



Quản Trị Dữ Liệu Với Microsoft SQL Server

Chương: 8

Truy xuất dữ liệu

- Mô tả câu lệnh SELECT, cú pháp và cách sử dụng.
- Giải thích các mệnh đề khác nhau được sử dụng với câu lệnh SELECT.
- Cách sử dụng mệnh đề ORDER BY.
- Mô tả làm việc với XML có định kiểu(typed) và không định kiểu(untyped).
- Giải thích các bước(procedure) tạo, sử dụng, và xem(view) lược đồ XML.
- Giải thích sử dụng Xquery để truy xuất dữ liệu XML.



Giới thiệu

- SELECT là câu lệnh cốt lõi được sử dụng để truy xuất dữ liệu trong SQL Server 2012.
- XML cho phép các nhà phát triển tự định nghĩa các thẻ(tag) của riêng mình và làm cho các chương trình khác có thể hiểu về các thẻ này.
- XML là biện pháp được ưa thích bởi các nhà phát triển để lưu trữ, định dạng, quản lý dữ liệu cho Web.

Câu lệnh SELECT 1-2

Có thể xem(view) dữ liệu của một bảng bằng câu lệnh SELECT.

Câu lệnh SELECT cho phép lấy các dòng và các cột từ một hoặc nhiều bảng.

Kết quả trả về của lệnh SELECT là một bảng hay còn được gọi là tập kết quả(resultset).

Câu lệnh SELECT cũng cho phép ghép(join) hai bảng hoặc lấy một tập nhỏ các cột từ một hoặc trong các bảng được ghép.

Câu lệnh SELECT xác định các cột được sử dụng cho một truy vấn.

Câu lệnh SELECT 2-2

Cú pháp của lệnh SELECT gồm có một dãy các biểu thức, được phân cách nhau bởi các dấu phẩy (commas).

Mỗi biểu thức trong câu lệnh sẽ tạo ra một cột trong tập kết quả(resultset).

Thứ tự các cột xuất hiện theo đúng thứ tự của biểu thức được chỉ ra trong câu lệnh SELECT.

➤ Cú pháp của câu lệnh SELECT như sau:

Cú pháp

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

Trong đó,

table_name: là tên bảng có chứa dữ liệu sẽ được lấy để hiển thị.

<column_name1>...<column_nameN>: là danh sách các cột được hiển thị.

SELECT không có FROM

Từ phiên bản SQL Server 2005 đến SQL Server 2012, có thể sử dụng câu lệnh SELECT mà không cần có phải mệnh đề FROM.

Đoạn code sau minh họa sử dụng câu lệnh SELECT không có mệnh đề FROM.

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

- Đoạn code chỉ lấy và hiển thị 5 kí tự đầu tiên của từ 'International'.
- Kết quả được thể hiện trong hình sau:

Results		Messages	
	(No column name)		
1	Inter		

Hiển thị tất cả các cột 1-2

Dấu hoa thị (*) thường được sử dụng trong câu lệnh SELECT khi muốn lấy tất cả các cột có trong bảng.

Nó được dùng như là một sự viết tắt (shorthand) để thay cho việc phải liệt kê ra toàn bộ tên các cột có trong bảng được chỉ ra ở mệnh đề FROM.

➤ Cú pháp để chọn tất cả các cột như sau:

Cú pháp

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

Trong đó,

*: Chỉ ra rằng sẽ lấy toàn bộ các cột của các bảng được chỉ ra ở mệnh đề FROM.
<table_name>: là tên của bảng mà thông tin được lấy từ đó. Có thể có bao nhiêu bảng cũng được. Khi hai hay nhiều bảng được sử dụng, dòng của bảng này được ánh xạ với dòng của các bảng khác. Hoạt động này mất nhiều thời gian nếu dữ liệu trong các bảng rất lớn(huge). Do vậy, khuyến khích nên sử dụng cú pháp cùng với điều kiện.

Hiển thị tất cả các cột 2-2

- Đoạn code sau minh họa việc sử dụng toán tử '*' trong câu lệnh SELECT :

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

- Hình sau cho thấy một phần kết quả với toàn bộ các cột của bảng HumanResources.Employee:

	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	On
1	1	295847284	adventure-works\ken0	0x	0
2	2	245797967	adventure-works\temi0	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

Hiển thị các cột được chọn 1-2

Câu lệnh `SELECT` hiển thị hoặc trả về các cột do người dùng chọn hoặc chỉ ra trong câu lệnh.

Để hiển thị các cột cụ thể, cần phải biết tên các cột có trong bảng.

➤ Dưới đây là cú pháp để chọn ra các cột cụ thể:

Cú pháp:

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

Trong đó,

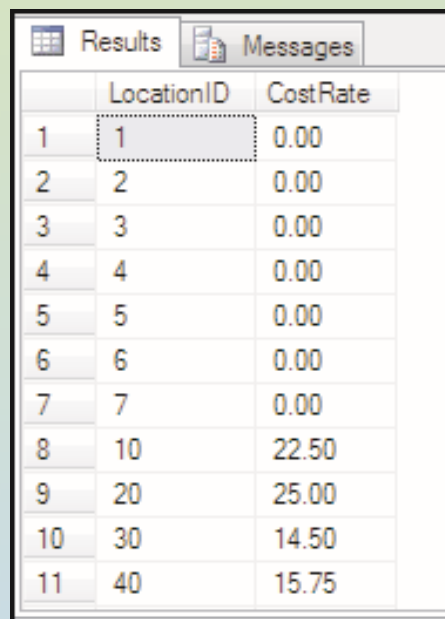
`<column_name1>..<column_nameN>`: là các cột được chọn để hiển thị

Hiển thị các cột được chọn 2-2

- Ví dụ, để hiển thị mức chi phí(cost rate) của nhiều nơi khác nhau trong bảng `Production.Location` của csdl `AdventureWorks2012`, câu lệnh `SELECT` được viết như trong đoạn code dưới đây:

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

- Hình dưới đây cho thấy hai cột `LocationID` và `CostRate` được lấy từ bảng `Production.Location` của csdl `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

Sử dụng các hằng trong tập kết quả 1-2

Các hằng chuỗi kí tự được sử dụng khi các cột kí tự được ghép (joined).

Chúng giúp định dạng đúng cách và dễ đọc.

Các hằng này không được chỉ ra như một cột riêng biệt trong tập kết quả.

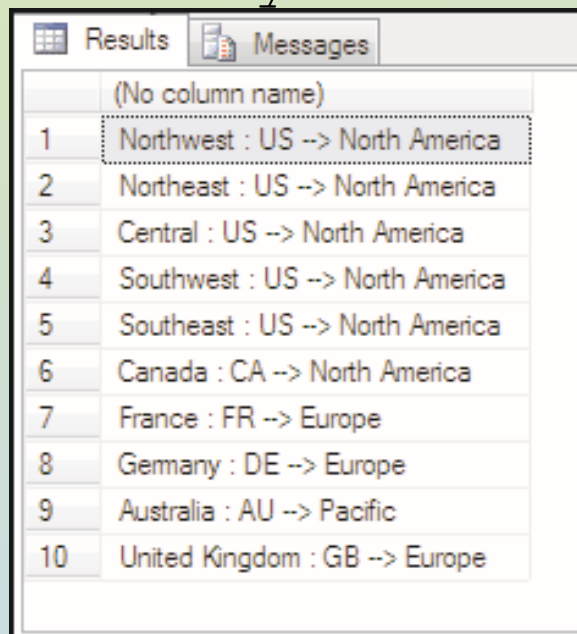
Thường sẽ hiệu quả hơn cho một ứng dụng để xây dựng các giá trị thành kết quả khi chúng được hiển thị, chứ không phải là cách sử dụng các máy chủ để kết hợp các giá trị không đổi.

Sử dụng các hằng trong tập kết quả 2-2

- Ví dụ để ghép thêm kí tự ':' 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 viết như đoạn code dưới đây:

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

- Hình sau đây hiển thị tên quốc gia, mã vùng quốc gia, và nhóm tương ứng từ bảng Sales.SalesTerritory của csdl 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

Đổi tên các cột trong tập kết quả 1-2

Khi các cột được hiển thị trong tập kết quả, chúng đi kèm với các tiêu đề tương ứng đã được xác định trong bảng.

Các tiêu đề này có thể được thay đổi, đổi tên, đặt lại tên, hoặc có thể gắn một tên mới bằng mệnh đề AS.

Do vậy, bằng cách tùy biến các tiêu đề, sẽ làm chúng trở nên dễ hiểu hơn và có ý nghĩa.

Đổi tên các cột trong tập kết quả 2-2

- Đoạn code dưới đây minh họa hiển thị tiêu đề '**ChangedDate**' cho cột **ModifiedDate** trong bảng **dbo.Individual**, câu lệnh SELECT:

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

- Kết quả hiển thị tiêu đề '**ChangedDate**' thay cho tiêu đề cột **ModifiedDate** trong bảng **dbo.Individual**.
- Hình dưới đây cho thấy tiêu đề ban đầu và tiêu đề sau khi được thay đổi:

Results	Messages												
<table><tr><th></th><th>ModifiedDate</th></tr><tr><td>1</td><td>1981-02-02</td></tr></table>		ModifiedDate	1	1981-02-02	<table><tr><th>Results</th><th>Messages</th></tr><tr><td><table><tr><th></th><th>ChangedDate</th></tr><tr><td>1</td><td>1981-02-02</td></tr></table></td><td></td></tr></table>	Results	Messages	<table><tr><th></th><th>ChangedDate</th></tr><tr><td>1</td><td>1981-02-02</td></tr></table>		ChangedDate	1	1981-02-02	
	ModifiedDate												
1	1981-02-02												
Results	Messages												
<table><tr><th></th><th>ChangedDate</th></tr><tr><td>1</td><td>1981-02-02</td></tr></table>		ChangedDate	1	1981-02-02									
	ChangedDate												
1	1981-02-02												

Giá trị tính toán trong tập kết quả 1-2

Một câu lệnh `SELECT` có thể chứa các biểu thức toán học bằng việc áp dụng các toán tử trên một hoặc nhiều cột.

Nó cho phép tập kết quả có thêm những giá trị không hề tồn tại trong bảng cơ sở (base table), mà là do được tính toán từ các giá trị có sẵn trong bảng cơ sở (base table).

Ví dụ xem xét bảng `Production.ProductCostHistory` từ csdl `AdventureWorks2012`.

Hãy xem xét ví dụ, trong đó người sản xuất quyết định giảm giá 15% trên trị giá chuẩn (standard costs) của tất cả sản phẩm.

Giá trị tính toán trong tập kết quả 2-2

- Hiện đang không có số tiền giảm giá, nhưng có thể tính được nó bằng cách thực hiện câu lệnh SELECT như trong đoạn code sau:

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

- Hình dưới đây cho thấy kết quả của số tiền giảm (Discount) được tính:

	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

Sử dụng từ khóa DISTINCT

Từ khóa DISTINCT dùng để ngăn việc lấy ra các dòng dữ liệu trùng nhau

Nó giúp loại bỏ các dòng trùng lặp trong tập kết quả của câu lệnh SELECT.

Ví dụ, nếu chọn cột StandardCost mà không có từ khóa DISTINCT, nó sẽ hiển thị toàn bộ các trị giá chuẩn(standard costs) hiện có trong bảng.

Khi sử dụng từ khóa DISTINCT trong truy vấn, SQL Server chỉ hiển thị một lần trong nhóm các bản ghi giống nhau của StandardCost như trong đoạn mã sau đây:

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

Sử dụng từ khóa TOP và PERCENT

Từ khóa TOP sẽ chỉ lấy ra một vài dòng đầu tiên trong tập kết quả.

Có thể giới hạn số dòng cần lấy từ tập kết quả bằng một số hoặc tỷ lệ phần trăm.

Từ khóa TOP cũng có thể sử dụng với những câu lệnh khác như INSERT, UPDATE, và DELETE.

➤ Cú pháp sử dụng từ khóa TOP:

Cú pháp:

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

Trong đó,

expression: là một số hoặc tỷ lệ phần trăm số dòng cần lấy từ tập kết quả.

PERCENT: chỉ ra số dòng cần lấy theo tỷ lệ phần trăm.

WITH TIES: lấy thêm thêm một số dòng cần hiển thị.

SELECT với INTO 1-3

Mệnh đề INTO được sử dụng cùng với SELECT để tạo một bảng mới.

Cấu trúc cột của bảng mới là các cột được liệt kê trong câu lệnh SELECT, và toàn bộ các dòng lấy được bởi SELECT được chèn(insert) vào bảng mới.

Để thực thi mệnh đề này với câu lệnh SELECT, người thực hiện phải có quyền sử dụng lệnh CREATE TABLE trên cơ sở dữ liệu đích.

➤ Cú pháp câu lệnh SELECT với mệnh đề INTO như sau:

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

Trong đó,

new_table: là tên của bảng mới được tạo.

SELECT với INTO 2-3

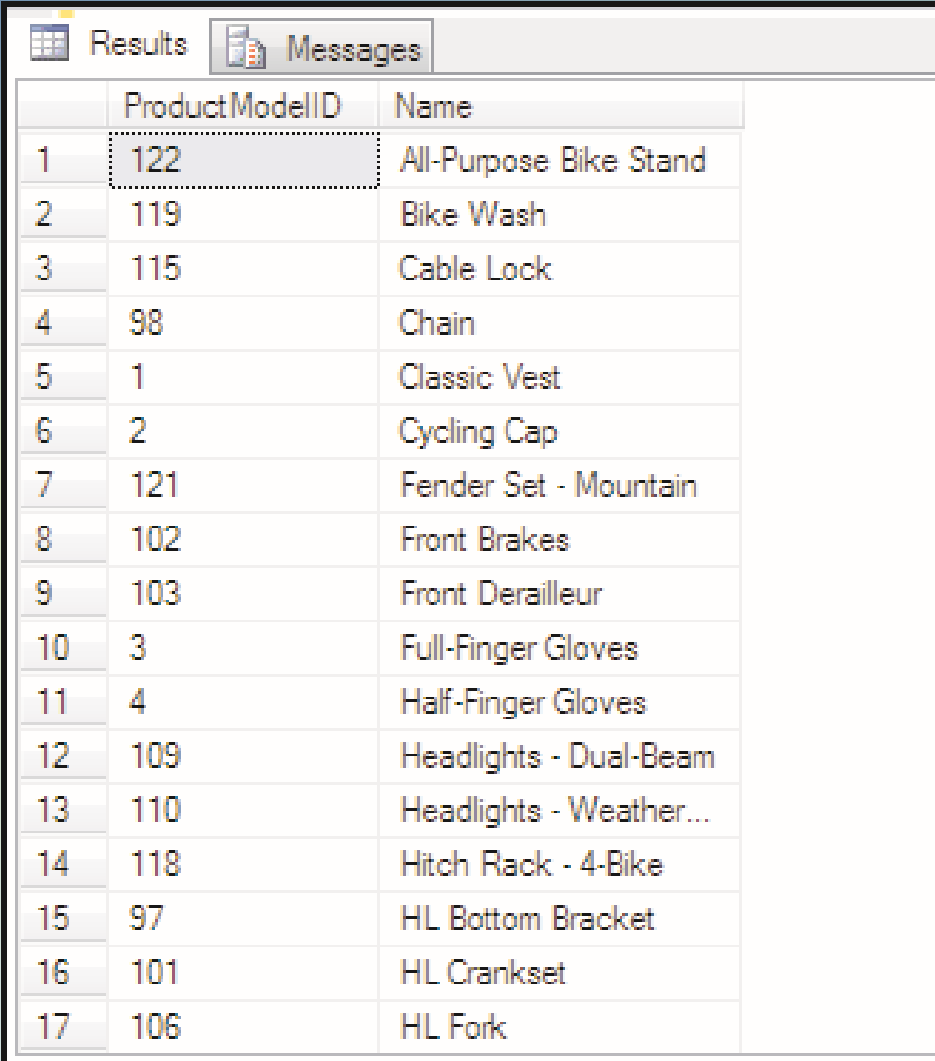
- Đoạn code dưới đây sử dụng mệnh đề INTO để tạo bảng mới có tên Production.ProductName với các chi tiết như ID của sản phẩm và tên của nó từ bảng Production.ProductModel:

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

- Sau khi thực thi đoạn code một thông báo được hiển thị: '(128 row(s) affected)' [(128 dòng được tác động)].

SELECT với INTO 3-3

- Nếu một truy vấn được viết để hiển thị các dòng của bảng mới vừa tạo, kết quả sẽ được hiển thị như trong hình dưới đây:



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 17 rows of data. The table has two columns: 'ProductModelID' and 'Name'. The first row is highlighted with a dotted border, indicating it is the current row. The data is as follows:

	ProductModelID	Name
1	122	All-Purpose Bike Stand
2	119	Bike Wash
3	115	Cable Lock
4	98	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

SELECT với WHERE 1-8

Mệnh đề WHERE được sử dụng với câu lệnh SELECT để chọn có điều kiện hoặc để giới hạn các bản ghi được lấy về bởi truy vấn.

Mệnh đề WHERE chỉ ra một biểu thức Boolean để kiểm tra(test) các dòng được trả về bởi truy vấn.

Dòng được trả về nếu biểu thức là true và bị loại bỏ nếu là false.

➤ Cú pháp câu lệnh SELECT với mệnh đề WHERE như sau:

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

Trong đó,

search_condition: là điều kiện được so khớp bởi các dòng.

SELECT với WHERE 2-8

- Bảng dưới đây liệt kê các toán tử khác nhau có thể được sử dụng trong mệnh đề WHERE:

Toán tử	Giải thích
=	Bằng
<>	Khác
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
!	Không (phủ định)
BETWEEN	Giữa một khoảng
LIKE	Tìm với các ký tự đại diện
IN	Bên trong một khoảng

SELECT với WHERE 3-8

Đoạn code sau đây minh họa sử dụng toán tử = (equal) trong mệnh đề WHERE để hiển thị dữ liệu có EndDate là 6/30/2007 12:00:00 AM:

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

➤ Kết quả sử dụng SELECT với mệnh đề WHERE như hình dưới:

	ProductID	StartDate	EndDate	StandardCost	ModifiedDate
1	707	2006-07-01 00:00:0...	2007-06-30 00:...	13.8782	2007-06-30 00:00:00
2	708	2006-07-01 00:00:0...	2007-06-30 00:...	13.8782	2007-06-30 00:00:00
3	711	2006-07-01 00:00:0...	2007-06-30 00:...	13.8782	2007-06-30 00:00:00
4	712	2006-07-01 00:00:0...	2007-06-30 00:...	5.2297	2007-06-30 00:00:00
5	713	2006-07-01 00:00:0...	2007-06-30 00:...	29.0807	2007-06-30 00:00:00
6	714	2006-07-01 00:00:0...	2007-06-30 00:...	29.0807	2007-06-30 00:00:00
7	715	2006-07-01 00:00:0...	2007-06-30 00:...	29.0807	2007-06-30 00:00:00
8	716	2006-07-01 00:00:0...	2007-06-30 00:...	29.0807	2007-06-30 00:00:00
9	717	2006-07-01 00:00:0...	2007-06-30 00:...	722.2568	2007-06-30 00:00:00

SELECT với WHERE 4-8

- Tất cả các truy vấn trong SQL đều dùng các nháy đơn (single quotes) để bao các giá trị văn bản (text values).
- Ví dụ, hãy xem truy vấn dưới đây lấy về tất cả các bản ghi từ bảng `Person.Address` có thành phố là `Bothell`.
- Đoạn code sau đây minh họa sử dụng toán tử `=` trong mệnh đề `WHERE` để hiển thị dữ liệu có thành phố(City) là `Bothell`.

```
USE AdventureWorks2012
SELECT * FROM Person.Address WHERE Person.City = 'Bothell'
GO
```

- Hình sau cho thấy kết quả của truy vấn:

	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

SELECT với WHERE 5-8

- Các giá trị số không bao trong trong bất kỳ dấu nháy nào, xem đoạn code minh họa dưới đây :

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

- Truy vấn hiển thị tất cả các bản ghi có giá trị của cột DepartmentID nhỏ hơn 10.
- Kết quả của đoạn code trên được thể hiện trong hình sau:

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

SELECT với WHERE 6-8

- Cũng có thể sử dụng mệnh đề `WHERE` với các kí tự đại diện (wildcard) cho trong bảng dưới đây:

Wildcard	Description	Example
<code>_</code>	It will display a single character	<code>SELECT * FROM Person. Contact WHERE Suffix LIKE 'Jr_'</code>
<code>%</code>	It will display a string of any length	<code>SELECT * FROM Person. Contact WHERE LastName LIKE 'B%'</code>
<code>[]</code>	It will display a single character within the range enclosed in the brackets	<code>SELECT * FROM Sales. CurrencyRate WHERE ToCurrencyCode LIKE 'C[AN][DY]'</code>
<code>[^]</code>	It will display any single character not within the range enclosed in the brackets	<code>SELECT * FROM Sales. CurrencyRate WHERE ToCurrencyCode LIKE 'A[^R][^S]'</code>

- Tất cả các kí tự đại diện(wildcard) được sử dụng cùng với từ khóa `LIKE` để làm cho truy vấn chính xác và cụ thể.

SELECT với WHERE 7-8

Mệnh đề WHERE cũng được sử dụng với các toán tử logic như AND, OR, và NOT. Những toán tử này được dùng ghép nhiều điều kiện trong mệnh đề WHERE.

Toán tử AND ghép hai hay nhiều điều kiện và trả về TRUE nếu tất cả các điều kiện cùng là TRUE.

Vậy nó sẽ trả về tất cả các dòng từ bảng nếu tất cả các điều kiện được liệt kê là đúng. Đoạn code sau đây minh họa sử dụng toán tử AND:

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

SELECT với WHERE 8-8

- Toán tử OR trả về TRUE nếu một thỏa mãn một trong các điều kiện, và tất cả các dòng sẽ được hiển thị. Đoạn code sau đây minh họa về sử dụng toán tử OR:

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

- Truy vấn hiển thị tất cả các dòng có AddressID < 900 hoặc AddressTypeID = 5.
- Toán tử phủ định điều kiện NOT.
- Đoạn code dưới đây minh họa sử dụng toán tử NOT:

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

- Đoạn code sẽ hiển thị tất cả các bản ghi có AddressTypeID của nó không bằng 5.
- Có thể sử dụng nhiều toán tử luận lý (logical) trong một câu lệnh SELECT.
- Khi sử dụng nhiều toán tử luận lý (logical) thì thứ tự ưu tiên thực hiện: đầu tiên là toán tử NOT, sau đó đến toán tử AND, và cuối cùng là toán tử OR.

Mệnh đề GROUP BY 1-2

Mệnh đề GROUP chia(partition) tập kết quả thành một hoặc nhiều nhóm(subset) con. Mỗi nhóm đều có các giá trị và các biểu thức chung.

Nếu sử dụng hàm thống kê(Sum, Count,...) trong mệnh đề GROUP BY, tập kết quả sẽ sinh ra giá trị tổng hợp duy nhất cho mỗi nhóm tập hợp (aggregate).

Mỗi cột được nhóm sẽ làm giới hạn số dòng của tập kết quả. Mỗi cột được nhóm sẽ chỉ có một dòng cho mỗi nhóm dòng.

Mệnh đề GROUP BY có thể có nhiều cột được nhóm. Cú pháp của mệnh đề GROUP BY:

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

Trong đó,

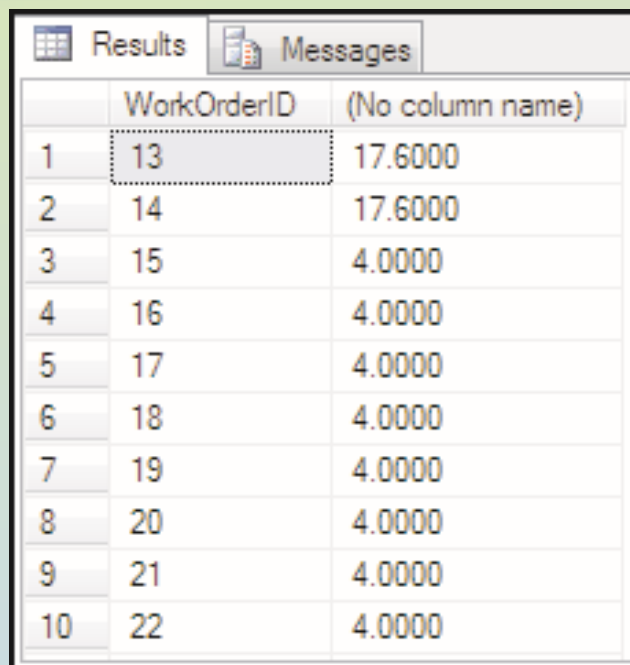
column_name1, ...: là tên các cột mà theo đó tập kết quả được nhóm lại.

Mệnh đề GROUP BY 2-2

- Ví dụ, nếu phải tính tổng số giờ tài nguyên cho mỗi đơn hàng làm việc, câu truy vấn trong đoạn mã sau đây sẽ trả về tập kết quả:

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

- Hình sau cho thấy kết quả của đoạn mã trên:



	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

Các mệnh đề và các câu lệnh 1-7

- Microsoft SQL Server 2012 cung cấp các thành phần cú pháp truy vấn nâng cao để việc truy xuất và xử lý dữ liệu mạnh mẽ hơn.

Common Table Expression (CTE) trong câu lệnh SELECT và INSERT

- Một CTE là một tập kết quả tạm thời dựa trên truy vấn SELECT và INSERT thông thường.
- Đoạn mã dưới đây minh họa sử dụng CTE trong câu lệnh INSERT:

```
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
```


Các mệnh đề và các câu lệnh 2-7

- Truy vấn trong đoạn mã chèn thêm dòng mới cho bảng NewEmployees và chuyển toàn bộ kết quả tạm thời tới bảng EmployeeTemp như hình dưới đây:

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

Mệnh đề OUTPUT trong các câu lệnh INSERT và UPDATE

- Mệnh đề OUTPUT trả về các dòng vừa được thêm mới bởi lệnh INSERT, hoặc bị xóa bởi lệnh DELETE, hoặc để xem các giá trị trong các dòng trước và sau khi thực hiện lệnh UPDATE .
- Đoạn code sau đây minh họa cách sử dụng câu lệnh UPDATE với câu lệnh INSERT:

Các mệnh đề và các câu lệnh 3-7

- Đoạn code sau đây minh họa cách sử dụng câu lệnh UPDATE với câu lệnh INSERT:

```
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
```

Các mệnh đề và các câu lệnh 4-7

- Kết quả trong hình dưới đây cho thấy các dòng bị tác động(affected) bởi câu lệnh INSERT và UPDATE:

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

Các mệnh đề và các câu lệnh 5-7

Mệnh đề .WRITE

- Mệnh đề .WRITE được dùng trong câu lệnh UPDATE khi muốn thực hiện thay thế giá trị trong cột có kiểu dữ liệu kích thước lớn (`varchar(max)`)
- Cú pháp mệnh đề .WRITE:

Cú pháp:

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

Trong đó,

`expression`: là chuỗi kí tự được dùng để thay thế trong cột chứa giá trị có kiểu dữ liệu lớn.

`@offset`: là giá trị vị trí bắt đầu mà việc thay thế sẽ thực hiện từ đó.

`@Length`: là độ dài của phần sẽ bị thay thế trong cột, bắt đầu tính từ vị trí `@offset`.

Các mệnh đề và các câu lệnh 6-7

- Đoạn code dưới đây minh họa cách sử dụng mệnh đề **.WRITE** trong câu lệnh UPDATE:

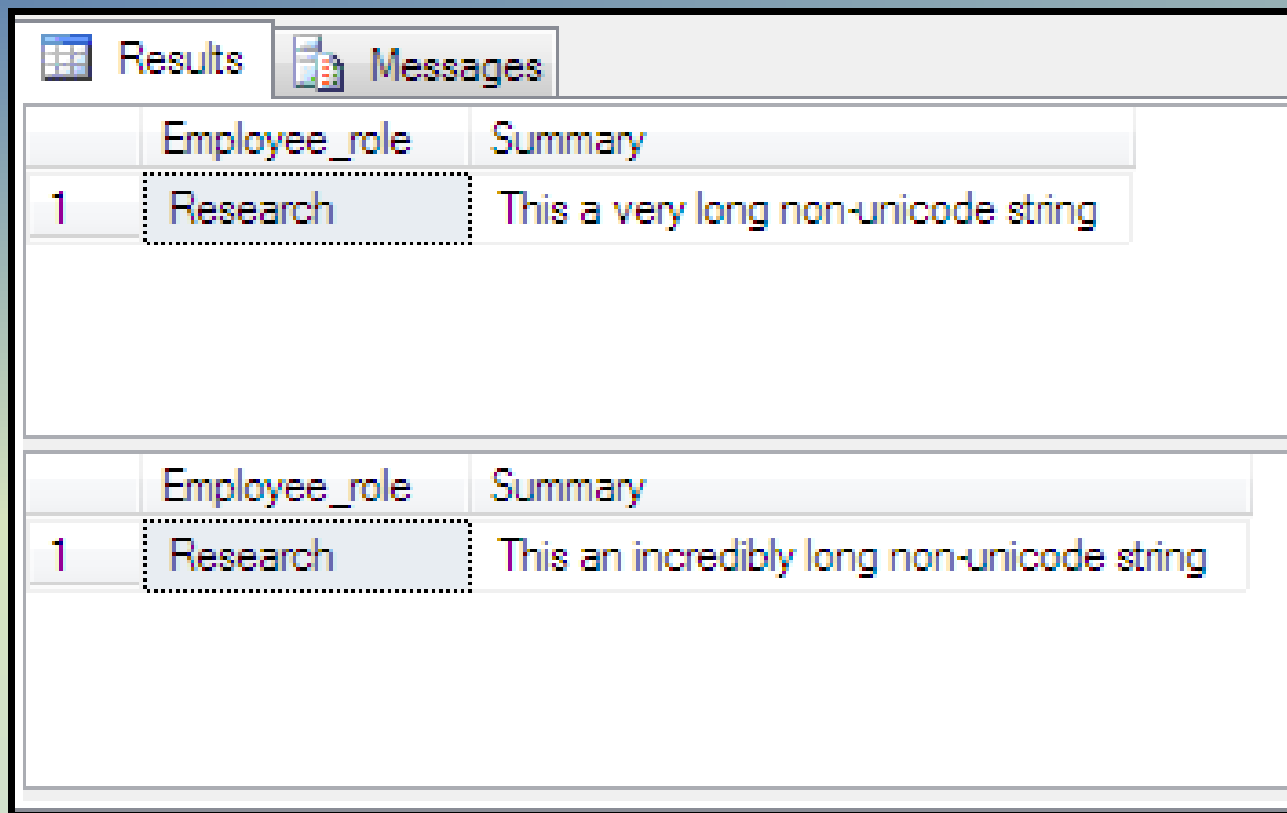
```
USE CUST_DB;
GO
CREATE TABLE dbo.table_5
(
    Employee_role VARCHAR(max),
    Summary VARCHAR(max)
)
INSERT INTO dbo.table_5(Employee_role, Summary)
VALUES ('Research', 'This a very long non-unicode string')
SELECT *FROM dbo.table_5

UPDATE dbo.table_5 SET Summary .WRITE('n incredibly', 6,5)
WHERE Employee_role LIKE 'Research'

SELECT *FROM dbo.table_5
```

Các mệnh đề và các câu lệnh 7-7

- Hình dưới đây hiển thị kết quả của mệnh đề .WRITE:



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

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

Mệnh đề ORDER BY 1-2

Dùng cho việc sắp xếp thứ tự các dòng trong tập kết quả theo giá trị của cột được chỉ ra.

Nó sắp xếp kết quả truy vấn theo một hoặc nhiều cột. Sắp xếp có thể theo thứ tự tăng [ascending (ASC)] hoặc giảm [descending (DESC)].

Mặc định, các bản ghi được sắp xếp theo chiều tăng dần ASC. Để sắp xếp theo chiều giảm sử dụng từ khóa tùy chọn DESC.

Khi có nhiều cột được chỉ ra, SQL Server sẽ ưu tiên thực hiện sắp xếp dữ liệu trong cột từ phía bên trái trước, sau đó lần lượt đến cột kế tiếp.

Cú pháp

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

Mệnh đề ORDER BY 2-2

- Câu lệnh SELECT trong đoạn code dưới đây sắp xếp kết quả của truy vấn theo cột SalesLastYear của bảng Sales.SalesTerritory:

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

- Hình sau cho thấy kết quả của truy vấn:

	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

Làm việc với XML 1-2

Ngôn ngữ đánh dấu mở rộng [Extensible Markup Language (XML)] cho phép các nhà phát triển tự phát triển bộ thẻ(tag) cho riêng mình và làm cho các chương trình khác có thể hiểu các thẻ này.

XML là phương tiện ưa chuộng dành cho các nhà phát triển để lưu trữ, định dạng và quản lý dữ liệu trên Web.

Các ứng dụng ngày nay có sự pha trộn(mix) của nhiều công nghệ như ASP, công nghệ Microsoft .NET, XML, và SQL Server 2012 làm việc song song (tandem) với nhau.

Trong bối cảnh như vậy, tốt hơn hết là hãy lưu trữ dữ liệu XML trong SQL Server 2012.

Làm việc với XML 2-2

- Cơ sở dữ liệu XML nguyên gốc (Native XML) trong SQL Server 2012 có một số thuận lợi được liệt kê dưới đây:

Dễ dàng quản lý và tìm kiếm dữ liệu

- Tất cả dữ liệu XML được lưu trữ cục bộ tại một nơi, do vậy dễ dàng cho việc quản lý và tìm kiếm.

Hiệu suất tốt hơn (Better Performance)

- Các truy vấn từ csdl XML được thực thi tốt (well-implemented XML) nhanh hơn việc truy vấn trên các tài liệu được lưu trữ trên hệ thống tập tin (file system).
- Ngoài ra, csdl phân tích cơ bản mỗi tài liệu khi lưu trữ nó.

Dễ dàng xử lý dữ liệu

- Các tài liệu lớn có thể xử lý dễ dàng.

- SQL Server 2012 hỗ trợ lưu trữ nguyên gốc dữ liệu XML (native storage of XML data) bằng cách dùng kiểu dữ liệu XML.

Làm việc với XML 1-3

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

Kiểu dữ liệu xml được sử dụng để lưu trữ các đoạn(fragments) và tài liệu XML trong một csdl SQL Server.

Một đoạn(fragment) XML là một thể hiện XML khuyết(thiếu) đi phần tử mức đỉnh(top) trong cấu trúc của nó.

➤ Cú pháp để tạo một bảng có các cột có kiểu xml như sau:

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

Làm việc với XML 2-3

- Đoạn mã dưới đây tạo bảng mới có tên và một trong các cột của nó có kiểu dữ liệu xml:



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

- Cột có kiểu dữ liệu XML có thể thêm ngay lúc tạo bảng hoặc sau khi bảng đã được tạo.
- Các cột kiểu dữ liệu xml cũng hỗ trợ DEFAULT và ràng buộc NOT NULL.
- Dữ liệu có thể được chèn vào cột xml trong bảng Person.PhoneBilling như trong đoạn code dưới đây:

```
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
```

Làm việc với XML 3-3

- Kết quả được thể hiện trong hình dưới đây:

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

- Câu lệnh DECLARE được sử dụng khai báo các biến có kiểu xml.
- Đoạn code sau minh họa cách tạo một biến kiểu xml:

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

- Các cột kiểu dữ liệu không sử dụng làm khóa chính(primary key), khóa ngoại (foreign key), hoặc ràng buộc unique.

Typed XML và Untyped XML 1-4

Có hai cách để lưu trữ các tài liệu XML trong các cột có kiểu dữ liệu xml là XML có định kiểu(typed XML) và XML không định kiểu (untyped XML).

Một thể hiện XML có lược đồ gắn với nó được gọi là thể hiện XML định kiểu(typed XML instance). Một lược đồ là tiêu đề cho thể hiện hoặc tài liệu XML.

Nó mô tả cấu trúc và những giới hạn nội dung của các tài liệu XML bằng cách gắn kiểu của dữ liệu xml với thuộc tính và các loại phần tử XML.

Việc gắn kết (Associating) các lược đồ XML(XML schemas) với các thể hiện hoặc các tài liệu XML được khuyến nghị vì dữ liệu có thể được kiểm tra khi nó được lưu trữ vào các cột có kiểu dữ liệu xml.

SQL Server không thực hiện bất kỳ kiểm tra hợp lệ nào cho dữ liệu được nhập vào cột xml. Tuy nhiên, nó đảm bảo rằng dữ liệu được lưu trữ là hợp khuôn dạng (well-formed).

Dữ liệu XML không định kiểu(Untyped XML data) cũng có thể được tạo ra và lưu trữ trong các cột của bảng hoặc thay đổi(variables) tùy theo nhu cầu và phạm vi của dữ liệu.

Typed và Untyped XML 2-4

- Bước thứ nhất trong việc dùng XML định kiểu là đăng ký một lược đồ(schema)
- Để thực hiện, hãy sử dụng câu lệnh CREATE XML SCHEMA

```
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" />
```

Typed và Untyped XML 3-4

```
</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 một tập(collection) của lược đồ, được sử dụng để kiểm tra tính hợp lệ dữ liệu XML định kiểu với tên của tập(collection).
- Ví dụ trên cho thấy một lược đồ mới CricketSchemaCollection được thêm vào csdl SampleDB.
- Khi một lược đồ được đăng ký, lược đồ có thể được sử dụng trong các thể hiện mới của kiểu dữ liệu xml.

Typed và Untyped XML 4-4

- Đoạn mã sau đây minh họa tạo một bảng có cột kiểu xml và chỉ ra một lược đồ cho cột đó:

```
USE SampleDB

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

GO
```

- Để thêm các dòng mới với dữ liệu XML có định kiểu, câu lệnh INSERT được sử dụng như trong đoạn code dưới đây:

```
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
```

- Cũng có thể tạo ra một biến có kiểu typed XML bằng cách chỉ ra tên collection của lược đồ như trong đoạn code minh họa sau:

```
USE SampleDB

DECLARE @team xml(CricketSchemaCollection)

SET @team = '<MatchDetails><Team
country="Australia"></Team></MatchDetails>'

SELECT @team

GO
```

XQuery 1-4

Sau khi dữ liệu XML được lưu trữ trong cột xml, nó có thể được truy vấn và lấy ra bằng ngôn ngữ XQuery.

Truy vấn XML(XML Query) hay XQuery là một ngôn ngữ truy vấn mới, là sự kết hợp cú pháp của ngôn ngữ csdl quan hệ và ngôn ngữ XPath.

XQuery có thể truy vấn dữ liệu XML có cấu trúc hoặc bán cấu trúc (semi-structured)

Để truy vấn một thể hiện XML(XML instance) được lưu trữ trong một biến hoặc cột kiểu xml, các phương thức kiểu dữ liệu xml được sử dụng.

Các nhà phát triển cần phải truy vấn các tài liệu XML, và điều này liên quan đến việc chuyển các tài liệu XML trong định dạng yêu cầu.

XQuery có thể giúp thực hiện các truy vấn phức tạp đối với một nguồn dữ liệu XML trên Web.

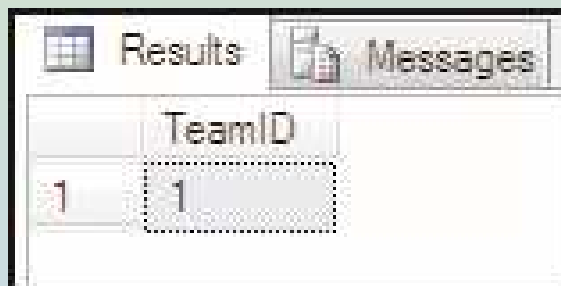
- Một số phương thức kiểu dữ liệu xml được sử dụng bằng XQuery được mô tả như sau:

exist()

- Đây là phương thức được sử dụng để xác định xem một hoặc nhiều nút(node) được chỉ ra có trong tài liệu XML hay không.
- Nó trả về 1 nếu biểu thức Xquery trả về ít nhất một nút, trả về 0 nếu biểu thức Xquery trả về kết quả trống(empty) và NULL nếu các ví dụ kiểu dữ liệu xml dựa vào đó các truy vấn được thực hiện là NULL.
- Đoạn code sau đây minh họa sử dụng phương thức `exist()` :

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

- Đoạn code này sẽ chỉ trả về những giá trị TeamID trong đó phần tử Team được chỉ ra trong TeamInfo. Kết quả được thể hiện trong hình sau đây:



The screenshot shows the 'Results' pane of SQL Server Enterprise Manager. It displays a single row of data with two columns: 'TeamID' and an unnamed column containing the value '1'. The row is highlighted with a dashed border.

	TeamID	
1	1	

query()

- Phương thức query() có thể được sử dụng để lấy toàn bộ nội dung hoặc chọn một phần(section) của tài liệu XML.
- Đoạn code dưới đây minh họa cách sử dụng phương thức query():

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

- Kết quả được thể hiện trong hình minh họa sau:

Results		Messages	
	Info		
1	<u><Team country="Australia" score="355" /><Team co...</u>		

value()

- Phương thức value() có thể được sử dụng để trích xuất các giá trị vô hướng từ một kiểu dữ liệu xml.
- Đoạn code dưới đây minh họa sử dụng phương thức này:

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

- Kết quả được thể hiện trong hình sau:



	Score
1	355

- Câu lệnh SELECT lấy hàng và cột từ bảng.
- Câu lệnh SELECT cho phép người sử dụng để xác định biểu thức khác nhau để xem các tập kết quả một cách có trật tự.
- Một câu lệnh SELECT có thể chứa biểu thức toán học bằng cách áp dụng các nhà khai thác một hoặc nhiều cột.
- Từ khóa DISTINCT ngăn lấy các bản ghi bị trùng lặp.
- XML cho phép các nhà phát triển có thể phát triển các thẻ cho riêng mình và làm cho các chương trình khác có thể hiểu các thẻ.
- Một ví dụ XML có định kiểu là một thể hiện XML trong đó có một lược đồ liên kết với nó.
- Dữ liệu XML có thể được truy vấn và lấy ra sử dụng ngôn ngữ XQuery.