

Session - 8

Accessing Data

Welcome to the Session, **Accessing Data**.

This session describes SELECT statement. It also explains the expressions and the various clauses used with SELECT statement. Finally, the session introduces the new xml data type and describes how to work with XML data in SQL Server 2012 tables. A brief look at the XQuery language, which is used to query XML data, is also discussed in the session.

In this Session, you will learn to:

- Describe the SELECT statement, its syntax, and use
- Explain the various clauses used with SELECT
- State the use of ORDER BY clause
- Describe working with typed and untyped XML
- Explain the procedure to create, use, and view XML schemas
- Explain use of XQuery to access XML data

8.1 Giới thiệu

Câu lệnh SELECT là một lệnh lõi được sử dụng để truy cập dữ liệu trong SQL Server 2012. XML cho phép các nhà phát triển xây dựng tập hợp các thẻ riêng của họ và làm cho nó có thể cho các chương trình khác hiểu những thẻ này. XML là phương tiện ưa thích dành cho các nhà phát triển để lưu trữ, định dạng và quản lý dữ liệu trên web.

8.2 Câu lệnh SELECT

Bảng với dữ liệu của nó có thể được xem bằng cách sử dụng câu lệnh SELECT. Câu lệnh SELECT trong một truy vấn sẽ hiển thị thông tin cần thiết trong bảng.

Câu lệnh SELECT lấy các hàng và cột từ một hoặc nhiều bảng. Đầu ra của câu lệnh SELECT là một bảng khác gọi là tập kết quả. Câu lệnh SELECT còn nối hai bảng hoặc lấy một tập hợp con các cột từ một hoặc nhiều bảng. Câu lệnh SELECT định nghĩa các cột được sử dụng cho một truy vấn. Cú pháp của câu lệnh SELECT có thể bao gồm một loạt các biểu thức cách nhau bằng dấu phẩy. Mỗi biểu thức trong câu lệnh là một cột trong tập kết quả. Những cột xuất hiện theo cùng một trình tự như thứ tự của biểu thức trong câu lệnh SELECT.

Câu lệnh SELECT lấy các hàng từ cơ sở dữ liệu và cho phép lựa chọn một hoặc nhiều hàng hoặc cột từ một bảng. Sau đây là cú pháp cho câu lệnh SELECT.

Cú pháp:

```
SELECT <column_name1>...<column_nameN> FROM <table_name>
```

trong đó,

table_name: là bảng từ đó dữ liệu sẽ được hiển thị.

<column_name1>...<column_nameN>: là những cột sẽ được hiển thị.

Ghi chú - Tất cả các lệnh trong SQL Server 2012 không kết thúc bằng dấu chấm phẩy.

8.2.1 SELECT không có FROM

Nhiều phiên bản SQL sử dụng FROM trong truy vấn, nhưng trong tất cả các phiên bản từ SQL Server 2005, bao gồm cả SQL Server 2012, có thể sử dụng các câu lệnh SELECT mà không sử dụng mệnh đề FROM. Code Snippet 1 cho thấy việc sử dụng câu lệnh SELECT mà không sử dụng mệnh đề FROM.

Code Snippet 1:

```
SELECT LEFT('International',5)
```

Đoạn mã này sẽ hiển thị chỉ 5 ký tự đầu tiên từ bên trái nhất của từ 'International'.

Kết quả được trình bày trong Hình 8.1.

(No column name)	
1	Inter

Hình 8.1: First Five Characters from the Extreme left of the Word

8.2.2 Displaying All Columns

Dấu sao (*) được sử dụng trong câu lệnh SELECT để lấy tất cả các cột từ bảng. Nó được sử dụng như một cách viết tắt để liệt kê tất cả các tên cột trong các bảng có tên trong mệnh đề FROM. Sau đây là cú pháp để chọn tất cả các cột.

Cú pháp :

```
SELECT * FROM <table_name>
```

trong đó,

*: chỉ ra tất cả các cột của bảng có tên trong mệnh đề FROM.

<table_name>: là tên của bảng mà từ đó thông tin sẽ được lấy ra. Có thể đưa vào số lượng bảng bất kỳ. Khi hai hoặc nhiều bảng được sử dụng, hàng của mỗi bảng được ánh xạ với hàng của các bảng khác. Hoạt động này mất rất nhiều thời gian nếu dữ liệu trong các bảng rất lớn. Do đó, khuyến khích nên sử dụng cú pháp này với một điều kiện.

Code Snippet 2 cho thấy việc sử dụng '*' trong câu lệnh SELECT.

Code Snippet 2:

```
USE AdventureWorks2012
SELECT * FROM HumanResources.Employee
GO
```

Kết quả một phần của Code Snippet 2 với một số cột của bảng HumanResources.Employee được trình bày trong Hình 8.2.

	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	On	
1	1	295847284	adventure-works\ken0	0x	0	
2	2	245797967	adventure-works\tem0	0x58	1	
3	3	509647174	adventure-works\roberto0	0x5AC0	2	
4	4	112457891	adventure-works\rob0	0x5AD6	3	
5	5	695256908	adventure-works\gail0	0x5ADA	3	
6	6	998320692	adventure-works\jossef0	0x5ADE	3	
7	7	134969118	adventure-works\dylan0	0x5AE1	3	
8	8	811994146	adventure-works\diane1	0x5AE158	4	

Hình 8.2: Displaying All Columns

8.2.3 Hiện thị các cột đã chọn

Câu lệnh SELECT hiển thị hoặc trả lại một số cột có liên quan được người dùng lựa chọn hoặc được đề cập trong câu lệnh. Để hiển thị các cột cụ thể, kiến thức về tên cột có liên quan trong bảng là cần thiết. Sau đây là cú pháp để chọn các cột cụ thể.

Cú pháp:

```
SELECT <column_name1>..<column_nameN> FROM <table_name>
```

trong đó,

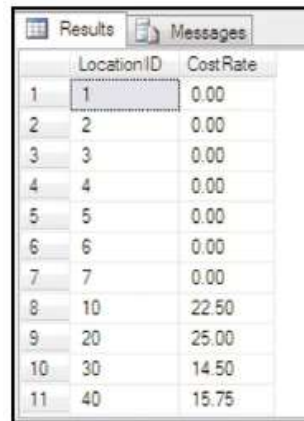
<column_name1>..<column_nameN>: là những cột sẽ được hiển thị.

Ví dụ, để hiển thị các tỷ lệ chi phí tại các địa điểm khác nhau từ bảng Production.Location trong cơ sở dữ liệu AdventureWorks2012, câu lệnh SELECT như được trình bày trong Code Snippet 3.

Code Snippet 3:

```
USE AdventureWorks2012
SELECT LocationID, CostRate FROM Production.Location
GO
```

Hình 8.3 trình bày các cột **LocationID** và **CostRate** từ cơ sở dữ liệu AdventureWorks2012.



	LocationID	CostRate
1	1	0.00
2	2	0.00
3	3	0.00
4	4	0.00
5	5	0.00
6	6	0.00
7	7	0.00
8	10	22.50
9	20	25.00
10	30	14.50
11	40	15.75

Hình 8.3: locationID and CastRate Columns

8.3 Các biểu thức khác nhau với SELECT

Câu lệnh SELECT cho phép người dùng chỉ ra các biểu thức khác nhau để xem tập kết quả theo một cách có trật tự. Những biểu thức này gán các tên gọi khác nhau cho các cột trong tập kết quả, tính toán các giá trị, và loại bỏ các giá trị trùng lặp.

8.3.1 Sử dụng các hằng số trong tập kết quả

Các hằng số dạng chuỗi ký tự được dùng khi các cột ký tự được ghép lại. Chúng giúp việc định dạng phù hợp hoặc khả năng đọc được. Những hằng số này không được chỉ định làm một cột riêng biệt trong tập kết quả

Thường có hiệu quả hơn cho ứng dụng để dựng các giá trị hằng số vào trong các kết quả khi chúng được hiển thị, hơn là sử dụng máy chủ để kết hợp các giá trị hằng số. Ví dụ, để đưa vào ' ' và '→' trong tập kết quả để hiển thị tên quốc gia, mã vùng quốc gia, và nhóm tương ứng của nó, câu lệnh **SELECT** được trình bày trong Code Snippet 4.

Code Snippet 4:

```
USE AdventureWorks2012
SELECT [Name] + ' : ' + CountryRegionCode + '→' + [Group] FROM Sales.SalesTerritory
GO
```

Hình 8.4 displays hiển thị tên quốc gia, mã vùng quốc gia, và nhóm tương ứng từ Sales.SalesTerritory của cơ sở dữ liệu AdventureWorks2012

	(No column name)
1	Northwest : US -> North America
2	Northeast : US -> North America
3	Central : US -> North America
4	Southwest : US -> North America
5	Southeast : US -> North America
6	Canada : CA -> North America
7	France : FR -> Europe
8	Germany : DE -> Europe
9	Australia : AU -> Pacific
10	United Kingdom : GB -> Europe

Hình 8.4: Country Name, Country Region Code, and Corresponding Group

8.3.2 Đổi tên các cột tập kết quả

Khi các cột được hiển thị trong tập kết quả chúng đi kèm với các đầu đề tương ứng được chỉ ra trong bảng. Những đầu đề này có thể được thay đổi, đổi tên, hoặc có thể được chỉ định một tên mới bằng cách sử dụng mệnh đề AS. Do đó, bằng cách tùy biến các đầu đề, chúng trở nên dễ hiểu hơn và có ý nghĩa.

Đoạn mã 5 trình bày cách hiển thị '**ChangedDate**' như đầu đề cho cột **ModifiedDate** trong bảng **dbo.Individual**, câu lệnh SELECT.

Code Snippet 5:

```
USE CUST_DB
SELECT ModifiedDate as 'ChangedDate' FROM dbo.Individual
GO
```

Kết quả hiển thị '**ChangedDate**' làm đầu đề cho cột **ModifiedDate** trong bảng **dbo.Individual**. Hình 8.5 trình bày đầu đề ban đầu và đầu đề đã thay đổi.

	ModifiedDate
1	1981-02-02

	ChangedDate
1	1981-02-02

Hình 8.5: Column Heading Modified to ChangedDate

8.3.3 Tính toán các giá trị trong tập kết quả

Câu lệnh SELECT có thể chứa các biểu thức toán học bằng cách áp dụng các toán tử cho một hoặc nhiều cột. Nó cho phép một tập kết quả để chứa các giá trị không tồn tại trong bảng cơ sở, nhưng được tính toán từ các giá trị được lưu trữ trong bảng cơ sở.

Ghi chú - Bảng được sử dụng trong mệnh đề FROM của một truy vấn được gọi là một bảng cơ sở

Ví dụ, hãy xem xét bảng **Production.ProductCostHistory** từ cơ sở dữ liệu **AdventureWorks2012**. Xem xét ví dụ trong đó người sản xuất quyết định đưa ra 15% giảm chi phí tiêu chuẩn của tất cả các sản phẩm. Số tiền giảm giá không tồn tại nhưng có thể được tính bằng cách thực hiện câu lệnh SELECT được trình bày trong Code Snippet 6.

Code Snippet 6:

```
USE AdventureWorks2012
SELECT ProductID, StandardCost, StandardCost * 0.15 as Discount FROM
Production.ProductCostHistory
GO
```

Hình 8.6 trình bày kết quả khi số tiền giảm giá được tính toán sử dụng câu lệnh **SELECT**.

	ProductID	StandardCost	Discount
1	707	12.0278	1.804170
2	707	13.8782	2.081730
3	707	13.0863	1.962945
4	708	12.0278	1.804170
5	708	13.8782	2.081730
6	708	13.0863	1.962945
7	709	3.3963	0.509445
8	710	3.3963	0.509445
9	711	12.0278	1.804170
10	711	13.8782	2.081730
11	711	13.0863	1.962945

Hình 8.6: Calculated Discount Amount

8.3.4 Sử dụng *DISTINCT*

Từ khóa *DISTINCT* ngăn chặn việc lấy ra các bản ghi trùng lặp. Nó giúp loại bỏ các hàng lặp lại từ tập kết quả của câu lệnh *SELECT*. Ví dụ, nếu cột *StandardCost* được chọn mà không sử dụng từ khóa *DISTINCT*, nó sẽ hiển thị tất cả các chỉ phí tiêu chuẩn hiện diện trong bảng. Khi sử dụng từ khóa *DISTINCT* trong truy vấn, SQL Server sẽ hiển thị mọi bản ghi của *StandardCost* chỉ một lần như được trình bày trong Code Snippet 7.

Code Snippet 7:

```
USE AdventureWorks2012
SELECT DISTINCT StandardCost FROM Production.ProductCostHistory
GO
```

8.3.5 Sử dụng *TOP* và *PERCENT*

Từ khóa *TOP* sẽ chỉ hiển thị tập hợp đầu tiên của các hàng như một tập kết quả. Tập hợp các hàng hoặc là hạn chế cho một số hoặc phần trăm các hàng. Biểu thức *TOP* cũng có thể được sử dụng với các câu lệnh khác như *INSERT*, *UPDATE*, và *DELETE*. Sau đây là cú pháp cho từ khóa *TOP*.

Cú pháp :

```
SELECT [ALL|DISTINCT] [TOP expression [PERCENT] [WITH TIES]]
```

trong đó,

expression: là số hoặc tỷ lệ phần trăm các hàng để được trả lại như là kết quả.

PERCENT: trả về số hàng bị giới hạn bởi tỷ lệ phần trăm.

WITH TIES: là số hàng bổ sung sẽ được hiển thị.

Câu lệnh SELECT có các mệnh đề khác nhau liên kết với nó. Trong phần này, từng mệnh đề sẽ được thảo luận chi tiết.

8.3.6 SELECT với INTO

Mệnh đề INTO tạo ra một bảng mới và chèn các hàng và cột được liệt kê trong câu lệnh SELECT vào nó. Mệnh đề INTO còn chèn các hàng hiện có vào bảng mới. Để thực hiện thi mệnh đề này với câu lệnh SELECT, người dùng phải có sự cho phép để CREATE TABLE trong cơ sở dữ liệu đích.

Cú pháp:

```
SELECT <column_name1> .. <column_nameN> [INTO new_table] FROM table_list
```

trong đó,

new_table: là tên của bảng mới sẽ được tạo ra.

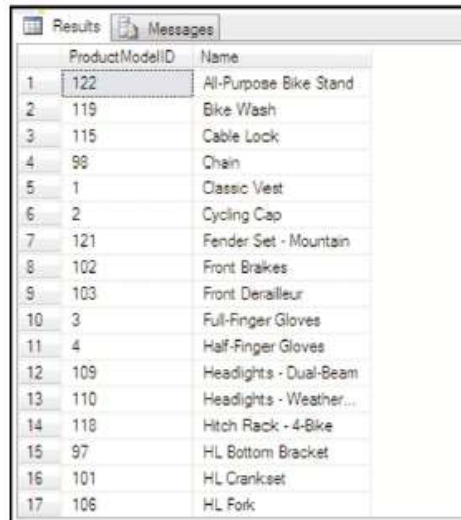
Code Snippet 8 sử dụng mệnh đề INTO tạo ra một bảng mới Production.ProductName với các chi tiết như mã sản phẩm và tên của nó từ bảng Production.ProductModel.

Code Snippet 8:

```
USE AdventureWorks2012
SELECT ProductModelID, Name INTO Production.ProductName FROM Production.
ProductModel
GO
```

Sau khi thực thi đoạn mã này, một thông báo nêu ra '(128 row(s) affected)' (128 hàng bị ảnh hưởng) được hiển thị.

Nếu một truy vấn được viết để hiển thị các hàng của bảng mới, đầu ra sẽ như được trình bày trong Hình 8.7.



	ProductModelID	Name
1	122	All-Purpose Bike Stand
2	119	Bike Wash
3	115	Cable Lock
4	99	Chain
5	1	Classic Vest
6	2	Cycling Cap
7	121	Fender Set - Mountain
8	102	Front Brakes
9	103	Front Derailleur
10	3	Full-Finger Gloves
11	4	Half-Finger Gloves
12	109	Headlights - Dual-Beam
13	110	Headlights - Weather...
14	118	Hitch Rack - 4-Bike
15	97	HL Bottom Bracket
16	101	HL Crankset
17	106	HL Fork

Hình 8.7: New Table Created

8.3.7 SELECT với WHERE

Mệnh đề WHERE với câu lệnh SELECT được sử dụng để chọn hoặc hạn chế có điều kiện các bản ghi lấy ra bằng truy vấn. Mệnh đề WHERE chỉ ra biểu thức Boolean để kiểm tra các hàng được trả lại bằng truy vấn. Hàng được trả lại nếu biểu thức là true và bị loại bỏ nếu là false.

Cú pháp:

```
SELECT <column_name1>...<column_nameN> FROM <table_name> WHERE <
search_condition>
```

trong đó,

search_condition: là điều kiện để được đáp ứng bằng các hàng .

Table 8.trình bày các toán tử khác nhau có thể được sử dụng với mệnh đề WHERE.

Toán tử	Mô tả
=	Bằng
<>	Không bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
!	Không

Toán tử	Mô tả
BETWEEN	Giữa một phạm vi
LIKE	Tìm kiếm một mẫu có thứ tự
IN	Ở trong một phạm vi

Table 8.1: Operators

Code Snippet 9 trình bày toán tử bằng với mệnh đề WHERE để hiển thị dữ liệu với **EndDate 6/30/2007 12:00:00 AM**.

Code Snippet 9:

```
USE AdventureWorks2012
SELECT * FROM Production.ProductCostHistory WHERE EndDate = '6/30/2007 12:00:00 AM'
GO
```

Code Snippet 9 sẽ trả lại tất cả các bản ghi từ bảng Production.ProductCostHistory trong đó có ngày kết thúc là **'6/30/2007 12:00:00 AM'**.

Kết quả SELECT với mệnh đề WHERE được trình bày trong hình 8.8.

	ProductID	StartDate	EndDate	StandardCost	ModifiedDate
1	707	2006-07-01 00:00:00	2007-06-30 00:00:00	13.8782	2007-06-30 00:00:00
2	708	2006-07-01 00:00:00	2007-06-30 00:00:00	13.8 82	2007-06-30 00:00:00
3	711	2006-07-01 00:00:00	2007-06-30 00:00:00	13.8782	2007-06-30 00:00:00
4	712	2006-07-01 00:00:00	2007-06-30 00:00:00	5.2297	2007-06-30 00:00:00
5	713	2006-07-01 00:00:00	2007-06-30 00:00:00	29.0807	2007-06-30 00:00:00
6	714	2006-07-01 00:00:00	2007-06-30 00:00:00	29.0807	2007-06-30 00:00:00
7	715	2006-07-01 00:00:00	2007-06-30 00:00:00	29.0807	2007-06-30 00:00:00
8	716	2006-07-01 00:00:00	2007-06-30 00:00:00	29.0807	2007-06-30 00:00:00
9	717	2006-07-01 00:00:00	2007-06-30 00:00:00	722.2568	2007-06-30 00:00:00

Hình 8.8: SELECT with WHERE clause

Tất cả các truy vấn trong SQL sử dụng ngoặc đơn để bao bọc các giá trị văn bản. Ví dụ, hãy xem xét truy vấn sau đây, mà lấy tất cả các bản ghi từ bảng Person.Address có Bothell là thành phố.

Code Snippet 10 trình bày toán tử bằng với mệnh đề WHERE để hiển thị dữ liệu với địa chỉ có thành phố là Bothell.

Code Snippet 10:

```
USE AdventureWorks2012
SELECT DISTINCT StandardCost FROM Production.ProductCostHistory
GO
```

Kết quả của truy vấn được trình bày trong Hình 8.9.

	AddressID	AddressLine1	AddressLine2	City	StateProvinceID	PostalCode
1	5	1226 Shoe St.	NULL	Bothell	79	98011
2	11	1318 Lasalle Street	NULL	Bothell	79	98011
3	6	1399 Firestone Drive	NULL	Bothell	79	98011
4	18	1873 Lion Circle	NULL	Bothell	79	98011
5	40	1902 Santa Cruz	NULL	Bothell	79	98011
6	1	1970 Napa Ct.	NULL	Bothell	79	98011
7	10	250 Race Court	NULL	Bothell	79	98011
8	868	25111 228th St Sw	NULL	Bothell	79	98011
9	19	3148 Rose Street	NULL	Bothell	79	98011

Hình 8.9: Query with Single Quotes

Các giá trị số không được đưa vào trong dấu ngoặc kép như được trình bày trong Code Snippet 11.

Code Snippet 11:

```
USE AdventureWorks2012
SELECT * FROM HumanResources.Department WHERE DepartmentID < 10
GO
```

Truy vấn trong Code Snippet 11 hiển thị tất cả những bản ghi này trong đó giá trị trong **DepartmentID** ít hơn 10.

Kết quả của truy vấn được trình bày trong hình 8.10.

Results Messages				
	DepartmentID	Name	GroupName	Modified
1	1	Engineering	Research and Development	2002-1
2	2	Tool Design	Research and Development	2002-1
3	3	Sales	Sales and Marketing	2002-1
4	4	Marketing	Sales and Marketing	2002-1
5	5	Purchasing	Inventory Management	2002-1
6	6	Research and Development	Research and Development	2002-1
7	7	Production	Manufacturing	2002-1
8	8	Production Control	Manufacturing	2002-1

Hình 8.10: Output of Where Clause with < Operator

Mệnh đề WHERE cũng có thể được sử dụng với các ký tự đại diện như được trình bày trong bảng 8.2. Tất cả các ký tự đại diện được sử dụng cùng với từ khóa LIKE để thực hiện truy vấn chính xác và cụ thể.

Wildcard	Mô tả	Ví dụ
-	Ký hiệu này sẽ hiển thị một ký tự đơn	SELECT* FROM Person . Contact WHERE Suffix LIKE ' Jr_ '
%	Ký hiệu này sẽ hiển thị một chuỗi với độ dài bất kỳ	SELECT * FROM Person . ContactWHERELastName LIKE ' B% '
[]	Ký tự này sẽ hiển thị một ký tự đơn trong một phạm vi được bao bọc trong dấu ngoặc vuông	SELECT* FROM Sales . CurrencyRate WHERE ToCurrencyCodeLIKE 'C [AN] [DY] '
[^]	Nó sẽ hiển thị một ký tự bất kỳ không nằm trong phạm vi, được đặt trong dấu ngoặc	SELECT* FROM Sales . CurrencyRateWHERE ToCurrencyCodeLIKE ' A[AR] [AS] '

Table 8.2: Wildcard Characters

Mệnh đề WHERE còn sử dụng các toán tử logic như là AND, OR, và NOT. Những toán tử này được sử dụng với các điều kiện tìm kiếm trong mệnh đề WHERE.

Toán tử AND nối hai hoặc nhiều điều kiện và trả lại TRUE chỉ khi cả hai điều kiện là TRUE. Do vậy, nó trả lại tất cả các hàng từ các bảng nơi cả hai điều kiện được liệt kê là true

Code Snippet 12 trình bày toán tử AND.

Code Snippet 12:

```
USE AdventureWorks2012
SELECT * FROM Sales.CustomerAddress WHERE AddressID > 900 AND AddressTypeID = 5
GO
```

Toán tử OR trả lại TRUE và hiển thị tất cả các hàng nếu nó thỏa mãn một trong những điều kiện này. Code Snippet 13 trình bày toán tử OR.

Code Snippet 13:

```
USE AdventureWorks2012
SELECT * FROM Sales.CustomerAddress WHERE AddressID < 900 OR AddressTypeID = 5
GO
```

Truy vấn trong Đoạn mã 13 sẽ hiển thị tất cả các hàng có AddressID ít hơn 900 hoặc có AddressTypeID bằng 5.

Toán tử NOT phủ định điều kiện tìm kiếm. Code Snippet 14 trình bày toán tử NOT.

Code Snippet 14:

```
USE AdventureWorks2012
SELECT * FROM Sales.CustomerAddress WHERE NOT AddressTypeID = 5
GO
```

Code Snippet 14 sẽ hiển thị tất cả các bản ghi với AddressTypeID không bằng 5. Có thể sử dụng nhiều toán tử logic trong câu lệnh SELECT đơn lẻ. Khi có nhiều hơn một toán tử logic được sử dụng, NOT được đánh giá đầu tiên, sau đó AND, và cuối cùng là OR.

8.3.8 Mệnh đề GROUP BY

Mệnh đề GROUP BY phân vùng tập kết quả vào một hoặc nhiều tập hợp con. Một tập hợp con có các giá trị và biểu thức chung. Nếu một hàm tổng hợp được sử dụng trong mệnh đề GROUP BY, tập kết quả tạo ra giá trị duy nhất cho mỗi tổng hợp.

Từ khóa GROUP BY được theo sau bằng một danh sách các cột, được gọi là cột được nhóm. Mỗi cột được nhóm hạn chế số lượng hàng của tập kết quả. Đối với mỗi cột được nhóm, chỉ có một hàng. Mệnh đề GROUP BY có thể có nhiều hơn một cột được nhóm.

Sau đây là cú pháp cho mệnh đề GROUP BY.

Cú pháp:

```
SELECT <column_name1>..<column_nameN> FROM <table_name> GROUP BY <column_name>
```

trong đó,

column_name1: là tên của cột theo đó tập kết quả nên được nhóm lại. Ví dụ, hãy xem xét rằng nếu tổng số giờ tài nguyên đã được tìm thấy cho mỗi lệnh sản xuất, truy vấn trong Code Snippet 15 sẽ lấy ra tập kết quả .

Code Snippet 15:

```
USE AdventureWorks2012
SELECT WorkOrderID, SUM(ActualResourceHrs) FROM Production.WorkOrderRouting
GROUP BY WorkOrderID
GO
```

Kết quả được trình bày trong hình 8.11.

	WorkOrderID	(No column name)
1	13	17.6000
2	14	17.6000
3	15	4.0000
4	16	4.0000
5	17	4.0000
6	18	4.0000
7	19	4.0000
8	20	4.0000
9	21	4.0000
10	22	4.0000

Hình 8.11: GROUP BY Clause

Mệnh đề GROUP BY có thể được sử dụng với các mệnh đề khác nhau.

8.3.9 Các mệnh đề và câu lệnh

Microsoft SQL Server 2012 cung cấp các phần tử cú pháp truy vấn nâng cao để truy cập và xử lý dữ liệu tốt hơn.

- Common Table Expression (CTE) trong câu lệnh **SELECT** và **INSERT**
CTE là một tập kết quả tạm thời có tên dựa trên truy vấn **SELECT** và **INSERT**.

Code Snippet 16 trình bày việc sử dụng CTE trong câu lệnh **INSERT**.

Code Snippet 16:

```
USE CUST_DB

CREATE TABLE NewEmployees (EmployeeID smallint, FirstName char(10), LastName
char(10), Department varchar(50), HiredDate datetime, Salary money);

INSERT INTO NewEmployees
VALUES (11, 'Kevin', 'Blaine', 'Research', '2012-07-31', 54000);

WITH EmployeeTemp (EmployeeID, FirstName, LastName, Department,
HiredDate, Salary)
AS
(
SELECT * FROM NewEmployees
)
SELECT * FROM EmployeeTemp
```

Câu lệnh trong Code Snippet 16 chèn một hàng mới cho bảng **NewEmployees** và chuyển tập kết quả tạm thời thành **EmployeeTemp** như được trình bày trong hình 8.12.

Results		Messages				
	EmployeeID	FirstName	LastName	Department	HiredDate	Salary
1	11	Kevin	Blaine	Research	2012-07-31 00:00:00.000	54000.00

Hình 8.12: Transferring Temporary Result to EmployeeTemp

- Mệnh đề **OUTPUT** trong câu lệnh **INSERT** và **UPDATE**

Mệnh đề **OUTPUT** trả về thông tin về các hàng bị ảnh hưởng bởi câu lệnh **INSERT** và câu lệnh **UPDATE**

Code Snippet 17 trình bày cách sử dụng câu lệnh **UPDATE** với câu lệnh **INSERT**.

Code Snippet 17:

```
USE CUST_DB;
GO
CREATE TABLE dbo.table_3
(
    id INT,
    employee VARCHAR(32)
)
go
INSERT INTO dbo.table_3 VALUES
    (1, 'Matt')
    , (2, 'Joseph')
    , (3, 'Renny')
    , (4, 'Daisy');
GO
DECLARE @updatedTable TABLE
(
    id INT, olddata_employee VARCHAR(32), newdata_employee VARCHAR(32)
);
UPDATE dbo.table_3
    Set employee= UPPER(employee)
OUTPUT
    inserted.id,
    deleted.employee,
    inserted.employee
INTO @updatedTable
SELECT * FROM @updatedTable
```

Sau khi thực thi Code Snippet 17, kết quả nơi các hàng bị ảnh hưởng bởi câu lệnh **INSERT** và câu lệnh **UPDATE** được trình bày trong hình 8.13.

	id	olddata_employee	newdata_employee
1	1	Matt	MATT
2	2	Joseph	JOSEPH
3	3	Renny	RENNY
4	4	Daisy	DAISY

Hình 8.13: Output of UPDATE Statement

➤ Mệnh đề .WRITE

Mệnh đề .WRITE được sử dụng trong câu lệnh UPDATE để thay thế giá trị trong một cột có kiểu dữ liệu có giá trị lớn. Sau đây là cú pháp cho mệnh đề .WRITE.

Cú pháp:

```
.WRITE(expression, @offset, @Length)
```

trong đó,

expression: là chuỗi ký tự sẽ được đặt vào cột loại dữ liệu có giá trị lớn.

@offset: là giá trị bắt đầu (đơn vị) nơi thay thế sẽ được thực hiện.

@Length: là chiều dài của phần trong cột, bắt đầu từ @offset được thay thế bằng biểu thức.

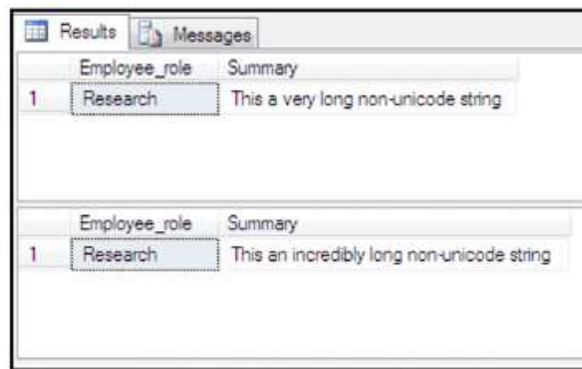
Code Snippet 18 trình bày mệnh đề .WRITE được sử dụng như thế nào trong câu lệnh UPDATE.

Code Snippet 18:

```
USE CUST_DB;
GO
CREATE TABLE dbo.table_5
(
    Employee_role VARCHAR(max),
    Summary VARCHAR(max)
)
```

```
INSERT INTO dbo.table_5(Employee_role, Summary) VALUES ('Research',
'This a very long non-unicode string')
SELECT *FROM dbo.table_5
UPDATE dbo.table_5 SET Summary .WRITE('n incredibly', 6,5)
WHERE Employee_role LIKE 'Research'
SELECT *FROM dbo.table_5
```

Hình 8.14 hiển thị kết quả của truy vấn mệnh đề .WRITE.



	Employee_role	Summary
1	Research	This a very long non-unicode string
2	Research	This an incredibly long non-unicode string

Hình 8.14: Output of .WRITE Clause Query

8.4 Mệnh đề ORDER BY

Nó chỉ ra thứ tự các cột nên được sắp xếp trong một tập kết quả. Nó phân loại các kết quả truy vấn bằng một hoặc nhiều cột. Phân loại có thể là theo thứ tự tăng dần (ASC) hoặc giảm dần (DESC). Theo mặc định, các bản ghi được sắp xếp theo thứ tự ASC. Để chuyển sang chế độ giảm dần, hãy sử dụng từ khóa tùy chọn DESC. Khi nhiều trường được sử dụng, SQL Server xem xét trường tận cùng bên trái làm mức chính của phân loại và những cái khác làm các mức phân loại thấp hơn.

Cú pháp :

```
SELECT <column_name> FROM <table_name> ORDER BY column_name {ASC|DESC}
```

Câu lệnh SELECT trong Code Snippet 19 sắp xếp các kết quả truy vấn trên cột SalesLastYear của bảng Sales.SalesTerritory

Code Snippet 19:

```
USE AdventureWorks2012
SELECT * FROM Sales.SalesTerritory ORDER BY SalesLastYear
GO
```

Kết quả được trình bày trong hình 8.15.

	TerritoryID	Name	Country...	Group	Sales...	SalesLastYear	Cos
1	8	Germany	DE	Europe	3805...	1307949.7917	0.0
2	10	United Kingdom	GB	Europe	5012...	1635823.3967	0.0
3	9	Australia	AU	Pacific	5977...	2278548.9776	0.0
4	7	France	FR	Europe	4772...	2396539.7601	0.0
5	3	Central	US	North America	3072...	3205014.0767	0.0
6	1	Northwest	US	North America	7887...	3298694.4938	0.0
7	2	Northeast	US	North America	2402...	3607148.9371	0.0
8	5	Southeast	US	North America	2538...	3925071.4318	0.0
9	4	Southwest	US	North America	1051...	5366575.7098	0.0

Hình 8.15: ORDER BY Clause

8.5 Làm việc với XML

Extensible Markup Language (XML) cho phép các nhà phát triển xây dựng tập hợp các thẻ riêng của họ và làm cho các chương trình khác có thể hiểu những thẻ này. XML là phương tiện ưa thích dành cho các nhà phát triển để lưu trữ, định dạng và quản lý dữ liệu trên web. Các ứng dụng của ngày nay có sự kết hợp các công nghệ như ASP, công nghệ Microsoft .NET, XML, và SQL Server 2012 làm việc song song. Trong kịch bản như vậy, tốt nhất là lưu trữ dữ liệu XML ở trong SQL Server 2012.

Các cơ sở dữ liệu XML nguyên gốc trong SQL Server 2012 có một số các ưu điểm. Một số trong số này được liệt kê như sau:

- Dễ tìm kiếm và quản lý dữ liệu - Tất cả các dữ liệu XML được lưu trữ cục bộ ở một nơi, do đó làm cho nó dễ tìm kiếm và quản lý hơn.
- Hiệu suất tốt hơn - Truy vấn từ một cơ sở dữ liệu XML được thực hiện tốt sẽ nhanh hơn so với các truy vấn trên các tài liệu được lưu trữ trong một hệ thống tập tin. Ngoài ra, cơ sở dữ liệu về bản chất phân tích từng tài liệu khi lưu trữ chúng.
- Dễ xử lý dữ liệu - Có thể xử lý các tài liệu lớn một cách dễ dàng.

SQL Server 2012 hỗ trợ lưu trữ thuần dữ liệu XML bằng cách sử dụng loại dữ liệu xml. Các phần sau khám phá kiểu dữ liệu xml, làm việc với XML được định kiểu và không định kiểu, lưu trữ chúng trong SQL Server 2012, và sử dụng XQuery để lấy dữ liệu từ các cột của kiểu dữ liệu xml.

8.5.1 Kiểu dữ liệu XML

Ngoài các kiểu dữ liệu thường được sử dụng thường xuyên, SQL Server 2012 cung cấp một kiểu dữ liệu hoàn mới ở dạng kiểu dữ liệu xml.

Kiểu dữ liệu xml được sử dụng để lưu trữ các tài liệu và phân đoạn XML trong một cơ sở dữ liệu SQL Server. Phân đoạn XML là một thể hiện XML với phần tử mức cao nhất từ cấu trúc của nó. Sau đây là cú pháp để tạo ra bảng với các cột thuộc loại xml.

Cú pháp:

```
CREATE TABLE <table_name> ([column_list], <column_name> xml [, column_list])
```

Code Snippet 20 tạo ra một bảng mới tên là **PhoneBilling** với một trong các cột thuộc về kiểu dữ liệu xml.

Code Snippet 20:

```
USE AdventureWorks2012
CREATE TABLE Person.PhoneBilling (Bill_ID int PRIMARY KEY, MobileNumber bigint
UNIQUE, CallDetails xml)
GO
```

Cột thuộc loại xml có thể được thêm vào bảng tại thời điểm tạo ra hoặc sau khi tạo ra nó. Các cột thuộc kiểu dữ liệu xml hỗ trợ các giá trị DEFAULT cũng như là ràng buộc NOT NULL.

Dữ liệu có thể được chèn vào cột xml trong bảng **Person.PhoneBilling** như được trình bày trong Code Snippet 21.

Code Snippet 21:

```
USE AdventureWorks2012
INSERT INTO Person.PhoneBilling VALUES (100, 9833276605,
'<Info><Call>Local</Call><Time>45 minutes</Time><Charges>200</Charges></Info>')
SELECT CallDetails FROM Person.PhoneBilling
GO
```

Kết quả được trình bày trong Hình 8.16.

Results		Messages
CallDetails		
1	<Info><Call>Local</Call><Time>45 minutes</Time><Charges>200</Charges></Info>	

Hình 8.16: XML Data in Columns

Câu lệnh **DECLARE** được sử dụng để tạo ra các biến thuộc loại xml

Code Snippet 22 trình bày cách tạo ra một biến thuộc kiểu xml.

Code Snippet 22:

```
DECLARE @xmlvar xml
SELECT @xmlvar='<Employee name="Joan" />'
```

Các cột thuộc kiểu dữ liệu xml không thể được sử dụng như một khóa chính, khóa ngoại, hoặc là một ràng buộc duy nhất.

8.5.2 XML định kiểu và không định kiểu

Có hai cách lưu trữ các tài liệu XML trong các cột thuộc kiểu dữ liệu xml, cụ thể là XML định kiểu và không định kiểu. Đối tượng định hình XML mà có một lược đồ được liên kết với nó được gọi là đối tượng định hình XML được phân loại. Lược đồ là một đầu đề cho một thể hiện hoặc tài liệu XML. Nó mô tả cấu trúc và giới hạn nội dung của các tài liệu XML bằng cách kết hợp các kiểu dữ liệu xml với các loại phần tử và thuộc tính XML. Nên kết hợp các lược đồ XML với các thể hiện hoặc tài liệu XML bởi vì dữ liệu có thể được xác thực trong khi nó đang được lưu trữ vào cột kiểu dữ liệu xml.

SQL Server không thực hiện bất kỳ xác thực nào đối với dữ liệu được nhập vào cột xml. Tuy nhiên, nó bảo đảm là dữ liệu sẽ được lưu được định hình tốt. Dữ liệu XML không định kiểu có thể được tạo ra và được lưu trữ trong các cột bảng hoặc các biến tùy theo nhu cầu và phạm vi của dữ liệu.

Bước đầu tiên trong việc sử dụng XML được phân loại là đăng ký một lược đồ. Điều này được thực hiện bằng cách sử dụng câu lệnh `CREATE XML SCHEMA COLLECTION` như được trình bày trong Code Snippet 23.

Code Snippet 23:

```
USE SampleDB

CREATE XML SCHEMA COLLECTION CricketSchemaCollection

AS N'<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
<xsd:element name="MatchDetails">
<xsd:complexType>
<xsd:complexContent>
<xsd:restriction base="xsd:anyType">
<xsd:sequence>
<xsd:element name="Team" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:complexContent>
<xsd:restriction base="xsd:anyType">
<xsd:sequence />

```

```
<xsd:attribute name="country" type="xsd:string" />
<xsd:attribute name="score" type="xsd:string" />
</xsd:restriction>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:restriction>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>
</xsd:schema>'
GO
```

Câu lệnh CREATE XML SCHEMA COLLECTION tạo ra một bộ sưu tập các lược đồ, bất kỳ cái nào trong đó có thể được sử dụng để xác thực dữ liệu XML định kiểu với tên của bộ sưu tập. Ví dụ này trình bày một lược đồ mới được gọi là **CricketSchemaCollection** đang được thêm vào cơ sở dữ liệu **SampleDB**. Một khi lược đồ được đăng ký, lược đồ đó có thể được sử dụng trong các thể hiện mới của kiểu dữ liệu xml.

Code Snippet 24 tạo ra một bảng với cột loại xml và chỉ ra một lược đồ cho cột đó.

Code Snippet 24:

```
USE SampleDB

CREATE TABLE CricketTeam ( TeamID int identity not null, TeamInfo xml (CricketSchemaCollection) )

GO
```

Để tạo ra các hàng mới với dữ liệu XML định kiểu, câu lệnh **INSERT** có thể được sử dụng như được trình bày trong **Code Snippet 25**.

Code Snippet 25:

```
USE SampleDB

INSERT INTO CricketTeam (TeamInfo) VALUES ( '<MatchDetails><Team
country="Australia" score="355"></Team><Team country="Zimbabwe"
score="200"></Team><Team country="England" score="475"></Team></
MatchDetails>' )

GO
```

Một biến XML định kiểu cũng có thể được tạo ra bằng cách chỉ ra tên bộ sưu tập lược đồ. Ví dụ, trong Code Snippet 26, một nhóm biến được khai báo là biến XML định kiểu với tên lược đồ là **CricketSchemaCollection**. Câu lệnh SET được sử dụng để gán biến làm một phân đoạn XML.

Code Snippet 26:

```
USE SampleDB
DECLARE @teamxml (CricketSchemaCollection)
SET @team = '<MatchDetails><Team country="Australia"></Team></MatchDetails>'
SELECT @team
GO
```

8.5.3 XQuery

Sau khi dữ liệu XML đã được lưu trữ bằng cách sử dụng kiểu dữ liệu xml, nó có thể được truy vấn và lấy ra sử dụng một ngôn ngữ có tên là XQuery. XML Query hay XQuery là một ngôn ngữ truy vấn mới, ngôn ngữ này kết hợp cú pháp quen thuộc với những người phát triển làm việc với cơ sở dữ liệu quan hệ, và ngôn ngữ XPath được dùng để chọn các phần riêng lẻ hoặc tập hợp các thành tố từ một tài liệu XML. XQuery có thể là truy vấn dữ liệu XML có cấu trúc hoặc bán cấu trúc. Để truy vấn một thể hiện XML được lưu trữ trong một biến hoặc cột thuộc loại xml, các phương thức kiểu dữ liệu xml được sử dụng. Ví dụ, biến thuộc loại xml được khai báo và truy vấn bằng cách sử dụng các phương thức của kiểu dữ liệu xml. Các nhà phát triển cần phải truy vấn các tài liệu XML, và điều này liên quan đến việc biến đổi các tài liệu XML ở định dạng yêu cầu. XQuery làm cho nó có thể thực hiện các truy vấn phức tạp đối với một nguồn dữ liệu XML trên web.

Một số phương thức kiểu dữ liệu xml được sử dụng với XQuery được mô tả như sau:

➤ exist()

Phương thức này được sử dụng để xác định xem một hay nhiều nút quy định có mặt trong tài liệu XML hay không. Nó trả về 1 nếu biểu thức XQuery đã trả về ít nhất một nút, 0 nếu biểu thức XQuery đã đánh giá thành kết quả rỗng, và NULL nếu thể hiện kiểu dữ liệu xml dựa vào đó truy vấn đã được thực hiện là NULL.

Code Snippet 27 trình bày việc sử dụng phương thức exist(). Giả sử là nhiều bản ghi đã được chèn vào bảng.

Code Snippet 27:

```
USE SampleDB
SELECT TeamID FROM CricketTeam WHERE TeamInfo.exist('(/MatchDetails/Team)') = 1
GO
```

Điều này sẽ chỉ trả lại những giá trị **TeamID** nơi phần tử **Team** đã được chỉ ra trong **TeamInfo**. Kết quả được trình bày trong hình 8.17.



TeamID
1

Hình 8.17: exist() Method

➤ query()

Có thể sử dụng phương thức query() để lấy toàn bộ nội dung của một tài liệu XML hoặc một phần đã chọn của tài liệu XML. Đoạn mã 28 trình bày việc sử dụng phương thức query().

Code Snippet 28:

```
USE SampleDB
SELECT TeamInfo.query('/MatchDetails/Team') AS Info FROM CricketTeam
GO
```

Kết quả được trình bày trong hình 8.18.



Info
<Team country="Australia" score="355" />

Hình 8.18: query() Method

➤ value()

Có thể sử dụng phương thức value() để trích xuất các giá trị vô hướng từ một kiểu dữ liệu xml. Code Snippet 29 trình bày phương thức này.

Code Snippet 29:

```
USE SampleDB
SELECT TeamInfo.value('/MatchDetails/Team/@score' [1], 'varchar(20)')
AS Score FROM CricketTeam where TeamID=1
GO
```

Kết quả của lệnh này được trình bày trong Hình 8.19.



	Score
1	355

Hình 8.19: value() Method

8.6 Kiểm tra tiến độ của bạn

1. Cái nào sau đây cho phép các nhà phát triển xây dựng tập hợp các thẻ riêng của họ và làm cho các chương trình khác có thể hiểu những thẻ này?

(A)	Xquery	(C)	DHTML
(B)	HTML	(D)	XML

2. Câu lệnh _____ lấy các hàng và cột từ một hoặc nhiều bảng.

(A)	SELECT	(C)	INSERT
(B)	DISPLAY	(D)	SHOW

3. Điều nào sau đây là định dạng chung của truy vấn mệnh đề .WRITE?

(A)	<pre>ADD INTO dbo.table_5 (Employee_role, Summary) VALUES ('Research', 'This a very long non-unicode string') SELECT * FROM dbo.table_5 UPDATE dbo.table_5 SET Summary .WRITE ('n incredibly')</pre>	(C)	<pre>INSERT INTO dbo.table_5 (Employee_role, Summary) VALUES ('Research', 'This a very long non-unicode string') SELECT * FROM dbo.table_5 UPDATE dbo.table_5 SET Summary .WRITE ('n incredibly', 6, 5) WHERE Employee_role LIKE 'Research'</pre>
(B)	<pre>INSERT INTO dbo.table_5 (Employee_role, Summary) VALUES ('Research', 'This a very long non-unicode string') SELECT * FROM dbo.table_5 UPDATE dbo.table_5 SET Summary .WRITE ('n incredibly', 6, 5) WHERE Employee_role LIKE 'Research'</pre>	(D)	<pre>INSERT INTO dbo.table_5 (Employee_role, Summary) VALUES ('Research', 'This a very long non-unicode string') SELECT * FROM dbo.table_5 dbo.table_5 SET Summary ('n incredibly', 6, 5) WHERE Employee_role LIKE 'Research'</pre>

4. Mệnh đề nào sau đây với câu lệnh SELECT được sử dụng để chỉ ra các bảng hoặc lấy ra các bản ghi?

(A)	WHERE	(C)	.VALUE
(B)	FROM	(D)	.WRITE

5. _____ được sử dụng để nâng cao hiệu quả của các truy vấn trên các tài liệu XML được lưu trữ trong một cột XML.

(A)	XML indexing	(C)	XML querying
(B)	XMLimport	(D)	XML export

8.6.1 Đáp án

1.	D
2.	A
3.	B
4.	B
5.	A



- Câu lệnh SELECT lấy các hàng và cột từ các bảng.
- Câu lệnh SELECT cho phép người dùng chỉ ra các biểu thức khác nhau để xem tập kết quả theo một cách có trật tự.
- Câu lệnh SELECT có thể chứa các biểu thức toán học bằng cách áp dụng các toán tử cho một hoặc nhiều cột.
- Từ khóa DISTINCT ngăn việc gọi ra các bản ghi trùng lặp.
- XML cho phép các nhà phát triển xây dựng tập hợp các thẻ riêng của họ và làm cho nó có thể cho các chương trình khác hiểu những thẻ này .
- Thẻ hiển XML định kiểu là một thẻ hiển XML trong đó có một sơ đồ liên kết với nó.
- Dữ liệu XML có thể được truy vấn và lấy ra sử dụng ngôn ngữ XQuery.



1. Transcorp United Inc là một công ty xuất nhập khẩu tại Mỹ. Cơ sở dữ liệu của công ty được tạo ra trong SQL Server 2012. Transcorp có khoảng 3.000 nhân viên trên toàn thế giới. Các chi tiết của nhân viên như là Mã nhân viên, Tên nhân viên, Phòng ban của nhân viên, Ngày gia nhập, và...vv được lưu trữ trong bảng **EMP _ Details**.

Là quản trị viên cơ sở dữ liệu của Transcorp, bạn phải thực hiện các nhiệm vụ sau :

- Lấy dữ liệu của nhân viên đã gia nhập công ty trước năm 2012 và sau năm 2010.
- Chỉnh sửa tên của nữ nhân viên Julia Drek thành Julia Dean bằng cách sử dụng thuộc tính .WRITE .
- Lấy dữ liệu của tất cả các nhân viên đến từ Houston.