...*/` để chú thích cho một nhóm

✍ Thực Thi Các Câu Lệnh SQL

Thực thi một câu lệnh đơn:

Một câu lệnh SQL được phân ra thành các thành phần cú pháp như trên bởi một parser, sau đó SQL Optimizer (một bộ phận quan trọng của SQL Server) sẽ phân tích và tìm cách thực thi (Execute Plan) tối ưu nhất ví dụ như cách nào nhanh và tốn ít tài nguyên của máy nhất... và sau đó SQL Server Engine sẽ thực thi và trả về kết quả.

Thực Thi một nhóm lệnh (Batches)

Khi thực thi một nhóm lệnh SQL Server sẽ phân tích và tìm biện pháp tối ưu cho các câu lệnh như một câu lệnh đơn và chứa execution plan đã được biên dịch (compiled) trong bộ nhớ sau đó nếu nhóm lệnh trên được gọi lại lần nữa thì SQL Server không cần biên dịch mà có thể thực thi ngay điều này giúp cho một batch chạy nhanh hơn.

Lệnh GO

Lệnh này chỉ dùng để gửi một tín hiệu cho SQL Server biết đã kết thúc một batch job và yêu cầu thực thi. Nó vốn không phải là một lệnh trong T-SQL.

2.2 Cách sử dụng biến, toán tử, biểu thức, điều kiện

Biến (Variable)

SQL Server cung cấp 2 loại biến trong T-SQL đó là: biến toàn cục **global variable** được bắt đầu bằng @@ và **local variable** (biến cục bộ) được bắt đầu bằng @

Global variable

```
SELECT @@VERSION AS SQL_SERVER_VERSION_DETAILS

SQL_SERVER_VERSION_DETAILS
-----
Microsoft SQL Server 2008 (RTM) - 10.0.1600.22 (Intel X86)
Jul  9 2008 14:43:34
Copyright (c) 1988-2008 Microsoft Corporation
Developer Edition on Windows NT 6.1 <X86> (Build 7600: )

(1 row(s) affected)
```

Một số globle variable trong SQL: @@CONNECTIONS, @@CPU_BUSY, @@CURSOR_ROWS, @@ERROR, @@FETCH_STATUS...

Local variable

```
USE Northwind

DECLARE @EmpIDVar INT

SET @EmpIDVar = 3

SELECT EmployeeID, LastName, FirstName FROM Employees

WHERE EmployeeID = @EmpIDVar + 1
```

	EmployeeID	LastName	FirstName
1	4	Peacock	Margaret

Lưu ý: Khi khai báo biến local variable ta dùng từ khóa DECLARE

Toán tử (Operator)

Toán tử bao gồm các phép tính: +, -, *, /

Ví dụ:

```
DECLARE @A INT, @B INT, @KetQua FLOAT  
SET @A=5  
SET @B=2  
SET @KetQua= @A + @B  
PRINT N'Cộng: ' + convert(nvarchar(40),@KetQua)
```

```
SET @KetQua= @A - @B  
PRINT N'Trừ: ' + convert(nvarchar(40),@KetQua)
```

```
SET @KetQua= @A * @B  
PRINT N'Nhân: ' + convert(nvarchar(40),@KetQua)
```

```
SET @KetQua= @A / @B  
PRINT N'Chia: ' + convert(nvarchar(40),@KetQua)
```

Kết quả:

Cộng: 7

Trừ: 3

Nhân: 10

Chia: 2

Cấu trúc điều khiển

- **IF...ELSE**

Cú pháp:

IF (Biểu_thức)

{ Câu lệnh hoặc nhóm lệnh được thực thi }

ELSE

{ Câu lệnh hoặc nhóm lệnh được thực thi }

Lưu ý: Trong SQL nếu ta muốn thực thi 1 nhóm lệnh thì nhóm lệnh đó phải nằm trong từ khóa BEGIN...END

Ví dụ:

```
DECLARE @CharGender Char(1),
        @Gender Varchar(20);
SET @CharGender = 'F';
IF (@CharGender<>'F')
    SET @Gender='Male'
ELSE
    SET @Gender='Female'
```

`SELECT @Gender AS [Giới Tính]`

▪ CASE...WHEN

Khi chúng ta sử dụng nhiều If..else thì có thể dùng Case..When để thay thế.

Cú pháp:

CASE Biểu thức

WHEN Giá trị 1 THEN kết quả

WHEN Giá trị 2 THEN kết quả

WHEN Giá trị n THEN kết quả

END

Ví dụ:

```
DECLARE @CharGender Char(1),
        @Gender Varchar(20);
SET @CharGender = 'F';
SET @Gender =
CASE @CharGender
    WHEN 'm' THEN 'Male'
    WHEN 'M' THEN 'Male'
    WHEN 'f' THEN 'Female'
    WHEN 'F' THEN 'Female'
END;
SELECT @Gender AS [Giới Tính]
```

- **Khôi lệnh : BEGIN...END**

Cú pháp:

```
BEGIN
    { Câu lệnh hoặc nhóm lệnh được thực thi}
END
```

- **Vòng lặp : WHILE**

Cú pháp:

```
WHILE Biểu thức
{
    Câu lệnh hoặc nhóm lệnh được thực thi
}
```

Ví dụ:

```
DECLARE @Number As int
```

```
SET @Number = 1
WHILE @Number < 5
BEGIN
    SELECT @Number AS Number
    SET @Number = @Number + 1
END
```

Number	
1	1
<hr/>	
1	2
<hr/>	
1	3
<hr/>	
1	4

- **FOR**

Mệnh đề For được dùng để chỉ định chọn lựa BROWSE hoặc XML. BROWSE và XML không liên quan trong cấu trúc lặp.

- **BREAK**

Thoát khỏi vòng lặp WHILE hoặc mệnh đề IF... ELSE được lồng bên trong vòng lặp WHILE. Các câu lệnh thực thi sau từ END được thực thi.

- **CONTINOUS**

Chạy lại vòng lặp WHILE. Các câu lệnh thực thi sau từ khóa CONTIOUS điều được bỏ qua.

- **GOTO**

Dùng GOTO để nhảy đến tên label trong khối lệnh đang được thực thi. Không có phát biểu nào ở giữa GOTO và lable được thực thi.

Cú pháp:

GOTO Tên_lable

- **RETURN**

Chúng ta có thể dùng RETURN bất kỳ thời điểm nào để thoát khỏi khối lệnh, thủ tục. Các phát biểu sau RETURN sẽ không được thực thi.

Cú pháp:

RETURN [integer_expression]

- **WAITFOR**

Thực thi một khối lệnh của một đợt, stored procedure hoặc transaction đến thời gian chỉ định hoặc thời gian được kích hoạt hoặc sửa một câu lệnh được chỉ định hoặc trả về ít nhất một dòng.

Cú Pháp

WAITFOR

{

 DELAY <'thời gian'>

 | TIME <'thời gian'>

}

Ví dụ

```
USE msdb;
EXECUTE sp_add_job @job_name = 'TestJob';
BEGIN
    WAITFOR TIME '10:00';
```

```
EXECUTE sp_update_job @job_name = 'TestJob',
@new_name = 'UpdatedJob';
END;
GO
```

2.3 Truy vấn dữ liệu sử dụng SELECT...FROM

Câu lệnh SELECT... FROM dùng để cho phép bạn có thể chọn lựa các dữ liệu cần thiết từ một hoặc nhiều bảng có quan hệ bên trong một cơ sở dữ liệu.

Cú pháp:

```
SELECT <tên_column_1, tên_column_2,..., tên_column_n>
```

```
FROM <tên_table>
```

hay

```
SELECT * FROM tên_table
```

(*) dùng để lấy tất cả các cột trong table

Ví dụ:

```
USE Northwind
```

```
SELECT RegionID, RegionDescription
```

```
FROM Region
```

Hoặc bạn có thể viết theo cách sau:

```
USE Northwind
```

```
SELECT * FROM Region
```

	RegionID	RegionDescription
1	1	Eastern
2	2	Western
3	3	Northern
4	4	Southern

2.3.1 Truy vấn dữ liệu sử dụng mệnh đề WHERE

Với cú pháp SELECT... FROM bên dưới kết hợp mệnh đề WHERE cho phép bạn có thể lọc các dòng dữ liệu bên trong một bảng phải thỏa điều kiện đưa ra trong mệnh đề WHERE.

Cú pháp:

```
SELECT [DISTINCT ][TOP số dòng[PERCENT]]<tên_column_1,  
       tên_column_2,..., tên_column_n>  
FROM <tên_table>  
WHERE <điều kiện>
```

Từ khóa DISTINCT : dùng để chỉ định truy vấn chỉ chọn ra các dòng dữ liệu duy nhất, không trùng lặp dữ liệu.

Từ khóa TOP : dùng để chỉ định truy vấn chỉ chọn ra chính xác bao nhiêu dòng dữ liệu đầu tiên. Nếu có thêm từ khóa PERCENT đi kèm theo thì truy vấn chỉ chọn ra bao nhiêu phần trăm mẫu tin đầu tiên, lúc bấy giờ con số mà bạn chỉ định phải nằm trong phạm vi từ 0 đến 100. Thông thường khi sử dụng từ khóa TOP thì bạn sẽ kết hợp mệnh đề ORDER BY để sắp xếp lại dữ liệu theo một thứ tự nào đó.

Điều kiện lọc : là điều kiện chỉ định việc lọc ra các mẫu tin bên trong bảng. Thông thường là một biểu thức luận lý.

Ví dụ:

```
USE Northwind  
SELECT RegionID, RegionDescription  
FROM Region  
WHERE RegionDescription='Southern'
```

	RegionID	RegionDescripti...
1	4	Southern

2.3.2 Truy vấn và sắp xếp dữ liệu sử dụng mệnh đề ORDER BY

Với cú pháp SELECT...FROM bên dưới kết hợp mệnh đề ORDER BY cho phép bạn có thể lấy dữ liệu của các cột bên trong một bảng, sau đó sắp xếp lại dữ liệu theo thứ tự chỉ định là tăng hoặc giảm.

Cú pháp:

```
SELECT <tên_column_1, tên_column_2,..., tên_column_n>
FROM <tên_table>
[WHERE <điều kiện>]
ORDER BY <Tên_Column> ASC/ DESC
```

Ví dụ:

```
USE Northwind
SELECT ProductID,ProductName,UnitPrice
FROM Products
WHERE UnitPrice>=34
ORDER BY UnitPrice ASC
```

	ProductID	ProductName	UnitPrice
1	60	Camembert Pierrot	34.00
2	72	Mozzarella di Giovanni	34.80
3	69	Gudbrandsdalsost	36.00
4	56	Gnocchi di nonna Alice	38.00
5	12	Queso Manchego La Pastora	38.00
6	17	Alice Mutton	39.00
7	8	Northwoods Cranberry Sauce	40.00
8	27	Schoggi Schokolade	43.90
9	63	Vegie-spread	43.90
10	28	Rössle Sauerkraut	45.60
11	43	Ipoh Coffee	46.00
12	62	Tarte au sucre	49.30
13	51	Manjimup Dried Apples	53.00
14	59	Raclette Courdavault	55.00
15	18	Carnarvon Tigers	62.50
16	20	Sir Rodney's Marmalade	81.00
17	9	Mishi Kobe Niku	97.00
18	29	Thüringer Rostbratwurst	123.79
19	38	Côte de Blaye	263.50

2.3.3 Sử dụng hàm của T-SQL trong truy vấn dữ liệu

Hàm MAX() : Hàm này sẽ trả về giá trị lớn nhất trong biểu thức. Nó có thể dùng với các kiểu dữ liệu số, chuỗi hay ngày tháng. Max trả về giá trị lớn nhất trong toàn bộ giá trị sau khi đã đổi chiều.

Lưu ý: Hàm MAX bỏ qua các giá trị NULL.

Ví dụ:

```
USE Northwind  
  
SELECT MAX(UnitPrice) AS MaxPrice  
  
FROM Products
```

Hàm MIN() : Ngược lại với hàm MAX. Hàm MIN trả về giá trị nhỏ nhất trong biểu thức. Hàm này có thể dùng với các trường số, chuỗi, ngày tháng. Ngoài ra hàm này bỏ qua giá trị NULL:

Ví dụ:

```
USE Northwind  
  
SELECT MIN(UnitPrice) AS MinPrice FROM Products
```

Hàm AVG() : Hàm này trả về giá trị trung bình của các giá trị trong các trường dữ liệu được chỉ ra trong biểu thức.

Lưu ý: Hàm AVG chỉ được dùng với các trường có kiểu dữ liệu là số. Ngoài ra nó có khả năng loại bỏ giá trị NULL

Ví dụ:

```
USE Northwind  
SELECT AVG(UnitPrice) AS AvgPrice  
FROM Products
```

Hàm SUM() : Hàm này trả về tổng của tất cả các giá trị của trường dữ liệu trong biểu thức. Ngoài ra, bạn có thể dùng tới DISTINCT với SUM để tính tổng cho các giá trị duy nhất của trường dữ liệu trong biểu thức. Các giá trị NULL sẽ bị bỏ qua.

Lưu ý : SUM chỉ dùng cho các trường dữ liệu là kiểu số.

Ví dụ:

```
USE Northwind  
SELECT SUM(UnitPrice) AS [Tổng Sản Phẩm]  
FROM Products
```

Hàm COUNT() : Hàm COUNT được sử dụng đếm các bản ghi được select trong chuỗi truy vấn. Hàm này có thể đếm được các giá trị NULL trong biểu thức. Nếu ta dùng nó với từ khóa DISTINCT, COUNT đếm được các giá trị duy nhất. Ngoài ra nó có thể được dùng với các trường số và ký tự.

Lưu ý: Các bạn có thể dùng ký tự * thay cho biểu thức trong hàm COUNT. Bằng cách này chúng ta có thể đếm được tất cả các bản ghi mà không cần quan tâm đến các trường dữ liệu.

Ví dụ: `USE Northwind`

```
SELECT COUNT(ProductID) FROM Products
```

Hàm SQUARE() : tính bình phương

Ví dụ:

```
DECLARE @A INT  
SET @A=5  
SELECT SQUARE(@A)AS [@A Bình Phương]
```

Hàm ROUND() :

ROUND luôn trả về một giá trị. Nếu chiều dài lớn hơn số lượng các chữ số trước dấu thập phân, ROUND trả về 0.

Round trả về một biểu thức số được làm tròn, bất kể loại dữ liệu, khi chiều dài là một số âm.

Ví dụ	Kết quả
ROUND(748.58, -1)	750.00
ROUND(748.58, -2)	700.00
ROUND(748.58, -3)	1000.00

Ví dụ sau minh họa cách sử dụng Round

```
SELECT ROUND(123.9994, 3), ROUND(123.9995, 3)
```

```
GO
```

	(No column name)	(No column name)
1	123.9990	124.0000

Ví dụ sau minh họa làm tròn và xấp xỉ

```
SELECT ROUND(123.4545, 2);
```

```
GO
```

```
SELECT ROUND(123.45, -2);
```

```
GO
```

(No column name)	
1	123.4500

(No column name)	
1	100.00

Ví dụ sau sử dụng hai câu SELECT để chứng minh sự khác biệt giữa làm tròn và cắt xén. Câu lệnh đầu tiên có kết quả làm tròn. Câu lệnh thứ hai có kết quả cắt xén.

The screenshot shows the SQL Server Management Studio interface with two result sets. The first result set has a header '(No column name)' and one row with value '151.00'. The second result set has a header '(No column name)' and one row with value '150.00'.

Results	
(No column name)	151.00
1	

Results	
(No column name)	150.00
1	

Hàm CHAR()

Ví dụ:

```
USE Northwind;
```

```
GO
```

```
SELECT FirstName + ' ' + LastName, + CHAR(13) + [Address] + CHAR(13) +  
HomePhone
```

```
FROM Employees
```

```
WHERE EmployeeID = 1;
```

```
GO
```

	(No column name)	(No column name)
1	Nancy Davolio	507 - 20th Ave. E. Apt. 2A (206) 555-9857

Hàm UPPER() , LOWER()

Với cú pháp chung bên dưới của các hàm UPPER, LOWER có kết quả trả về là một chuỗi sau khi đã được chuyển đổi các ký tự bên trong chuỗi thành chữ in (upper), hoặc chữ thường (lower).

Ví dụ : hàm UPPER

```
USE Northwind;
GO
SELECT UPPER(FirstName) + ' ' + UPPER(LastName) AS Fullname
FROM Employees
WHERE EmployeeID=1
```

	Fullname
1	NANCY DAVOLIO

Ví dụ : hàm LOWER

```
USE Northwind;
GO
SELECT LOWER(UPPER(FirstName)) + ' ' +
LOWER(UPPER(LastName)) AS Fullname
FROM Employees
WHERE EmployeeID=1
```

	Fullname
1	nancy davolio

Hàm LEN()

Với cú pháp đơn giản của hàm LEN bên dưới có kết quả trả về là một số nguyên dương dùng để chỉ định chiều dài của một chuỗi chứa bao nhiêu ký tự.

Ví dụ:

```
USE Northwind;
GO
SELECT LEN(FirstName) AS [Length], FirstName
FROM Employees
WHERE EmployeeID=1
```

	Length	FirstName
1	5	Nancy

Hàm LTRIM(), RTRIM()

Với cú pháp chung bên dưới của các hàm LTRIM, RTRIM có kết quả trả về là một chuỗi đã được cắt bỏ các khoảng trắng ở đầu chuỗi (ltrim) hoặc các khoảng trắng ở cuối chuỗi (rtrim).

Cú pháp :

LTRIM (chuỗi)

RTRIM (Chuỗi)

Ví dụ : hàm LTRIM

```
DECLARE @string_to_trim varchar(60)
SET @string_to_trim = '    Five spaces are at the beginning of
this string.'
```

```
SELECT 'Here is the string without the leading spaces: ' +
LTRIM(@string_to_trim)
```

```
GO
```

(No column name)
1 Here is the string without the leading spaces: Five spaces are at the beginning of this string.

Ví dụ: hàm RTRIM()

```
DECLARE @string_to_trim varchar(60);

SET @string_to_trim = 'Four spaces are after the period in this sentence.
';
SELECT @string_to_trim + ' Next string.';

SELECT RTRIM(@string_to_trim) + ' Next string.';

GO
```

(No column name)	
1	Four spaces are after the period in this sentence. Next string.
(No column name)	
1	Four spaces are after the period in this sentence. Next string.

Hàm LEFT(),RIGHT(),SUBSTRING

Với cú pháp chung bên dưới của các hàm LEFT, RIGHT, SUBSTRING có kết quả trả về là một chuỗi con được trích ra từ chuỗi nguồn. Chuỗi con được trích ra tại vị trí bắt đầu từ bên trái (left), bên phải (right) hoặc tại bất kỳ vị trí nào (substring) và lấy ra bao nhiêu ký tự.

Cú pháp :

LEFT (chuỗi nguồn , số ký tự)

RIGHT (chuỗi nguồn , số ký tự)

SUBSTRING (chuỗi nguồn ,vị trí, số ký tự)

Trong đó :

- Chuỗi nguồn : là chuỗi ký tự nguồn chứa các ký tự muốn được chọn lựa để trích ra.
- Số ký tự : là một số nguyên dương chỉ định số ký tự bên trong chuỗi nguồn sẽ được trích ra.

- Vị trí : là số nguyên dương chỉ định tại vị trí bắt đầu trích được áp dụng cho hàm SUBSTRING.
- Chuỗi con : là chuỗi kết quả trả về sau khi thực hiện việc trích các ký tự đã chỉ định trong các tham số trên.

Ví dụ: hàm LEFT

`SELECT LEFT('abcdefg',2) -> Kết quả : ab`

Ví dụ: hàm RIGHT

`SELECT RIGHT('abcdefg',2)-> Kết quả fg`

Ví dụ: hàm SUBSTRING

`SELECT SUBSTRING ('abcdefg',4,3)-> Kết quả def`

Hàm GETDATE() : lấy ngày hiện hành

Ví dụ:

```
SELECT GETDATE() AS [Ngày giờ hiện tại]
      , CONVERT (date, GETDATE()) AS [Ngày hiện tại]
      , CONVERT (time, GETDATE()) AS [Giờ hiện tại]
```

	Ngày giờ hiện tại	Ngày hiện tại	Giờ hiện tại
1	2009-09-29 06:43:32.233	2009-09-29	06:43:32.2330000

Hàm DATEPART(YY,getdate()) : lấy 1 phần (ngày , tháng hoặc năm,...) của ngày

Ví dụ:

```
SELECT DATEPART(year, '12:10:30.123') AS [Năm]
      ,DATEPART(month, '12:10:30.123') AS [Tháng]
      ,DATEPART(day, '12:10:30.123') AS [Ngày]
      ,DATEPART(dayofyear, '12:10:30.123') AS [Ngày trong năm]
```

,DATEPART(weekday, '12:10:30.123') AS [Thứ]

	Năm	Tháng	Ngày	Ngày trong năm	Thứ
1	1900	1	1	1	2

Hàm DATEDIFF(X,Y,Z) : tính khoảng cách giữa hai ngày

Ví dụ:

```
DECLARE @startdate datetime2 = '2007-05-05 12:10:09.3312722';
DECLARE @enddate datetime2 = '2007-05-07 12:10:09.3312722';
SELECT DATEDIFF(day, @startdate, @enddate) AS [Số ngày]
```

Số ngày
2

Hàm DAY() : lấy ngày

Ví dụ:

```
SELECT DAY(GETDATE()) AS 'Ngày'
```

Hàm MONTH() : lấy tháng

Ví dụ:

```
SELECT MONTH(GETDATE()) AS 'Tháng'
```

Hàm YEAR() : lấy năm

Ví dụ:

```
SELECT YEAR(GETDATE()) AS 'Năm'
```

Hàm CAST() : chuyển đổi kiểu

Ví dụ:

```
DECLARE @NgaySinh datetime, @Tuoi int
SET @NgaySinh='1986-12-19'
```

```
SET @Tuoi = YEAR(GETDATE())- YEAR(@NgaySinh)
```

```
SELECT N'Bạn được: ' + CAST(@Tuoi AS varchar(10)) + N' tuổi'
```

(No column name)	
1	Bạn được: 23 tuổi

Hàm CONVERT() : chuyển đổi kiểu có định dạng

Ví dụ:

```
DECLARE @NgaySinh datetime, @Tuoi int
```

```
SET @NgaySinh='1986-12-19'
```

```
SET @Tuoi = YEAR(GETDATE())- YEAR(@NgaySinh)
```

```
SELECT N'Bạn được: ' + CONVERT(varchar(10), @Tuoi) + N' tuổi'
```

(No column name)	
1	Bạn được: 23 tuổi

Chương 3:

Truy Vấn Dữ Liệu Trên Nhiều Bảng

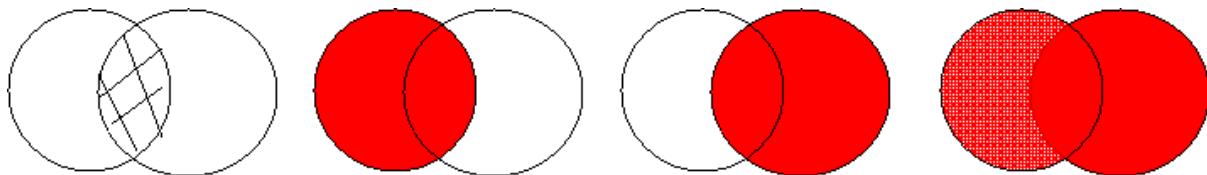
Kết thúc chương này các bạn có thể :

- Trình bày và thực hành được truy vấn trên nhiều bảng với Inner Join
- Trình bày và thực hành được truy vấn trên nhiều bảng với Outer Join
- Trình bày và thực hành được truy vấn trên nhiều bảng với Cross Join
- Trình bày và thực hành được truy vấn kết hợp dữ liệu sử dụng Union
- Trình bày và thực hành được truy vấn tạo bảng với Select...Into
- Trình bày và thực hành được truy vấn sử dụng mệnh đề : Group By, Having, Merge, Intersect...

Trong chương này chúng ta sẽ tìm hiểu về cách truy vấn dữ liệu trên nhiều bảng sử dụng các loại mệnh đề JOIN trong SQL Server.

Với mệnh đề SELECT... FROM kết hợp mệnh đề JOIN cho phép bạn liên kết hai bảng có quan hệ với nhau để lấy ra các dữ liệu chung. Điểm quan trọng giữa những bảng này phải có các cột quan hệ chung nhau và thứ tự quan hệ khi bạn chỉ định giữa các bảng cũng sẽ làm ảnh hưởng đến kết quả của truy vấn.

Bằng cách sử dụng JOIN bạn có thể lấy dữ liệu từ nhiều table dựa trên mối quan hệ logic giữa các table (logical relationship). Có những loại JOIN như sau:



Thứ tự từ trái sang phải: Inner Join, Left Outer Join, Right Outer Join, Full Outer Join

3.1 Truy vấn dữ liệu sử dụng INNER JOIN

Dùng Inner Join để select data từ 2 hay nhiều tables trong đó giá trị của các cột được join phải xuất hiện ở cả 2 tables tức là phần gạch chéo trên hình và

dùng để chỉ định việc so sánh giá trị trong các cột của các bảng là tương đương (dữ liệu đều có ở cả hai bảng). Hệ thống sẽ trả về các mẫu tin thỏa điều kiện quan hệ ở cả hai bảng.

Ví dụ:

```
SELECT TOP 10 ProductName, CategoryName
FROM Products AS P INNER JOIN Categories AS C
ON P.CategoryID=C.CategoryID
ORDER BY ProductName
```

3.2 Truy vấn dữ liệu sử dụng LEFT JOIN

Dùng Left Outer Join để select data từ 2 hay nhiều tables trong đó tất cả cột bên table thứ nhất và không tồn tại bên table thứ hai sẽ được select cộng với các giá trị của các cột được inner join. Số cột select được sẽ bằng với số cột của table thứ nhất. Tức là phần tô màu đỏ trên hình.

Ví dụ:

```
SELECT ProductName, CategoryName
FROM Products AS P LEFT JOIN Categories AS C
ON P.CategoryID=C.CategoryID
ORDER BY ProductName
```

3.3 Truy vấn dữ liệu sử dụng RIGHT OUTER JOIN

Dùng Right Outer Join để select data từ 2 hay nhiều tables trong đó tất cả cột bên table thứ hai và không tồn tại bên table thứ nhất sẽ được select cộng với các giá trị của các cột được inner join. Số cột select được sẽ bằng với số cột của table thứ hai. Tức là phần tô màu đỏ trên hình.

Ví dụ:

```
SELECT ProductName, CategoryName
FROM Products AS P RIGHT JOIN Categories AS C
ON P.CategoryID=C.CategoryID
ORDER BY ProductName
```

3.4 Truy vấn dữ liệu sử dụng CROSS JOIN

Dùng Cross Join ghép data từ hai table trong đó số dòng thu được bằng với số dòng của table thứ nhất nhân với số dòng của table thứ hai.

Ví dụ:

```
SELECT TOP 5 * FROM dbo.Products
CROSS JOIN dbo.Categories
```

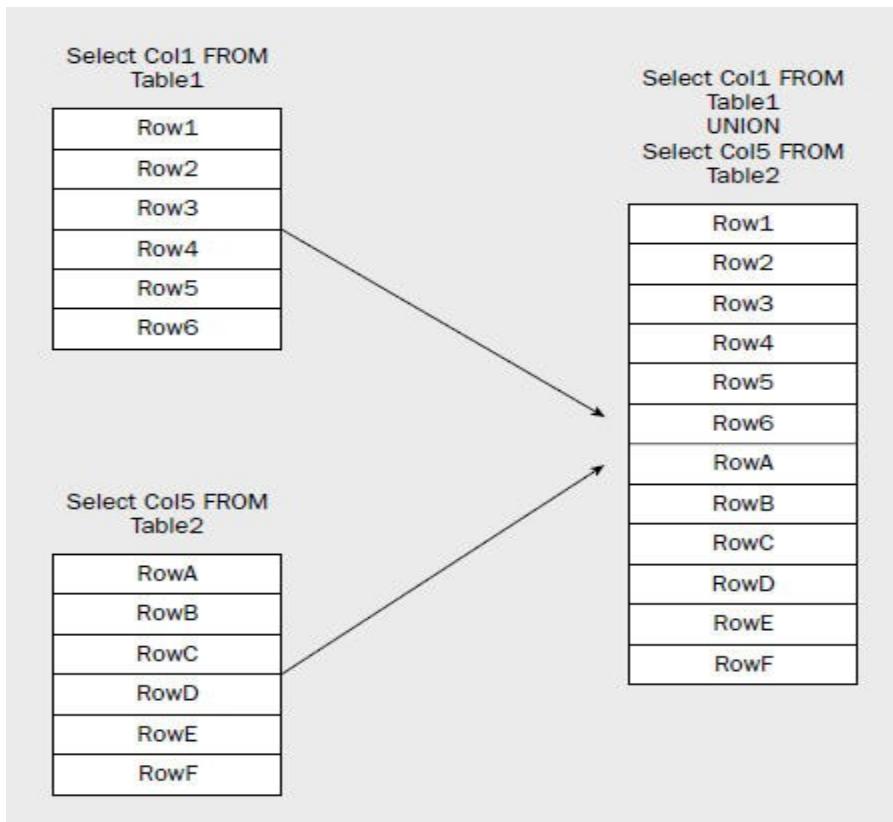
Lưu ý : trong câu lệnh này không có keyword "On".

3.5 Truy vấn dữ liệu và kết hợp dữ liệu sử dụng UNION

Việc kết hợp dữ liệu của hai truy vấn SELECT FROM bằng mệnh đề UNION cho phép bạn có thể tạo ra một tập hợp các mẫu tin từ các mẫu tin có trong câu lệnh SELECT FROM thứ nhất và các mẫu tin có trong câu lệnh SELECT FROM thứ hai. Khác với việc liên kết dữ liệu bằng mệnh đề JOIN, mệnh đề UNION thực ra chỉ thực hiện việc thêm vào các dòng dữ liệu trong câu lệnh SELECT FROM thứ nhất vào cuối các dòng dữ liệu trong câu lệnh SELECT FROM thứ hai.

Thông thường bạn sử dụng mệnh đề UNION dùng để nối dữ liệu từ các bảng khác nhau trong cơ sở dữ liệu thành một bộ các mẫu tin liên tục nhau. Các cột chỉ định trong hai câu lệnh SELECT FROM phải có cùng kiểu dữ liệu tương thích thứ tự như nhau, tổng số các cột phải bằng nhau. Việc định dạng tiêu đề của các cột tính toán chỉ cần thực hiện trong câu lệnh truy vấn đầu tiên.

Kết hợp các kết quả của hai hay nhiều truy vấn thành một tập kết quả duy nhất mà bao gồm tất cả các dòng thuộc về tất cả các truy vấn trong union. Các hoạt động UNION khác nhau từ cách sử dụng join để kết hợp cột từ hai table.



Ví dụ:

```
CREATE TABLE UnionTest1
```

```
(  
    idcol int IDENTITY,  
    col2 char(3),  
)
```

```
CREATE TABLE UnionTest2
```

```
(  
    idcol int IDENTITY,  
    col4 char(3),  
)
```

```
INSERT INTO UnionTest1 VALUES ('AAA');
```

```
INSERT INTO UnionTest1 VALUES ('BBB');
```

```
INSERT INTO UnionTest1 VALUES ('CCC');
```

```
SELECT * FROM UnionTest1;
```

	idcol	col2
1	1	AAA
2	2	BBB
3	3	CCC

```
INSERT INTO UnionTest2 VALUES ('CCC');
```

```
INSERT INTO UnionTest2 VALUES ('DDD');
```

```
INSERT INTO UnionTest2 VALUES ('EEE');
```

Regular UNION

```
SELECT col2 FROM UnionTest1
```

```
UNION
```

```
SELECT col4 FROM UnionTest2;
```

	col2
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

```
UNION ALL
```

```
SELECT col2 FROM UnionTest1
```

```
UNION ALL
```

```
SELECT col4 FROM UnionTest2;
```

	col2
1	AAA
2	BBB
3	CCC
4	CCC
5	DDD
6	EEE

3.6 Truy vấn tạo table sử dụng SELECT ... INTO

Với cú pháp SELECT...FROM bên dưới kết hợp mệnh đề INTO cho phép bạn sao chép dữ liệu và cấu trúc từ kết quả của một truy vấn cho ra một bảng dữ liệu mới bên trong cơ sở dữ liệu hiện hành hoặc các bảng dữ liệu tạm thời dùng để tính toán các xử lý phức tạp. Trong trường hợp nếu bạn muốn tạo ra bảng dữ liệu mới thì bắt buộc tên của bảng phải duy nhất trong cơ sở dữ liệu.

Bạn có thể chỉ định các ký tự dấu thăng (#) hoặc hai ký tự dấu thăng (##) phía trước tên bảng được tạo trong câu lệnh SELECT INTO dùng để tạo ra các bảng tạm cục bộ (local) hoặc các bảng tạm toàn cục (global). Bảng tạm cục bộ chỉ được sử dụng bởi người tạo ra nó và hệ thống sẽ tự động hủy bỏ bảng tạm cục bộ khi người tạo ra bảng ngưng nối kết vào Microsoft SQL Server. Ngược lại bảng tạm toàn cục được sử dụng cho nhiều người khác nhau và hệ thống tự động hủy bảng tạm toàn cục khi không còn người sử dụng nào nối kết vào Microsoft SQL Server.

SELECT ... INTO tạo ra một table mới trong filegroup mặc định và chèn các kết quả dòng từ truy vấn vào table mới.

Cú pháp :

```
SELECT Danh_sách_các_cột INTO Tên_bảng_mới
FROM Tên_bảng_dl
```

Trong đó :

- Tên bảng mới : là tên của bảng mới sẽ được tạo lập có cấu trúc và dữ liệu từ truy vấn.
- Tên bảng dữ liệu : là tên của bảng chứa dữ liệu nguồn cho việc sao chép.

Ví dụ:

```
SELECT TOP 10 [CustomerID] ,[Address],[City]      ,[Region],[PostalCode]
,[Country],[Phone] ,[Fax]
INTO KhachHang
FROM [Northwind].[dbo].[Customers]
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, under the 'QuanLyBanHang' database, there is a table named 'KhachHang' which is highlighted with a green oval. The table has columns: CustomerID, Address, City, Region, PostalCode, Country, Phone, and Fax. The data grid shows 10 rows of customer information. Below the table, the Output window displays a message about auto-attach.

CustomerID	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Obere Str. 57	Berlin	NULL	12209	Germany	030-0074321	030-0076545
ANATR	Avda. de la ...	México D.F.	NULL	05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Mataderos 2...	México D.F.	NULL	05023	Mexico	(5) 555-3932	NULL
AROUT	120 Hanove...	London	NULL	WA1 1DP	UK	(171) 555-7788	(171) 555-6...
BERGS	Berguvsväg...	Luleå	NULL	S-958 22	Sweden	0921-12 34 65	0921-12 34 ...
BLAUS	Forsterstr. 57	Mannheim	NULL	68306	Germany	0621-08460	0621-08924
BLONP	24, place Klé...	Strasbourg	NULL	67000	France	88.60.15.31	88.60.15.32
BOLID	C/ Araquí, 67	Madrid	NULL	28023	Spain	(91) 555 22 82	(91) 555 91 ...
BONAP	12, rue des ...	Marseille	NULL	13008	France	91.24.45.40	91.24.45.41
BOTTM	23 Tsawasse...	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 555-3...
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Output

```
Auto-attach to process '[1164] [SQL] ANHPHUONG' on machine 'ANHPHUONG' succeeded.
The thread 'ANHPHUONG [57]' (0x17a0) has exited with code 0 (0x0).
The program '[1164] [SQL] ANHPHUONG: ANHPHUONG' has exited with code 0 (0x0).
```

Ready

3.7 Truy vấn dữ liệu sử dụng mệnh đề WITH

Xác định một kết quả được đặt tên tạm thời, được biết đến như là một bảng biểu chung.

Ví dụ:

```
USE Northwind
WITH Emp(EmployeeID, FirstName, LastName)
AS
(
    SELECT EmployeeID, FirstName, LastName
    FROM dbo.Employees
)
```

```
SELECT EmployeeID, FirstName, LastName  
FROM Emp
```

	EmployeeID	FirstName	LastName
1	1	Nancy	Davolio
2	2	Andrew	Fuller
3	3	Janet	Leverling
4	4	Margaret	Peacock
5	5	Steven	Buchanan
6	6	Michael	Suyama
7	7	Robert	King
8	8	Laura	Callahan
9	9	Anne	Dodsworth

3.8 Truy vấn dữ liệu sử dụng mệnh đề ORDER BY

Bạn có thể sắp xếp kết quả theo các ký tự đầu tiên bằng cách sử dụng mệnh đề Order by.

Ví dụ:

```
USE Northwind  
SELECT FirstName, LastName, [Address]  
FROM dbo.Employees  
ORDER BY FirstName ASC
```

	FirstName	LastName	Address
1	Andrew	Fuller	908 W. Capital Way
2	Anne	Dodsworth	7 Houndstooth Rd.
3	Janet	Leverling	722 Moss Bay Blvd.
4	Laura	Callahan	4726 - 11th Ave. N.E.
5	Margaret	Peacock	4110 Old Redmond Rd.
6	Michael	Suyama	Coventry House Miner Rd.
7	Nancy	Davolio	507 - 20th Ave. E. Apt. 2A
8	Robert	King	Edgeham Hollow Winchester Way
9	Steven	Buchanan	14 Garrett Hill

3.9 Truy vấn dữ liệu sử dụng mệnh đề MERGE

Một điểm mới trong SQL Server 2008 là mệnh đề MERGE. Bạn có thể kết hợp 2 hay nhiều table với nhau.

Ví dụ:

`USE Northwind`

`SELECT EM.FirstName, E.LastName, E.[Address]`

`FROM dbo.Employees E`

`INNER MERGE JOIN dbo.Employees EM`

`ON E.EmployeeID=EM.EmployeeID`

`ORDER BY EM.FirstName ASC`

	FirstName	LastName	Address
1	Andrew	Fuller	908 W. Capital Way
2	Anne	Dodsworth	7 Houndstooth Rd.
3	Janet	Leverling	722 Moss Bay Blvd.
4	Laura	Callahan	4726 - 11th Ave. N.E.
5	Margaret	Peacock	4110 Old Redmond Rd.
6	Michael	Suyama	Coventry House Miner Rd.
7	Nancy	Davolio	507 - 20th Ave. E. Apt. 2A
8	Robert	King	Edgeham Hollow Winchester Way
9	Steven	Buchanan	14 Garrett Hill

3.10 Truy vấn dữ liệu sử dụng mệnh đề GROUP BY

Với cú pháp SELECT FROM bên dưới kết hợp mệnh đề GROUP BY cho phép bạn có thể nhóm dữ liệu của các dòng bên trong một bảng và được phép sử dụng các hàm thống kê đi kèm theo để tính toán các dữ liệu có tính chất thống kê tổng hợp. Thông thường, sau khi nhóm dữ liệu, bạn nên sắp xếp lại dữ liệu để hiển thị theo một thứ tự nào đó. Do vậy bạn sẽ sử dụng mệnh đề ORDER BY sau mệnh đề GROUP BY. Mệnh đề group by dùng để gom nhóm khi tính toán.

Cú pháp :

```
SELECT Danh_sách_các_cột | Hàm_thống_kê AS Bí_danh
FROM Tên_bảng
[WHERE Điều_kiện_lọc]
GROUP BY Danh_sách_cột_nhómdl
[ORDER BY Tên_cột [DESC] [, ...]]
```

Ví dụ:

```
USE Northwind
SELECT Country, COUNT(EmployeeID) AS 'Tổng'
FROM dbo.Employees
GROUP BY Country
```

	Country	Tổng
1	UK	4
2	USA	5

3.11 Truy vấn dữ liệu sử dụng mệnh đề HAVING

Với mệnh đề SELECT... FROM bên dưới kết hợp mệnh đề HAVING cho phép bạn có thể lọc lại dữ liệu sau khi đã nhóm dữ liệu của các dòng bên trong một bảng. Khác với mệnh đề WHERE dùng để lọc các dòng dữ liệu hiện đang có bên trong bảng, mệnh đề HAVING chỉ được phép sử dụng đi kèm theo mệnh đề GROUP BY dùng để lọc lại dữ liệu sau khi đã nhóm. Điều này có nghĩa là mệnh đề HAVING chỉ được dùng kèm với mệnh đề GROUP BY.

Mệnh đề HAVING dùng để xác định một điều kiện tìm kiếm cho một nhóm hoặc một tập hợp. HAVING thường được sử dụng trong một mệnh đề GROUP BY.

Cú pháp :

```
SELECT Danh_sách_các_cột | Hàm_thống_kê AS Bí_danh
  FROM Tên_bảng
 [WHERE Điều_kiện_lọc]
 GROUP BY Danh_sách_cột_nhóm
 HAVING Điều_kiện_lọc_nhóm
 [ORDER BY Tên_cột [DESC] [, ...]]
```

Trong đó :

- **Hàm thống kê** : là tên của các hàm thống kê và các tham số tương ứng dùng để tính tổng (SUM), tính giá trị thấp nhất (MIN), tính giá trị cao nhất (MAX), đếm các mẫu tin (COUNT), tính giá trị trung bình (AVG) của các dữ liệu bên trong bảng.
- **Bí danh** : là tiêu đề mới của các cột tính toán. Các tiêu đề này chỉ có hiệu lực lúc hiển thị dữ liệu trong câu lệnh truy vấn mà không làm ảnh hưởng đến cấu trúc bên trong của bảng.
- **Danh sách cột nhóm dữ liệu** : là danh sách tên các cột được nhóm dữ liệu để tính toán.

Ví dụ: Lấy danh sách các hóa đơn có trị giá > 10000 và kết quả được sắp xếp tăng dần theo trị giá

```
SELECT OD.OrderID, SUM(UnitPrice*Quantity)AS Subtotal
  FROM [Order Details] OD JOIN dbo.Orders O
    ON OD.OrderID=O.OrderID
 GROUP BY OD.OrderID
 HAVING SUM(UnitPrice*Quantity)>10000
 ORDER BY SUM(UnitPrice*Quantity) ASC
```

Kết quả :

	OrderID	Subtotal
1	10691	10164.80
2	10540	10191.70
3	10479	10495.60
4	10515	10588.50
5	10353	10741.60
6	10897	10835.24
7	10417	11283.20
8	10889	11380.00
9	10817	11490.70
10	10424	11493.20
11	10372	12281.20
12	10981	15810.00
13	11030	16321.90
14	10865	17250.00

3.12 Truy vấn dữ liệu sử dụng mệnh đề INTERSECT

INTERSECT tự động loại bỏ các dữ liệu trùng từ 2 câu lệnh truy vấn.

Ví dụ:

```
USE Northwind  
  
SELECT RegionID, RegionDescription  
FROM dbo.Region  
  
INTERSECT  
  
SELECT RegionID, RegionDescription  
FROM dbo.Region
```

	RegionID	RegionDescripti...
1	1	Eastern
2	2	Western
3	3	Northern
4	4	Southern

USE Northwind

SELECT OrderID FROM dbo.Orders

INTERSECT

SELECT OrderID FROM [Order Details]

3.13 Truy vấn dữ liệu sử dụng mệnh đề EXCEPT

EXCEPT tự động loại bỏ các dữ liệu trùng từ 2 câu lệnh truy vấn.

Ví dụ:

USE Northwind

SELECT OrderID FROM dbo.Orders

EXCEPT

SELECT OrderID FROM [Order Details]

```
OrderID
-----

```

```
(0 row(s) affected)
```

3.14 Truy vấn dữ liệu sử dụng mệnh đề COMPUTE BY

Với cú pháp SELECT FROM bên dưới kết hợp mệnh đề COMPUTE cho phép bạn có thể tạo ra dòng thống kê dữ liệu ở bên cuối kết quả truy vấn. Tuy nhiên nếu bạn sử dụng thêm mệnh đề COMPUTE BY tiếp theo thì hệ thống sẽ thống kê dữ liệu theo từng nhóm dữ liệu.

Cú pháp :

```
SELECT Danh_sách_các_cột
FROM Tên_bảng
COMPUTE COUNT |MIN|MAX|SUM|AVG (Tên_cột)
[ BY Tên_cột_nhóm ]
```

Trong đó :

- Count, Min, Max, Sum, Avg : là các hàm thống kê tính toán dữ liệu mà kết quả sẽ xuất hiện ở cuối kết quả truy vấn hoặc từng nhóm dữ liệu.
- Tên cột : tên các cột hoặc biểu thức được tính toán kèm với các hàm thống kê chỉ định trước đó.

Ví dụ:

USE Northwind

```
SELECT * FROM Suppliers COMPUTE count(SupplierID)
```

USE Northwind

```
SELECT * FROM Suppliers ORDER BY SupplierID COMPUTE count(SupplierID) BY
SupplierID
```

3.15 Truy vấn dữ liệu sử dụng mệnh đề FOR

Mệnh đề FOR được sử dụng chỉ định trong sự chọn lựa BROWSE hay XML. BROWSE hay XML không liên quan trong cấu trúc lặp.

Ví dụ:

USE Northwind

```
SELECT EmployeeID, FirstName, LastName, City, Country
```

```
FROM dbo.Employees
```

```
ORDER BY FirstName, LastName
```

```
FOR XML AUTO
```

Kết quả :

```
XML_F52E2B61-18A1-11d1-B105-00805F49916B
1 <dbo.Employees EmployeeID="2" FirstName="Andrew" LastName="Ful...
```

Chương 4

Truy Vấn Dữ Liệu Nâng Cao

Kết thúc chương này các bạn có thể :

- Trình bày được khái niệm cơ bản SubQuery (truy vấn con)
- Thực hành được truy vấn dữ liệu sử dụng SubQueries như Table (bảng)
- Thực hành được truy vấn dữ liệu sử dụng SubQueries như Expression (biểu thức)
- Thực hành được truy vấn dữ liệu sử dụng mệnh đề EXISTS và NOT EXIITS
- Thực hành được truy vấn dữ liệu Sử dụng từ khóa DISTINCT

Trong khi lập trình bên trong Transaction-SQL, có đôi lúc bạn sẽ sử dụng đến truy vấn con để tính toán dữ liệu. Truy vấn con chỉ là một câu lệnh truy vấn chọn lựa (SELECT) được lồng vào các câu lệnh truy vấn khác nhằm thực hiện các truy vấn tính toán phức tạp.

Khi sử dụng đến truy vấn con, bạn cần lưu tâm đến một vài yếu tố sau :

- Cần mở và đóng ngoặc đơn cho câu lệnh truy vấn con.
- Bạn chỉ được phép tham chiếu đến tên một cột hoặc một biểu thức sẽ trả về giá trị trong truy vấn con.
- Kết quả của truy vấn con có thể trả về là một giá trị đơn lẻ hoặc một danh sách các giá trị.
- Cấp độ lồng nhau của các truy vấn con bên trong Microsoft SQL Server là 32 mức.

4.1 Sử dụng SubQueries như Table

Chúng ta có thể sử dụng một câu lệnh **SELECT** để trả về các bản ghi mà sẽ được sử dụng bởi câu **SELECT** khác. Câu lệnh bao ở bên ngoài gọi là **parent query** và câu lệnh bên trong gọi là **subquery**.

Select <Column Name> from <table> Where

Ví dụ: giả sử chúng muốn biết các sản phẩm mà đã được đặt hàng, chúng ta có thể sử dụng câu lệnh , trong ví dụ này chúng ta sử dụng cơ sở dữ liệu Northwind.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid titled 'Results' showing a list of product names from the 'Products' table. The grid has a header row with 'productname' and 12 data rows. The data includes: 1 Chai, 2 Chang, 3 Aniseed Syrup, 4 Chef Anton's Cajun Seasoning, 5 Chef Anton's Gumbo Mix, 6 Grandma's Boysenberry Spread, 7 Uncle Bob's Organic Dried Pears, 8 Northwoods Cranberry Sauce, 9 Mishi Kobe Niku, 10 Ikura, 11 Queso Cabrales, and 12 Queso Manchego La Pastora. To the left of the results grid, a query editor window displays the following SQL code:

```
select productname from Products where ProductID in
(select ProductID from [Order Details] )
```

H4.1 sử dụng subqueries

4.2 Sử dụng SubQueries như mệnh đề

Chúng ta có thể sử dụng một subquery để thay thế cho một giá trị trong mệnh đề thay thế cho một giá trị trong mệnh đề SELECT, như là một phần của mệnh đề WHERE. Điều này sẽ được chỉ ra ví dụ sau đây.

Khi sử dụng các toán tử so sánh với sub query có một số giới hạn với việc trả lại số lượng dòng và cột của sub-query. Các giới hạn được liệt kê như bảng dưới.

Một cột	Nhiều cột
Một dòng	Sử dụng <code>=,>,<</code> và các toán tử so sánh khác.
Nhiều dòng	Sử dụng ANY, ALL, IN và EXISTS

4.3 Sử dụng SubQueries như biểu thức

Ta có thể sử dụng kết quả trả về của SubQuery như là 1 biểu thức

Ví dụ: Lấy số lượng mặt hàng thuộc loại 1

```
Select distinct CategoryID , (Select count(ProductID)
                           from Products
                           where CategoryID = 1
                           group by CategoryID ) as ProductQuantity
From Products
where CategoryID = 1
```

Một số nguyên tắc của SubQuery:

- Theo sau một toán tử so sánh có thể bao gồm một biểu thức hoặc cột (ngoại trừ **EXISTS...IN** trong câu lệnh **SELECT**).
- Nếu mệnh đề **WHERE** trong câu lệnh bao gồm tên các cột, sẽ phải phù hợp với các cột trong danh sách SubQuery.
- Kiểu dữ liệu **ntext, image, text** không thể sử dụng trong SubQuery.
- Bởi vì phải trả về một giá trị đơn, subquery không bao gồm mệnh đề **GROUP BY** và **HAVING**, từ khóa **ANY** hay **ALL**.
- Từ khóa **DISTINCT** không thể sử dụng với subquery.
- Mệnh đề **COMPUTE** và **INTO** không thể được dùng trong câu lệnh.
- Mệnh đề **ORDER BY** chỉ có thể dùng khi từ khóa **TOP** được nêu trong câu lệnh.
- View được tạo ra bởi Subquery không thể được cập nhật.

- Subquery với từ khóa **EXISTS**, theo quy ước, dấu * thay cho tên cột, subquery được tạo ra nhằm tạo sự tồn tại để kiểm tra sự tồn tại và trả về giá trị TRUE hay FALSE thay vì trả về dữ liệu.

Ví dụ : Liệt kê sản phẩm có ProductID=1, trong đó subquery dùng để tính tổng số lượng sản phẩm đó.

The screenshot shows a SQL query window titled "TRIDUNG-PC...\BaitapSubSuery.sql*". The query itself is:

```
SELECT ProductID , ProductName,
(
    SELECT SUM(Quantity) AS SaleAmount
    FROM [Order Details]
    WHERE ProductID = 1
    GROUP BY ProductID
) AS SumQuantity

FROM Products
WHERE ProductID = 1
```

Below the query window is a results grid:

ProductID	ProductName	SumQuantity
1	Chai	828

At the bottom of the interface, there are tabs for "Results" and "Messages", and status information: "TRIDUNG-PC\SQLEXPRESS (9.0 SP2)", "TRIDUNG-PC\TRIDUNG (52)", "Northwind", "00:00:00", and "1 row".

4. 2 Dùng Subquery tính tổng số lượng

Ví dụ : Liệt kê danh sách các nhà phân phối nhiều mặt hàng nhất

```
TRIDUNG-PC...\BaitapSubSuery.sql*
SELECT supplierID, Count(ProductID) AS MaxProductAmount
FROM Products
GROUP BY SupplierID
HAVING Count(ProductID) >= All
    (SELECT Count(ProductID)
     FROM Products
     GROUP BY SupplierID)
```

supplierID	MaxProductAmount
1	7
2	12

Results Messages

TRIDUNG-PC\SQLEXPRESS (9.0 SP2) TRIDUNG-PC\TRIDUNG (52) Northwind 00:00:00 2 rows

H 4.3 Liệt kê nhà phân phối

Ví dụ : Tính tổng số lượng các mặt hàng đã bán theo từng nhà cung cấp

```
TRIDUNG-PC...\BaitapSubSuery.sql*
SELECT T.SupplierID , SUM(T.SaleQuantity) AS SumSaleQuantity
FROM (
    SELECT SupplierID, sum(Quantity) AS SaleQuantity
    FROM Products p inner join [Order Details] od
    ON p.ProductID = od.ProductID
    GROUP BY p.ProductID, p.SupplierID
) AS T
GROUP BY T.SupplierID
```

SupplierID	SumSaleQuantity
1	2213
2	1735
3	1436
4	1134
5	1050

Results Messages

TRIDUNG-PC\SQLEXPRESS (9.0 SP2) TRIDUNG-PC\TRIDUNG (52) Northwind 00:00:00 29 rows

H 4.4 Tổng hàng đã bán theo nhà cung cấp

4.4 Sử dụng từ khóa EXISTS và NOT EXISTS

Khi một subquery có từ EXISTS, nó có chức năng kiểm tra sự tồn tại. Từ khóa EXISTS được sử dụng để kiểm tra sự tồn tại của các dòng trả về bởi subquery. Subquery lúc này không thực sự trả lại dữ liệu, mà nó trả về một giá trị TRUE hoặc FALSE.

Một subquery bao gồm từ EXISTS có cú pháp như sau:

WHERE (NOT) EXISTS (Subquery)

Ví dụ: giả sử chúng ta muốn biết chỉ những sản phẩm nào có mã 1 đã được đặt hàng. Chúng ta có thể sử dụng từ khóa EXISTS để kiểm tra nếu thông tin về sản phẩm tồn tại trong bảng Order Details.

Câu lệnh và kết quả của nó được chỉ ra trong hình 4.5

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery1.sql...E\Thanh (52)*" displays the following SQL code:

```
select productname from Products where
    exists (select ProductID from [Order Details] where CategoryID = 1 )
```

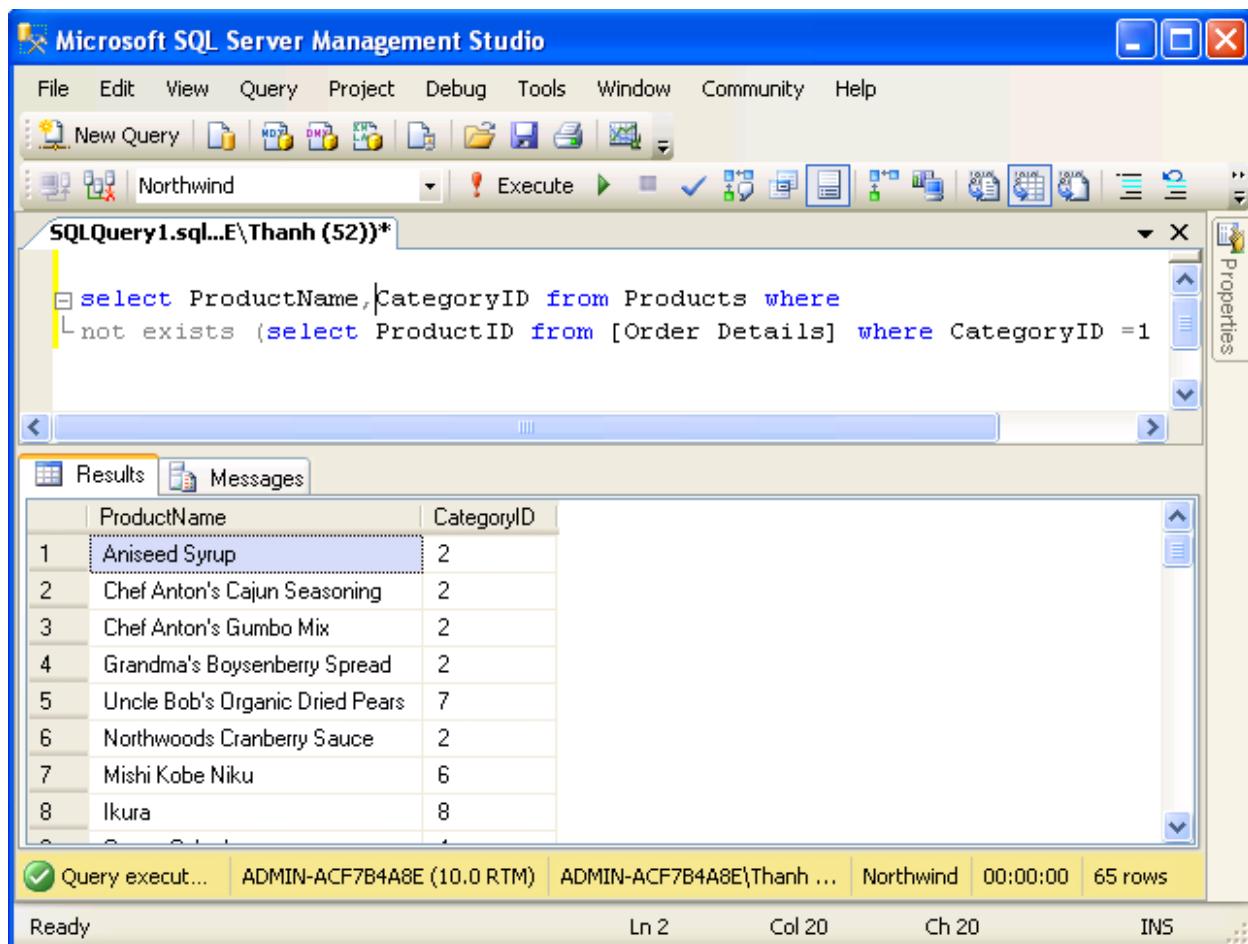
The results grid shows the following data:

	productname
1	Chai
2	Chang
3	Guaraná Fantástica
4	Sasquatch Ale
5	Steeleye Stout
6	Côte de Blaye
7	Chartreuse verte
8	Ipoh Coffee
...	...

At the bottom of the interface, the status bar shows: Ready, Ln 3, Col 17, Ch 17, INS.

H 4.5 Sử dụng mệnh đề EXISTS

Ví dụ: nếu chúng ta muốn chỉ ra những sản phẩm loại 1 mà chưa bao giờ đặt hàng. Chúng ta sẽ sử dụng từ khóa NOT EXISTS để truy vấn.



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid titled 'Results' showing data from a query. The query itself is:

```
select ProductName, CategoryID from Products where
not exists (select ProductID from [Order Details] where CategoryID = 1)
```

The results grid displays the following data:

	ProductName	CategoryID
1	Aniseed Syrup	2
2	Chef Anton's Cajun Seasoning	2
3	Chef Anton's Gumbo Mix	2
4	Grandma's Boysenberry Spread	2
5	Uncle Bob's Organic Dried Pears	7
6	Northwoods Cranberry Sauce	2
7	Mishi Kobe Niku	6
8	Ikura	8

At the bottom of the screen, the status bar shows: 'Query execut...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', '65 rows'. The bottom right corner of the status bar also shows 'Ready', 'Ln 2', 'Col 20', 'Ch 20', and 'INS'.

H 4.6 sử dụng mệnh đề NOT EXISTS

- ❖ Một subqueries có thể lồng nhiều subqueries (có thể lên tới 32 mức) nhưng khi đó sẽ không có hiệu suất thi hành như mong muốn.

Ví dụ: giả sử rằng chúng ta muốn tìm chi tiết những hóa đơn mà bao gồm sản phẩm được cung cấp từ thành phố London.(H4.7)

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql...E\Thanh (52)*' contains the following SQL code:

```
select OrderID, ProductID, UnitPrice from [Order Details] where
    ProductID in (select ProductID from Products where
        SupplierID = (select SupplierID from Suppliers where City = 'LONDON'))
```

The results pane displays a table with columns 'OrderID', 'ProductID', and 'UnitPrice'. The data is as follows:

	OrderID	ProductID	UnitPrice
1	10255	2	15.20
2	10258	2	15.20
3	10264	2	15.20
4	10285	1	14.40
5	10289	3	8.00
6	10294	1	14.40
7	10298	2	15.20
8	10317	1	14.40

The status bar at the bottom indicates: 'Query execut...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:01', '94 rows'.

H 4.7 Sử dụng nested **subqueries**

4.5 Sử dụng từ khóa DISTINCT

Từ khóa **DISTINCT** có tác dụng khử các dòng trùng nhau được trả về từ tập kết quả của một chuỗi **Select**. Trong trường hợp câu lệnh có từ khóa **DISTINCT** thì tất cả các dòng trùng nhau cũng hiển thị.

Ví dụ, chúng ta muốn hiện thị các ProductID không trùng nhau từ bản Order Details với Discount là 0. Câu lệnh và kết quả như trong hình 4.5.

Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Community Help

New Query | Databases | Logins | DMVs | XML | File | Folder | Script | Properties

SQLQuery1.sql...E\Thanh (52)*

```
select distinct productid from [Order Details]
  where Discount =0
```

Properties

Results Messages

productid
1 23
2 46
3 69
4 15
5 3
6 72
7 26
8 6
9 49

Query executed... ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... Northwind 00:00:00 77 rows

Ready Ln 1 Col 1 INS

H 4.8 Sử dụng mệnh đề DISTINCT.

Chương 5

Modifying Data

Kết thúc chương này các bạn có thể :

- Thực hành được thêm dữ liệu vào bảng sử dụng lệnh *INSERT*
- Thực hành được thêm dữ liệu vào bảng sử dụng lệnh *INSERT...SELECT*
- Thực hành được thêm dữ liệu vào bảng với từ khóa *DEFAULT*
- Thực hành được xóa dữ liệu trên bảng liệu sử dụng lệnh *DELETE*
- Thực hành được cập nhật dữ liệu trong bảng sử dụng lệnh *UPDATE*
- Trình bày khái niệm cơ bản về *TRANSACTION*

5.1 Thêm dữ liệu sử dụng **INSERT**

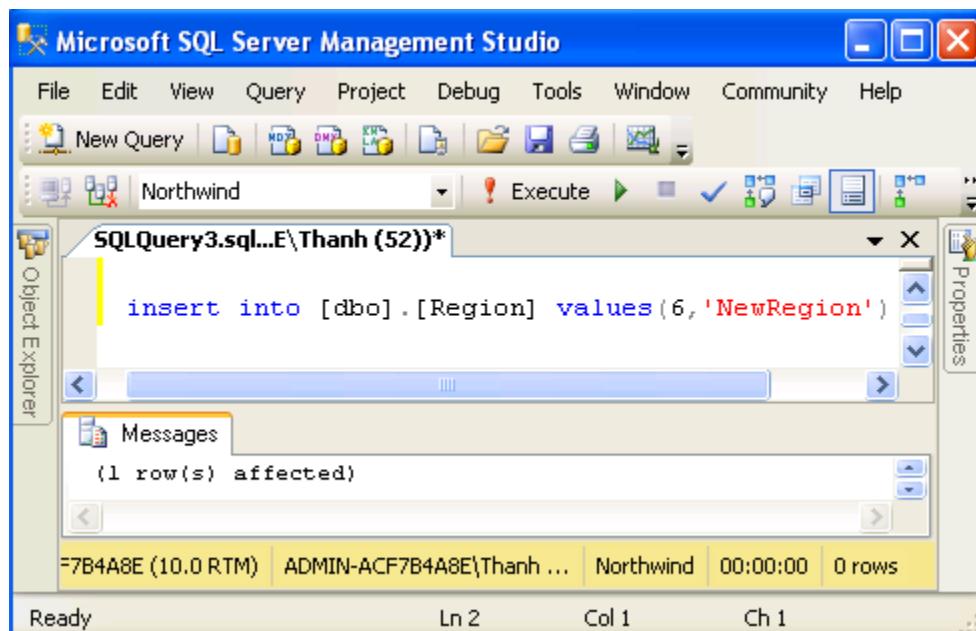
Đây là câu lệnh đơn giản nhất để thêm một dòng dữ liệu mới vào table sử dụng câu lệnh Insert Into .. Values, với câu lệnh này bạn phải chắc chắn rằng tất cả các giá trị thêm vào phải tương ứng với các cột trong bản cần thêm dữ liệu

Cú pháp

Insert Into [table_name] Values ("value 1","value 2", "value 3",...)

Ví dụ: chúng ta muốn thêm một giá trị mới vào bảng Region trong cơ sở dữ liệu Northwind ,
Kết quả như trong hình 5.1

Insert into [Region] Values (6,"NewRegion")



H 5.1 Thêm Mới

5.2 Thêm dữ liệu sử dụng INSERT...SELECT

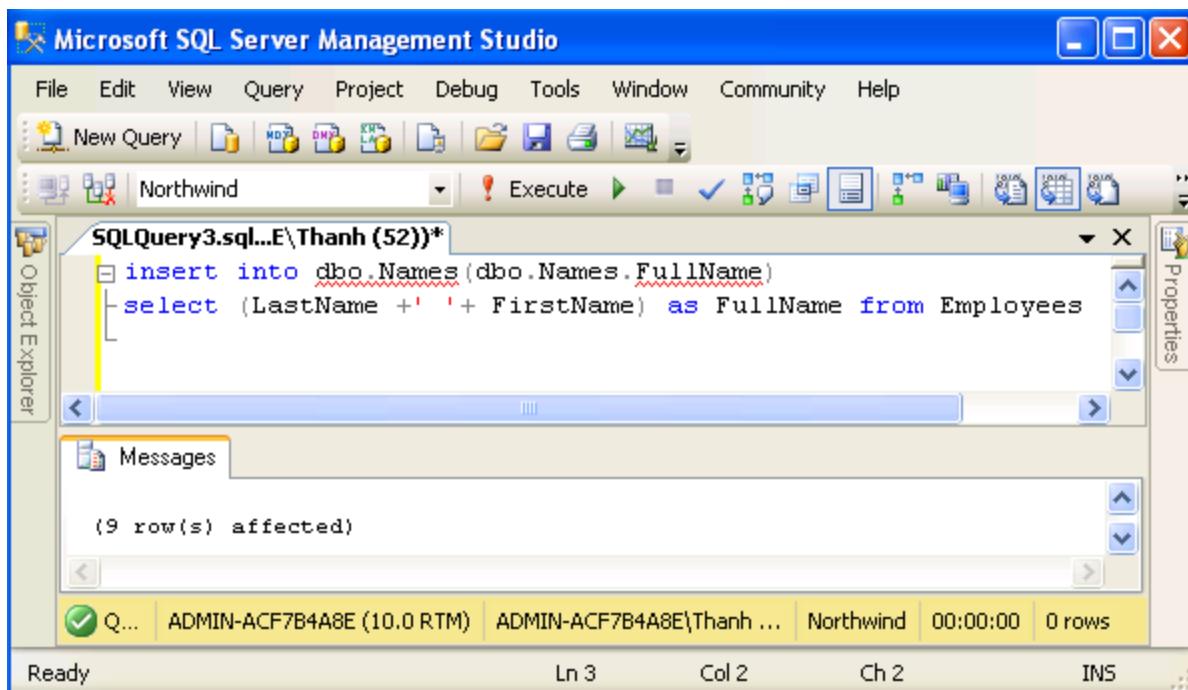
Với câu lệnh **Insert into .. values** chúng ta chỉ thêm được một dòng dữ liệu tại một thời điểm thêm mới dữ liệu. Để có thể thêm nhiều dòng dữ liệu tại cùng một thời điểm insert chúng ta có thể sử dụng câu lệnh **Insert ... select**

Cú pháp

INSERT INTO [table_name] ([column1], [column2], [column3], ...)

SELECT statement.

Ví dụ: chúng ta tạo mới một bảng Names có hai cột NameID ,FullName và lấy tất cả tên của các nhân viên trong bảng Employees thêm vào bản Names, chúng ta sẽ dùng câu lệnh Insert ... Select như hình bên dưới (H 5.2)



H 5.2 Insert...Select

5.3 Thêm dữ liệu với từ khóa DEFAULT

Để không phải nhập cùng một giá trị cho nhiều cột dữ liệu, chúng ta có thể gán cho một hay nhiều cột trong bảng với một giá trị mặc định là DEFAULT value.

Cú pháp:

INSERT INTO [table_name] DEFAULT VALUES

Ví dụ:

Chúng ta tạo một bảng tên History, gồm 3 trường ID, CreateDate, CreateName. Trong đó ta gán thuộc tính DEFAULT cho CreateDate

```
CREATE TABLE History
(
    ID int IDENTITY,
    CreateDate DateTime DEFAULT (GETDATE()),
    CreateName varchar(20) NULL
)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "Microsoft SQL Server Management Studio". The menu bar includes File, Edit, View, Project, Debug, Table Designer, Tools, Window, Community, and Help. The toolbar has various icons for New Query, Object Explorer, and other management tools. The main window displays the "ADMIN-ACF7B4A... - dbo.History" table definition. The "Table Designer" tab is selected. The table structure is shown in a grid:

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
CreateDate	datetime	<input checked="" type="checkbox"/>
CreateName	varchar(20)	<input checked="" type="checkbox"/>

Below the table grid, the "Column Properties" pane is open, showing the properties for the "ID" column under the "(General)" section:

Name	Value
(Name)	ID
Allow Nulls	No

H 5.3 Bảng History

Chúng ta Insert vào bảng vừa tạo một dòng dữ liệu theo cách thông thường. Ta sẽ dùng hàm GETDATE() để lấy ngày giờ hệ thống

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the Northwind database is selected. In the center pane, a query window titled 'SQLQuery5.sql...E\Thanh (55)*' contains the following SQL code:

```

INSERT INTO History (CreateDate, CreateName) VALUES (GETDATE(), 'Admin')
GO
SELECT ID, CreateDate, CreateName FROM History

```

The Results tab displays the output of the SELECT statement:

ID	CreateDate	CreateName
1	2009-09-15 07:54:50.933	Admin

At the bottom, the status bar shows: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '1 rows'.

H 5.4 Insert dữ liệu không dùng DEFAULT VALUES

Bây giờ ta thêm vào một dòng dữ liệu và sử dụng DEFAULT VALUES, các bạn vẫn có thể lấy giờ hệ thống.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the Northwind database is selected. In the center pane, a query window titled 'SQLQuery5.sql...E\Thanh (55)*' contains the following SQL code:

```

INSERT INTO History (CreateDate, CreateName) VALUES (GETDATE(), 'Admin')
GO
INSERT INTO History DEFAULT VALUES
GO
SELECT ID, CreateDate, CreateName FROM History

```

The Results tab displays the output of the SELECT statement:

ID	CreateDate	CreateName
1	2009-09-15 07:54:50.933	Admin
2	2009-09-15 08:06:11.210	NULL

At the bottom, the status bar shows: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '2 rows'.

H 5.5 Insert dữ liệu dùng DEFAULT VALUES

5.4 Xóa dữ liệu sử dụng câu lệnh DELETE

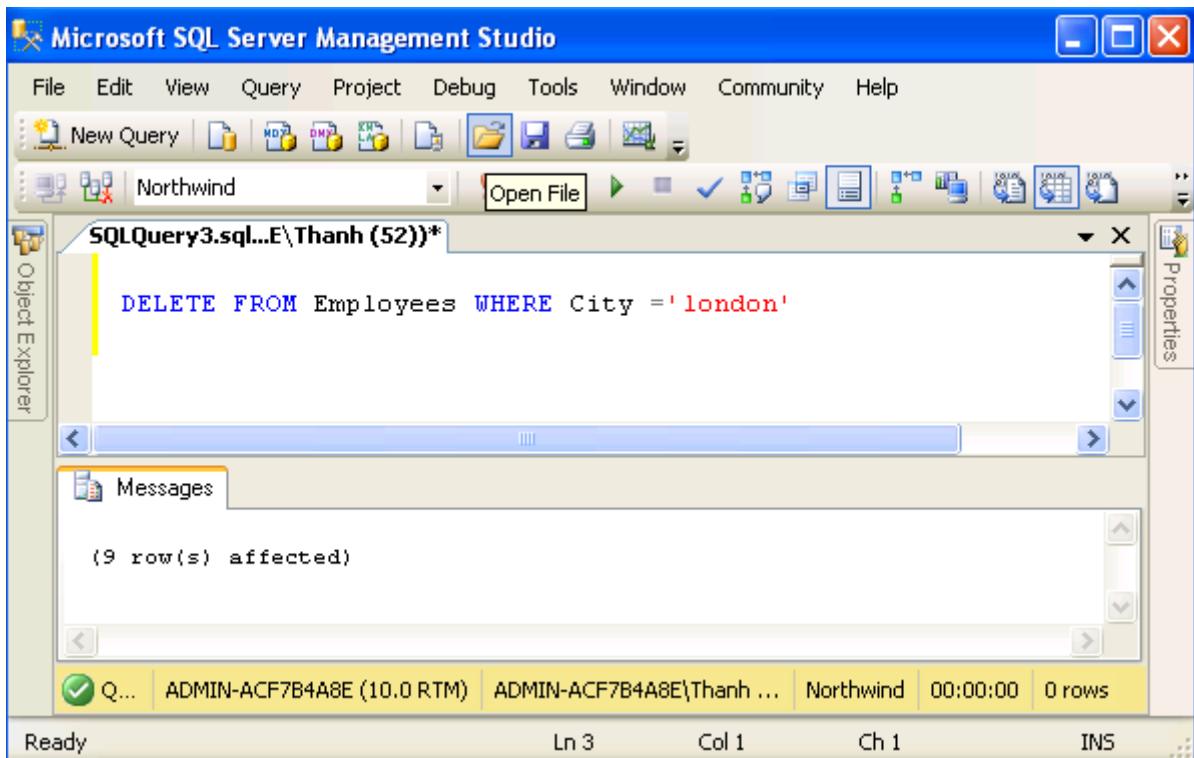
Câu lệnh Delete dùng để xóa tất cả các dòng ra khỏi bảng.

Cú pháp

DELETE [table_name] WHERE [column]=[value]

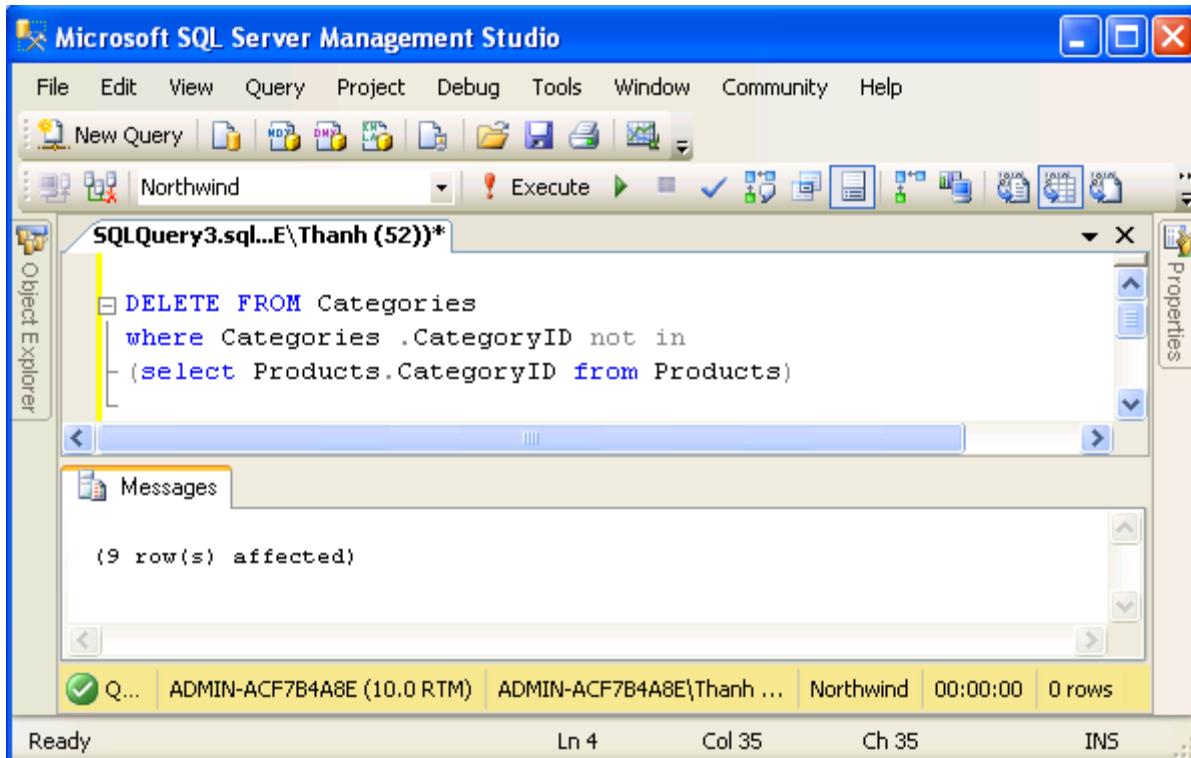
Lưu ý : Khi sử dụng lệnh Delete mà không có mệnh đề Where thì SQL Server sẽ xóa toàn bộ các dòng trong bảng .

Ví dụ : trong bảng Employee chúng ta muốn xóa những nhân viên nào ở thành phố LonDon thì ta có thể sử dụng câu lệnh như sau (H 5.6)



H 5.6 Xóa nhân viên

Ví dụ 2: xóa tất cả các loại sản phẩm chưa có sản phẩm nào (H 5.7)



H 5.7 Xóa loại sản phẩm

5.5 Cập nhật dữ liệu sử dụng UPDATE

Câu lệnh UPDATE được sử dụng để cập nhập/sửa đổi dữ liệu đã có trong bảng

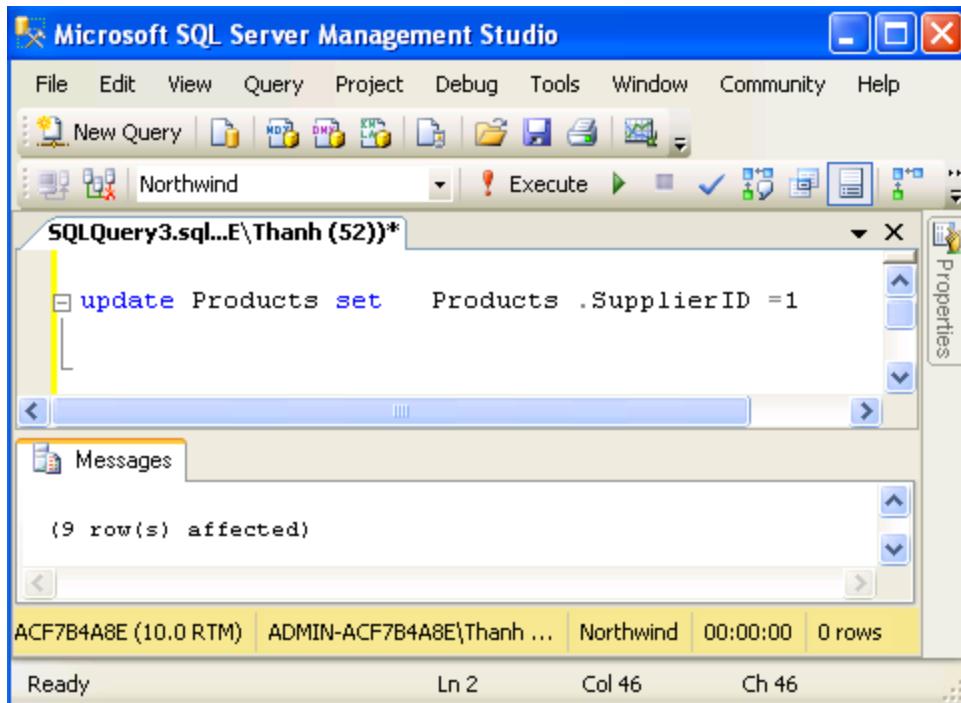
Cú pháp

Update [table_name] SET [column]=[new value]

WHERE [column]=[value]

Lưu ý : Khi sử dụng lệnh Update mà không có mệnh đề Where thì SQL Server sẽ cập nhật toàn bộ các dòng trong bảng có cùng giá trị của cột được update

Ví dụ : nếu như bạn muốn cập nhập lại tất cả các sản phẩm có cùng một nhà cung cấp , bạn có thể thực hiện như sau (H 5.8)



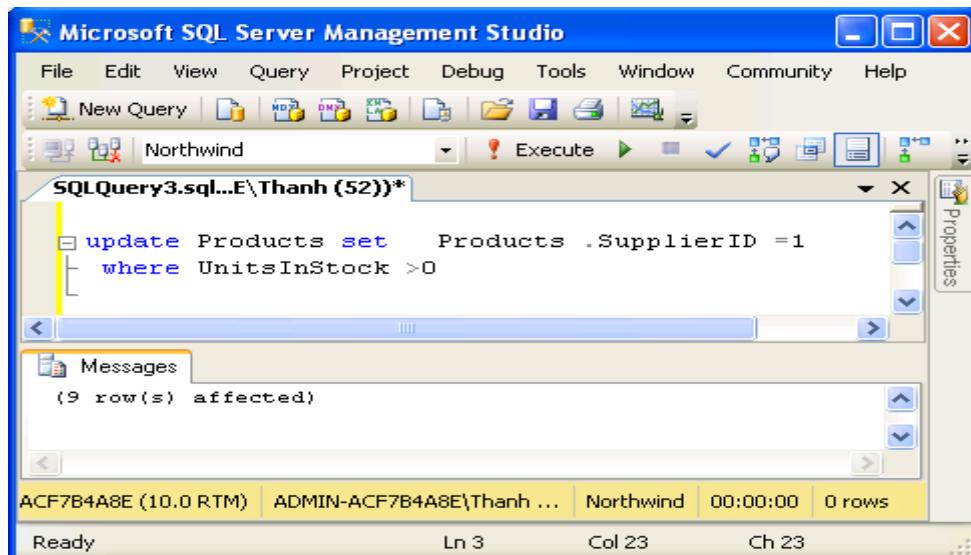
The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery3.sql...E\Thanh (52)*" contains the following SQL code:

```
update Products set Products .SupplierID =1
```

The "Messages" pane below the query window displays the result: "(9 row(s) affected)". The status bar at the bottom indicates "Ready", "Ln 2", "Col 46", and "Ch 46".

H 5.8 Cập nhập nhà cung cấp

Ví dụ : nếu như bạn muốn cập nhập lại tất cả các sản phẩm có cùng một nhà cung cấp và những sản phẩm này có số lượng hàng trong kho lớn hơn 0 , bạn có thể thực hiện như ví dụ trên và thêm vào mệnh đề **Where** như sau (H 5.9)



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery3.sql...E\Thanh (52)*" contains the following SQL code:

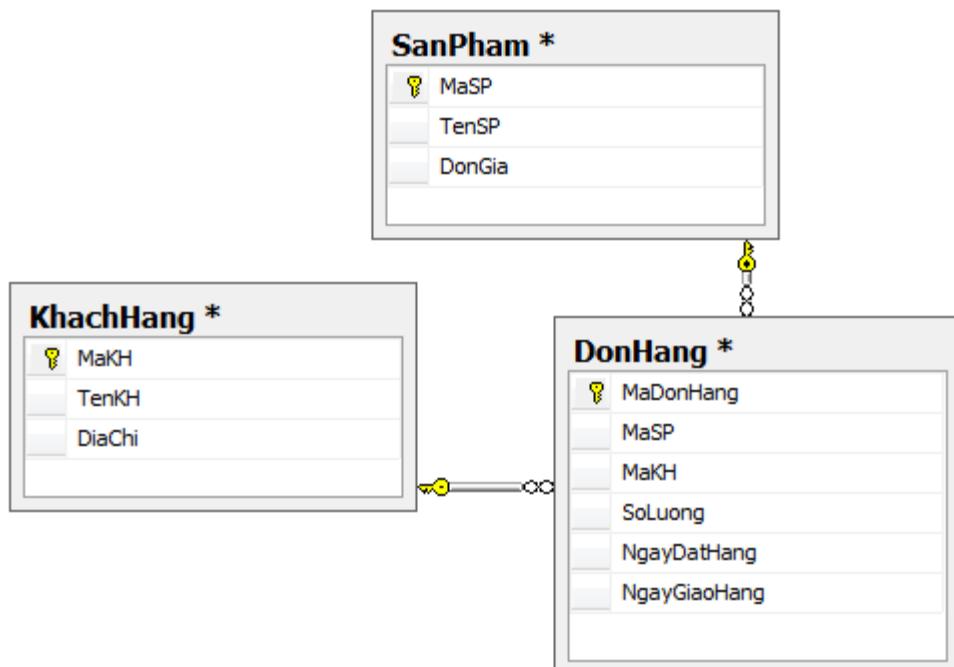
```
update Products set Products .SupplierID =1  
where UnitsInStock >0
```

The "Messages" pane displays the result: "(9 row(s) affected)". The status bar at the bottom indicates "Ready", "Ln 3", "Col 23", and "Ch 23".

H 5.9 Cập nhập sản phẩm với mệnh đề WHERE

Để áp dụng các câu lệnh trên chúng ta thực hành bài tập sau đây :

Đầu tiên , chúng ta tạo một database BanHang gồm bảng như sau: SanPham, KhachHang, DonHang.



H 5.10 Tạo CSDL BanHang

Thêm dữ liệu vào các bảng dùng câu lệnh INSERT

```
TRIDUNG-PC\SQL...g - dbo.SanPham TRIDUNG-PC\SQL... SQLQuery2.sql* TRIDUNG-PC\SQL...ng - Diagram_0* 
INSERT INTO KhachHang (TenKH,DiaChi) VALUES ('Nguyen Van A','123 To Hien Thanh')
go
INSERT INTO KhachHang (TenKH,DiaChi) VALUES ('Tran Van B','456 Nguyen Dinh Chieu')
go
INSERT INTO KhachHang (TenKH,DiaChi) VALUES ('Le Thi C','789 Su Van Hanh')
go
INSERT INTO DonHang (MaSP,MaKH,SoLuong,NgayDatHang,NgayGiaoHang)
VALUES (1,1,5,1/9/2009,3/9/2009)
go
```

H 5.12 Thêm dữ liệu bằng INSERT

TRIDUNG-PC\SQLLE... - dbo.SanPham* TRIDUNG-PC\SQLLE... SQLQuery2.sql*

```
VALUES (1, 2, 3)
go
```

1	1	coffee	12000
2	2	tea	10000
3	3	milk	15000
4	4	beer	20000

	MaKH	TenKH	DiaChi
1	1	Nguyen Van A	123 To Hien Thanh
2	2	Tran Van B	456 Nguyen Dinh Chieu
3	3	Le Thi C	789 Su Van Hanh

	MaDonHang	MaSP	MaKH	SoLuong
1	1	1	1	5
2	2	1	2	3
3	3	2	1	5
4	4	3	4	2
5	5	4	3	5

H 5.13 Dữ liệu sau khi thêm

Dùng UPDATE để thay đổi giá trị các cột MaSP, SoLuong của đơn hàng thứ nhất.

TRIDUNG-PC\SQLLE... - dbo.SanPham* TRIDUNG-PC\SQLLE... SQLQuery2.sql*

```
go
UPDATE DonHang
SET
    MaSP=4, SoLuong=10
WHERE MaDonHang=1
```

1	1	4	10
2	2	1	3
3	3	2	5
4	4	3	2
5	5	4	5

H 5.14 Cập nhật dữ liệu của MaDonHang=1

Ta xóa thông tin khách hàng có mã là 3 bằng lệnh DELETE

Để xóa dữ liệu của các bảng có quan hệ với nhau, chúng ta phải xóa từ bảng con trước (DonHang), sau đó mới xóa trong bảng cha (KhachHang).

```
TRIDUNG-PC\SQL... SQLQuery2.sql* [TRIDUNG-PC\SQL...ng - Diagram_0*]
DELETE FROM DonHang WHERE MaDonHang=5
GO
DELETE FROM KhachHang WHERE MaKH=3

Messages
(1 row(s) affected)
```

H 5.15 Xóa dữ liệu 2 bảng liên quan

5.6 Sử dụng TRANSACTION

Transaction là một tập hợp các câu lệnh được kết hợp lại để thực một công việc. Transaction được dùng để đảm bảo rằng các câu lệnh được thực thi thành công hoặc thất bại.

Có 3 phần chính trong một Transaction:

BEGIN TRANSACTION: bắt đầu một Transaction. Dữ liệu sẽ không được cập nhật đến CSDL cho đến khi COMMIT TRANSACTION được gọi.

COMMIT TRANSACTION: được gọi khi tất cả các câu lệnh ngay sau BEGIN TRANSACTION thực hiện thành công, dữ liệu sẽ được ghi xuống CSDL.

ROLLBACK TRANSACTION: trả tất cả dữ liệu về trạng thái ban đầu trước khi BEGIN TRANSACTION được gọi.

Cú Pháp:

BEGIN TRANSACTION

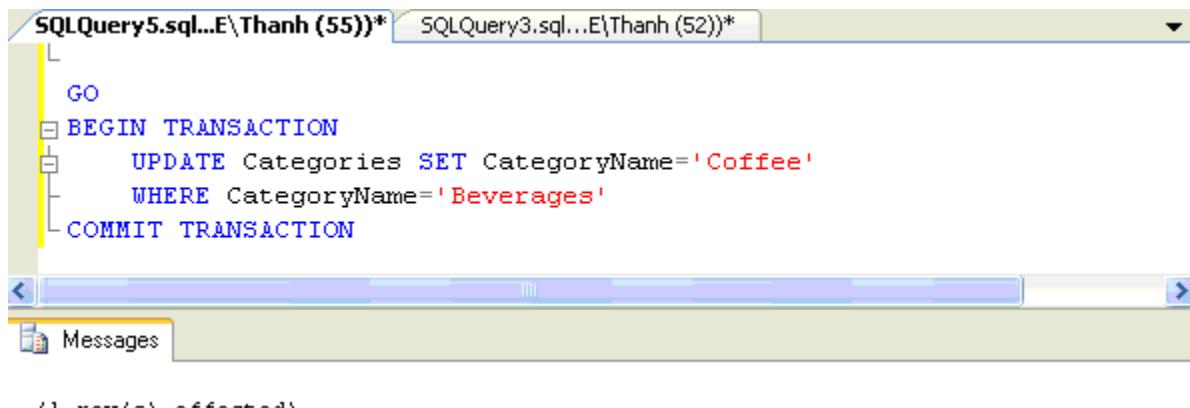
SQL Statements

COMMIT | ROLLBACK TRANSACTION

- Chúng ta có thể sử dụng TRY...CATCH hoặc IF cùng với TRANSACTION

BEGIN TRY**BEGIN TRAN****-- Code for your transaction****COMMIT TRAN****END TRY****BEGIN CATCH****-- output an error message****ROLLBACK TRAN****END CATCH**

Ví dụ: Chúng ta sử dụng bảng Categories trong CSDL Northwind để tạo 1 Transaction Update đơn giản.



The screenshot shows a SQL Server Management Studio window with two tabs: 'SQLQuery5.sql...E\Thanh (55)*' and 'SQLQuery3.sql...E\Thanh (52)*'. The code in the visible tab is:

```
GO
BEGIN TRANSACTION
    UPDATE Categories SET CategoryName='Coffee'
    WHERE CategoryName='Beverages'
COMMIT TRANSACTION
```

In the 'Messages' pane at the bottom, the output is:

```
(1 row(s) affected)
```

H 5.16 TRANSACTION UPDATE CategoryName

Sau khi Update, sản phẩm Beverages đã được đổi thành Coffee , kết quả như hình sau .

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid titled 'Results' showing data from the 'Categories' table. The columns are 'CategoryID' and 'CategoryName'. The data is as follows:

CategoryID	CategoryName
1	Coffee
2	Condiments
3	Confections
4	Dairy Products
5	Grains/Cereals
6	Meat/Poultry
7	Produce

At the top of the screen, there is a menu bar with File, Edit, View, Query, Project, Debug, Tools, Window, Community, Help. Below the menu bar is a toolbar with various icons. The title bar says 'Microsoft SQL Server Management Studio'. The status bar at the bottom shows 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '8 rows'.

H 5.17 Truy vấn bảng Categories

Chương 6

Giới Thiệu Các Thành Phần Khác Trong SQL Server 2008

Kết thúc chương này các bạn có thể :

- Mô tả được khái niệm và sử dụng được Trigger để ràng buộc dữ liệu
- Mô tả được khái niệm và sử dụng được Store Procedure
- Trình bày khái niệm cơ bản và sử dụng Function và User-Defined Function
- Trình bày khái niệm cơ bản và sử dụng View

6.1 Giới thiệu về STORE PROCEDURE

• STORE PROCEDURE

Là một nhóm các câu lệnh T-SQL đã được biên dịch từ trước(pre-compiled). Một Store Procedure có thể không chứa hoặc chứa nhiều tham số truyền vào; đồng thời có thể trả về một giá trị, một bảng hoặc không trả về 1 giá trị nào đó. Sử dụng Store Procedure độ thực thi nhanh hơn , dễ thay đổi , nâng cấp và bảo mật,....

Có 2 dạng Store Procedure:

- System Store Procedure
- User-defined Store Procedure

Các System Store Procedure có sẵn khi chúng ta cài đặt SQL Server. Tất cả các System Store Procedure đều bắt đầu bằng tiền tố `sp_`

6.1.1. Một số nhóm System Store Procedure:

- **Database Engine stored procedures:** bao gồm các câu lệnh queries lấy thông tin của SQL Server và CSDL.

Ví dụ: `sp_helpdb` lấy thông tin của tất cả các CSDL hiện có trên Server

- **Database Mail stored procedures:** dùng cho các thao tác về e-mail trong CSDL (*sp_send_dbmail*)
- **Security stored procedures:** dùng cho mục đích bảo mật như thêm/ xóa User, đăng nhập... (*sp_addlogin*)
- **XML stored procedures:** dùng quản lý các tài liệu XML (*sp_xml_preparedocument*).

6.1.2 User-defined Store Procedure

Cú pháp:

```
CREATE PROC | PROCEDURE <store name>
    @variablename datatype
    @variablename datatype
    ...
AS
    SQL Statement
```

Ví dụ: chúng ta tạo một Store Procedure tên GetEmployees để lấy thông tin các nhân viên theo thành phố từ bảng Employees theo tham số truyền vào là @City.

```
CREATE PROC GetEmployees
    @City nvarchar(15)
AS
    SELECT EmployeeID, LastName, FirstName, City FROM Employees
    WHERE City=@City
```

Sau đó, để gọi Store Procedure vừa tạo, ta dùng lệnh **EXEC <store name>**

	EmployeeID	LastName	FirstName	City
1	5	Buchanan	Steven	London
2	6	Suyama	Michael	London

H 6.2 Gọi SP GetEmployees

Để thay đổi tên hay nội dung một Store Procedure ta dùng **ALTER** thay cho **CREATE**

```

GO
ALTER PROC GetEmployees
    @City nvarchar(15)
AS
    SELECT EmployeeID, LastName, FirstName, City FROM Employees
    WHERE City=@City
  
```

Các ví dụ minh họa thực hành về Store Procedure: Tạo các store procedure sử dụng CSDL Northwind

- Tạo store procedure lấy danh sách tất cả các mặt hàng

```

/*
1. Viet 1 SP lay danh sach tat ca cac mat hang
*/
create proc spGetAllProducts
as
SELECT ProductID, ProductName, UnitPrice
FROM Products
  
```

- Tạo store procedure lấy danh sách các mặt hàng có UnitPrice >= giá trị bất kỳ

```
create proc spGetAllProductsByUnitPrice
(
    @Price money
)
as
SELECT ProductID, ProductName, UnitPrice
FROM Products
where UnitPrice >=@Price
order by UnitPrice desc
```
- Tạo store procedure cập nhật UnitPirce với ProductID và UnitPrice do người dùng nhập vào

```
create proc spUpdateUnitPrice
(
    @ProID int,
    @NewPrice money
)
As
update Products set UnitPrice = @NewPrice
where ProductID = @ProID
```

6.2 Giới thiệu về Function và User-Defined Function

Function được dùng tương tự như Store Procedure giúp tối ưu hoạt động của CSDL; giảm thời gian viết lại các lệnh SQL thường dùng.Ta có thể truyền vào các tham số cho Function.

Tuy nhiên, Function có những đặc điểm khác với Store Procedure:

- Function luôn trả về một giá trị
- Function phải có tham số kèm theo khi gọi, ngoại trừ 1 số function như GETDATE(), PI()...
- Function có thể được gọi bên trong câu lệnh SELECT

Những Function sẵn có khi ta cài SQL Server gọi là Built-in Function.

➤ Một số ví dụ về Built-in Function:

Ví dụ 1

Tính tổng số lượng từng mặt hàng (Quantity) có mã đơn hàng (OrderID) là 10248.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query results grid titled 'Results' showing the following data:

	OrderID	ProductID	UnitPrice	TotalQuantity
1	10248	11	14.00	12
2	10248	42	9.80	10
3	10248	72	34.80	5

At the bottom of the results grid, it says '3 rows'. Below the results grid, the status bar displays: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '3 rows'.

H 6.7 Tính tổng bằng Sum()

Ví Dụ 2: Đôi khi, chúng ta muốn có kết quả truy vấn mà dữ liệu trả về được kết hợp từ nhiều cột trong bảng; khi đó ta có thể dùng CONCAT(). Mỗi hệ CSDL cung cấp 1 cách khác nhau

- MySQL: CONCAT()
- Oracle: CONCAT(), ||
- SQL Server: +

Trong ví dụ này chúng ta chỉ áp dụng CONCAT cho SQL Server. Chú ý : Ta chỉ có thể kết hợp các cột có cùng kiểu dữ liệu.

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "Microsoft SQL Server Management Studio". The menu bar includes File, Edit, View, Query, Project, Debug, Tools, Window, Community, and Help. The toolbar contains icons for New Query, Import, Export, and various database management tools. The object explorer on the left shows a connection to "Northwind". The main query editor window has a tab titled "SQLQuery7.sql...E\Thanh (51)*" containing the following T-SQL code:

```
go
SELECT CompanyName, ContactName FROM Customers WHERE CustomerID='ALFKI'
```

The results pane below shows the output of the query:

	CompanyName	ContactName
1	Alfreds Futterkiste	Maria Anders

At the bottom, a status bar indicates "Query executed..." and "1 rows".

H 6.8 Truy vấn không dùng CONCAT

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "Microsoft SQL Server Management Studio". The menu bar includes File, Edit, View, Query, Project, Debug, Tools, Window, Community, and Help. Below the menu is a toolbar with various icons for tasks like New Query, Save, and Execute. A connection dropdown shows "Northwind". The main area displays a query window titled "SQLQuery7.sql...E\Thanh (51)*" containing the following T-SQL code:

```
go
SELECT CompanyName + ' ' + ContactName FROM Customers
WHERE CustomerID='ALFKI'
```

The results pane below shows one row of data:

	(No column name)
1	Alfreds Futterkiste Maria Anders

A status bar at the bottom indicates "Query executed..." and "1 rows".

H 6.9 Truy vấn dùng CONCAT

Ví dụ 3: Hàm Built-in **STUFF()**Cú pháp:**STUFF (character_expression , start , length ,character_expression)**

Chúng ta tạo 2 bảng tên ContactPersons và MyCustomers có dữ liệu như hình bên dưới:

	ContactPersonID	CustomerID	ContactPersonName
1	0001	0001	ContactPerson1.1
2	0002	0001	ContactPerson1.2
3	0003	0001	ContactPerson1.3
4	0004	0002	ContactPerson2.1
5	0005	0002	ContactPerson2.2

Query executed successfully. ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... master 00:00:00 5 rows

H 6.10 Bảng ContactPersons

	CustomerID	CustomerName
1	0001	mycustomer1
2	0002	mycustomer2

Query executed successfully. ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... master 00:00:00 2 rows

H 6.11 Bảng MyCustomers

Đề thu được kết quả như hình sau

	CustomerID	CustomerName	ContactPersons
1	0001	mycustomer1	ContactPerson1.1,ContactPerson1.2,ContactPerson1.3
2	0002	mycustomer2	ContactPerson2.1,ContactPerson2.2

H 6.12 Kết quả truy vấn

Chúng ta sử dụng hàm STUFF

```
SELECT c.CustomerID, c.CustomerName,
       (SELECT DISTINCT STUFF (( SELECT ',' + ContactPersonName
      FROM dbo.ContactPersons
      WHERE dbo.ContactPersons.CustomerID= c.CustomerID
      FOR XML PATH('')),1,1,'') AS ContactPersonName FROM ContactPersons)
  AS ContactPersons
FROM MyCustomers c
```

H 6.13 Gọi hàm STUFF()

Ví dụ 4: Hàm UNPIVOT()

Chúng ta tạo một bảng tên **Clients**

```
create table Clients
(
    clientID int primary key,
    clientName varchar(100),
    contact1 int,
    contact2 int,
    contact3 int,
    contact4 int
)
```

Kết quả :

	clientID	clientName	contact1	contact2	contact3	contact4
1	1	ABC Corp	1	34	2	NULL
2	2	DEF Foundation	6	2	8	9
3	3	GHI Inc.	5	9	NULL	NULL
4	4	XYZ Industries	24	NULL	6	NULL

H 6.14 Dữ liệu Bảng Clients

Với cách tạo bảng như trên, chúng ta khó có thể đếm tất cả các lần liên hệ của mỗi khách hàng theo một cột với cách truy vấn thông thường. Chúng ta có nhiều cách để thực hiện

Truy vấn dùng UNION ALL

```
select clientID, contact1 as ContactID
from clients
where contact1 is not null
union all
select clientID, contact2 as ContactID
from clients
where contact2 is not null
union all
select clientID, contact3 as ContactID
from clients
where contact3 is not null
union all
select clientID, contact4 as ContactID
from clients
where contact4 is not null
```

clientID	ContactID
1	1
2	6
3	5
4	24
5	34
6	2
7	9
8	2
9	8
10	6

Query execute... ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... Northwind 00:00:00 11 rows

H 6.15 Kết quả truy vấn dùng Union All

Tuy nhiên cách làm trên tương đối dài và khó hiểu, do đó SQL Server 2008 cung cấp cho ta một hàm Built-in đơn giản và dễ sử dụng là **UNPIVOT()**.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window titled "SQLQuery11.sql...E\Thanh (51)*". The query is:

```

GO
SELECT clientID, Contact.ContactID
FROM clients c
UNPIVOT (ContactID FOR ContactNumber
IN(contact1,contact2,contact3,contact4))
AS Contact
  
```

The results pane displays a table with four rows:

	clientID	ContactID
1	1	1
2	1	34
3	1	2
4	2	6

At the bottom of the results pane, it says "11 rows". Below the results pane, the status bar shows: "Query execute..." ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... Northwind 00:00:00 11 rows".

H 6.16 Gọi hàm Unpivot()

Lưu ý:

Trước khi sử dụng hàm **Unpivot()**, bắt buộc ta phải gọi 1 Store hệ thống là **sp_dbcmptlevel**

```
EXEC sp_dbcmptlevel Northwind, 90
```

Với cú pháp:

```
Sp_dbcmptlevel [@dbname] [ @new_cmptlevel]
```

@dbname: tên CSDL

@new_cmptlevel = 80 (SQL Server 2000)

= 90 (SQL Server 2005)

= 100 (SQL Server 2008)

- **User Defined Function:** ta có thể tự viết các Function cho riêng mình nếu như các Built-in Function không phù hợp với yêu cầu.

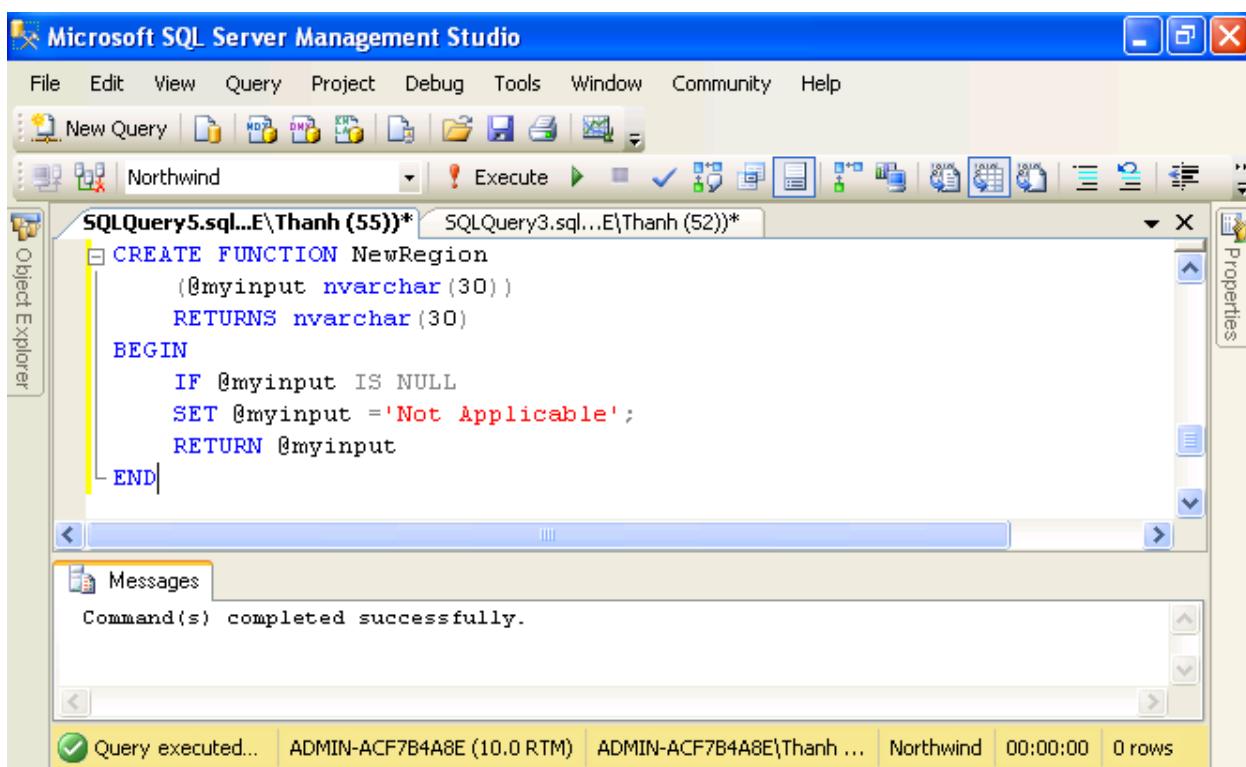
Có 2 dạng User Defined Function: **Scalar Function** và **Table-valued Function**

- **Scalar Function:** luôn trả về một giá trị cụ thể.

Cú pháp:

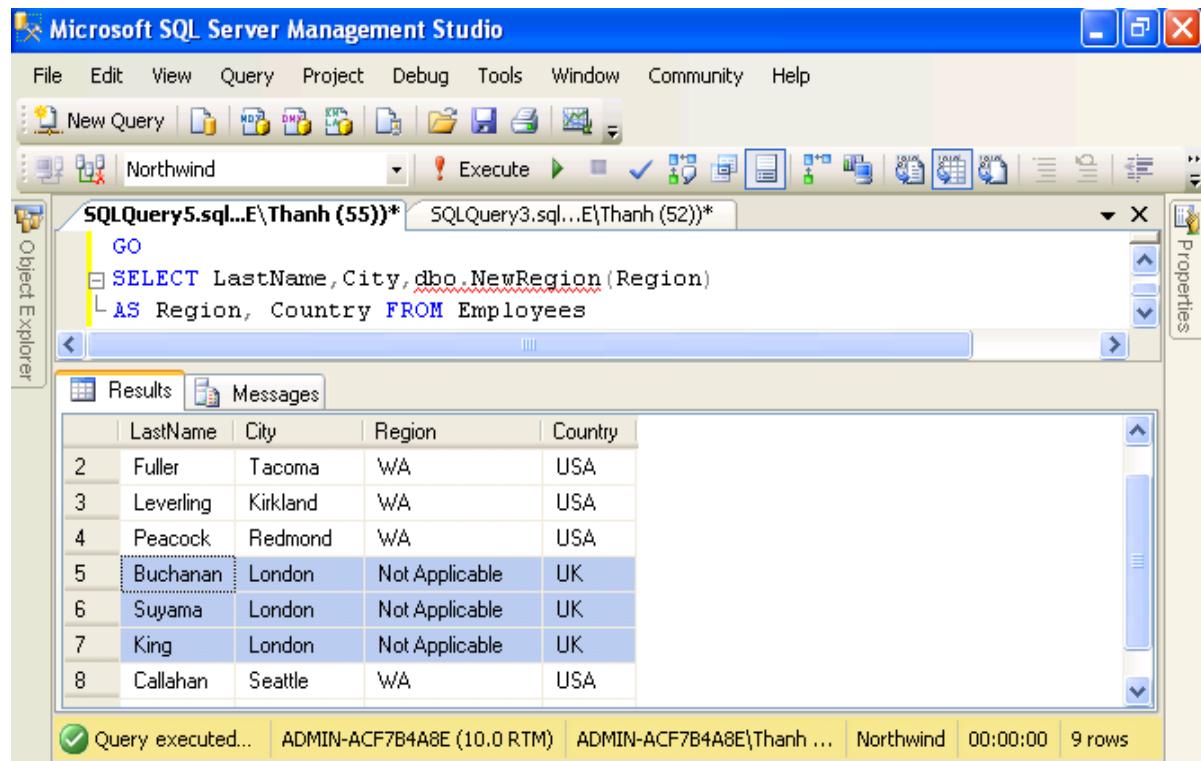
```
CREATE FUNCTION FunctionName
    (@parameter datatype)
    RETURNS type
AS
BEGIN
    SQL Statements
    RETURN Scalar expression
END
```

Ví dụ: chúng ta sẽ tạo 1 hàm dùng thay thế các giá trị NULL thành NOT APPLICABLE



H 6.17 Tạo hàm NewRegion

Sau đó ta gọi hàm NewRegion vừa tạo trong câu lệnh SELECT từ bảng Employees.



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid titled 'Results' showing data from a query. The columns are labeled 'LastName', 'City', 'Region', and 'Country'. The data consists of 9 rows:

	Last Name	City	Region	Country
2	Fuller	Tacoma	WA	USA
3	Leverling	Kirkland	WA	USA
4	Peacock	Redmond	WA	USA
5	Buchanan	London	Not Applicable	UK
6	Suyama	London	Not Applicable	UK
7	King	London	Not Applicable	UK
8	Callahan	Seattle	WA	USA

At the bottom of the interface, a status bar displays: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '9 rows'.

H 6.18 Gọi hàm NewRegion

Table-valued Function: trả về kiểu giá trị: kiểu bảng dữ liệu giống như View (sẽ được trình bày bên dưới).

Cú pháp:

```
CREATE FUNCTION FunctionName
    (@parameter datatype)
    RETURNS Table
AS
    RETURN Statement
```

Ví dụ: tạo 1 function liệt kê các Nhân viên có City=@City

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a node for 'SQLQuery5.sql...E\Thanh (55)*'. The main query window contains the following T-SQL code:

```
GO
CREATE FUNCTION GetEmployeeByCity
    (@City nvarchar(15))
RETURNS TABLE
AS
RETURN
    (SELECT LastName, FirstName, City FROM Employees WHERE City=@City)
```

The 'Messages' pane at the bottom shows the message: 'Command(s) completed successfully.'

At the bottom of the interface, the status bar displays: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '0 rows'.

H 6.19 Tạo function GetEmployeeByCity

Gọi hàm GetEmployeeByCity vừa tạo với tham số @City='London'

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a node for 'SQLQuery5.sql...E\Thanh (55)*'. The main query window contains the following T-SQL code:

```
GO
SELECT * FROM GetEmployeeByCity('London')
```

The 'Results' pane at the bottom displays the following data:

	LastName	FirstName	City
1	Buchanan	Steven	London
2	Suyama	Michael	London
3	King	Robert	London
4	Dodsworth	Anne	London

At the bottom of the interface, the status bar displays: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '4 rows'.

H 6.20 Gọi hàm GetEmployeeByCity

Tuy nhiên ta có thể thu được kết quả tương tự bằng cách kết hợp giữa 1 hàm Built-in là COALESCE() và 1 hàm User Defined.

Tạo hàm User Defined tên MySubstring()

```
SQLQuery7.sql..E\Thanh (51)*
CREATE FUNCTION MySubstring (@CustomerID int)
RETURNS nvarchar(1000)
AS
BEGIN
DECLARE @str nvarchar(1000)
SELECT @str= COALESCE(@str + ' , ' , '') + rtrim( [ContactPersonName])
FROM ContactPersons ct WHERE ct.CustomerID =@CustomerID
RETURN @str
END
```

H 6.21 Hàm MySubString

Sau khi gọi hàm MySubString trong câu lệnh SELECT ta cũng thu được kết quả tương tự ví dụ ở phần hàm Built-in

```
Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Community Help
New Query Northwind Execute
SQLQuery7.sql..E\Thanh (51)*
GO
SELECT c.CustomerID ,c.CustomerName ,dbo.MySubstring(c.CustomerID )
AS ContactPerson FROM MyCustomers AS c
```

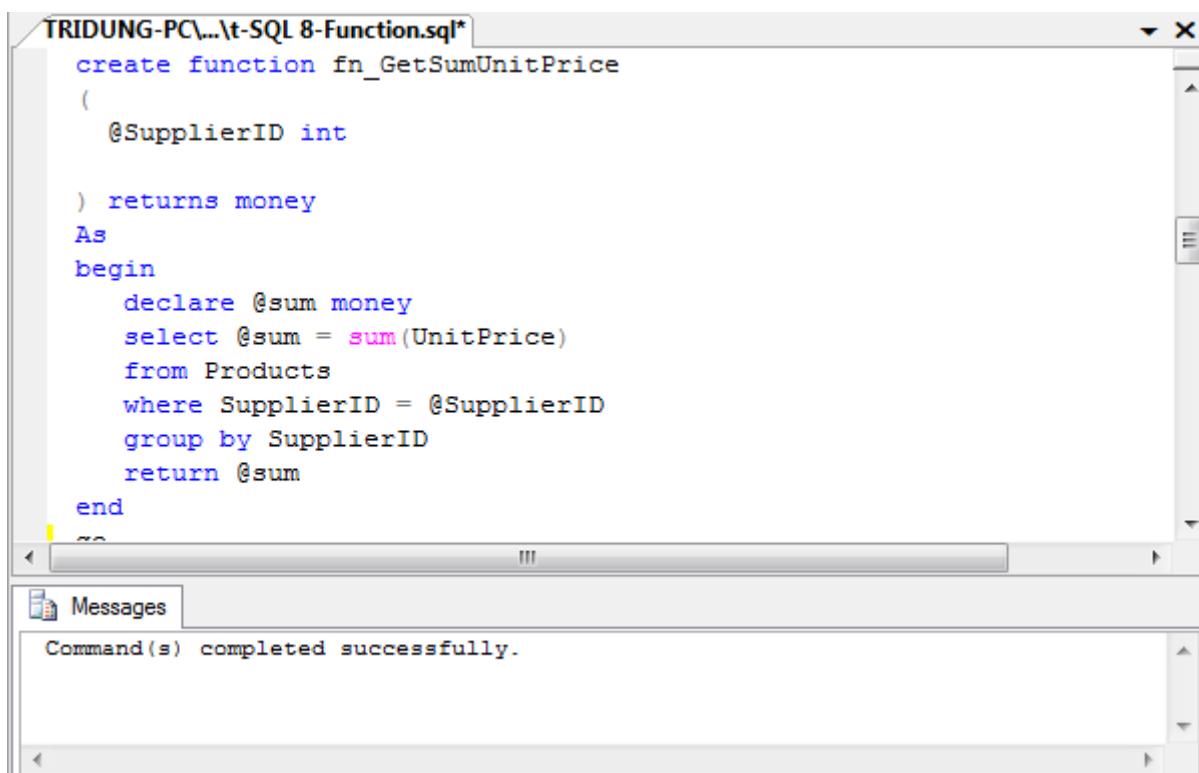
CustomerID	CustomerName	ContactPerson
1	mycustomer1	ContactPerson1.1 , ContactPerson1.2 , ContactPerson1.3
2	mycustomer2	ContactPerson2.1 , ContactPerson2.2

Query executed... ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... Northwind 00:00:00 2 rows

H 6.22 Gọi hàm MySubString()

- Các ví dụ minh họa

Viết hàm trả về tổng UnitPrice của các mặt hàng thuộc nhà cung cấp có mã số @SupplierID



The screenshot shows the SQL Server Management Studio interface. In the top pane, a script named 'TRIDUNG-PC\...\t-SQL 8-Function.sql' is open, containing the following T-SQL code:

```
create function fn_GetSumUnitPrice
(
    @SupplierID int

) returns money
As
begin
    declare @sum money
    select @sum = sum(UnitPrice)
    from Products
    where SupplierID = @SupplierID
    group by SupplierID
    return @sum
end
```

In the bottom pane, the 'Messages' tab is selected, showing the message: 'Command(s) completed successfully.'

H6.23 fn_GetSumUnitPrice

Gọi hàm và xem kết quả:

```
select dbo.fn_GetSumUnitPrice(2) as SumUnitPrice
```

	SumUnitPrice
1	81,40

Viết hàm tính tổng tiền của một khách hàng

The screenshot shows a SQL script window titled 'TRIDUNG-PC\...\t-SQL 8-Function.sql*'. The script creates a function named 'fn_GetTotalAmount' that takes a parameter '@CusID' and returns a float. It uses a SELECT statement with a GROUP BY clause to calculate the total amount for a given customer ID by joining the 'Order Details' and 'Orders' tables.

```
create function fn_GetTotalAmount
(
    @CusID varchar(30)

) returns float
As
begin
    declare @TotalAmount float
    select @TotalAmount = sum(UnitPrice*Quantity)
    from [Order Details] od inner join Orders o
    on od.OrderID = o.OrderId
    group by CustomerID
    having CustomerID = @CusID |

    return @TotalAmount
end
```

Below the script window is a 'Messages' pane showing the message: 'Command(s) completed successfully.'

H 6.24 Hàm fn_GetTotalAmount

Kết quả

The screenshot shows a SQL query window with the same title 'TRIDUNG-PC\...\t-SQL 8-Function.sql*'. The query is 'SELECT dbo.fn_GetTotalAmount ('QUICK')'. Below the query window is a 'Results' pane displaying the output: a single row with one column containing the value '117483,39'.

(No column name)
1 117483,39

H 6.24 gọi hàm fn_GetTotalAmount

6.3 Giới thiệu về VIEW

View là một đối tượng cho phép ta có thể xem chính xác những dữ liệu cần thiết, không chỉ một vài trường trong 1 bảng mà còn có thể từ nhiều trường từ nhiều bảng khác nhau.

Có 2 cách để tạo View:

❖ Cách 1: tạo View trong View Designer

Khởi động SQL Server Management Studio (SSMS)

- Kết nối với Server
- Chọn CSDL cần làm việc
- Click phải vào **Views Container** -> chọn **New View**
- Trong hộp thoại **Add Table** -> chọn các bảng cần thiết -> **Add**
- Click chọn các trường trong bảng vừa thêm vào
- Chọn **Execute SQL** để xem kết quả
- Ctrl + S để lưu View

❖ Cách 2: tạo View bằng T-SQL

Cú Pháp:

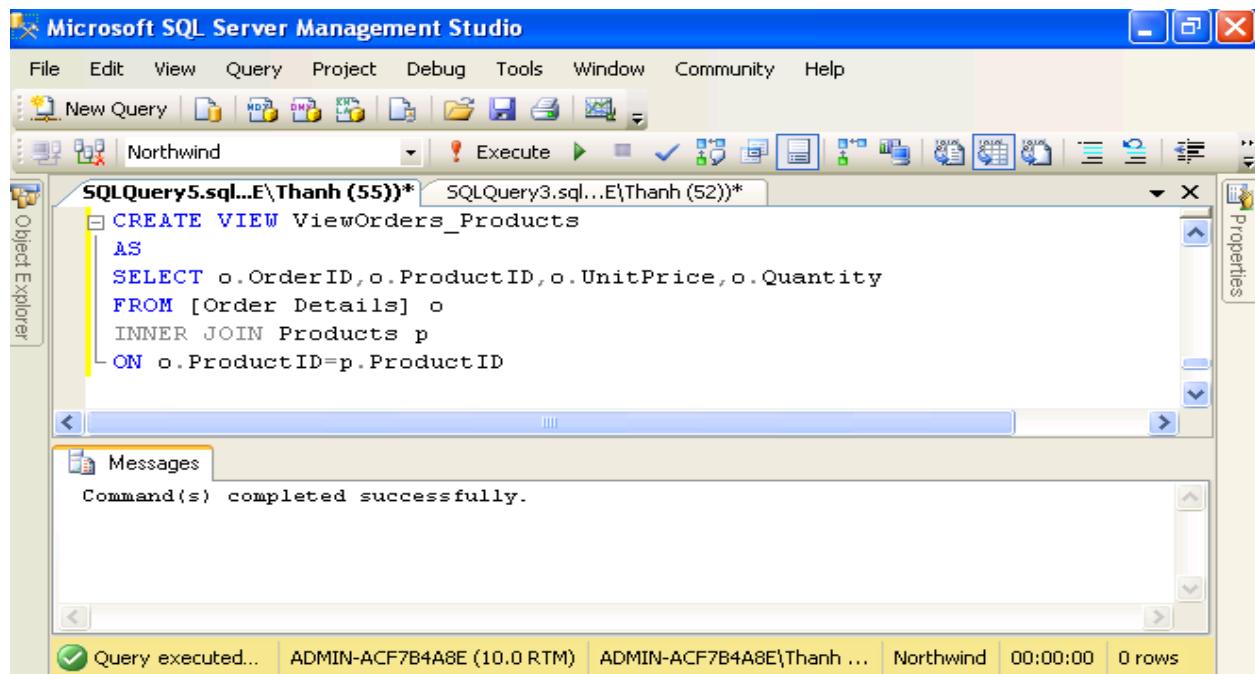
CREATE VIEW <viewName>

AS

SQL Statements

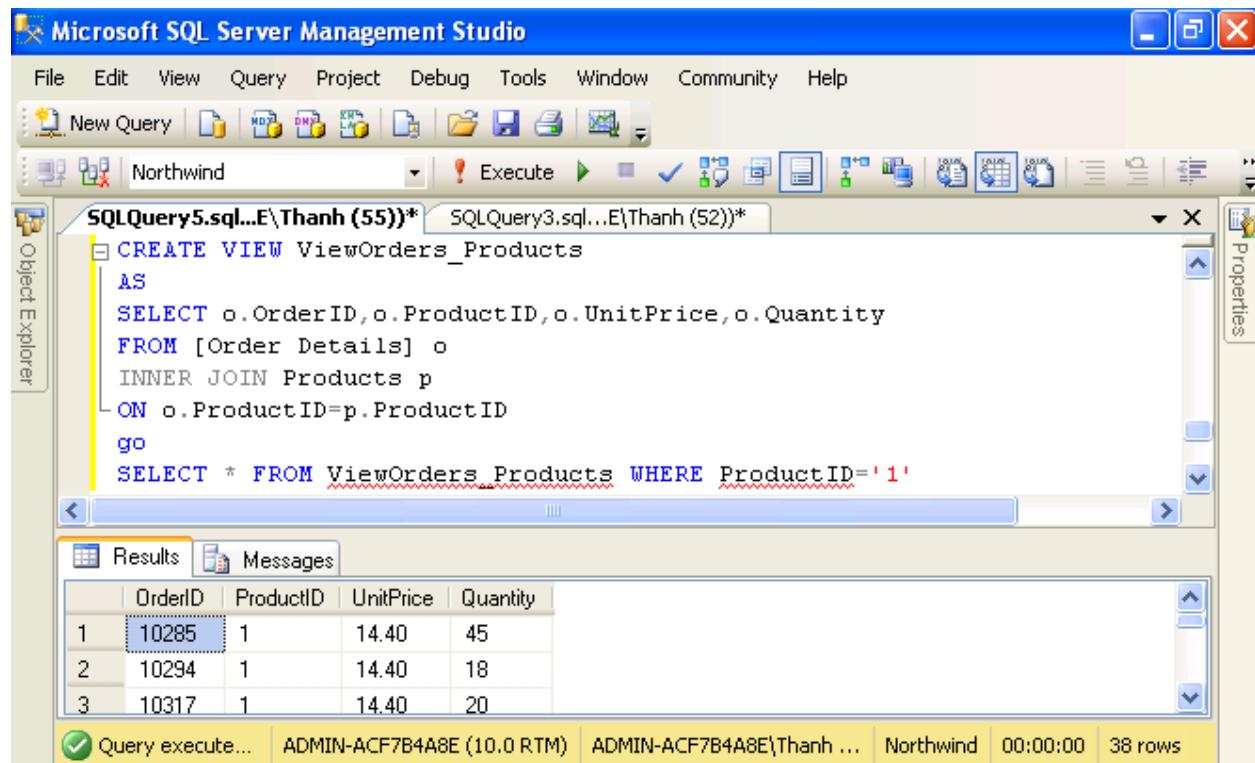
Để chỉnh sửa hoặc xóa View ta dùng **ALTER/ DROP <ViewName>**

Ví dụ: tạo 1 View tên ViewOrders_Products với dữ liệu lấy từ 2 bảng Order Details và Products



H 6.25 Tạo ViewOrders_Details

Gọi View vừa tạo với điều kiện ProductID=1



H 6.26 Gọi view ViewOrders_Products

Các ví dụ minh họa

- **Tạo View thống kê số lượng mặt hàng theo từng nhà cung cấp**

```
TRIDUNG-PC...\t-SQL 6-View.sql*
create view vGetProductBySupplier
as
SELECT SupplierID, COUNT(ProductID) AS AmountProduct
FROM Products
GROUP BY SupplierID
go
SELECT * FROM vGetProductBySupplier
```

SupplierID	AmountProduct
1	3
2	4
3	3
4	3
5	2

H 6.27 View vGetProductBySupplier

- **Tạo view lấy danh sách các hóa đơn có giá trị >= 10000**

```
TRIDUNG-PC...\t-SQL 6-View.sql*
create view vGetOrder
as
SELECT OrderID, SUM(UnitPrice * Quantity) AS TotalAmount
FROM [Order Details]
GROUP BY OrderID
Having SUM(UnitPrice * Quantity) >10000

GO
SELECT * FROM vGetOrder
```

OrderID	TotalAmount
10353	10741,60
10372	12281,20
10417	11283,20
10424	11493,20
10479	10495,60
10515	10588,50

H 6.28 View vGetOrder

6.4 Giới thiệu TRIGGER

Trigger là đối tượng gắn liền với một bảng, tự động thực hiện khi xảy ra sự thay đổi dữ liệu trong bảng như Update, Insert hay Delete. Trigger được dùng để đảm bảo Data Integrity (tồn vẹn dữ liệu) hay thực hiện các Business Rule(ràng buộc dữ liệu) nào đó.

Định nghĩa Trigger cần chú ý:

- Trigger được tạo trong bảng nào?
- Trigger được kích hoạt khi câu lệnh nào được thực thi (Insert ,Update hay Delete).

Cú pháp

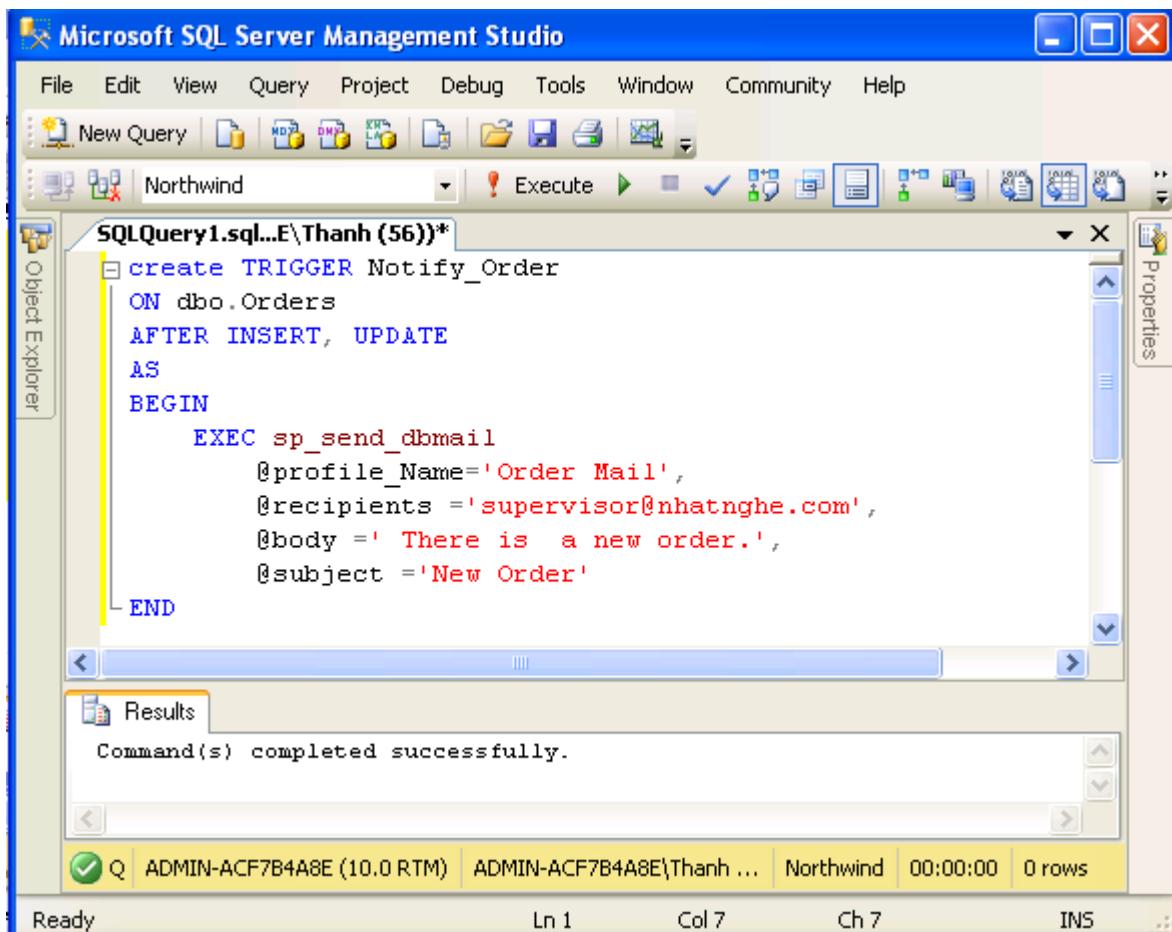
```
CREATE TRIGGER <Trigger Name>
ON <Scope>
<Trigger Timing> <Trigger Condition>
As
Begin
<Trigger Action>
End
```

Sử dụng trigger khi chúng ta cập nhật dữ liệu trên bảng

Cú pháp

```
CREATE TRIGGER <Trigger name>
ON <table>
FOR UPDATE
AS
BEGIN
Statements
END
```

Giả sử bạn muốn tạo ra một Trigger tự động thông báo tới người quản lý khi có một đơn hàng mới được cập nhập.

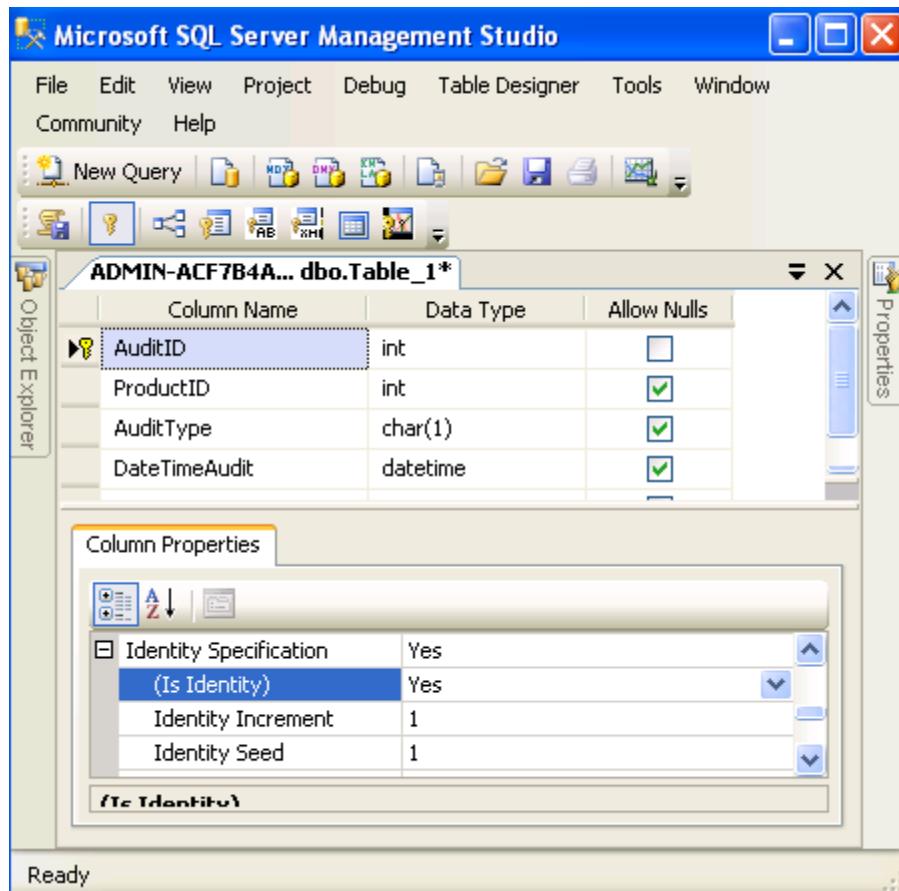


H 6.29 Tạo trigger send mail

Chú ý: để thực hiện được câu lệnh trên chúng ta phải Enable chức năng "[Database Mail XPs](#)"

Sau khi bạn tạo thành công Trigger sẽ tự động giám sát và thông báo khi có sự thay đổi trong bản Orders.

Ví dụ 1: Để giám sát các hoạt động thêm ,chỉnh sửa và cập nhập hàng trong table Products chúng ta tạo ra một table Audit , với các cột AuditID, AuditType,DateTimeAudit,ProductID. Trong đó AuditType có các giá trị 'I','U','D' tương ứng với các hoạt động Insert, Update hay Delete.



H 6.30 Bảng Audit

Sau đó chúng ta tạo ra Trigger dùng cho việc audit như sau

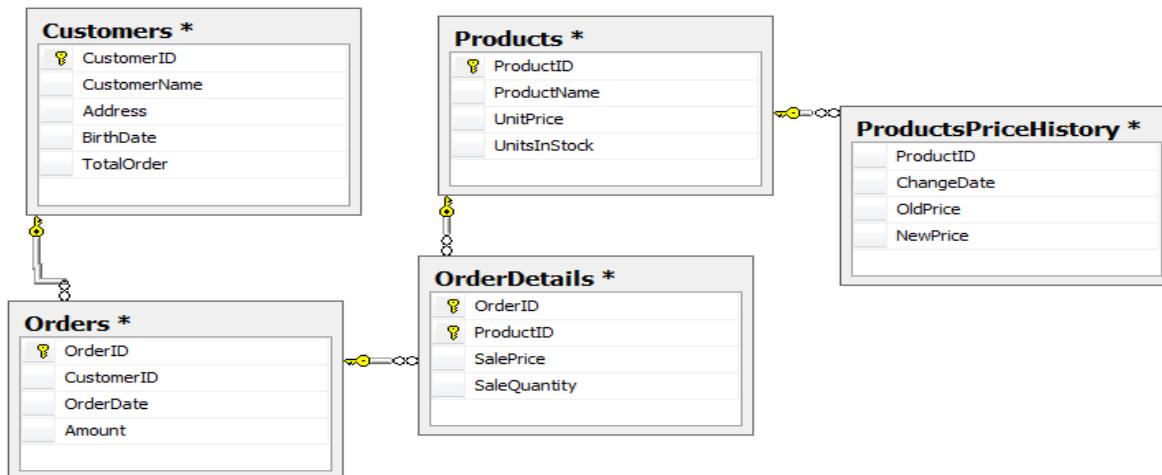
```
--Insert Trigger
create trigger [AuditInsertProduct]
on [Products]
for Insert
as
begin
    insert into Audit(AuditType,DateTimeAudit,ProductID)
    values ('I',getdate(),(select productID from inserted))
end
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery3.sql...E\Thanh (52)*" displays the provided T-SQL script. The status bar at the bottom shows the following information: Qu | ADMIN-ACF7B4A8E (10.0 RTM) | ADMIN-ACF7B4A8E\Thanh ... | Northwind | 00:00:01 | 0 rows. The bottom toolbar includes buttons for Matches, Ln, Col, Ch, and INS.

H 6.31 Trigger Insert Audit

➤ Các ví dụ minh họa :

Chúng ta tạo một database như bên dưới, sau đó tạo các Trigger để quản lý việc thay đổi dữ liệu



H 6.32 Tạo Database

- Tạo Trigger tg_UpdateOrder

The screenshot shows the SQL Server Management Studio interface. In the top window, a script named 'trigger.sql' is being run against a database named 'Diagram_0'. The script creates a trigger 'tg_UpdateOrder' on the 'OrderDetails' table that updates the 'Amount' column in the 'Orders' table whenever a new row is inserted into 'OrderDetails'. The 'Messages' pane at the bottom indicates that the command(s) completed successfully.

```
CREATE TRIGGER tg_UpdateOrder ON [OrderDetails]
FOR INSERT
AS
DECLARE @Subtotal float
SELECT @Subtotal = SalePrice* SaleQuantity
FROM inserted

UPDATE Orders SET Amount = Amount + @Subtotal
FROM Orders o inner join inserted i
ON o.OrderID = i.OrderID
```

Messages

Command(s) completed successfully.

- Tạo trigger tg_DeleteOrderDetail

The screenshot shows the SQL Server Management Studio interface. A script named 'trigger.sql' is being run against a database named 'Diagram_0'. The script creates a trigger 'tg_DeleteOrderDetail' on the 'OrderDetails' table that updates the 'Amount' column in the 'Orders' table whenever a row is deleted from 'OrderDetails'. It uses a temporary variable to store the subtotal of the deleted row and then subtracts it from the 'Amount' in the 'Orders' table. The 'Messages' pane at the bottom indicates that the command(s) completed successfully.

```
CREATE TRIGGER tg_DeleteOrderDetail ON [OrderDetails]
FOR Delete
AS
declare @Subtotal float

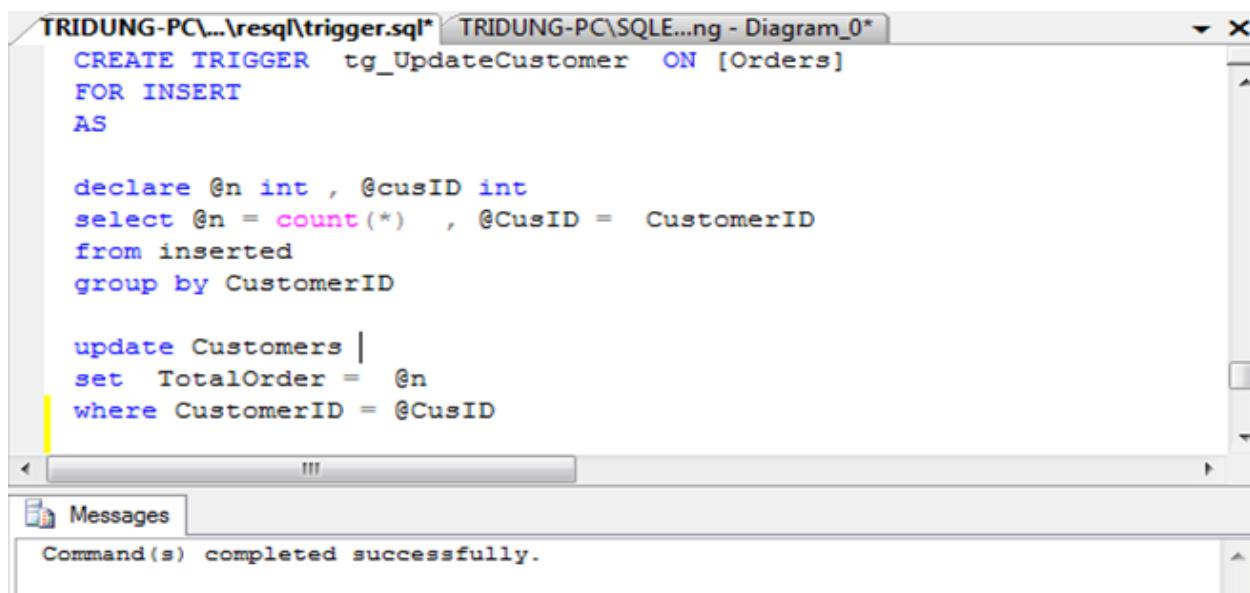
select @Subtotal = SalePrice* SaleQuantity
from deleted

update Orders set Amount = Amount- @Subtotal
from Orders o inner join deleted d
on o.OrderID = d.OrderID
```

Messages

Command(s) completed successfully.

- Tạo trigger tg_UpdateCustomer



The screenshot shows a SQL Server Management Studio window with the following details:

- Title Bar:** TRIDUNG-PC...\resql\trigger.sql* | TRIDUNG-PC\SQLExpress - Diagram_0*
- Text Editor Content:**

```
CREATE TRIGGER tg_UpdateCustomer ON [Orders]
FOR INSERT
AS

declare @n int , @cusID int
select @n = count(*) , @CusID = CustomerID
from inserted
group by CustomerID

update Customers |
set TotalOrder = @n
where CustomerID = @CusID
```
- Messages Tab:** Command(s) completed successfully.

Chương 7:

SQL SERVER 2008 và XML

Kết thúc chương này các bạn có thể :

- *Trình bày được các khái niệm cơ bản về XML và SQL Server 2008*
- *Trình bày và thực hiện được việc lưu trữ dữ liệu dạng XML trong bảng*
- *Thực hiện được truy vấn dữ liệu XML*
- *Thực hiện được Thêm, xóa, sửa dữ liệu XML trong bảng*

7.1 Giới thiệu về XML và SQL Server 2008

❖ Khái niệm về XML

XML (Extensible Markup Language) được W3C tạo ra để trở thành một dạng dữ liệu đơn giản, linh hoạt dựa trên **StandardGeneralized Markup Language (SGML)**. Ngôn ngữ XML được W3C đề xuất vào năm 1996 và công bố vào năm 1998.

W3C XML 1.0 đưa ra một tập hợp các quy tắc cho việc thêm cấu trúc và nội dung dữ liệu bằng cách đánh dấu, tiêu chuẩn hóa việc truyền và chia sẻ dữ liệu giữa các ứng dụng.

Một số ưu điểm của XML:

- **Đơn giản cho việc xử lý:** dữ liệu dạng XML có thể được xử lý bởi các chương trình chuyển đổi.
- **Được truyền trực tiếp trên Internet:** XML được tạo bởi các tập hợp ký tự được định nghĩa như UTF-8, UTF-16... Được thiết kế để dễ dàng đi qua tường lửa, sử dụng các chuẩn giao thức trên Internet như HTTP
- **Đơn giản với người đọc:** do được tạo bởi dạng ký tự và cấu trúc của tài liệu XML rõ ràng, nên con người có thể đọc và hiểu nội dung bên trong dễ dàng, giúp cho việc kiểm tra lỗi đơn giản hơn kiểu dữ liệu dạng nhị phân.
- **Được tạo dễ dàng:** không giống dạng dữ liệu nhị phân, XML có thể được tạo bởi một trình soạn thảo đơn giản.

Cấu trúc của một tài liệu XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <child 1>...</child 1>
    <child 2>...</child 2>
    <child 3>...</child 3>
</root>
```

Ví dụ :

```
<?xml version="1.0" encoding="utf-8"?>
<ProductList>
    <ProductDetail>
        <ProductID>1</ProductID>
        <ProductName>coffee</ProductName>
        <UnitPrice>10</UnitPrice>
        <Quantity>20</Quantity>
    </ProductDetail>
    <ProductDetail>
        <ProductID>2</ProductID>
        <ProductName>sugar</ProductName>
        <UnitPrice>10</UnitPrice>
        <Quantity>3</Quantity>
    </ProductDetail>
<ProductList>
```

❖ XML trong SQL Server 2008

Một số đặc điểm chính mà SQL Server 2008 hỗ trợ XML:

- Kiểu dữ liệu XML (XML data type)
- Lượt đồ XML (XML Schema collections)
- Chỉ mục XML (XML indexes)
- XQuery và XML DML

▪ **Kiểu dữ liệu XML (XML data type)**

Đây là một trong những điểm quan trọng trong SQL Server 2008. XML data type hỗ trợ lưu trữ các tài liệu có định dạng XML, đồng thời được dung để khai báo các cột trong một bảng, các biến T-SQL, tham số, kiểu tra về của 1 hàm. Hơn nữa, XML data type mang đến một tập hợp các phương thức dùng truy vấn dữ liệu.

Giới hạn:

- Dữ liệu dạng Xml không thể vượt quá 2GB
- Không hỗ trợ chuyển đổi sang kiểu dữ liệu dạng **text** hoặc **ntext**. Có thể dùng **varchar(max)** hoặc **nvarchar(max)** thay thế.
- Không thể so sánh hoặc sắp xếp, không thể dùng GROUP BY.

▪ **Lượt đồ XML (XML Schema)**

SQL Server 2008 hỗ trợ tạo lược đồ XML phía server để có thể kiểm tra cấu trúc của tài liệu XML.

▪ **Chỉ mục XML (XML index)**

Trong SQL Server, khi ta truy vấn dữ liệu XML, trước hết dữ liệu sẽ được chuyển sang một định dạng khác, quá trình đó gọi là **shredding**. Quá trình này có thể mất nhiều thời gian khi truy vấn một số lượng lớn dữ liệu dạng XML. SQL Server 2008 hỗ trợ tạo chỉ mục cho các cột có kiểu dữ liệu XML trong bảng, giúp tối ưu hóa và nâng cao khả năng truy vấn dữ liệu.

▪ **XQuery và XML DML**

Kiểu dữ liệu XML cung cấp một số phương thức cho phép ta truy vấn hoặc chỉnh sửa dữ liệu dạng XML. Nhưng phương thức như **query()**, **value()**, **exist()**, **nodes()**, **modify()** hỗ trợ truy vấn XQuery và XML DML..

7.2 Cách xây dựng bảng, lưu dữ liệu dạng XML

Tạo Bảng: Để tạo một cột dạng XML trong bảng ta dùng lệnh CREATE TABLE

Cú pháp:

CREATE TABLE <table name>

(

Col1 <data type>,

Col2 <xml>

)

Ví dụ: Chúng ta tạo 1 bảng tên XmlProducts trong CSDL Northwind gồm 2 cột ID và xmlCol; trong đó cột xmlCol có kiểu dữ liệu XML

```
CREATE TABLE xmlProducts
(
    ID int NOT NULL,
    xmlCol xml,
    CONSTRAINT PK_xmlProducts
    PRIMARY KEY CLUSTERED (ID)
)|
```

7.3 Thêm, Xóa, Sửa dữ liệu XML trong bảng

▪ Thêm dữ liệu dạng XML vào bảng

Để thêm dữ liệu vào các cột có kiểu dữ liệu XML trong bảng, ta dùng lệnh INSERT INTO tương tự như cách thêm dữ liệu thông thường

Cú pháp:

```
INSERT INTO <TableName>
(
    <Column 1>
    <Column 2>
    ...
)
VALUE
(
    <Value 1>
    <Value 2>
    ...
)
```

Chúng ta cũng có thể thêm dữ liệu dạng 1 tài liệu XML :

Cú pháp:

```
INSERT INTO <TableName>
(
    <Xml Column>
    ...
)
VALUE
(
    <root>
        <child node1></ child node1>
        <child node2></ child node2>
```

```
</root>
...
)
```

Ví dụ: Thêm dữ liệu vào bảng XmlProducts vừa tạo

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is an SQL query window titled "SQLQuery13.sql...E\Thanh (51)*". The query is as follows:

```
go
INSERT INTO xmlProducts (ID, xmlCol)
VALUES ('001','<ProductList><ProductID>P001</ProductID>
        <ProductName>Sugar</ProductName></ProductList>')
go
INSERT INTO xmlProducts (ID, xmlCol)
VALUES ('002','<ProductList><ProductID>P002</ProductID>
        <ProductName>Milk</ProductName></ProductList>')
```

Below the query window, there is a "Results" tab showing the output of the query:

ID	xmlCol
1	<ProductList><ProductID>P001</ProductID><ProductName>Sugar</ProductName></ProductList>
2	<ProductList><ProductID>P002</ProductID><ProductName>Milk</ProductName></ProductList>

At the bottom of the interface, there is a status bar with the message "Query executed..." and other system information.

H 7.2 Thêm dữ liệu vào bảng XmlProducts

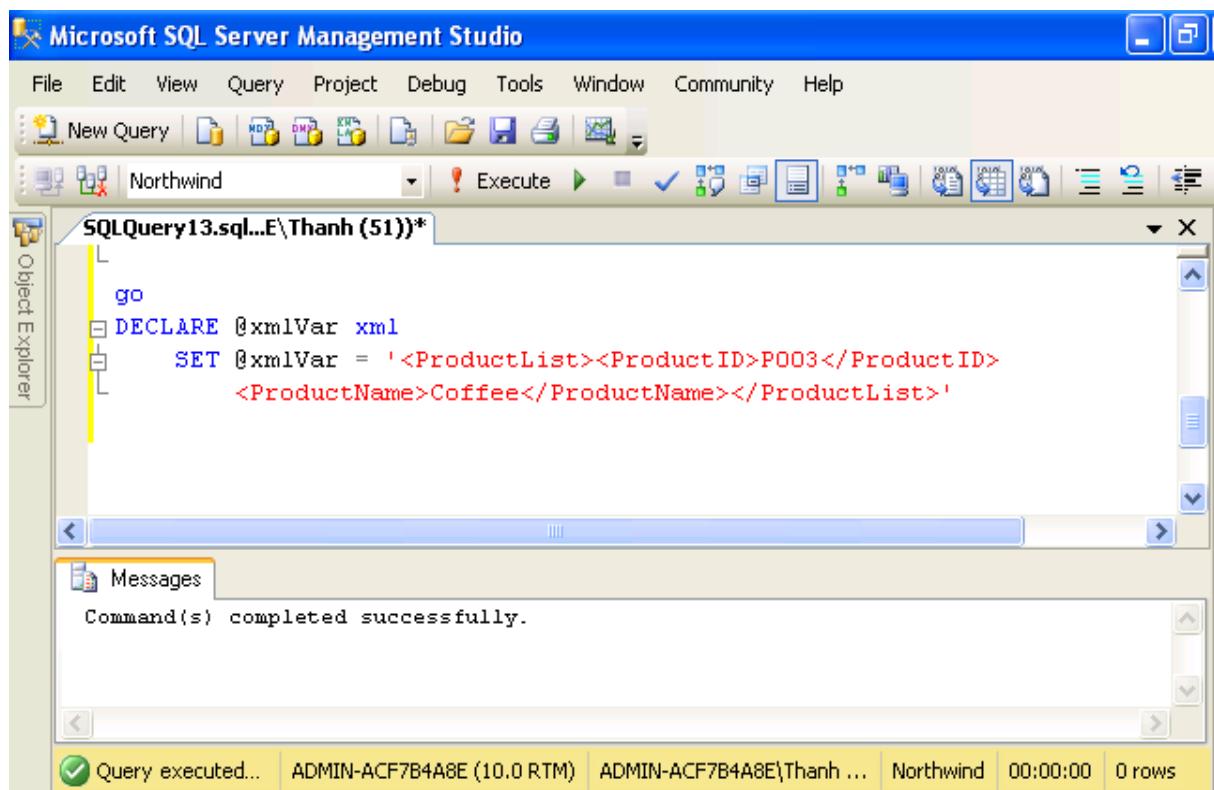
▪ Thêm dữ liệu vào bảng dùng DECLARE

Chúng ta có thể dùng DECLARE để khai báo biến nhằm giúp cho câu lệnh INSERT được đơn giản rõ ràng hơn

Cú pháp:

```
DECLARE <variable_name> [AS] xml
SET <variable_name> = [xml Document]
```

Ví dụ: dùng DECLARE khai báo 1 biến tên @xmlVar thay thế cho cột xmlCol

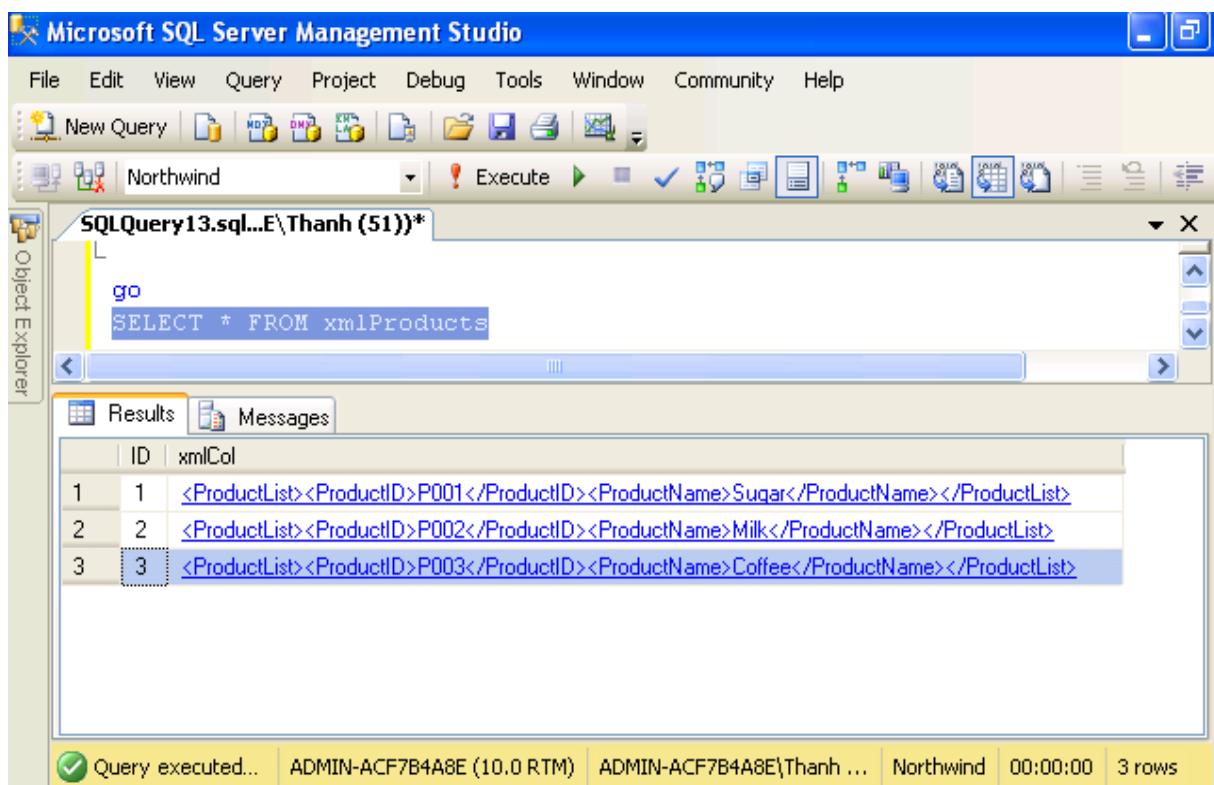


H 7.3 Dùng DECLARE khai báo biến @xmlVar

Sau khi khai báo biến chúng ta thực hiện câu lệnh INSERT, khi đó dùng biến @xmlVar thay cho cột xmlCol

```
DECLARE @xmlVar xml
SET @xmlVar = '<ProductList><ProductID>P003</ProductID>
<ProductName>Coffee</ProductName></ProductList>'

INSERT INTO xmlProducts (ID, xmlCol)
VALUES ('003', @xmlVar)
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the Northwind database is selected. A query window titled "SQLQuery13.sql...E\Thanh (51)*" contains the following SQL code:

```
go
SELECT * FROM xmlProducts
```

The Results pane displays the output of the query:

ID	xmlCol
1	<ProductList><ProductID>P001</ProductID><ProductName>Sugar</ProductName></ProductList>
2	<ProductList><ProductID>P002</ProductID><ProductName>Milk</ProductName></ProductList>
3	<ProductList><ProductID>P003</ProductID><ProductName>Coffee</ProductName></ProductList>

At the bottom of the interface, a status bar shows: "Query executed..." ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... Northwind 00:00:00 3 rows".

H 7.5 Kết quả sau khi thêm dữ liệu

▪ Cập nhật, Xóa dữ liệu

Đối với các thao tác chỉnh sửa nội dung vào xóa dữ liệu, chúng ta cũng thực hiện với câu lệnh UPDATE và DELETE

Ví dụ: Thay đổi tên sản phẩm có ID=3.

```
go
UPDATE xmlProducts
SET xmlCol='<ProductList><ProductID>P003</ProductID>
      <ProductName>Beer</ProductName></ProductList>'
WHERE ID='3'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window titled "SQLQuery13.sql...E\Thanh (51)*". The query is:

```
go
SELECT * FROM xmlProducts WHERE ID='3'
```

The results pane shows a single row of data:

ID	xmlCol
1	<ProductList><ProductID>P003</ProductID><ProductName>Beer</ProductName></ProductList>

At the bottom, a status bar indicates: "Query executed..." with a green checkmark, "ADMIN-ACF7B4A8E (10.0 RTM)", "ADMIN-ACF7B4A8E\Thanh ...", "Northwind", "00:00:00", and "1 rows".

H 7.7 Kết quả sau khi cập nhật

Ví dụ: Xóa dữ liệu vừa cập nhật

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window titled "SQLQuery13.sql...E\Thanh (51)*". The query itself is:

```

go
DELETE FROM xmlProducts WHERE ID='3'
go
SELECT * FROM xmlProducts

```

Below the query window is a results grid titled "Results". It has two rows of data:

ID	xmlCol
1	<ProductList><ProductID>P001</ProductID><ProductName>Sugar</ProductName></ProductList>
2	<ProductList><ProductID>P002</ProductID><ProductName>Milk</ProductName></ProductList>

At the bottom of the interface, a status bar displays: "Query executed..." ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... Northwind 00:00:00 2 rows".

H 7.8 Lệnh DELETE và kết quả

7.4 Truy vấn dữ liệu XML

Ngoài câu lệnh SELECT đơn giản đã được giới thiệu, SQL Server 2008 còn hỗ trợ một số cách truy vấn cho kiểu dữ liệu XML

FOR XML: trả về dữ liệu dạng một tài liệu XML

Cú pháp:

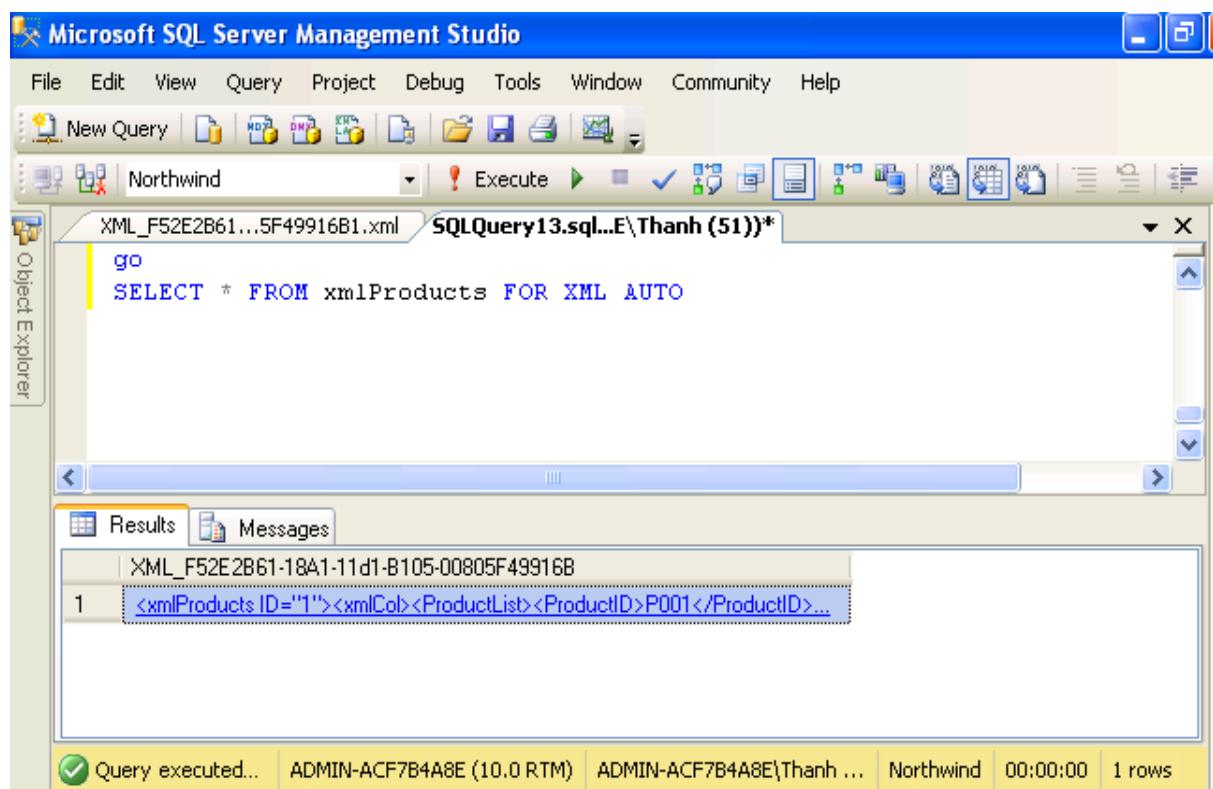
```

SELECT
    <column1>,<column2>...
FROM <table_name>
FOR XML <mode>

```

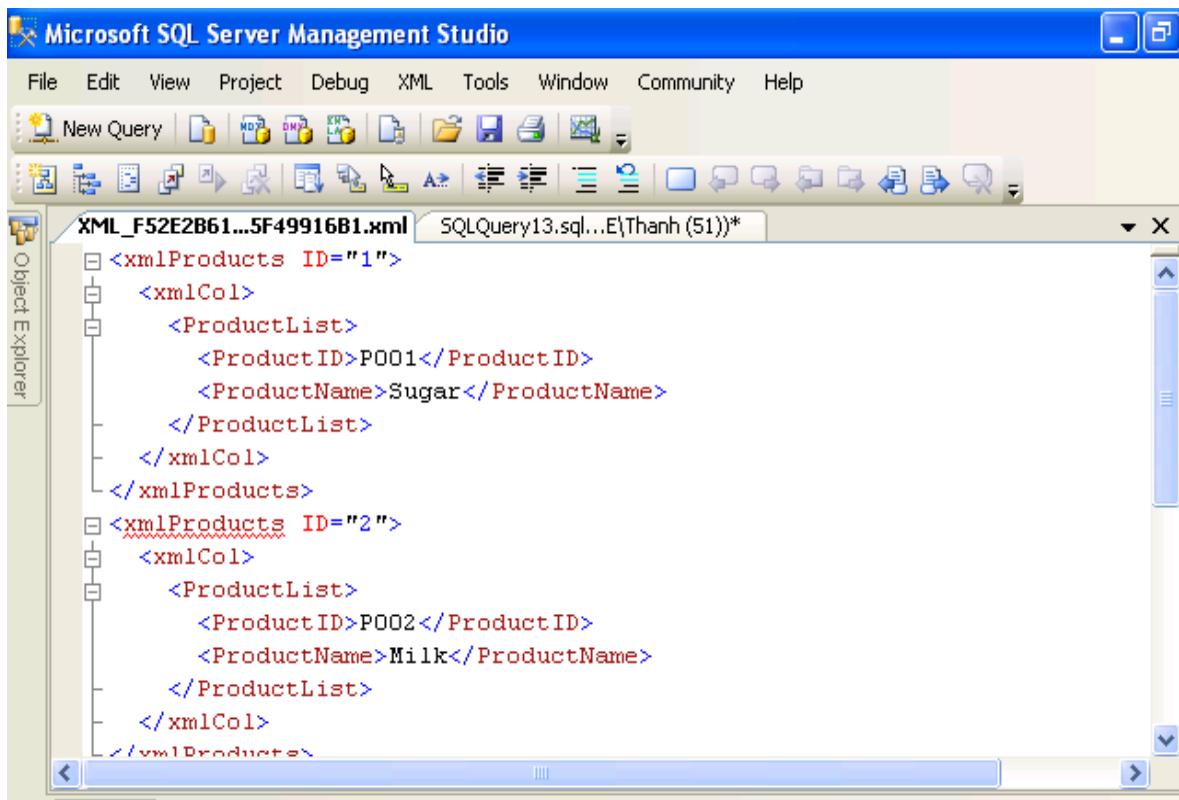
Với **mode:** RAW, AUTO, EXPLICIT, PATH

Ví Dụ: truy vấn dữ liệu của bảng xmlProducts



H 7.9 Lệnh SELECT...FOR XML

Sau đó, chúng ta click vào dòng dữ liệu trả về, SQL Server sẽ cho thấy kết quả



H 7.10 Kết quả truy vấn dạng XML

Tuy nhiên, khi dùng FOR XML, dữ liệu dạng XML không đúng cấu trúc vì thiếu một tag mà ta thường gọi là Root. Để thêm Root vào ta có thể dùng FOR XML PATH

Cú pháp:

```
SELECT
    <column1>,<column2>...
FROM <table_name>
FOR XML PATH, ROOT<root>
```

VÍ Dụ: Ta thêm một root ProductList vào tài liệu XML khi truy vấn FOR XML

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid displaying XML data. The XML structure is as follows:

```

<ProductList>
  <row ID="1">
    <xmlCol>
      <ProductList>
        <ProductID>P001</ProductID>
        <ProductName>Sugar</ProductName>
      </ProductList>
    </xmlCol>
  </row>
  <row ID="2">
    <xmlCol>
      <ProductList>
        <ProductID>P002</ProductID>
        <ProductName>Milk</ProductName>
      </ProductList>
    </xmlCol>
  </row>

```

The results grid has two columns: 'Results' and 'Messages'. At the bottom, a status bar indicates: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '1 rows'.

H 7.11 Truy vấn dùng PATH

Kết quả:

The screenshot shows the Microsoft SQL Server Management Studio interface with the XML data from the previous screenshot. The XML structure is identical:

```

<ProductList>
  <row ID="1">
    <xmlCol>
      <ProductList>
        <ProductID>P001</ProductID>
        <ProductName>Sugar</ProductName>
      </ProductList>
    </xmlCol>
  </row>
  <row ID="2">
    <xmlCol>
      <ProductList>
        <ProductID>P002</ProductID>
        <ProductName>Milk</ProductName>
      </ProductList>
    </xmlCol>
  </row>

```

The interface includes an Object Explorer on the left and a Properties pane on the right.

H 7.12 ProductList được thêm vào kết quả truy vấn

Ngoài việc truy vấn các bảng có dữ liệu dạng XML, ta có thể dùng FOR XML để truy vấn và xem dữ liệu từ các bảng thông thường dưới dạng 1 tài liệu XML.

Ví dụ: truy vấn dữ liệu các khách hàng trong bảng Customers của CSDL Northwind có CustomerID bắt đầu bằng 'A'

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query results window titled "SQLQuery13.sql...E\Thanh (51)*". The query itself is:

```
XML_F52E2B61...49916B15.xml
SELECT c.CustomerID AS "Customers/@ID",
       c.CompanyName AS "Customers/CustomerName",
       c.Address AS "Customers/Address/Add",
       c.City AS "Customers/Address/City"
  FROM Customers c WHERE c.CustomerID LIKE 'A%'
FOR XML PATH, ROOT('CustomerList'),
ELEMENTS XSINIL
```

Below the query, the results pane shows one row of XML data:

```
1 <CustomerList xmlns:xsi='http://www.w3.org/2001/X...
```

At the bottom of the results pane, there is a status bar with the following information:

Query executed... ADMIN-ACF7B4A8E (10.0 RTM) ADMIN-ACF7B4A8E\Thanh ... Northwind 00:00:00 1 rows

H 7.13 Truy vấn từ Customers

Kết quả:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is an XML document titled "XML_F52E2B61...49916B15.xml". The XML content represents a list of customers from the Northwind database. The root element is "CustomerList" with the namespace "http://www.w3.org/2001/XMLSchema-instance". It contains two "row" elements, each representing a customer. The first customer has the ID "ALFKI" and the name "Alfreds Futterkiste". Its address is "Obere Str. 57, Berlin". The second customer has the ID "ANATR" and the name "Ana Trujillo Emparedados y helados". Its address is "Avda. de la Constitución 2222, México D.F.". The XML code is as follows:

```
<CustomerList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <Customers ID="ALFKI">
      <CustomerName>Alfreds Futterkiste</CustomerName>
      <Address>
        <Add>Obere Str. 57</Add>
        <City>Berlin</City>
      </Address>
    </Customers>
  </row>
  <row>
    <Customers ID="ANATR">
      <CustomerName>Ana Trujillo Emparedados y helados</CustomerName>
      <Address>
        <Add>Avda. de la Constitución 2222</Add>
        <City>México D.F.</City>
      </Address>
    </Customers>
  </row>
</CustomerList>
```

H 7.14 Danh sách Khách hàng

Để tìm hiểu rõ hơn, các bạn có thể tham khảo tại <http://msdn.microsoft.com/en-us/library/ms191268.aspx>

❖ Một số phương thức dùng cho kiểu dữ liệu XML

SQL Server cung cấp cho chúng ta một số phương thức dùng truy vấn XML, các phương thức này được xem tương tự như Subquery, do đó không thể trong các câu lệnh PRINT hay trong các mệnh đề GROUP BY.

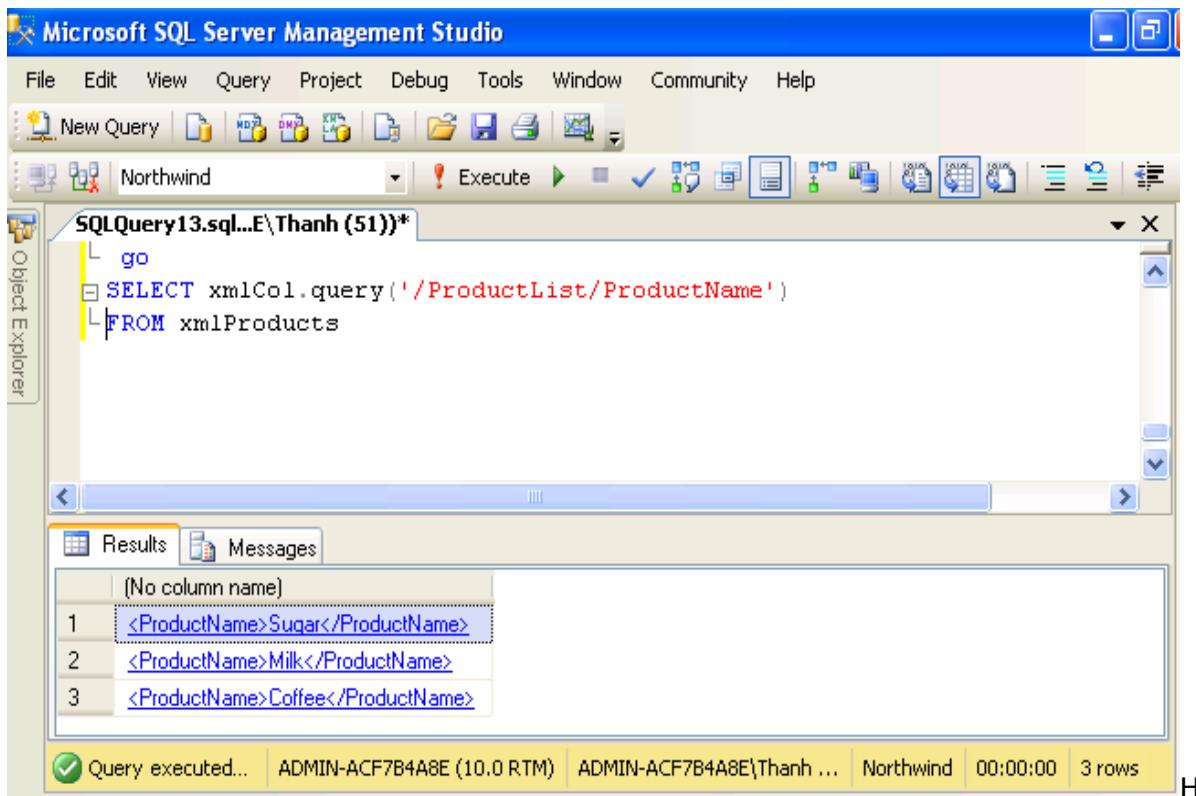
Cách gọi các phương thức này tương tự khi ta gọi phương thức/hàm trong lập trình hướng đối tượng

- **Hàm QUERY()**

Cú pháp:

xml_obj.query (xquery) : Với XQuery là node trong tài liệu xml

Ví dụ: Dùng Query() để lấy tên các sản phẩm trong bảng xmlProducts



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a results grid titled 'Results' showing three rows of XML data. The data consists of three entries under the column '(No column name)'. Each entry contains an XML node: row 1 has <ProductName>Sugar</ProductName>, row 2 has <ProductName>Milk</ProductName>, and row 3 has <ProductName>Coffee</ProductName>. Below the results grid, the status bar displays: 'Query executed...', 'ADMIN-ACF7B4A8E (10.0 RTM)', 'ADMIN-ACF7B4A8E\Thanh ...', 'Northwind', '00:00:00', and '3 rows'.

(No column name)
<ProductName>Sugar</ProductName>
<ProductName>Milk</ProductName>
<ProductName>Coffee</ProductName>

7.15 Liệt kê ProductName dùng Query()

- **Hàm Value():** trả về giá trị của các node.

Cú pháp:

xml_obj.value (xquery, data_type)

Ví Dụ: Lấy giá trị ProductID, ProductName trong bảng xmlProducts

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery13.sql...E\Thanh (51)*" displays the following XML query:

```
go

SELECT xmlCol.value(N'(/ProductList/ProductID)[1]', 'char(10)') AS ID,
       xmlCol.value(N'(/ProductList/ProductName)[1]', 'char(10)') AS Name
  FROM xmlProducts
```

The results pane shows a table with two rows:

	ID	Name
1	P001	Sugar
2	P002	Milk

At the bottom, a status bar indicates: "Query executed..." | ADMIN-ACF7B4A8E (10.0 RTM) | ADMIN-ACF7B4A8E\Thanh ... | Northwind | 00:00:00 | 2 rows

H 7.16 Liệt kê các sản phẩm

- **Hàm Exist():** kiểm tra sự tồn tại giá trị của các node

Cú pháp:

xml_obj.exist (xquery)

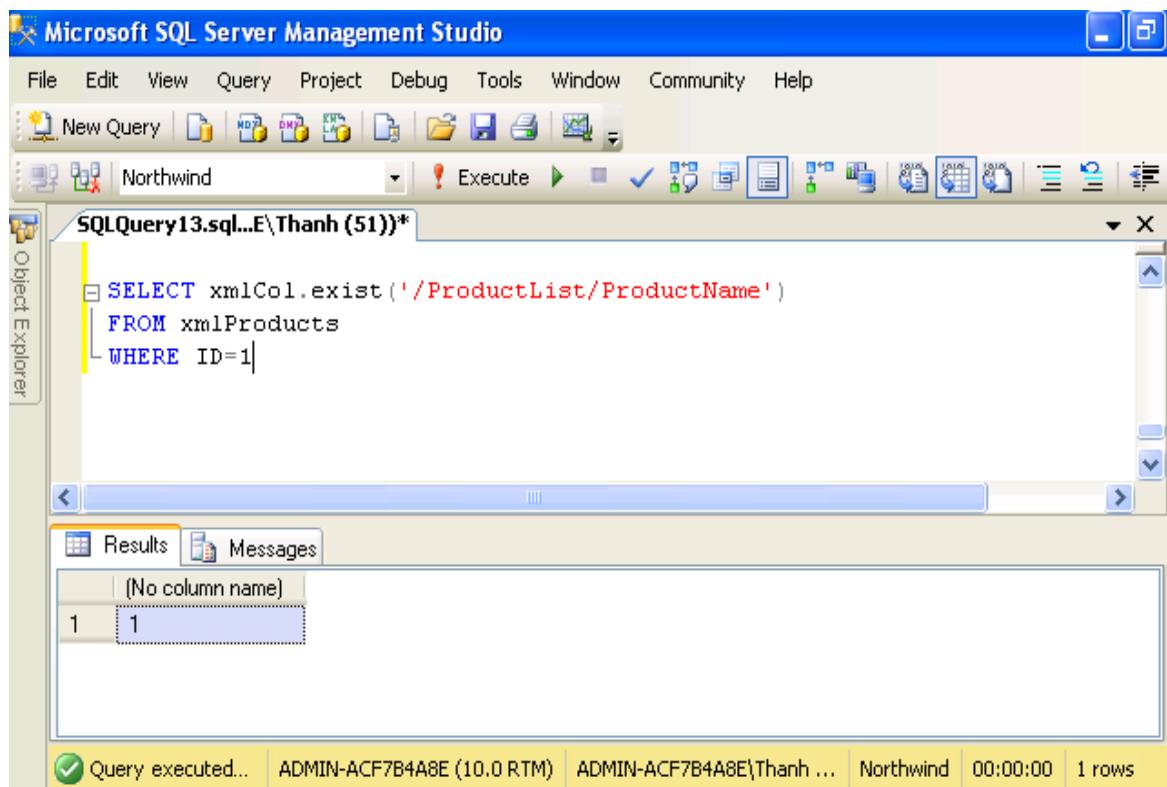
Các giá trị trả về:

0: không tồn tại giá trị bên trong node

1: tồn tại giá trị bên trong node

NULL: *xml_obj* không tồn tại

Ví dụ: Kiểm tra giá trị *ProductName* có *ID*=1



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window titled "SQLQuery13.sql...E\Thanh (51)*" containing the following XML query:

```
SELECT xmlCol.exist('/ProductList/ProductName')
FROM xmlProducts
WHERE ID=1
```

Below the query window, the "Results" tab is selected, displaying a single row of results:

	[No column name]
1	1

At the bottom of the interface, a status bar shows the following information: "Query executed..." with a green checkmark, "ADMIN-ACF7B4A8E (10.0 RTM)", "ADMIN-ACF7B4A8E\Thanh ...", "Northwind", "00:00:00", and "1 rows".

H 7.17 Kiểm tra tồn tại giá trị ProductName

- **Hàm Nodes()**

Cú pháp:

xml_obj.nodes (xquery) AS table (column)

Table(column): 1 bảng tạm được tạo và chứa cột dạng xml mà dữ liệu có được là giá trị trả về của **nodes()**

Ví dụ: lấy tên sản phẩm trong bảng xmlProducts truyền vào bảng myTable với cột myCol dạng xml

```
SQLQuery13.sql...E\Thanh (51)*
SELECT myTable.myCol.value
(N'(/ProductList/ProductName)[1]', 'char(10)') AS NAME
FROM xmlProducts
CROSS APPLY xmlCol.nodes(N'/ProductList') AS myTable(myCol)|
```

NAME
1 Sugar
2 Milk

Query executed... | ADMIN-ACF7B4A8E (10.0 RTM) | ADMIN-ACF7B4A8E\Thanh ... | Northwind | 00:00:00 | 2 rows

H 7.18 Sử dụng Nodes()

- **Chỉ mục XML (XML Index)**

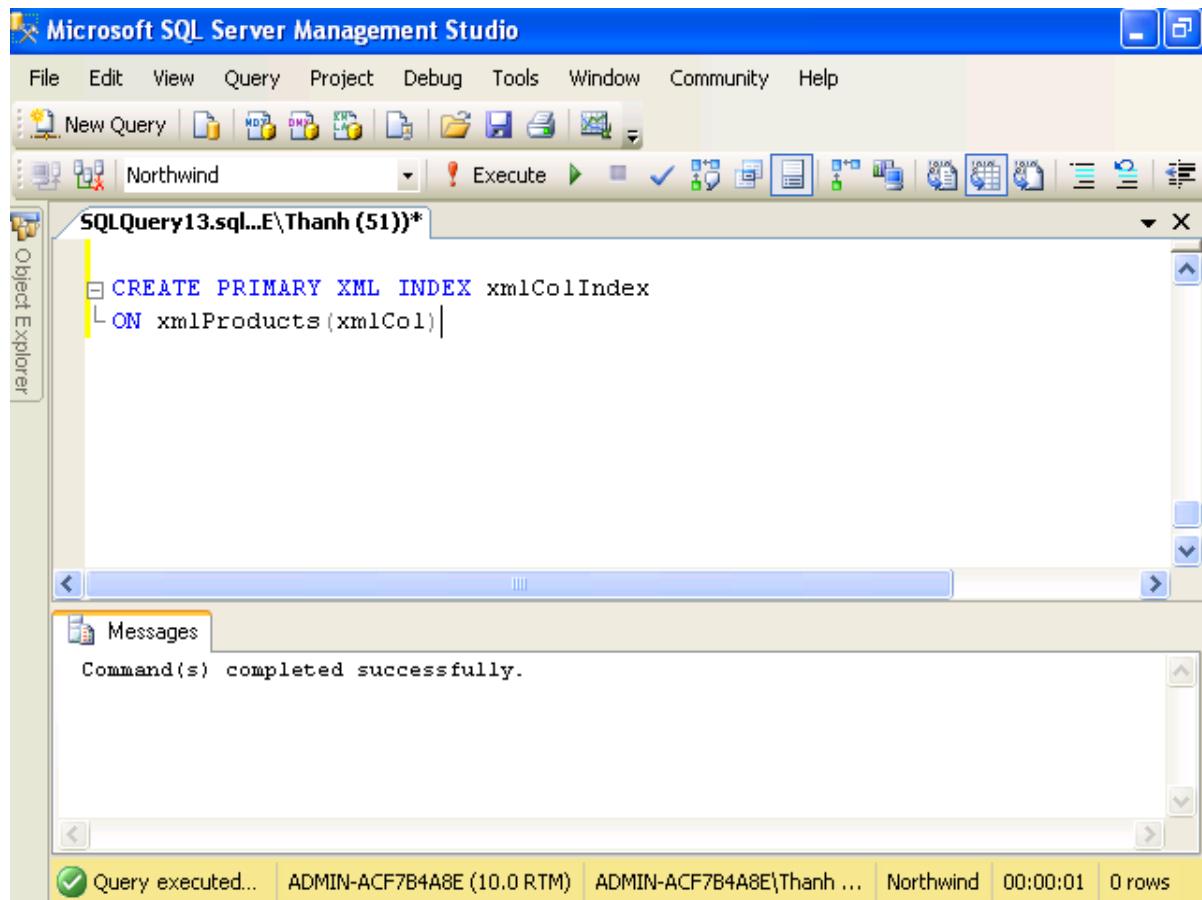
Chỉ mục được tạo trên các cột có kiểu dữ liệu xml, giúp nâng cao hiệu quả truy vấn khi ta có một bảng chứa một lượng lớn các dữ liệu dạng xml.

Có 2 loại chỉ mục: Primary XML Index và Secondary XML Index

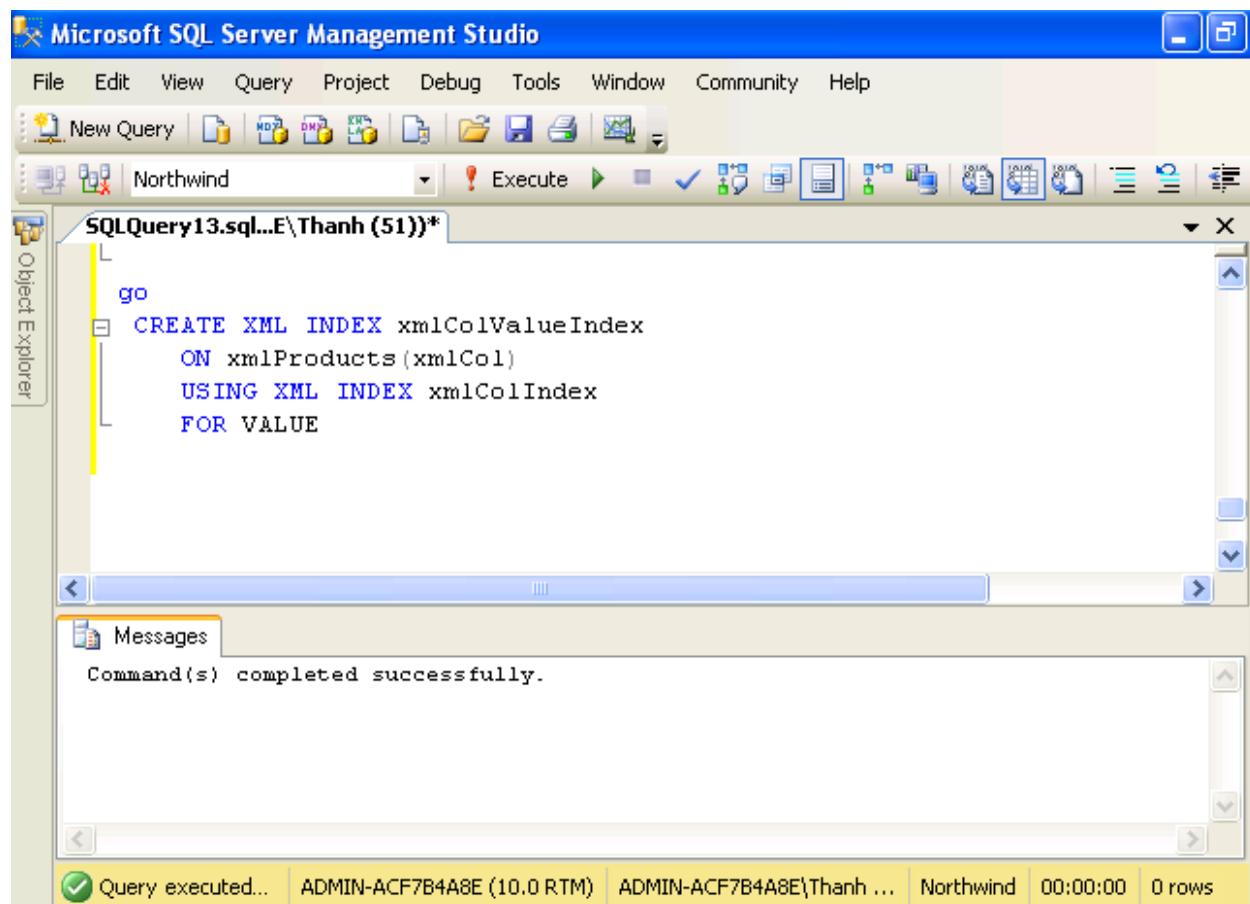
Cú pháp:

```
CREATE [ PRIMARY ] XML INDEX index_name
    ON <object> ( xml_column_name )
    [ USING XML INDEX xml_index_name
    [ FOR { VALUE | PATH | PROPERTY } ] ]
```

Ví dụ: Tạo Index cho cột xmlCol trong bảng xmlProducts



H 7.19 Tạo Primary Index tên xmlColIndex



H 7.20 Tạo Secondary Index tên xmlColValueIndex

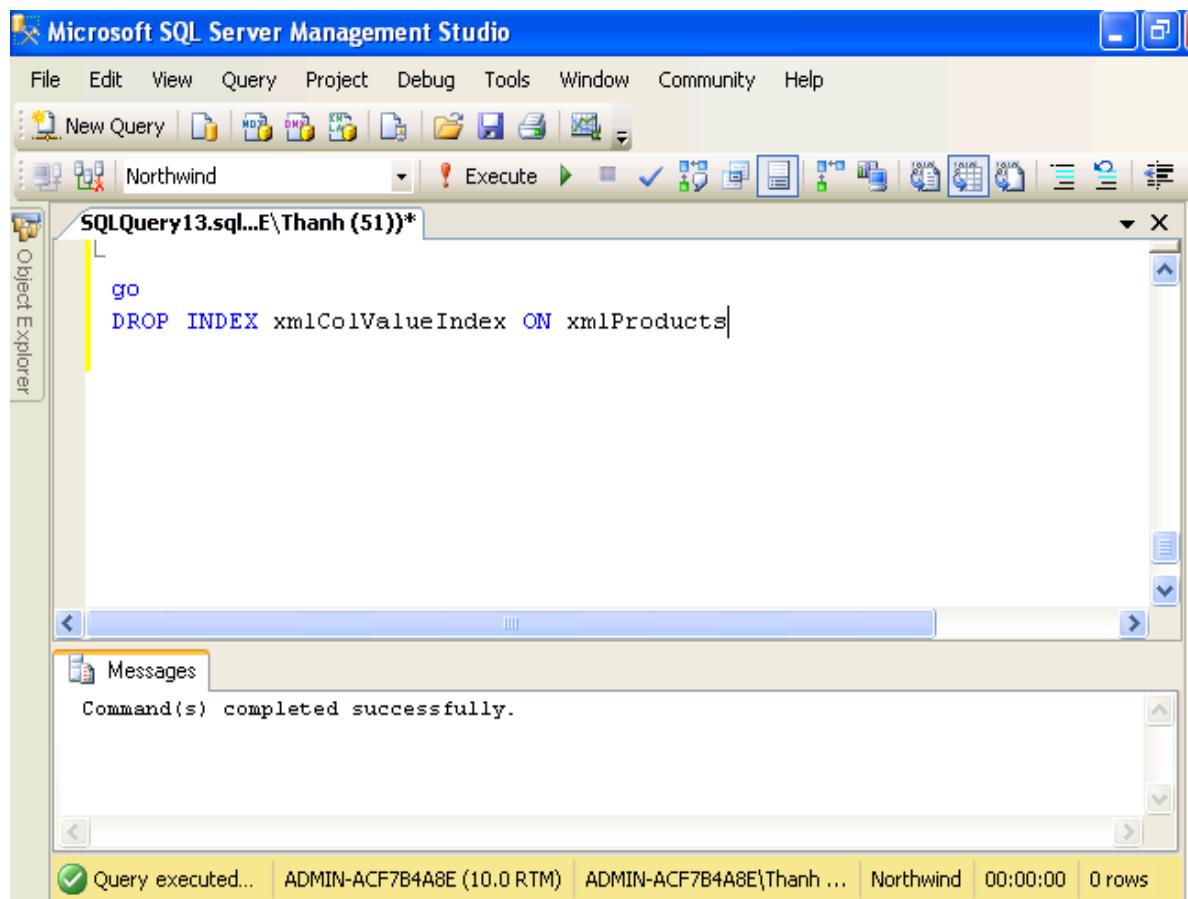
Xóa, Chính sửa Index

Để xóa Index, ta dùng lệnh DROP INDEX

Cú pháp:

DROP INDEX { *index_name* ON <object> [,...n] }

Ví dụ: xóa Secondary Index vừa tạo



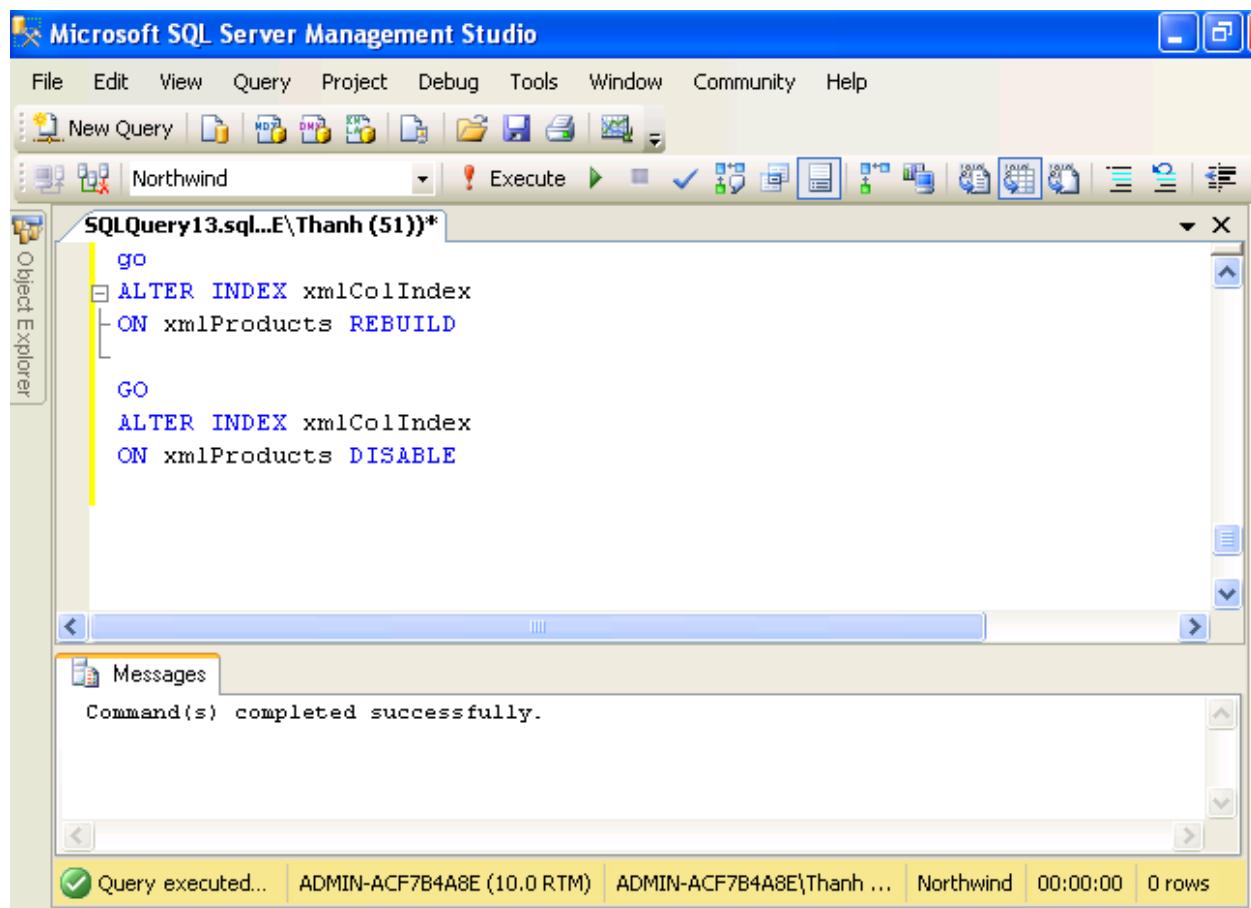
H 7.21 Xóa Index

Để chỉnh sửa ta dùng ALTER INDEX

Cú pháp:

ALTER INDEX <index_name>
ON <object> REBUILD | REORGANIZE | DISABLE

VÍ Dụ: chỉnh sửa Index dung Rebuild hoặc Disable



H 7.22 Chính sửa Index

▪ Lược đồ XML (XML Schema)

XML Schema mô tả cấu trúc của tài liệu dạng XML; sử dụng cú pháp của XML

- Định nghĩa các thành phần, thuộc tính có trong tài liệu.
- Định nghĩa kiểu dữ liệu của các thành phần bên trong.
- Xác định và sắp xếp các thuộc tính con.
- Xác định thành phần nào rỗng hoặc có thể thêm dữ liệu vào đó.

Cú pháp:

Tạo Schema:

CREATE XML SCHEMA COLLECTION <schema_name>
AS Expression

Chỉnh sửa Schema:

ALTER XML SCHEMA COLLECTION <schema_name>

Xóa Schema:

DROP XML SCHEMA COLLECTION <schema_name>

Ví dụ: Tạo 1 schema tên ContactSchemaCollection, sau đó tạo bảng Contacts và them dữ liệu vào bảng theo cấu trúc Schema vừa tạo

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window titled "SQLQuery13.sql...E\Thanh (51)*" containing the following XML schema definition:

```
CREATE XML SCHEMA COLLECTION ContactSchemaCollection AS
N'<?xml version="1.0" encoding="utf-16"?>
<xs:schema targetNamespace="http://tempuri.org/ContactSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    xmlns="http://tempuri.org/ContactSchema"
    xmlns:mstns="http://tempuri.org/ContactSchema"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Contact">
        <xs:complexType>
            <xs:sequence>
                </xs:sequence>
                <xs:attribute name="FirstName" type="xs:string" />
                <xs:attribute name="LastName" type="xs:string" />
                <xs:attribute name="PhoneNumber" type="xs:string" />
            </xs:complexType>
        </xs:element>

```

In the bottom status bar, it says "Query executed..." and "0 rows".

H 7.23 Tạo Schema ContactSchemaCollection

Tạo bảng Contacts gồm 1 cột Contact

```
CREATE TABLE Contacts
(
    Contact xml (ContactSchemaCollection)
)
```

Sau khi đã tạo bảng, chúng ta có thể thêm dữ liệu

```
INSERT INTO Contacts (Contact)
VALUES ('<Contact xmlns="http://tempuri.org/ContactSchema"
        FirstName="John" LastName="Doe" Phone="555-555-5555" />')
```

Do ta đã định nghĩa trước các thành phần của XML trong Schema, nên khi ta thêm dữ liệu không phù hợp, SQL Server sẽ kiểm tra với Schema và báo lỗi

The screenshot shows a SQL query window titled "SQLQuery13.sql...E\Thanh (51)*". The query is:

```
go
INSERT INTO Contacts (Contact)
VALUES ('<Contact xmlns="http://tempuri.org/ContactSchema"
        FirstName="John" LastName="Doe" Phone="555-555-5555" />')
```

In the "Messages" pane below, there is an error message:

```
Msg 6905, Level 16, State 3, Line 1
XML Validation: Attribute 'Phone' is not permitted in this context. Location: /*:Co
```

Để tìm hiểu rõ hơn về XML Schema cũng như các cách sử dụng khác của Schema, các bạn có thể tham khảo tại: <http://msdn.microsoft.com/en-us/library/ms176009.aspx>

Chương 8

.NET Integration & SQL Server

Kết thúc chương này các bạn có thể :

- Trình bày được các khái niệm về .NET Integration với SQL Server 20008
- Trình bày và xây dựng được CLR Store Procedure
- Trình bày và xây dựng được CLR User-Defined Function
- Trình bày và xây dựng CLR User-Defined Type

8.1 Giới thiệu về .NET Integration & SQL Server 2008

Common Language Runtime (CLR) cung cấp các phương thức quản lý mã nguồn như tích hợp các ngôn ngữ, bảo mật bằng mã truy cập, quản lý vòng đời các đối tượng, gỡ lỗi... Đối với người dùng SQL Server và các nhà phát triển ứng dụng, tích hợp CLR (CLR Integration) trong .Net nghĩa là ta có thể viết và lưu trữ Store Procedure, triggers, user-defined types, user-defined function bằng cách sử dụng bất kỳ .Net Framework. *Lưu ý rằng, CLR Integration không dùng trong Visual Studio 2003 (Framework 1.0).*

Những lợi ích chính của CLR Integration:

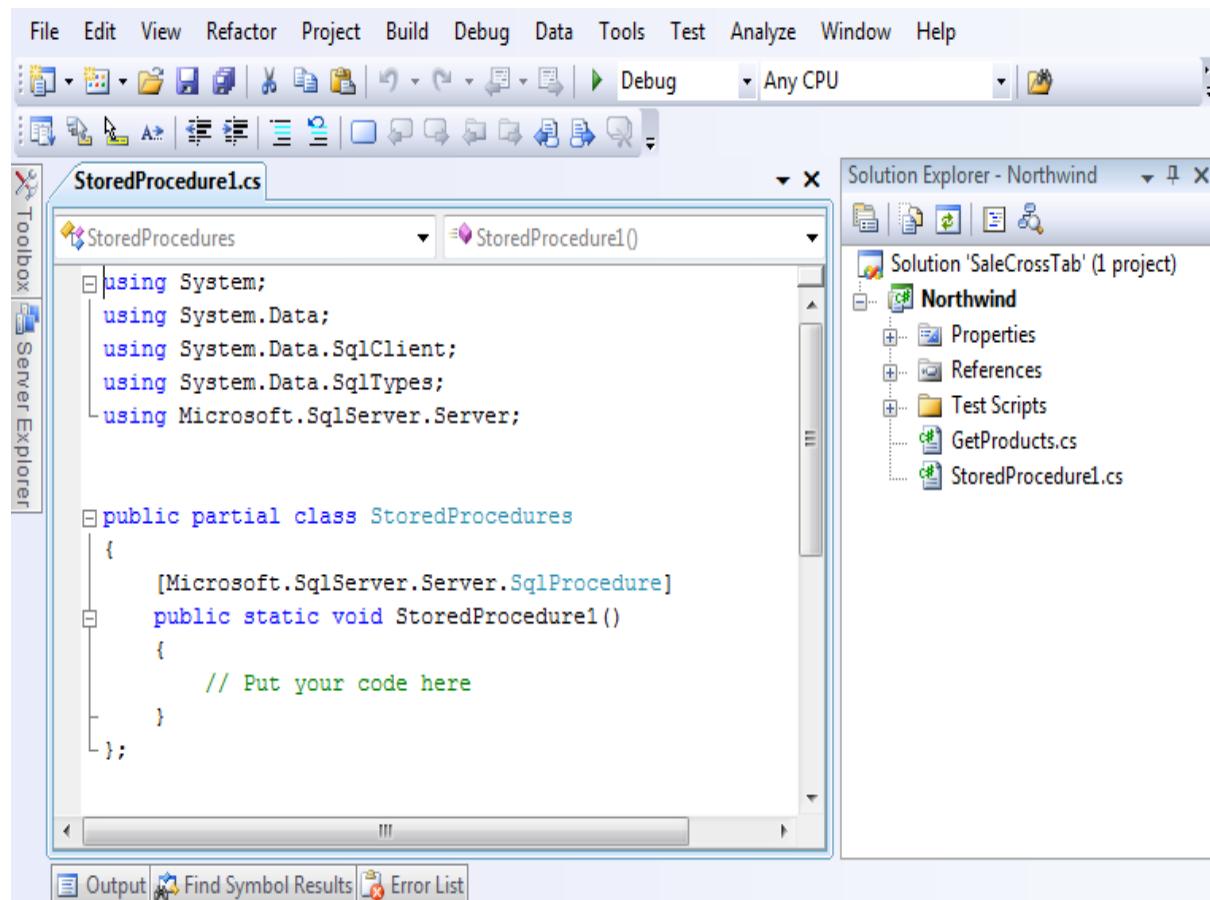
- **Mô hình lập trình tốt hơn:** các nhà phát triển có thể tận dụng sức mạnh của thư viện .Net Framework, trong đó cung cấp một tập hợp rộng rãi các class để sử dụng nhanh chóng và hiệu quả nhằm giải quyết các vấn đề lập trình.
- **Cải thiện tính an toàn và bảo mật:** quản lý các mã trong môi trường CLR được tổ chức trong cơ sở dữ liệu. SQL Server cung cấp một sự thay thế an toàn và bảo mật hơn các phiên bản trước.
- **Khả năng xác định kiểu dữ liệu:** User-defined type và user-defined aggregates là 2 đối tượng quản lý dữ liệu mới, mở rộng khả năng lưu trữ và truy vấn SQL Server.

8.2 Xây dựng CLR Store Procedure

CLR Store Procedure được sử dụng như các phương thức dạng *public static*. Các phương thức *static* có thể khai báo dạng *void* hay trả về một giá trị kiểu *integer*. Nếu khai báo dạng *void*, giá trị trả về là 0.

Giá trị trả về của Store Procedure có thể là tham số, bảng kết quả hoặc một thông báo.

Màn hình làm việc với CLR Store Procedure trong Visual Studio 2008



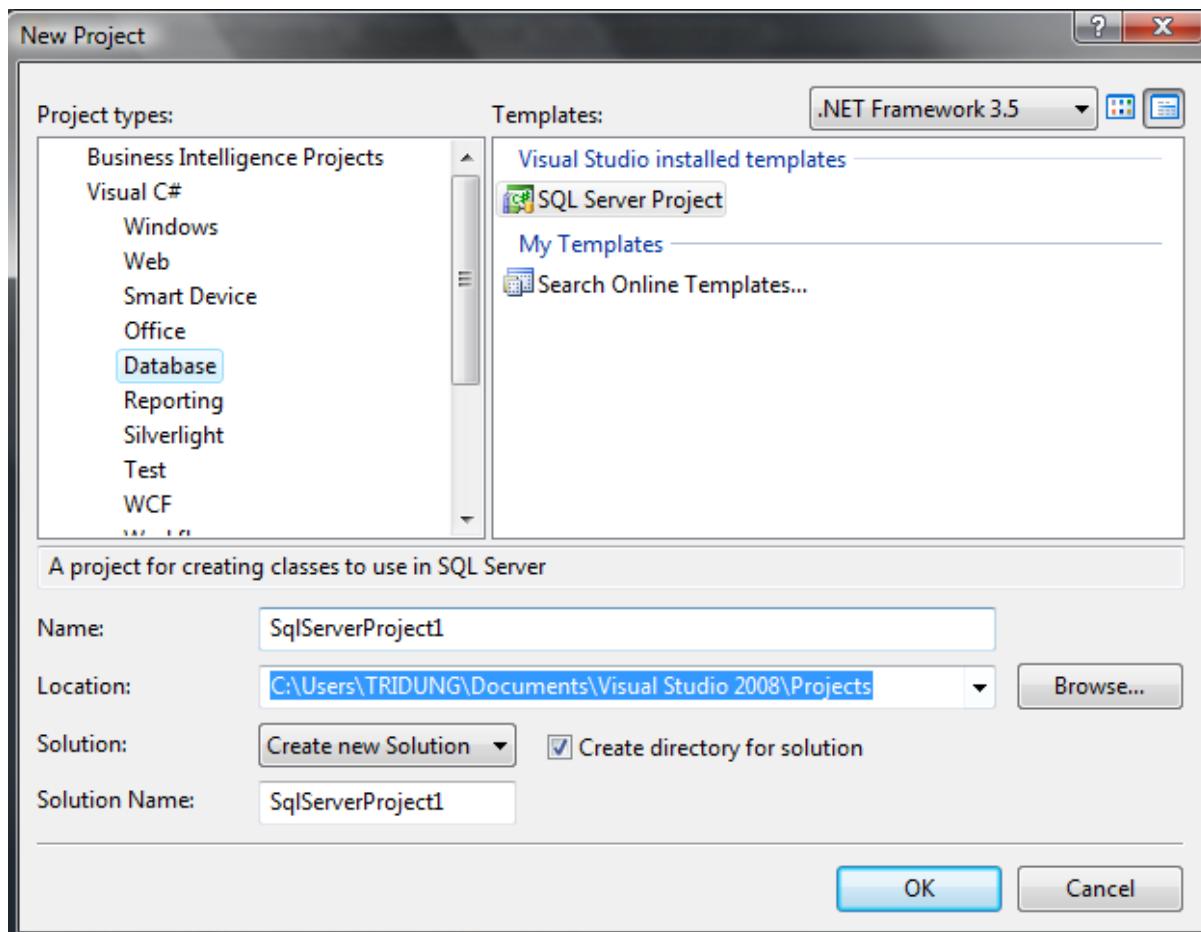
H 8.1 Giao diện viết Store Procedure

Để tìm hiểu rõ hơn về cách tạo một CLR Store Procedure Integration, chúng ta sẽ tạo một store procedure tên GetProducts trong CSDL Northwind theo các bước sau :

Bước 1: mở VS 2008 chọn File -> New Project.

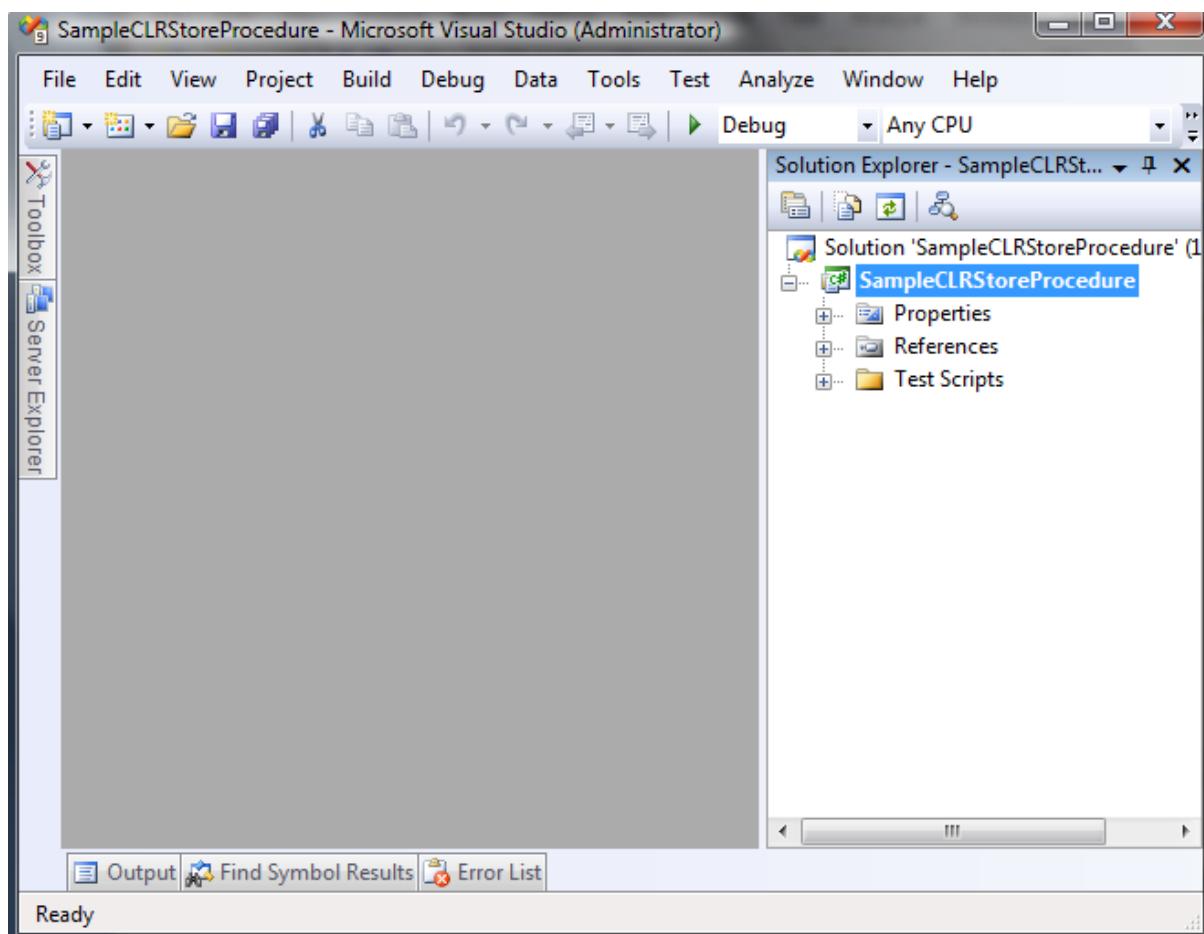
Chọn Database -> SQL Server Project

Đặt tên Project: SampleCLRStoreProcedure



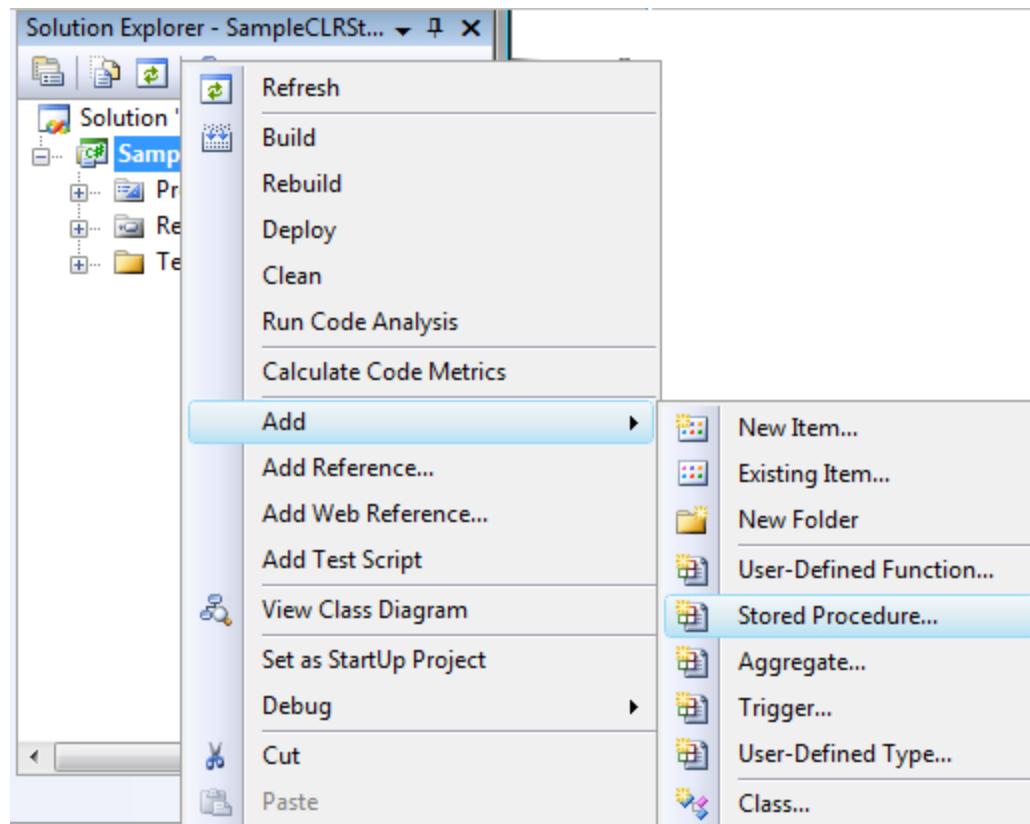
H 8.2 Tạo Project mới

Sau khi tạo Project, ta chọn Database cần làm việc; trong ví dụ này ta chọn Northwind, VS 2008 mở cửa sổ chính của project.



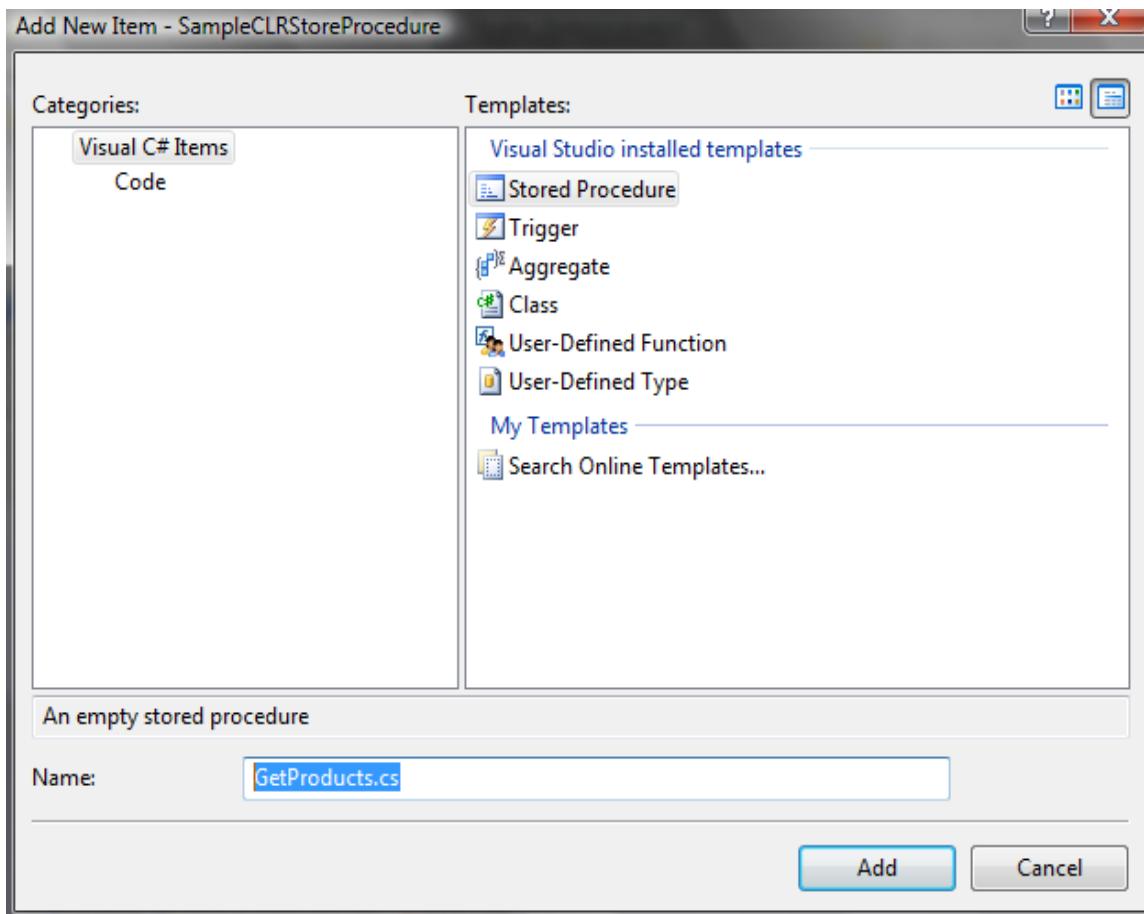
H 8.3 Cửa sổ chính Project

Bước 2: Thêm vào project 1 class dùng để tạo store procedure
Right click tên Project -> Add -> Store Procedure



H 8.3 Thêm Store Procedure

Bước 4: Đặt tên Store Procedure là GetProducts



H 8.4 Tạo store GetProducts

Bước 5: Chúng ta tạo store GetProducts có 1 tham số truyền vào kiểu int để nhận giá trị CategoryID do người dùng nhập vào

```
GetProducts.cs*
StoredProcedures
GetProducts(SqlInt32 CategoryID)

public partial class StoredProcedures
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void GetProducts(SqlInt32 CategoryID)
    {
        try
        {
            SqlConnection cnn = new SqlConnection("Context connection=true");

            SqlCommand cmd = new SqlCommand("SELECT ProductID,ProductName,CategoryID FROM Products" +
                "WHERE CategoryID=@CategoryID", cnn);
            SqlParameter param = new SqlParameter();
            param = cmd.Parameters.AddWithValue("@CategoryID", CategoryID);

            cnn.Open();
            SqlDataReader reader;
            reader = cmd.ExecuteReader();
            SqlContext.Pipe.Send(reader);

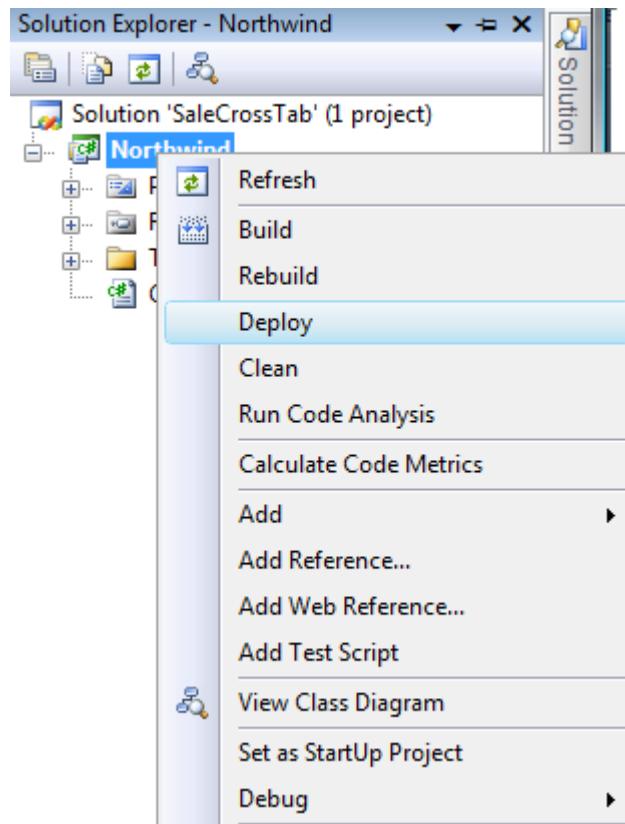
            cnn.Close();
        }
        catch (SqlException ex)
        {
            SqlContext.Pipe.Send(ex.Message);
        }
    }
};
```

H 8.5 Viết code Store GetProducts

Chú ý: kiểu dữ liệu dùng cho @CategoryID là SqlInt32

Bước 6: Deploy Store GetProducts

Right click Project -> Deploy

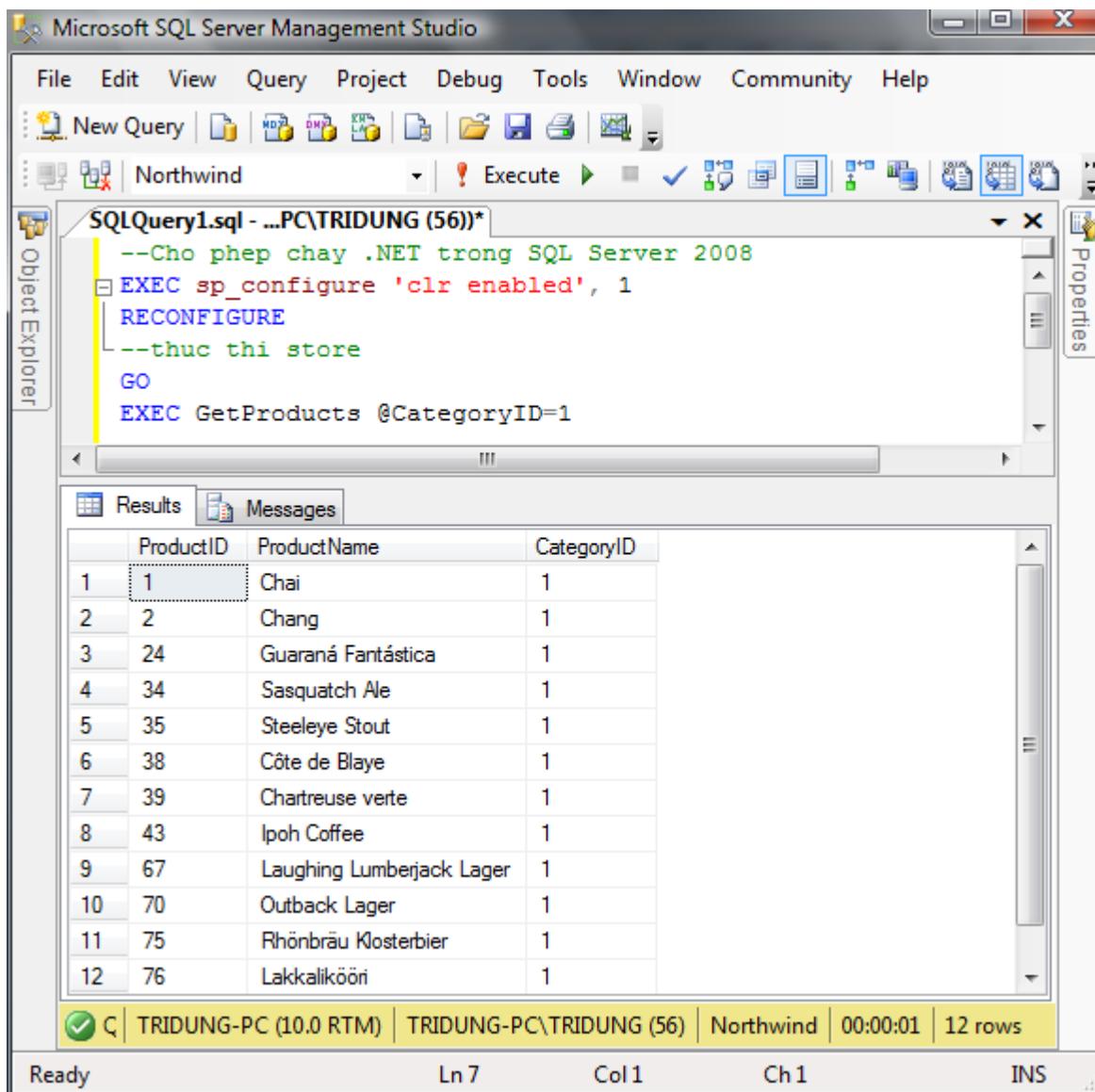


H 8.6 Deploy store

Nếu kết nối được với Server, Visual Studio sẽ thông báo Deploy Succeeded. Nếu thất bại sẽ xuất thông báo trong mục Output: Deploy Failed.

Bước 7: Để thực thi store procedure vừa tạo chúng ta sẽ thực hiện trong SQL Server 2008.

Chúng ta mở SQL Server Management Studio -> chọn New Query



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the Northwind database is selected. A query window titled "SQLQuery1.sql" contains the following T-SQL code:

```
--Cho phép chạy .NET trong SQL Server 2008
EXEC sp_configure 'clr enabled', 1
RECONFIGURE
--thực thi store
GO
EXEC GetProducts @CategoryID=1
```

The Results tab displays the output of the query, which is a table of products from the Northwind database:

	ProductID	ProductName	CategoryID
1	1	Chai	1
2	2	Chang	1
3	24	Guaraná Fantástica	1
4	34	Sasquatch Ale	1
5	35	Steeleye Stout	1
6	38	Côte de Blaye	1
7	39	Chartreuse verte	1
8	43	Ipoh Coffee	1
9	67	Laughing Lumberjack Lager	1
10	70	Outback Lager	1
11	75	Rhönbräu Klosterbier	1
12	76	Lakkalikööri	1

The status bar at the bottom shows: Ready | Ln 7 | Col 1 | Ch 1 | INS | TRIDUNG-PC (10.0 RTM) | TRIDUNG-PC\TRIDUNG (56) | Northwind | 00:00:01 | 12 rows.

H 8.7 Thực thi Store

8.3 Xây dựng CLR User-Defined Function

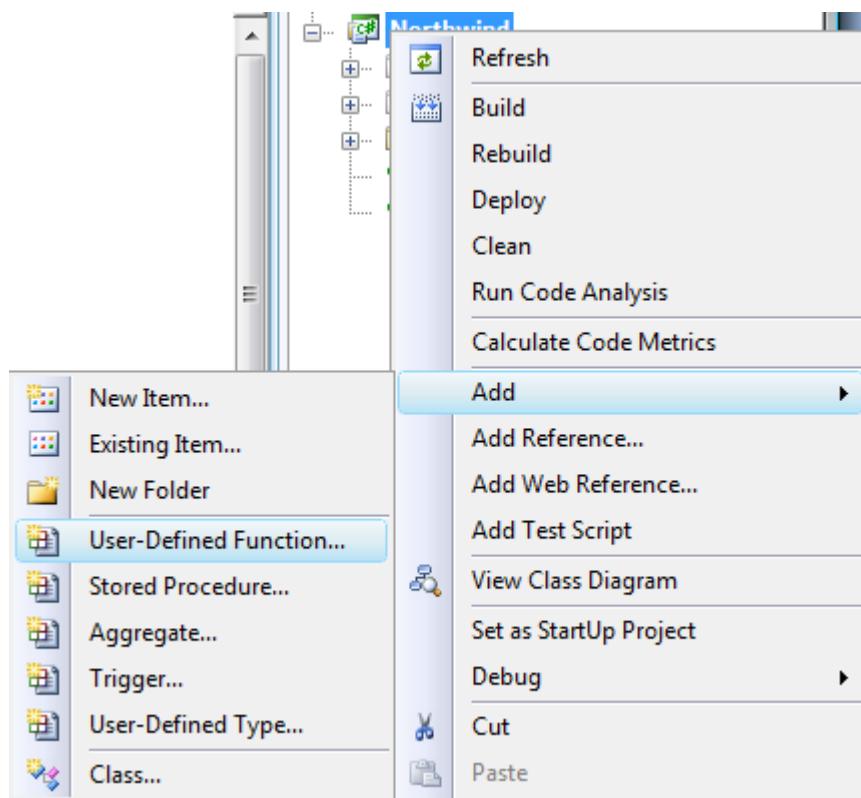
User-Defined Function có thể nhận tham số, thực hiện tính toán, các thao tác khác và trả về một giá trị nào đó.

Tương tự T-SQL Function, CLR User-Defined Function có 2 loại Function:

- Scalar function trả về một giá trị đơn.
- Table Valued function: trả về một tập hợp các dòng dữ liệu.

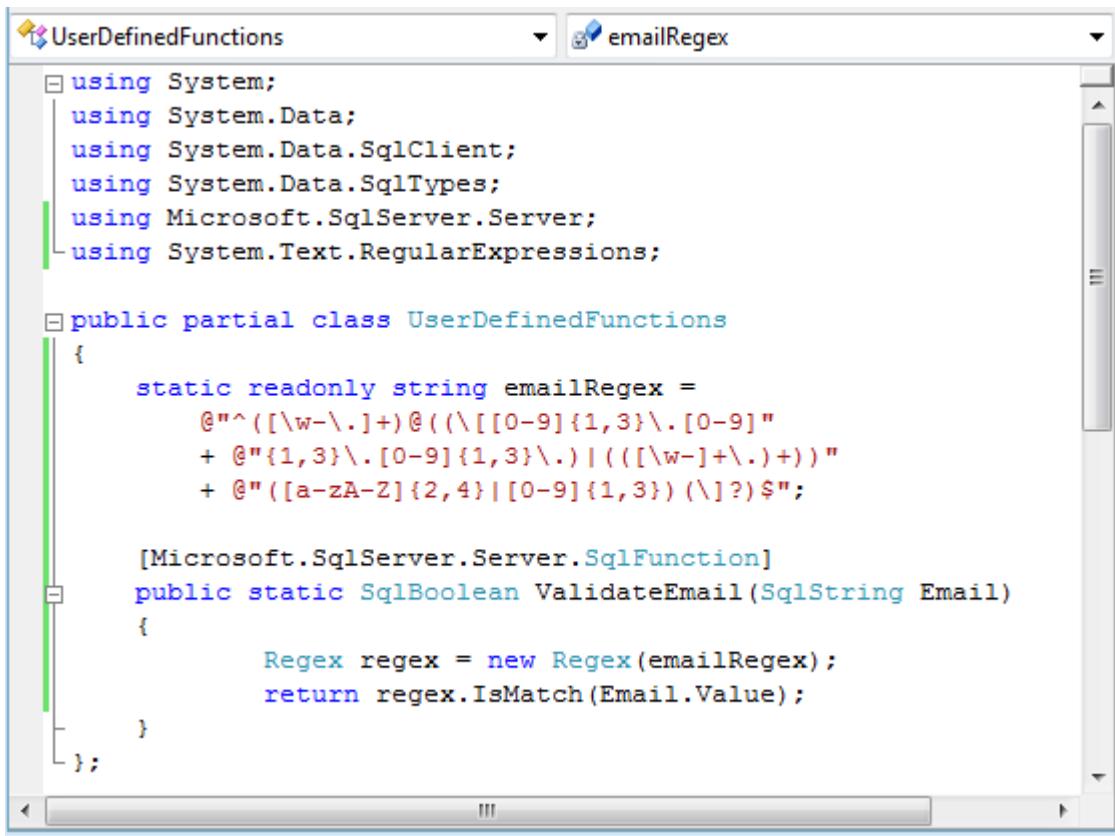
Ví dụ: Chúng ta sẽ tạo một User-defined Function dùng kiểm tra Email có hợp lệ hay không để hiểu rõ hơn về cách tạo và sử dụng Function.

Bước 1: Add ->User-Defined Function



H 8.8 Thêm mới Function

Bước 2 : Viết code cho hàm kiểm tra Email tên ValidateEmail



```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Text.RegularExpressions;

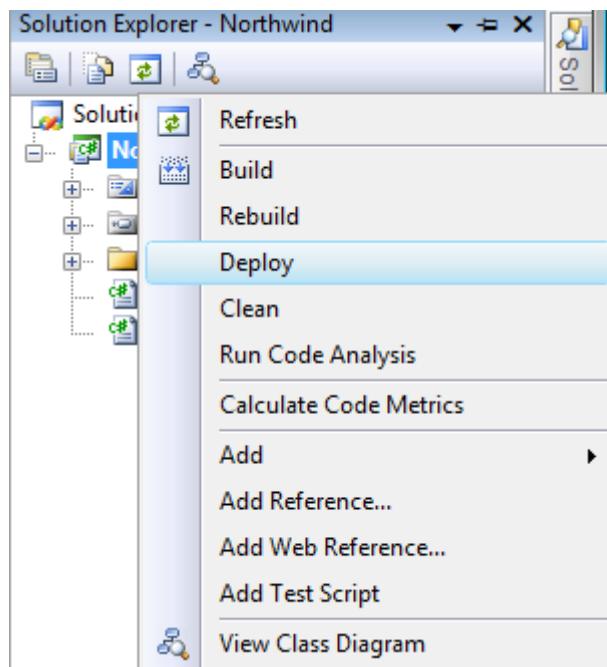
public partial class UserDefinedFunctions
{
    static readonly string emailRegex =
        @"^([\w\-.]+\@([[[0-9]{1,3}\.][0-9]" +
        + @"{1,3}\.][0-9]{1,3}\.)|(([[\w-]+\.)+))" +
        + @"([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";

    [Microsoft.SqlServer.Server.SqlFunction]
    public static SqlBoolean ValidateEmail(SqlString Email)
    {
        Regex regex = new Regex(emailRegex);
        return regex.IsMatch(Email.Value);
    }
};
```

H 8.9 Nội dung hàm ValidateEmail

Bước 3 :Nhấn phải chuột | Deploy thì VS.Net sẽ cài đặt hàm này vào SQL Server tự động .

*Lưu ý: Khi deploy thì VS.Net sẽ tạo một tập tin *.dll trong thư mục Bin\Debug để chúng ta có thể cài đặt trên cơ sở dữ liệu khác bằng cách sử dụng các lệnh T-SQL .*



H 8.10 Deploy hàm vừa tạo

Bước 4: Đăng ký thư viện .dll để sử dụng trong SQL Server 2008 (đăng ký bằng tay sử dụng lệnh T-SQL). Nếu chúng ta đã chọn Deploy ở bước 3 thì có thể sang bước 5.

```
--Dang ky thu vien .NET su dung trong SQL Server 2008
sp_Configure 'CLR Enabled', 1 Reconfigure With Override
GO
DROP ASSEMBLY Validate
GO
CREATE ASSEMBLY Validate
FROM 'C:\Users\TRIDUNG\Documents\Visual Studio 2008\Projects\SaleCrossTab\' +
'Northwind\bin\Debug\Northwind.dll'
WITH PERMISSION_SET = SAFE
GO
```

H 8.11 Đăng ký dll

Bước 5: Trong trường hợp ta không sử dụng tên hàm có sẵn trong .Net , ta có thể tạo hàm mới tên RegEx trong SQL Server 2008 dựa trên hàm đã tạo trong .Net.

Sau khi tạo hàm, chúng ta có thể gọi hàm này trong lệnh SELECT

The screenshot shows the SQL Server Management Studio interface. In the top query editor window, titled 'Query.sql - TRIDUNG\TRIDUNG (53)', the following T-SQL code is displayed:

```
GO
--Dinh nghia ham moi dua tren ham ValidateEmail trong .NET
CREATE FUNCTION dbo.RegEx
(
    @EmailAddress AS nvarchar(100)
)
RETURNS Bit
AS EXTERNAL NAME Validate.CountOrderByCustomer.ValidateEmail

GO
SELECT dbo.RegEx('daithanh@hotmail.com')
```

The results pane below shows the output of the 'SELECT' statement:

(No column name)
1

The status bar at the bottom indicates: Quer... | TRIDUNG-PC (10.0 RTM) | TRIDUNG-PC\TRIDUNG (52) | Northwind | 00:00:00 | 1 rows.

H 8.12 Tạo và gọi hàm RegEx

8.4 Xây dựng CLR User-Defined Types

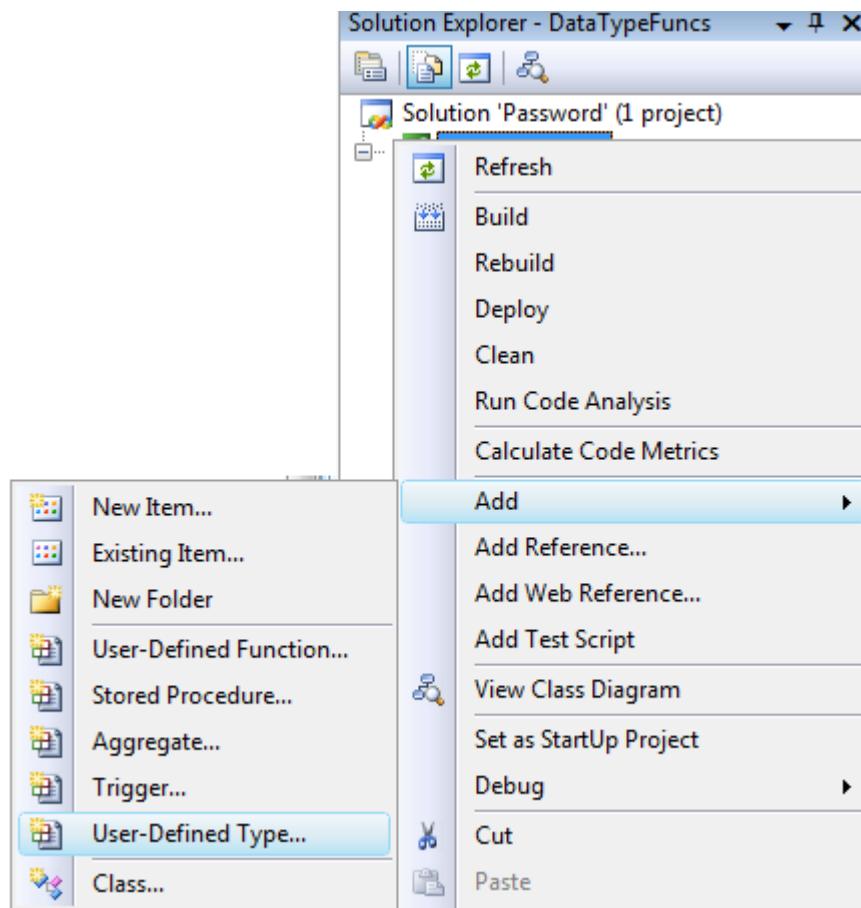
SQL Server 2008 cho phép ta tạo những đối tượng cơ sở dữ liệu trong .NET Framework CLR. Chúng ta có thể dùng User-Defined Types (UDTs) để tạo ra các kiểu dữ liệu mới để lưu trữ các đối tượng của CLR trong SQL Server. UDTs có thể chứa nhiều thành phần và có nhiều thuộc tính khác với những kiểu dữ liệu sẵn có trong SQL Server.

Để xây dựng UDTs, ta thực hiện tương tự nhưCLR User-defined Function:

- Viết code và biên dịch thư viện tạo UDTs trong .NET.
- Đăng ký thư viện vừa tạo trong SQL Server (CREATE ASSEMBLY)
- Tạo UDTs trong SQL Server
- Tạo bảng, tham số sử dụng UDTs.

Ví dụ : Tạo một User-defined Types dùng để kiểm tra Mật khẩu nhập vào trong CSDL

Bước 1: Thêm class dạng User-defined Type vào project (tên project là ThuVien)



H 8.13 Thêm class User-defined Type

Bước 2 : Viết code cho User-defined vừa thêm vào

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Text.RegularExpressions;
```

```
using System.IO;
```

```
[Serializable]
[SqlUserDefinedType(Format.UserDefined,
    IsByteOrdered=true, MaxByteSize=8000)]
public struct MatKhau : INullable, IBinarySerialize
{
    private bool m_Null;
    public string m_ChuoIMK;

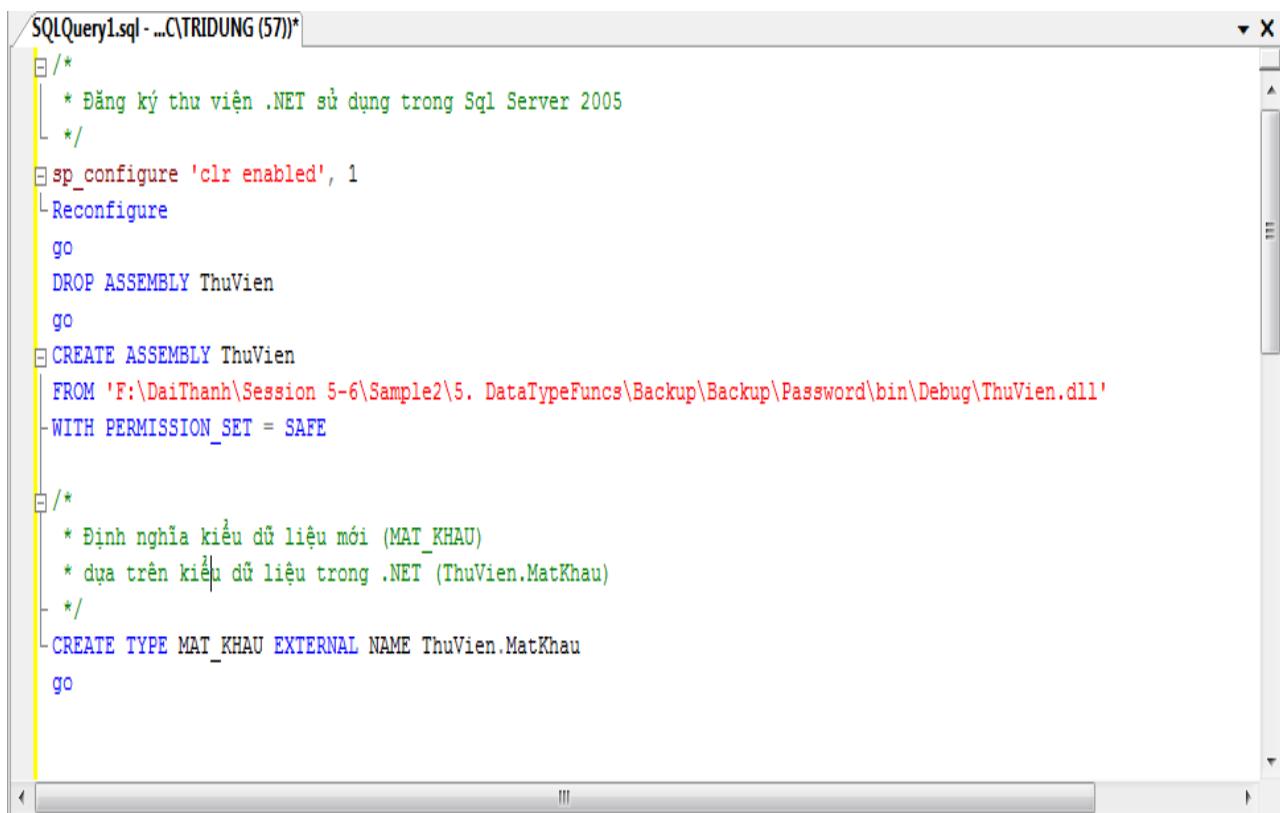
    public override string ToString()
    {
        return m_ChuoIMK;
    }
    public bool IsNull
    {
        get
        {
            return m_Null;
        }
    }
    public static MatKhau Null
    {
        get
        {
            MatKhau matKhau = new MatKhau();
            matKhau.m_Null = true;
            return matKhau;
        }
    }
}

public static MatKhau Parse(SqlString chuo)
{
    if (chuo.IsNull)
    {
        return Null;
    }
}
```

```
MatKhau matKhau = new MatKhau();
bool hopLe = Regex.IsMatch(chuoi.ToString(), "[a-zA-Z]{4,}[0-9]");
if (hopLe == true)
{
    matKhau.m_ChuoIMK = chuoi.ToString();
    return matKhau;
}
throw new SqlTypeException("Mat khau khong hop le.");
}
public void Write(BinaryWriter writer)
{
    writer.Write(m_ChuoIMK);
}
public void Read(BinaryReader reader)
{
    m_ChuoIMK = reader.ReadString();
}
```

Bước 3 : Build và Deploy User-defined Type tương tự như phần CLR User-defined Function.

Bước 4: Đăng ký .dll vào SQL Server



```
SQLQuery1.sql - ...C\TRIDUNG (57)*
/*
 * Đăng ký thư viện .NET sử dụng trong Sql Server 2005
 */
sp_configure 'clr enabled', 1
Reconfigure
go
DROP ASSEMBLY ThuVien
go
CREATE ASSEMBLY ThuVien
FROM 'F:\DaiThanh\Session 5-6\Sample2\5. DataTypeFuncs\Backup\Backup\Password\bin\Debug\ThuVien.dll'
WITH PERMISSION_SET = SAFE

/*
 * Định nghĩa kiểu dữ liệu mới (MAT_KHAU)
 * dựa trên kiểu dữ liệu trong .NET (ThuVien.MatKhau)
 */
CREATE TYPE MAT_KHAU EXTERNAL NAME ThuVien.MatKhau
go
```

H 8.14 Đăng ký dll

Bước 5 : Tạo bảng Users và thêm 3 dòng dữ liệu, trong đó có dòng dữ liệu đầu tiên không đúng với kiểu MAT_KHAU mà ta đã định nghĩa là phải bao gồm cả chữ lẫn số. Trong ví dụ này , chúng ta tạo ra một kiểu dữ liệu tên MAT_KHAU có dạng 4 ký tự đầu là chữ cái sau đó là các số nguyên từ 0-9

SQLQuery1.sql - ...C\TRIDUNG (57)*

```

CREATE TABLE Users
(
    UserName VARCHAR(50) NOT NULL,
    MyPassword MAT_KHAU NOT NULL, -- sử dụng kiểu dữ liệu mới
    CONSTRAINT [PK_UserName] PRIMARY KEY(UserName)
)
go
INSERT INTO Users VALUES ('Roger','jinglebook') --Mật khẩu không hợp lệ
INSERT INTO Users VALUES ('Roger','jingle49')
INSERT INTO Users VALUES ('Marcus','steelman009')

```

Messages

```

A .NET Framework error occurred during execution of user-defined routine or aggregate "MAT_KHAU".
System.Data.SqlTypes.SqlTypeException: Mat khau khong hop le.
System.Data.SqlTypes.SqlTypeException:
    at MatKhau.Parse(SqlString chuo)
.

The statement has been terminated.

(1 row(s) affected)

(1 row(s) affected)

```

Query completed wi... | TRIDUNG-PC (10.0 RTM) | TRIDUNG-PC\TRIDUNG (57) | AdventureWorks | 00:00:00 | 0 rows

H 8.15 Thông báo lỗi sau khi nhập dữ liệu

Theo như thông báo lỗi ở trên thì chỉ có 2 dòng dữ liệu 2 và 3 được thêm vào bảng Users, dòng đầu tiên không đúng với kiểu dữ liệu.

SQLQuery1.sql - ...C\TRIDUNG (57)*

```

go
SELECT * FROM Users

```

Results

	UserName	MyPassword
1	Marcus	0x0B737465656C6D616E303039
2	Roger	0x086A696E676C653439

-PC (10.0 RTM) | TRIDUNG-PC\TRIDUNG (57) | AdventureWorks | 00:00:00 | 2 rows

H 8.16 Liệt kê dữ liệu vừa thêm

Dữ liệu trong cột MyPassword đã được chuyển sang dạng Binary trong câu lệnh

```
public void Write(BinaryWriter writer)
{
    writer.Write(m_Chuoimk);
}
```

Để có thể thấy được mật khẩu ta có thể sử dụng hàm Cast trong câu lệnh Select

```
Select UserName,Cast(MyPassword as varchar) as MyPassword from Users
```

Kết quả

	UserName	MyPassword
1	Marcus	steelman009
2	Roger	jingle49

Trên đây là 2 ví dụ đơn giản về User-defined Function và User-defined Type, để tìm hiểu chi tiết hơn về CLR Integration trong SQL Server 2008. Chúng ta sử dụng kỹ thuật này để viết các hàm phức tạp bằng cách sử dụng các ngôn ngữ .Net (C#, VB.Net,...) thay vì phải viết bằng các lệnh T-SQL .

Các bạn có thể tham khảo theo Book Online 2008 và MSDN của Microsoft.