Session - 9

Advanced Queries and Joins

Welcome to the Session, **Advanced Queries and Joins**.

This session explains the various techniques to group and aggregate data and describes the concept of subqueries, table expressions, joins, and explores various set operators. The session also covers pivoting and grouping set operations.

In this Session, you will learn to:

- Explain grouping and aggregating data
- Describe subqueries
- Describe table expressions
- Explain joins
- Describe various types of joins
- Explain the use of various set operators to combine data
- Describe pivoting and grouping set operations

9.1 Giới thiệu

SQL Server 2012 đưa vào một số tính năng truy vấn mạnh mẽ giúp bạn lấy dữ liệu một cách hiệu quả và nhanh chóng. Dữ liệu có thể được nhóm lại và/hoặc tập hợp lại với nhau để trình bày thông tin tóm tắt. Sử dụng khái niệm các truy vấn con, một tập kết quả của SELECT có thể được sử dụng làm tiêu chí cho câu lệnh SELECT hoặc truy vấn khác. Phép nối giúp bạn kết hợp dữ liệu cột từ hai cột hoặc nhiều bảng dựa trên mối quan hệ logic giữa những bảng này. Mặt khác, các toán tử tập hợp như UNION và INTERSECT giúp bạn kết hợp dữ liệu hàng từ hai hoặc nhiều bảng. Các toán tử PIVOT và UNPIVOT được sử dụng để chuyển đổi hướng của dữ liệu từ hướng cột sang hướng hàng và ngược lại. Mệnh đề con GROUPING SET của mệnh đề GROUP BY giúp chỉ ra nhiều nhóm trong một truy vấn đơn lẻ.

9.2 Grouping Data

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. 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 của mệnh đề GROUP BY.

Cú pháp:

```
CREATE TYPE [ schema_name. ] type_name { FROM base_type [ (precision [ , scale ] ) ] [
NULL | NOT NULL ] } [ ; ]
```

trong đó,

column_name: là tên của cột theo đó tập kết quả nên được nhóm lại.

Xem xét bảng WorkOrderRouting trong cơ sở dữ liệu AdventureWorks2012. Tổng số giờ tài nguyên cho mỗi lệnh sản xuất cần phải được tính toán. Để đạt được điều này, các bản ghi cần phải được nhóm lại theo số thứ tự công việc, đó là WorkOrderID.

Code Snippet 1 lấy và hiển thị tổng số giờ tài nguyên mỗi đơn hàng làm việc cùng với số thứ tự công việc. Trong truy vấn này, hàm dựng sẵn có tên là SUM() được sử dụng để tính toán tổng số. SUM() là một hàm tổng hợp. Các hàm tổng hợp sẽ được trình bày chi tiết trong phần sau.

Code Snippet 1:

```
SELECT WorkOrderID, SUM(ActualResourceHrs) AS TotalHoursPerWorkOrder FROM
Production.WorkOrderRouting GROUP BY WorkOrderID
```

Thực hiện truy vấn này sẽ trả lại tất cả những số lệnh công tác cùng với tổng số giờ tài nguyên cho mỗi lệnh công tác.

Concepts

Một phần của kết quả được trình bày trong hình 9.1.

	WorkOrderID	TotalHoursPerWorkOrder
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 9.1: Using the GROUP BY Clause

Mệnh đề GROUP BY cũng có thể được sử dụng kết hợp với các mệnh đề khác. Những mệnh đề này như sau:

- **GROUP BY với WHERE**

  Mệnh đề WHERE cũng có thể được sử dụng với mệnh đề GROUP BY để hạn chế các hàng để gộp nhóm. Các hàng đáp ứng điều kiện tìm kiếm được xem xét để tạo nhóm. Các hàng không đáp ứng các điều kiện trong mệnh đề WHERE được loại bỏ trước khi bất kỳ gộp nhóm nào được thực hiện.

  Code Snippet 2 trình bày một truy vấn tương tự như Đoạn mã 1 nhưng hạn chế các hàng được hiển thị, bằng cách xem xét chỉ các bản ghi với WorkOrderID nhỏ hơn 50.

  Code Snippet 2:

```
SELECT WorkOrderID, SUM(ActualResourceHrs) AS TotalHoursPerWorkOrder
FROM Production.WorkOrderRouting WHERE WorkOrderID<50 GROUP BY WorkOrderID
```

  Bởi số bản ghi trả lại lớn hơn 25, một phần của kết quả được trình bày trong Hình 9.2.

	WorkOrderID	TotalHoursPerWorkOrder
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

Hình 9.2: GROUP BY with Where

- **GROUP BY với NULL**

  Nếu cột gộp nhóm chứa giá trị NULL, dòng đó sẽ trở thành một nhóm riêng biệt trong tập kết quả. Nếu cột gộp nhóm có nhiều hơn một giá trị NULL, giá trị NULL được đưa vào một hàng đơn lẻ. Xem xét bảng Production.Product. Có một số hàng trong đó có các giá trị NULL trong cột Class.

Concepts

Sử dụng GROUP BY trên một truy vấn cho bảng này cũng sẽ được xem xét các giá trị NULL. Ví dụ, Code Snippet 3 lấy và hiển thị giá trị trung bình của giá niêm yết cho mỗi Class.

Code Snippet 3:

```
SELECT Class, AVG (ListPrice) AS 'AverageListPrice' FROM
Production.Product GROUP BY Class
```

Như được trình bày trong hình 9.3, các giá trị NULL được nhóm lại thành một hàng đơn lẻ trong kết quả.

	Class	AverageListPrice
1	NULL	16.314
2	H	1679.4964
3	L	370.6887
4	M	635.5816

Hình 9.3: GROUP BY with NULL

- **GROUP BY with All**

Cũng có thể được sử dụng từ khóa **ALL** với mệnh đề **GROUP BY**. Nó chỉ có ý nghĩa khi **SELECT** có mệnh đề **WHERE**. Khi ALL được sử dụng, nó đưa vào tất cả các nhóm mà mệnh đề **GROUP BY** tạo ra. Nó thậm chí bao gồm cả các nhóm mà không đáp ứng các điều kiện tìm kiếm. Sau đây là cú pháp của việc sử dụng **GROUP BY** với **ALL**.

Cú pháp:

```
SELECT <column_name> FROM <table_name> WHERE <condition> GROUP BY ALL
<column_name>
```

Xem xét bảng Sales.SalesTerritory. Bảng này có một cột tên là Group chỉ ra khu vực địa lý khi lãnh thổ bán hàng thuộc vào nơi đó. Đoạn mã 4 tính toán và hiển thị tổng doanh số cho mỗi nhóm. Kết quả cần để hiển thị tất cả các nhóm cho dù họ có bất kỳ doanh số nào hay không. Để đạt được điều này, đoạn mã sử dụng GROUP BY với ALL.

Code Snippet 4:

```
SELECT [Group], SUM(SalesYTD) AS 'TotalSales' FROM Sales.SalesTerritory
WHERE [Group] LIKE 'N%' OR [Group] LIKE 'E%' GROUP BY ALL [Group]
```

Ngoài các hàng được hiển thị trong Code Snippet 4,nó cũng sẽ hiển thị nhóm 'Pacific' với các giá trị null như được trình bày trong hình 9.4. Điều này là do khu vực Thái Bình Dương không có bất kỳ doanh số nào.

	Group	TotalSales
1	Europe	13590506.0212
2	North America	33182889.0168
3	Pacific	NULL

Hình 9.4: GROUP BY with All

- **GROUP BY với HAVING**

  Mệnh đề HAVING chỉ được sử dụng với câu lệnh SELECT để chỉ ra một điều kiện tìm kiếm cho một nhóm. Mệnh đề HAVING hoạt động như mệnh đề WHERE ở những nơi mệnh đề WHERE không thể được sử dụng đối với các hàm tổng hợp như là SUM(). Một khi bạn đã tạo ra các nhóm với mệnh đề GROUP BY, bạn có thể muốn lọc các kết quả hơn nữa. Mệnh đề HAVING hoạt động như một bộ lọc trên các nhóm, tương tự như cách mệnh đề WHERE hoạt động như một bộ lọc trên các hàng trả về từ mệnh đề FROM. Sau đây là cú pháp của GROUP BY với HAVING.

  Cú pháp:

```
SELECT <column_name> FROM <table_name> GROUP BY <column_name> HAVING
<search_condition>
```

  Code Snippet 5 hiển thị hàng với nhóm 'Pacific' vì nó có tổng doanh thu ít hơn 6000000.

  Code Snippet 5:

```
SELECT [Group], SUM(SalesYTD) AS 'TotalSales' FROM Sales.SalesTerritory
WHERE [Group] LIKE 'N%' OR [Group] LIKE 'E%' GROUP BY ALL [Group]
```

  Đầu ra của mã này là hàng duy nhất, với tên Group là Pacific và tổng doanh thu là 5.977.814,9154.

9.3 Tổng kết dữ liệu

Mệnh đề GROUP BY còn sử dụng các toán tử như là CUBE và ROLLUP để trả lại dữ liệu tóm tắt. Số lượng các cột trong mệnh đề GROUP BY xác định số hàng tóm tắt trong tập kết quả. Các toán tử được mô tả như sau:

- **CUBE:** CUBE là một toán tử tổng hợp tạo ra một hàng siêu tổng hợp. Ngoài các hàng thông thường được cung cấp bằng GROUP BY, nó còn cung cấp bản tóm tắt các hàng mà mệnh đề GROUP BY tạo ra. Hàng tóm tắt được hiển thị cho tất cả các tổ hợp có thể có của các nhóm trong tập kết quả. Hàng tóm tắt hiển thị NULL trong tập kết quả nhưng đồng thời trả về tất cả những giá trị cho những cái này. Sau đây là cú pháp của CUBE.

  Cú pháp:

```
SELECT <column_name> FROM <table_name> GROUP BY <column_name> WITH CUBE
```

  Code Snippet 6 trình bày việc sử dụng CUBE.

  Code Snippet 6:

```
SELECT Name, CountryRegionCode, SUM(SalesYTD) AS TotalSales FROM Sales.
SalesTerritory WHERE Name <> 'Australia' AND Name<> 'Canada' GROUP BY
Name, CountryRegionCode WITH CUBE
```

  Code Snippet 6 lấy và hiển thị tổng doanh thu của mỗi quốc gia và ngoài ra, tổng doanh thu của tất cả các vùng của các nước.

Kết quả được trình bày trong Hình 9.5.

	Name	CountryRegionCode	TotalSales
1	Germany	DE	3805202.3478
2	NULL	DE	3805202.3478
3	France	FR	4772398.3078
4	NULL	FR	4772398.3078
5	United Kingdom	GB	5012905.3656
6	NULL	GB	5012905.3656
7	Central	US	3072175.118
8	Northeast	US	2402176.8476
9	Northwest	US	7887186.7882
10	Southeast	US	2538667.2515

Hình 9.5: Using Group By with CUBE

- **ROLLUP:** Ngoài những hàng thông thường được tạo ra bằng GROUP BY, nó còn đưa các hàng tóm tắt vào tập kết quả. Nó tương tự như toán tử CUBE, nhưng tạo ra một tập kết quả cho thấy các nhóm được sắp xếp theo một trật tự thứ bậc. Nó sắp xếp các nhóm từ thấp lên cao. Hệ thống phân cấp nhóm trong kết quả phụ thuộc vào thứ tự trong đó các cột được nhóm lại được chỉ ra.
  Sau đây là cú pháp của ROLLUP.

Cú pháp:

```
SELECT <column_name> FROM <table_name> GROUP BY <column_name> WITH ROLLUP
```

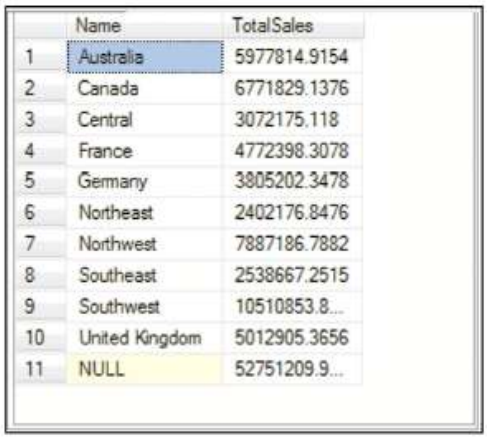
Code Snippet 7 trình bày việc sử dụng ROLLUP. Nó lấy và hiển thị tổng doanh thu của mỗi quốc gia và ngoài ra, tổng doanh thu của tất cả các vùng của các nước và sắp xếp chúng theo thứ tự.

Code Snippet 7:

```
SELECT Name, CountryRegionCode, SUM(SalesYTD) AS TotalSales
FROM Sales.SalesTerritory
WHERE Name <> 'Australia' AND Name<> 'Canada'
GROUP BY Name, CountryRegionCode
WITH ROLLUP
```

Concepts

Kết quả được trình bày trong hình Hình 9.6.



	Name	TotalSales
1	Australia	5977814.9154
2	Canada	6771829.1376
3	Central	3072175.118
4	France	4772398.3078
5	Germany	3805202.3478
6	Northeast	2402176.8476
7	Northwest	7887186.7882
8	Southeast	2538667.2515
9	Southwest	10510853.8...
10	United Kingdom	5012905.3656
11	NULL	52751209.9...

Hình 9.6: Using Group By with ROLLUP

9.4 Các hàm tổng hợp

Thỉnh thoảng, các nhà phát triển cũng có thể cần phải thực hiện phân tích trên các hàng, chẳng hạn như đếm số hàng đáp ứng các tiêu chí cụ thể hoặc tóm tắt tổng doanh số cho tất cả các đơn đặt hàng. Các hàm tổng hợp cho phép thực hiện điều này.

Do các hàm tổng hợp trả lại một giá trị duy nhất, chúng có thể được sử dụng trong các câu lệnh SELECT nơi biểu thức đơn lẻ được sử dụng, như là các mệnh đề SELECT, HAVING, và ORDER BY. Các hàm tổng hợp bỏ qua NULL, ngoại trừ khi sử dụng COUNT(*).

Các hàm tổng hợp trong danh sách SELECT không tạo ra bí danh cột. Bạn có thể muốn sử dụng mệnh đề AS để cung cấp một cái.

Các hàm tổng hợp trong mệnh đề SELECT hoạt động trên tất cả các hàng được truyền vào giai đoạn SELECT. Nếu không có mệnh đề GROUP BY, tất cả các hàng sẽ được tóm tắt lại.

SQL Server cung cấp nhiều hàm tổng hợp dựng sẵn. Các hàm thường sử dụng được đưa vào trong table 9.1 :

Tên hàm	Cú pháp	Mô tả
AVG	AVG(<biểu thức>)	Tính giá trị trung bình của tất cả các giá trị số không phải là NULL trong một cột.
COUNT or COUNT_BIG	COUNT(*) or COUNT(<biểu thức>)	Khi (*) được sử dụng, hàm này đếm tất cả các hàng, bao gồm cả những hàng có NULL. Hàm trả về số hàng không phải là NULL cho cột đó khi một cột được quy định là <biểu thức> Giá trị trả về của hàm COUNT là int. Giá trị trả về của COUNT_BIG là một big int.

Tên hàm	Cú pháp	Mô tả
MAX	MAX(<biểu thức>)	Trả về số lớn nhất, ngày/giờ mới nhất, hoặc chuỗi xảy ra cuối cùng.
MIN	MlN(<biểu thức>)	Trả về số nhỏ nhất, ngày/giờ mới nhất, hoặc chuỗi xảy ra đầu tiên.
SUM	SUM(<biểu thức>)	Tính tổng của tất cả các giá trị số không phải là NULL trong một cột.

Table 9.1: Commonly used Aggregate Functions

Để sử dụng tổng hợp dựng sẵn trong mệnh đề SELECT, xem xét truy vấn sau đây trong Code Snippet 8.

Code Snippet 8:

```
SELECT AVG([UnitPrice]) AS AvgUnitPrice,
MIN([OrderQty])AS MinQty,
MAX([UnitPriceDiscount]) AS MaxDiscount

FROMSales.SalesOrderDetail;
```

Do truy vấn không sử dụng mệnh đề GROUP BY, tất cả các hàng trong bảng sẽ được tóm tắt bằng các công thức tổng hợp trong mệnh đề SELECT.

Kết quả được trình bày trong hình Hình 9.7.

	AvgUnitPrice	MinQty	MaxDiscount
1	465.0934	1	0.40

Hình 9.7: Using Aggregate Functions

Khi sử dụng các giá trị tổng hợp trong mệnh đề SELECT, tất cả các cột được tham chiếu trong danh sách SELECT phải được sử dụng làm đầu vào cho hàm tổng hợp hoặc phải được tham chiếu trong mệnh đề GROUP BY. Không làm này, sẽ có một lỗi. Ví dụ, truy vấn trong Code Snippet 9 sẽ trả về một lỗi.

Code Snippet 9:

```
SELECT SalesOrderID, AVG(UnitPrice) AS AvgPrice

FROMSales.SalesOrderDetail;
```

Concepts

Cái này trả về một lỗi chỉ ra rằng cột Sales.SalesOrderDetail.SalesOrderID không hợp lệ trong danh sách SELECT bởi vì nó không được chứa trong hàm tổng hợp hoặc mệnh đề GROUP BY. Bởi truy vấn không sử dụng mệnh đề GROUP BY, tất cả các hàng sẽ được đối xử như một nhóm duy nhất. Do đó, tất cả các cột phải được sử dụng làm đầu vào cho các hàm tổng hợp. Để sửa chữa hoặc ngăn chặn lỗi, cần phải loại bỏ SalesOrderID khỏi truy vấn.

Ngoài việc sử dụng dữ liệu số, biểu thức tổng hợp cũng có thể bao gồm ngày, giờ, và dữ liệu ký tự cho việc tổng hợp.

Code Snippet 10 trả về ngày đặt hàng sớm nhất và mới nhất, sử dụng MIN và MAX.

Code Snippet 10:

```
SELECT MIN(OrderDate) AS Earliest,
MAX(OrderDate) AS Latest
FROM Sales.SalesOrderHeader;
```

Hình 9.8 shows the output.

	Earliest	Latest
1	2005-07-01 00:00:00.000	2008-07-31 00:00:00.000

Hình 9.8: Using Aggregate Functions with Non-Numeric Data

Các hàm khác cũng có thể được sử dụng kết hợp với các hàm tổng hợp.

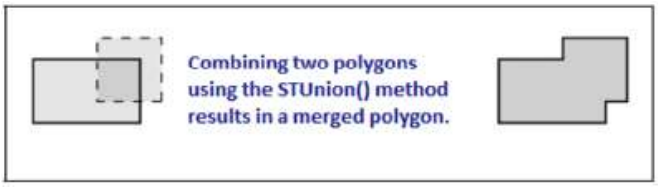
9.5 Các tập hợp không gian

SQL Server cung cấp một số phương pháp có thể giúp tổng hợp hai mục riêng lẻ của dữ liệu hình học hoặc địa lý. Những phương thức này được liệt kê trong hình 9.2.

Phương thức	Mô tả
STUnion	Trả về một đối tượng đại diện cho phép hợp của một thể hiện hình học/địa lý với một thể hiện hình học/địa lý khác.
STintersection	Trả về một đối tượng đại diện cho các điểm nơi một thể hiện hình học/địa lý giao cắt với một thể hiện hình học/địa lý khác.
STConvexHull	Trả về một đối tượng trình bày vỏ lồi của một thể hiện hình học/địa lý. Tập hợp các điểm được gọi là lồi nếu đối với bất kỳ hai điểm nào, toàn bộ đoạn được chứa trong tập hợp. Vỏ lồi của tập hợp điểm là tập lồi nhỏ nhất chứa tập hợp này. Đối với bất kỳ tập hợp các điểm nào, chỉ có một vỏ lồi.

Table 9.2: Spatial Aggregate Methods

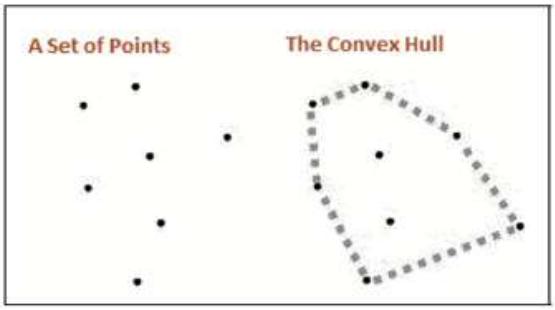
Hình 9.9, 9.10, và 9.11 mô tả trực quan ví dụ về những phương pháp này.



Hình 9.9: STUnion()

Concepts

Hình 9.10: STintersection()



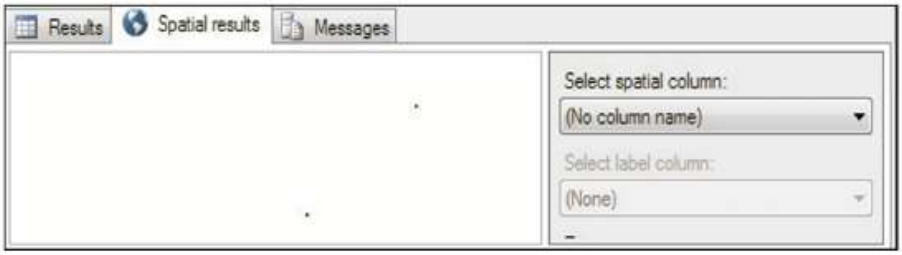
Hình 9.11: STConvexHull()

Code Snippet 11 trình bày việc sử dụng STUnion ( ).

Code Snippet 11:

```
SELECT geometry::Point(251, 1, 4326).STUnion(geometry::Point(252,2,4326));
```

Kết quả được trình bày trong hình 9.12. Nó trình bày hai điểm.



Hình 9.12: Using STUnion() with a geometry Type

Một ví dụ khác được đưa ra trong Code Snippet 12.

Code Snippet 12:

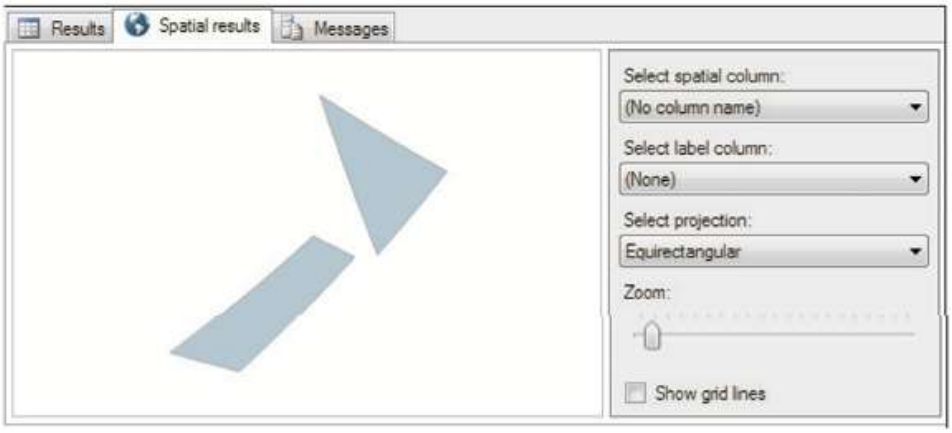
```
DECLARE @City1 geography
SET @City1 = geography::STPolyFromText(
 'POLYGON((175.3 -41.5, 178.3 -37.9, 172.8 -34.6, 175.3 -41.5))',
 4326)
DECLARE @City2 geography
```

Concepts

```
SET @City2 = geography::STPolyFromText(
 'POLYGON((169.3 -46.6, 174.3 -41.6, 172.5 -40.7, 166.3 -45.8, 169.3 -46.6))',
 4326)
DECLARE @CombinedCity geography = @City1.STUnion(@City2)
SELECT @CombinedCity
```

Ở đây, hai biến được khai báo thuộc loại geography và giá trị thích hợp được gán cho chúng. Sau đó, chúng được kết hợp thành một biến thứ ba thuộc loại geography bằng cách sử dụng phương pháp STUnion().

Kết quả của đoạn mã này được trình bày trong hình 9.13.



Hình 9.13: Using STUnion() with a geography Type

9.5.1 Các tổng hợp không gian mới

Thật dễ dàng để tính toán phép hợp của dữ liệu số thông thường bằng cách sử dụng các toán tử cơ bản với các truy vấn như là SELECT x + y hoặc bằng cách sử dụng toán tử UNION. Bạn có thể tính toán phép hợp của hai hình học riêng lẻ hoặc hai khu vực địa lý bằng cách sử dụng toán tử STUnion(). Điều gì xảy ra nếu có nhu cầu phải tính toán phép hợp của một tập hợp các đối tượng hình học/địa lý hoặc tất cả các phần tử trong một cột không gian? Điều gì xảy ra nếu có nhu cầu phải tìm ra giá trị trung bình của một tập hợp các phần tử Point? Sẽ không thể sử dụng hàm AVG() ở đây. Trong trường hợp này, bạn sẽ sử dụng các hàm tổng hợp không gian mới trong SQL Server 2012.

SQL Server 2012 đã giới thiệu bốn tổng hợp mới cho bộ toán tử không gian trong SQL Server:

- Union Aggregate (Tổng hợp phép hợp)
- Envelope Aggregate (Tổng hợp phong bì)
- Collection Aggregate (Tổng hợp bộ sưu tập)
- Convex Hull Aggregate (Tổng hợp vỏ lồi)

Concepts

Những tổng hợp này được thực hiện như là các phương pháp tĩnh, làm việc cho một trong kiểu dữ liệu geography hoặc geometry. Mặc dù các tổng hợp được áp dụng cho tất cả các lớp dữ liệu không gian, chúng có thể được mô tả tốt nhất với đa giác.

- **Union Aggregate – Tổng hợp phép hợp**

  Nó thực hiện phép hợp trên một tập hợp các đối tượng geometry. Nó kết hợp nhiều đối tượng không gian vào một đối tượng không gian, loại bỏ các ranh giới bên trong, nếu có thể áp dụng. Sau đây là cú pháp của UnionAggregate.

  Cú pháp:

```
UnionAggregate ( geometry_operand or geography_operand)
```

  trong đó,

  geometry_operand: là cột bảng loại geometry bao gồm tập hợp các đối tượng geometry trên đó có một phép hợp sẽ được thực hiện.

  geography_operand: là cột bảng loại geography bao gồm tập hợp các đối tượng geography trên đó có một phép hợp sẽ được thực hiện.

  Code Snippet 13 trình bày ví dụ đơn giản của việc sử dụng tổng hợp Union. Nó sử dụng bảng Person.Address trong cơ sở dữ liệu AdventureWorks2012.

  Code Snippet 13:

```
SELECT Geography::UnionAggregate(SpatialLocation)
AS AVGLocation
FROM Person.Address
WHERE City = 'London';
```

  Kết quả của mã này được trình bày trong Hình 9.14.

	AVGLocation
1	0xE61000000104A00100003DA82EC605C249407D37109CAD...

Hình 9.14: Using Spatial Aggregates

Để xem một phần trình bày trực quan của dữ liệu không gian, bạn có thể bấm vào tab **Spatial results** trong cửa sổ đầu ra. Điều này sẽ hiển thị đầu ra như được trình bày trong hình 9.15.

Hình 9.15: Viewing Spatial Results

- **Tổng hợp phong bì**

Nó trả về một khu vực giới hạn cho một tập hợp đã cho của các đối tượng geometry hoặc geography.

Dựa trên loại đối tượng được áp dụng, nó sẽ trả về các kết quả khác nhau. Đối với loại geometry, kết quả là một đa giác hình chữ nhật "truyền thống", gắn chặt chẽ các đối tượng đầu vào đã chọn. Đối với loại geography, kết quả là một đối tượng hình tròn, gắn lỏng lẻo các đối tượng đầu vào đã chọn. Hơn nữa, đối tượng hình tròn được định nghĩa sử dụng tính năng CurvePolygon mới. Sau đây là cú pháp của EnvelopeAggregate.

Cú pháp:

```
EnvelopeAggregate (geometry_operand or geography_operand)
```

trong đó,

geometry_operand: là cột bảng loại geometry bao gồm tập hợp các đối tượng geometry.

geography_operand: là cột bảng loại geography bao gồm tập hợp các đối tượng geography.

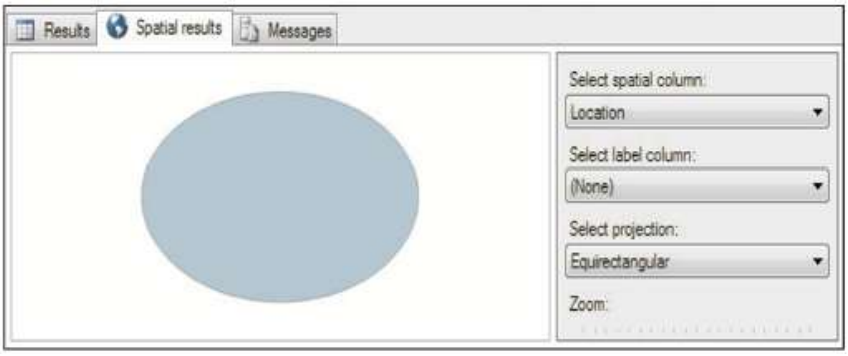
Code Snippet 14 trả về một khung giới hạn cho tập hợp các đối tượng trong một cột biến bảng.

Code Snippet 14:

```
SELECT Geography::EnvelopeAggregate(SpatialLocation)
AS Location
FROM Person.Address
WHERE City = 'London'
```

Concepts

Trình bày trực quan của kết quả này được trình bày trong Hình 9.16.



Hình 9.16: EnvelopeAggregate

- **Collection Aggregate** *(Tổng hợp bộ sưu tập)*

  Nó trả về thể hiện GeometryCollection/GeographyCollection với một phần geometry/ geography cho từng đối tượng không gian trong tập lựa chọn. Sau đây là cú pháp của CollectionAggregate.

  Syntax:

```
CollectionAggregate (geometry_operand or geography_operand)
```

  trong đó,

  geometry_operand: là cột bảng loại geometry bao gồm tập hợp các đối tượng geometry.

  geography_operand: là cột bảng loại geography bao gồm tập hợp các đối tượng geography.

  Code Snippet 15 trả về thể hiện GeometryCollection có chứa CurvePolygon và Polygon.

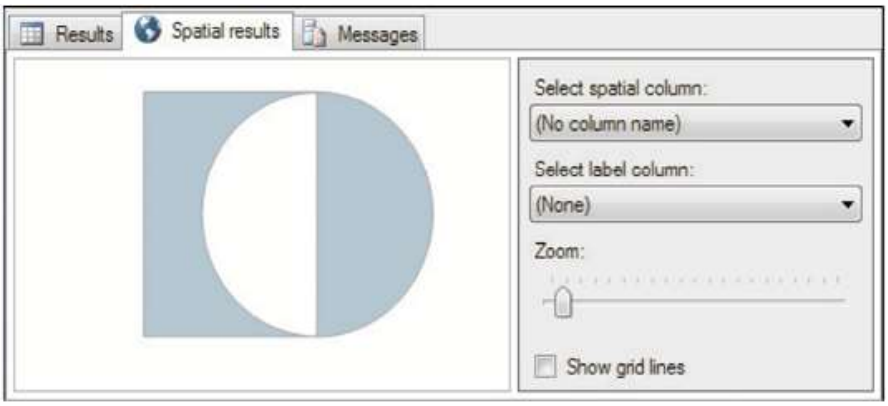
  Code Snippet 15:

```
DECLARE @CollectionDemo TABLE
(
shape geometry,
shapeType nvarchar(50)
)
INSERT INTO @CollectionDemo(shape,shapeType) VALUES('CURVEPOLYGON(CIRC
ULARSTRING(2 3, 4 1, 6 3, 4 5, 2 3))', 'Circle'),
('POLYGON((1 1, 4 1, 4 5, 1 5, 1 1))', 'Rectangle');

SELECT geometry::CollectionAggregate(shape)

FROM @CollectionDemo;
```

Kết quả của đoạn mã này được trình bày trong Hình 9.17.



Hình 9.17: Using CollectionAggregate

- **Convex Hull Aggregate** *(Tổng hợp vỏ lồi)*

Nó trả về một đa giác vỏ lồi, bao quanh một hoặc nhiều đối tượng không gian cho một tập hợp đã cho của các đối tượng geometry/geography. Sau đây là cú pháp của ConvexHullAggregate.

Cú pháp:

```
ConvexHullAggregate (geometry_operand or geography_operand)
```

where,

geometry_operand: là cột bảng loại geometry bao gồm tập hợp các đối tượng geometry.

geography_operand: là cột bảng loại geography bao gồm tập hợp các đối tượng geography.

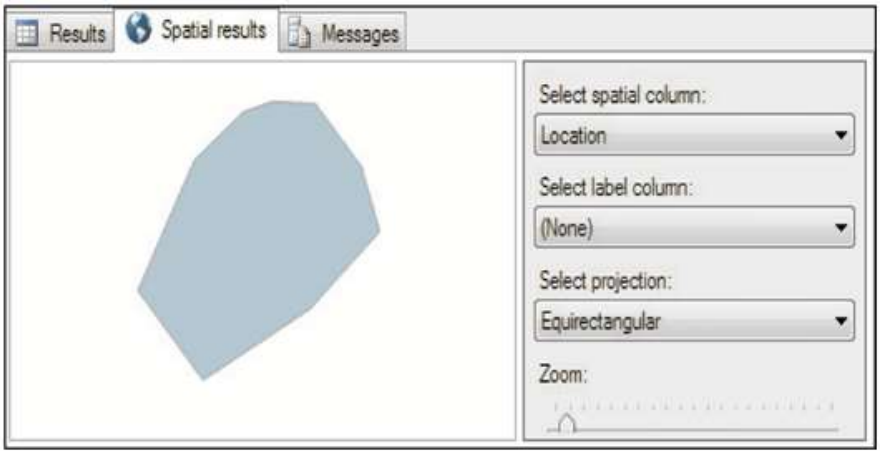
Code Snippet 16 trình bày việc sử dụng ConvexHullAggregate.

Code Snippet 16:

```
SELECT Geography::ConvexHullAggregate(SpatialLocation)
AS Location
FROM Person.Address
WHERE City = 'London'
```

Concepts

Kết quả được trình bày trong Hình 9.18.



Hình 9.18: Using ConvexHuiiAggregate

9.6 Các truy vấn con

Bạn có thể sử dụng câu lệnh SELECT hoặc một truy vấn để trả lại các bản ghi sẽ được sử dụng làm tiêu chí cho câu lệnh SELECT hoặc truy vấn khác. Truy vấn bên ngoài được gọi là truy vấn cha và truy vấn bên trong được gọi là truy vấn con. Mục đích của một truy vấn con là để trả về các kết quả cho truy vấn bên ngoài. Nói cách khác, câu lệnh truy vấn bên trong nên trả về cột này hoặc các cột được sử dụng trong tiêu chí của câu lệnh truy vấn bên ngoài.

Hình thức đơn giản nhất của một truy vấn con là cái trả về chỉ một cột. Truy vấn cha có thể sử dụng các kết quả của truy vấn con này dùng dấu =.

Cú pháp cho dạng cơ bản nhất của truy vấn con dùng chỉ một cột với dấu = được hiển thị.

Cú pháp:

```
SELECT <ColumnName> FROM <table>

WHERE <ColumnName> = ( SELECT <ColumnName> FROM <Table> WHERE <ColumnName> =
<Condition> )
```

Trong một truy vấn con, câu lệnh SELECT trong cùng được thực hiện đầu tiên và kết quả của nó được chuyển làm tiêu chí cho câu lệnh SELECT bên ngoài.

Xem xét một kịch bản trong đó cần phải xác định ngày đến hạn và ngày chuyển hàng cho các đơn đặt hàng gần đây nhất.

Concepts

Code Snippet 17 trình bày đoạn mã này để đạt được điều này.

Code Snippet 17:

```
SELECT DueDate, ShipDate
FROM Sales.SalesOrderHeader
WHERE Sales.SalesOrderHeader.OrderDate =
(SELECT MAX(OrderDate)
FROM Sales.SalesOrderHeader)
```

Ở đây, truy vấn con đã được sử dụng để đạt được kết quả mong muốn. Truy vấn hoặc truy vấn con bên trong lấy ngày đặt hàng gần đây nhất. Sau đó cái này được truyền cho truy vấn bên ngoài, hiển thị ngày đến hạn và ngày chuyển hàng cho tất cả các đơn đặt hàng đã được đặt vào ngày cụ thể đó.

Một phần của kết quả của đoạn mã này được trình bày trong Hình 9.19.

	DueDate	ShipDate
1	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
2	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
4	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
5	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
6	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
7	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
8	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
9	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
10	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000

Hình 9.19: Using a Simple Subquery

Dựa trên kết quả trả về của truy vấn bên trong, một truy vấn con có thể được phân loại là một truy vấn con vô hướng hoặc một truy vấn con có nhiều giá trị. Những công cụ này được mô tả như sau:

- Truy vấn con vô hướng trả lại một giá trị duy nhất. Ở đây, truy vấn bên ngoài cần phải được viết để xử lý một kết quả duy nhất.

- Truy vấn con có nhiều giá trị trả về kết quả tương tự như một bảng có một cột. Ở đây, truy vấn bên ngoài cần phải được viết để xử lý nhiều kết quả khả năng.

9.6.1 Làm việc với các truy vấn đa giá trị

Nếu toán tử = được sử dụng với truy vấn con, truy vấn con phải trả lại một giá trị vô hướng duy nhất. Nếu có nhiều hơn một giá trị được trả về, sẽ có lỗi và truy vấn sẽ không được xử lý. Trong kịch bản như vậy, những từ khóa ANY, ALL, IN, và EXISTS có thể được sử dụng với mệnh đề WHERE của câu lệnh SELECT khi truy vấn trả về một cột nhưng có một hoặc nhiều hàng.

Concepts

Những từ khóa này, còn được gọi là các xác nhận, được sử dụng với các truy vấn có nhiều giá trị.

Ví dụ, hãy xem xét tất cả các tên họ và tên gọi của người lao động có chức danh công việc là 'Giám đốc Nghiên cứu và Phát triển' cần được hiển thị. Ở đây, truy vấn bên trong có thể trả về nhiều hơn một dòng bởi có thể có nhiều hơn một nhân viên với chức danh công việc đó. Để đảm bảo rằng truy vấn bên ngoài có thể sử dụng các kết quả của truy vấn bên trong này, từ khóa IN sẽ phải được sử dụng.

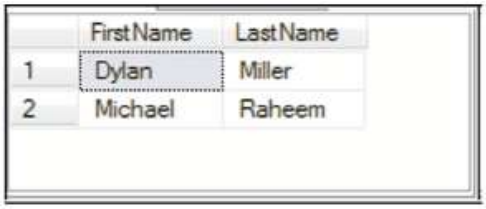
Code Snippet 18 trình bày phương thức này.

Code Snippet 18:

```
SELECT FirstName, LastName FROM Person.Person
WHERE Person.Person.BusinessEntityID IN (SELECT BusinessEntityID
FROM HumanResources.Employee WHERE JobTitle='Research and Development
Manager');
```

Ở đây, truy vấn bên trong lấy BusinessEntityID từ bảng HumanResources.Employee cho những bản ghi có chức danh công việc là 'Giám đốc Nghiên cứu và Phát triển'. Những kết quả này sau đó được truyền đến cho truy vấn bên ngoài, so khớp BusinessEntityID với cái trong bảng Person.Person. Cuối cùng, từ các bản ghi đang so khớp, tên họ và tên gọi được trích xuất và hiển thị.

Kết quả được hiển thị trong Hình 9.20.



	FirstName	LastName
1	Dylan	Miller
2	Michael	Raheem

Hình 9.20: Output of Subquery with IN keyword

Những từ khóa SOME hoặc ANY đánh giá thành true nếu kết quả là một truy vấn bên trong có chứa ít nhất một hàng đáp ứng sự so sánh này. Chúng so sánh một giá trị vô hướng với một cột các giá trị. SOME và ANY là tương đương, cả hai trả về cùng một kết quả. Chúng ít khi được sử dụng.

Có một số hướng dẫn cần phải được tuân thủ khi làm việc với các truy vấn con. Bạn nên nhớ các điểm sau đây khi sử dụng các truy vấn con :

- Không thể sử dụng kiểu dữ liệu ntext, text, và image trong danh sách các truy vấn con **SELECT**.
- Danh sách các truy vấn con **SELECT** được giới thiệu với toán tử so sánh có thể chỉ có một biểu thức hoặc tên cột.
- Truy vấn con được giới thiệu bằng toán tử so sánh không theo sau bằng từ khóa ANY hoặc ALL không thể đưa vào mệnh đề **GROUP BY** và **HAVING**.
- Bạn không thể sử dụng từ khóa **DISTINCT** với các truy vấn con bao gồm **GROUP BY**.

Concepts

- Bạn có thể chỉ ra ORDER BY chỉ khi TOP cũng được chỉ ra.

Bên cạnh các truy vấn con vô hướng và có nhiều giá trị, bạn cũng có thể lựa chọn giữa các truy vấn con khép kín và truy vấn con tương quan. Điều này được định nghĩa như sau :

- Truy vấn con khép kín được viết làm truy vấn độc lập, không có bất kỳ phụ thuộc nào vào truy vấn bên ngoài. Truy vấn con khép kín được xử lý một lần khi truy vấn bên ngoài chạy và chuyển các kết quả của nó cho truy vấn bên ngoài.

- Truy vấn con tương quan tham khảo một hoặc nhiều cột từ truy vấn bên ngoài và do đó, phụ thuộc vào truy vấn bên ngoài. Truy vấn con tương quan không thể chạy riêng rẽ với truy vấn bên ngoài.

  Từ khóa EXISTS được sử dụng với một truy vấn con để kiểm tra sự tồn tại của các hàng được trả về từ truy vấn con đó. Truy vấn con không thực sự trả về bất kỳ dữ liệu nào, nó trả về giá trị TRUE hoặc FALSE.

  Sau đây là cú pháp của một truy vấn con có chứa từ EXISTS.

  Cú pháp:

```
SELECT <ColumnName> FROM <table>
WHERE [NOT] EXISTS
(
   <Subquery_Statement>
)
```

  trong đó,

  Subquery_Statement : chỉ ra truy vấn con này.

  Mã trong Code Snippet 18 có thể được viết lại như được trình bày trong Code Snippet 19 sử dụng từ khóa EXISTS để mang lại cùng một kết quả.

  Code Snippet 19:

```
SELECT FirstName, LastName FROM Person.Person AS A
WHERE EXISTS (SELECT *
FROM HumanResources.Employee As B WHERE JobTitle = 'Research and Development
Manager' AND A.BusinessEntityID=B.BusinessEntityID);
```

  Ở đây, truy vấn con bên trong lấy tất cả các bản ghi so khớp chức danh công việc là 'Giám đốc Nghiên cứu và Phát triển' và BusinessEntityId khớp với cái trong bảng Person. Nếu không có các bản ghi phù hợp với cả hai điều kiện này, truy vấn con bên trong sẽ không trả về bất kỳ hàng nào. Như vậy, trong trường hợp đó, EXISTS sẽ trả về false và truy vấn bên ngoài cũng sẽ không trả về bất kỳ hàng nào. Tuy nhiên, mã trong Code Snippet 19 sẽ trở về hai hàng vì những điều kiện đã cho được thỏa mãn. Kết quả sẽ giống như Hình 9.20.

  Tương tự như vậy, người ta có thể sử dụng từ khóa NOT EXISTS. Mệnh đề WHERE trong đó nó được sử dụng được thỏa mãn nếu không có hàng được trả về từ truy vấn con này.

9.6.2 Các truy vấn con xếp lồng

Truy vấn con mà được được định nghĩa bên trong truy vấn con khác được gọi là truy vấn con xếp lồng.

Xem xét rằng bạn muốn lấy ra và hiển thị tên của những người từ Canada. Không có cách trực tiếp nào để lấy thông tin này từ bảng Sales.SalesTerritory không liên quan đến bảng Person.Person. Do đó, truy vấn con lồng nhau được sử dụng ở đây như được trình bày trong Code Snippet 20.

Code Snippet 20:

```
SELECT LastName, FirstName
FROM Person.Person
WHERE BusinessEntityID IN
 (SELECT BusinessEntityID
 FROM Sales.SalesPerson
 WHERE TerritoryID IN
  (SELECT TerritoryID
  FROM Sales.SalesTerritory
        WHERE Name='Canada')
 )
```

Kết quả được trình bày trong Hình 9.21.

	LastName	FirstName
1	Vargas	Garrett
2	Saraiva	José

Hình 9.21: Output of Nested Subqueries

9.6.3 Các truy vấn tương quan

Trong nhiều truy vấn có chứa các truy vấn con, truy vấn con cần được cho giá trị chỉ một lần để cung cấp các giá trị mà truy vấn cha cần. Điều này là do trong hầu hết những truy vấn, truy vấn con làm cho không có tham chiếu đến truy vấn cha mẹ, vì vậy giá trị trong truy vấn con vẫn không đổi.

Tuy nhiên, nếu truy vấn con tham chiếu tới một truy vấn cha, truy vấn con cần được đánh giá lại cho mỗi bước lặp trong truy vấn cha. Điều này là do tiêu chuẩn tìm kiếm trong truy vấn con phụ thuộc vào giá trị của một bản ghi cụ thể trong truy vấn cha.

Concepts

Khi truy vấn con lấy các tham số từ truy vấn cha của nó, cái này được biết như là truy vấn con tương quan. Xem xét rằng bạn muốn lấy tất cả các id thực thể kinh doanh của những người có thông tin liên lạc được sửa đổi lần cuối cùng không sớm hơn năm 2012. Để làm điều này, bạn có thể sử dụng truy vấn con tương quan như được trình bày trong Code Snippet 21.

Code Snippet 21:

```
SELECT e.BusinessEntityID
FROM Person.BusinessEntityContact e
WHERE e.ContactTypeID IN
(
SELECT c.ContactTypeID
FROM Person.ContactType c
WHERE YEAR(e.ModifiedDate) >=2012
)
```

Trong Code Snippet 21, truy vấn bên trong lấy các mã loại liên hệ cho tất cả những người có thông tin liên lạc đã được sửa đổi vào hoặc trước năm 2012. Những kết quả này sau đó được truyền cho truy vấn bên ngoài, so khớp những mã loại liên hệ với những người trong bảng Person.BusinessEntityContact và hiển thị các mã thực thể kinh doanh của những bản ghi đó. Hình 9.22 trình bày một phần của kết quả.

	BusinessEntityID
1	292
2	294
3	296
4	298
5	300
6	302
7	304

Hình 9.22 : Output of Correlated Queries

9.7 Các phép nối

Các phép nối được dùng để gọi ra dữ liệu từ hai hoặc nhiều bảng dữ liệu mối quan hệ logic giữa các bảng. Phép nối thông thường chỉ ra mối quan hệ khóa ngoại giữa các bảng. Nó định nghĩa cách thức trong đó hai bảng được liên kết vào một truy vấn bằng cách :

- Chỉ ra cột từ mỗi bảng để được dùng cho phép nối. Phép nối điển hình chỉ ra khóa ngoại từ một bảng và khóa được liên kết của nó trong bảng kia.
- Chỉ ra một toán tử logic như là =, <> để được sử dụng trong việc so sánh các giá trị từ những cột này.

Concepts

Phép nối có thể được quy định trong mệnh đề FROM hoặc WHERE.

Sau đây là cú pháp của câu lệnh JOIN.

Cú pháp:

```
SELECT <ColumnName1>, <ColumnName2>...<ColumnNameN>
FROM Table_A AS Table_Alias_A
 JOIN
Table_B AS Table_Alias_B
 ON
Table_Alias_A.<CommonColumn> = Table_Alias_B.<CommonColumn>
```

trong đó,

<ColumnName1>, <ColumnName2>: Là một danh sách các cột cần được hiển thị.

Table_A: Là tên của bảng bên trái của từ khóa JOIN.

Table_B: Là tên của bảng bên phải của từ khóa JOIN.

AS Table_Alias: Là cách gán tên bí danh cho bảng. Có thể sử dụng bí danh được định nghĩa cho bảng để biểu thị bảng để không cần phải dùng tên đầy đủ của bảng.

<CommonColumn>: Là cột chung cho cả hai bảng. Trong trường hợp này, việc nối chỉ thành công khi các cột có các giá trị so khớp.

Xem xét rằng bạn muốn liệt kê tên họ, tên gọi của nhân viên và các chức danh công việc của họ từ HumanResources.Employee và Person.Person. Để trích xuất thông tin này từ hai bảng, bạn cần phải nối chúng lại dựa trên BusinessEntityID như được trình bày trong Code Snippet 22.

Code Snippet 22:

```
SELECT A.FirstName, A.LastName, B.JobTitle
FROM Person.Person A
JOIN
HumanResources.Employee B
ON
A.BusinessEntityID = B.BusinessEntityID;
```

Ở đây, các bảng HumanResources.Employee và Person.Person được cho bí danh A và B. Chúng được nối với nhau trên cơ sở các id thực thể kinh doanh của chúng. Câu lệnh SELECT sau đó lấy các cột mong muốn thông qua các bí danh.

Concepts

Hình 9.23 trình bày kết quả.

	FirstName	LastName	JobTitle
1	Ken	Sánchez	Chief Executive Officer
2	Terri	Duffy	Vice President of Engineering
3	Roberto	Tamburello	Engineering Manager
4	Rob	Walters	Senior Tool Designer
5	Gail	Erickson	Design Engineer
6	Jossef	Goldberg	Design Engineer
7	Dylan	Miller	Research and Development Manager

Hình 9.23: Output of Join

Có ba loại phép nối như sau:

- Inner Joins *(Phép nối trong)*
- Outer Joins *(Phép nối ngoài)*
- Self-Joins *(Phép tự nối)*

9.7.1 Inner Join

Phép nối trong được hình thành khi các bản ghi từ hai bảng được kết hợp chỉ khi các hàng từ cả hai bảng được so khớp dựa trên một cột chung. Sau đây là cú pháp của phép nối bên trong.

Cú pháp:

```
SELECT <ColumnName1>, <ColumnName2>...<ColumnNameN> FROM
Table_A AS Table_Alias_A
 INNER JOIN
Table_B AS Table_Alias_B
 ON
Table_Alias_A.<CommonColumn> = Table_Alias_B.<CommonColumn>
```

Code Snippet 23 trình bày việc sử dụng phép nối bên trong. Kịch bản cho tính năng này tương tự như Code Snippet 22.

Code Snippet 23:

```
SELECT A.FirstName, A.LastName, B.JobTitle
FROM Person.Person A
INNER JOIN HumanResources.Employee B
          ON
A.BusinessEntityID = B.BusinessEntityID;
```

Concepts

Trong Code Snippet 23, phép nối bên trong được xây dựng giữa Person.Person và HumanResources.Employee dựa trên các id thực thể kinh doanh thông thường. Ở đây một lần nữa, hai bảng được cho các bí danh tương ứng là A và B. Kết quả giống như được trình bày trong hình 9.23.

9.7.2 Outer Join

Phép nối ngoài là các câu lệnh phép nối trả về tất cả các hàng từ ít nhất một trong các bảng đã chỉ định trong mệnh đề FROM, miễn là những hàng này đáp ứng bất kỳ điều kiện WHERE hoặc HAVING nào của câu lệnh SELECT.

Hai loại phép nối bên ngoài thường được sử dụng như sau :

- Left Outer Join
- Right Outer Join

Mỗi loại phép nối này sẽ được giải thích dưới đây.

- **Left Outer Join**

  Left outer join (Nối ngoài bên trái) trả lại tất cả các bản ghi từ bảng bên trái và chỉ so khớp các bản ghi từ bảng bên phải. Sau đây là cú pháp của phép nối bên ngoài.

  Cú pháp:

```
SELECT <ColumnList> FROM
Table_A AS Table_Alias_A
 LEFT OUTER JOIN
Table_B AS Table_Alias_B
 ON
Table_Alias_A.<CommonColumn> = Table_Alias_B.<CommonColumn>
```

  Xem xét rằng bạn muốn lấy tất cả các id khách hàng từ bảng Sales.Customers và thông tin đơn đặt hàng như là ngày gửi hàng và ngày đến hạn, ngay cả khi khách hàng đã không đặt bất kỳ đơn đặt hàng nào. Do số bản ghi sẽ rất lớn, nó sẽ được giới hạn chỉ những đơn đặt hàng được đặt trước năm 2012. Để đạt được điều này, bạn thực hiện phép nối bên ngoài bên trái như được trình bày trong Code Snippet 24.

  Code Snippet 24:

```
SELECT A.CustomerID, B.DueDate, B.ShipDate
FROM Sales.Customer A LEFT OUTER JOIN
Sales.SalesOrderHeader B
ON
A.CustomerID = B.CustomerID AND YEAR(B.DueDate)<2012;
```

Concepts

Trong Code Snippet 24, phép nối bên ngoài bên trái được xây dựng giữa các bảng Sales.Customer và Sales.SalesOrderHeader. Những bảng này được nối trên cơ sở các id khách hàng. Trong trường hợp này, tất cả các bản ghi từ bảng bên trái Sales.Customer và chỉ các bản ghi so khớp từ bảng bên phải Sales.SalesOrderHeader, được trả về. Hình 9.24 trình bày kết quả.

	CustomerID	DueDate	ShipDate
3...	18178	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3...	13671	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3...	11981	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3...	18749	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3...	15251	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3...	15868	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3...	18759	2008-08-12 00:00:00.000	2008-08-07 00:00:00.000
3...	215	NULL	NULL
3...	46	NULL	NULL
3...	169	NULL	NULL
3...	507	NULL	NULL
3...	630	NULL	NULL

Hình 9.24: Output of Left Outer Join

Như được trình bày trong kết quả, một số bản ghi trình bày các ngày đến hạn và ngày gửi hàng là NULL. Điều này là do cho một số khách hàng, không có đơn đặt hàng nào được đặt, do đó, các bản ghi của họ sẽ hiển thị những ngày này là NULL.

- **Right Outer Join**

Nối ngoài bên phải gọi ra tất cả các bản ghi từ bảng thứ hai trong phần nối bất kể là có dữ liệu so khớp trong bảng thứ nhất hay không. Sau đây là cú pháp của phép nối bên ngoài bên phải.

Cú pháp:

```
SELECT <ColumnList>
FROM Left_Table_Name
AS
Table_A AS Table_Alias_A
  RIGHT OUTER JOIN
Table_B AS Table_Alias_B
  ON
Table_Alias_A.<CommonColumn> = Table_Alias_B.<CommonColumn>
```

Concepts

Xem xét rằng bạn muốn lấy tất cả các tên sản phẩm từ bảng Product và tất cả các id đơn đặt hàng tương ứng từ bảng SalesOrderDetail ngay cả khi không có bản ghi phù hợp cho những sản phẩm trong bảng SalesOrderDetail. Để làm điều này, bạn sẽ sử dụng phép nối bên ngoài bên phải như được trình bày trong Code Snippet 25.

Code Snippet 25:

```
SELECT P.Name, S.SalesOrderID
FROM Sales.SalesOrderDetail S
RIGHT OUTER JOIN
Production.Product P
ON P.ProductID = S.ProductID;
```

Trong đoạn mã này, tất cả các bản ghi từ bảng Product được hiển thị bất kể chúng đã được bán hay chưa.

9.7.3 Self-Join

Phép tự nối được dùng để tìm ra các bản ghi trong một bảng và có liên quan đến các bản ghi khác trong cùng một bảng. Bảng được nối vào bản thân nó trong một phép tự nối.

Hãy xem xét bảng **Employee** trong cơ sở dữ liệu có tên là **Sterling** có cột tên là **mgr_id** để biểu thị thông tin cho các nhà quản lý nơi nhân viên phải báo cáo cho họ. Giả sử rằng bảng có các bản ghi thích hợp được đưa vào nó.

Người quản lý cũng là một nhân viên. Điều này có nghĩa là **mgr_id** trong bảng là **emp_id** của một nhân viên.

Concepts

Ví dụ, **Anabela** với **emp_id** là **ARD36773F** là một nhân viên nhưng **Anabela** cũng là một người quản lý cho Victoria, Palle, Karla và các nhân viên khác như được trình bày trong hình 9.25.

	emp_id	fname	minit	lname	job_id	job_lvl	pub_id	hire_date	mgr_id
1	PMA42628M	Paolo	M	Accorti	13	35	0877	1992-08-27 00:00:00.000	POK93028M
2	PSA89086M	Pedro	S	Afonso	14	89	1389	1990-12-24 00:00:00.000	POK93028M
3	VPA30890F	Victoria	P	Ashworth	6	140	0877	1990-09-13 00:00:00.000	ARD36773F
4	H-B39728F	Helen		Bennett	12	35	0877	1989-09-21 00:00:00.000	POK93028M
5	L-B31947F	Lesley		Brown	7	120	0877	1991-02-13 00:00:00.000	ARD36773F
6	F-C16315M	Francisco		Chang	4	227	9952	1990-11-03 00:00:00.000	MAS70474F
7	PTC11962M	Philip	T	Cramer	2	215	9952	1989-11-11 00:00:00.000	MAS70474F
8	A-C71970F	Aria		Cruz	10	87	1389	1991-10-26 00:00:00.000	POK93028M
9	AMD15433F	Ann	M	Devon	3	200	9952	1991-07-16 00:00:00.000	MAS70474F
10	ARD36773F	Anabela	R	Doming...	8	100	0877	1993-01-27 00:00:00.000	NULL
11	PHF38899M	Peter	H	Franken	10	75	0877	1992-05-17 00:00:00.000	POK93028M
12	PXH22250M	Paul	X	Henriot	5	159	0877	1993-08-19 00:00:00.000	MAS70474F

Hình 9.25: Employee Table

Để có được danh sách các tên của người quản lý cùng với các chi tiết khác, bạn có thể sử dụng phép tự nối để nối bảng nhân viên với chính nó và sau đó, trích xuất các bản ghi mong muốn. Code Snippet 26 trình bày cách sử dụng phép tự nối.

Code Snippet 26:

```
SELECT TOP 7 A.fname + ' ' + A.lname AS 'Employee Name', B.fname + ' '+B.lname AS
'Manager'
FROM
Employee AS A
INNER JOIN
Employee AS B
ON A.mgr_id = B.emp_id
```

Trong Code Snippet 26, bảng **Employee** được nối vào chính nó dựa trên các cột **mgr_id** và **emp_id**.

Kết quả của đoạn mã này được trình bày trong Hình 9.26.

	Employee Name	Manager
1	Paolo Accorti	Pirkko Koskitalo
2	Pedro Afonso	Pirkko Koskitalo
3	Victoria Ashworth	Anabela Domingues
4	Helen Bennett	Pirkko Koskitalo
5	Lesley Brown	Anabela Domingues
6	Francisco Chang	Margaret Smith
7	Philip Cramer	Margaret Smith

Hình 9.26: Using Self-Join

9.7.4 Câu lệnh MERGE

Câu lệnh MERGE cho phép bạn duy trì một bảng đích dựa trên một số điều kiện phép nối trên bảng nguồn sử dụng một câu lệnh duy nhất. Bây giờ bạn có thể thực hiện những hành động sau trong một câu lệnh MERGE:

- Đưa vào một hàng mới từ nguồn nếu hàng bị thiếu trong bảng đích
- Cập nhật một hàng đích nếu bản ghi đã tồn tại trong bảng nguồn
- Xóa một hàng đích nếu hàng bị thiếu trong bảng nguồn

Ví dụ, giả sử bạn có bảng **Products** duy trì các bản ghi của tất cả các sản phẩm. Bảng **NewProducts**duy trì các bản ghi các sản phẩm mới. Bạn muốn cập nhật bảng **Products** với các bản ghi từ bảng **NewProducts**. Ở đây, bảng **NewProducts** là bảng nguồn và **Products** là bảng đích. Bảng **Products** chứa các bản ghi của các sản phẩm hiện có với các dữ liệu đã cập nhật và các sản phẩm mới. Hình 9.27 trình bày hai bảng này.

Products

	ProductID	Name	Type	PurchaseDate
1	101	Rivets	Hardware	2012-12-01
2	102	Nuts	Hardware	2012-12-01
3	103	Washers	Hardware	2011-01-01
4	104	Rings	Hardware	2013-01-15
5	105	Paper Clips	Stationery	2013-01-01

NewProducts

	ProductID	Name	Type	PurchaseDate
1	102	Nuts	Hardware	2012-12-01
2	103	Washers	Hardware	2011-01-01
3	107	Rings	Hardware	2013-01-15
4	108	Paper Clips	Stationery	2013-01-01

Hình 9.27: Products and NewProducts Tables

Concepts

Xem xét việc bạn muốn :

- So sánh tên họ và tên gọi của khách hàng từ cả bảng nguồn và bảng đích
- Cập nhật thông tin khách hàng trong bảng đích nếu tên họ và tên gọi khớp nhau
- Đưa các bản ghi mới vào bảng đích nếu tên họ và tên gọi trong bảng nguồn không tồn tại trong bảng đích
- Xóa các bản ghi hiện hữu trong bảng đích nếu tên họ và tên gọi không khớp với những cái của bảng nguồn

Câu lệnh MERGE hoàn thành những tác vụ trong một câu lệnh duy nhất. MERGE còn cho phép bạn hiển thị một cách tùy chọn những bản ghi đã được đưa vào, cập nhật hay xóa bằng cách sử dụng mệnh đề OUTPUT. Sau đây là cú pháp của câu lệnh MERGE.

Cú pháp:

```
MERGE target_table
USING source_table
ON match_condition
WHEN MATCHED THEN UPDATE SET Col1 = val1 [, Col2 = val2...]
WHEN [TARGET] NOT MATCHED THEN INSERT (Col1 [,Col2...] VALUES (Val1 [,
Val2...])
WHEN NOT MATCHED BY SOURCE THEN DELETE
[OUTPUT $action, Inserted.Col1, Deleted.Col1,...];
```

Concepts

trong đó,

target_table : là bảng WHERE những thay đổi đang được thực hiện.

source_table : là bảng mà từ đó các hàng sẽ được chèn, cập nhật hay xóa vào bảng đích.

match_conditions : là những điều kiện JOIN và bất kỳ toán tử so sánh nào khác.

MATCHED : true nếu một hàng trong target_table và source_table phù hợp với match_condition.

NOT MATCHED : true nếu một hàng từ source_table không tồn tại trong target_table.

SOURCE NOT MATCHED : true nếu một hàng tồn tại trong target_table nhưng không phải trong source_table.

OUTPUT : Mệnh đề tùy chọn cho phép xem những bản ghi đã được chèn/xóa/cập nhật trong target_table.

Các câu lệnh MERGE được kết thúc bằng dấu chấm phẩy (;).
Code Snippet trình bày cách sử dụng câu lệnh MERGE. Nó sử dụng cơ sở dữ liệu **Sterling**.

Code Snippet 27:

```
MERGE INTO Products AS P1
USING
NewProducts AS P2
ON P1.ProductId = P2.ProductId
WHEN MATCHED THEN
     UPDATE SET
          P1.Name = P2.Name,
          P1.Type = P2.Type,
          P1.PurchaseDate = P2.PurchaseDate
WHEN NOT MATCHED THEN
      INSERT (ProductId, Name, Type, PurchaseDate)
       VALUES (P2.ProductId, P2.Name, P2.Type, P2.PurchaseDate)
WHEN NOT MATCHED BY SOURCE THEN
      DELETE

OUTPUT $action, Inserted.ProductId, Inserted.Name, Inserted.Type, Inserted.
PurchaseDate, Deleted.ProductId, Deleted.Name, Deleted.Type, Deleted.
PurchaseDate;
```

Concepts

Hình 9.28 trình bày kết quả.

	$action	ProductId	Name	Type	PurchaseDate	ProductId	Name	Type	PurchaseDate
1	INSERT	107	Rings	Hardware	2013-01-15	NULL	NULL	NULL	NULL
2	INSERT	108	Paper Clips	Stationery	2013-01-01	NULL	NULL	NULL	NULL
3	DELETE	NULL	NULL	NULL	NULL	101	Rivets	Hardware	2012-12-01
4	UPDATE	102	Nuts	Hardware	2012-12-01	102	Nuts	Hardware	2012-12-01
5	UPDATE	103	Washers	Hardware	2011-01-01	103	Washers	Hardware	2011-01-01
6	DELETE	NULL	NULL	NULL	NULL	104	Rings	Hardware	2013-01-15
7	DELETE	NULL	NULL	NULL	NULL	105	Paper Clips	Stationery	2013-01-01

Hình 9.28: Using MERGE

Bảng **NewProducts** là bảng nguồn và bảng **Products** là bảng đích. Điều kiện so khớp là cột **ProductId** của cả hai bảng. Nếu điều kiện so khớp đánh giá thành false (NOT MATCHED), khi đó các bản ghi mới được chèn vào bảng đích.

Nếu điều kiện so khớp đánh giá thành true (MATCHED), khi đó các bản ghi được cập nhật vào bảng đích từ bảng nguồn.

Nếu các bản ghi hiện diện trong bảng đích không khớp với những cái trong bảng nguồn (NOT MATCHED BY SOURCE), khi đó chúng bị xóa khỏi bảng đích. Câu lệnh cuối cùng sẽ hiển thị một báo cáo bao gồm các hàng đã được chèn vào/cập nhật/xóa như được trình bày trong kết quả.

9.8 Các biểu thức bảng chung (CTE)

Biểu thức bảng chung (CTE) tương tự như tập kết quả tạm thời được định nghĩa trong phạm vi thực hiện của câu lệnh SELECT, INSERT, UPDATE, DELETE, hoặc CREATE VIEW đơn lẻ. CTE là một biểu thức có tên được định nghĩa trong một truy vấn. CTE được định nghĩa ở đoạn đầu truy vấn và có thể được tham chiếu nhiều lần trong truy vấn bên ngoài. CTE bao gồm các tham chiếu tới chính nó được gọi là CTE đệ quy.

Ghi chú - Biểu thức bảng chung (CTE) lần đầu tiên được giới thiệu trong SQL Server 2005.

Lợi thế chính của CTE là cải thiện khả năng đọc và dễ dàng trong việc bảo trì các truy vấn phức tạp.

Sau đây là cú pháp để tạo ra CTE.

Cú pháp :

```
WITH <CTE_name>
AS ( <CTE_definition> )
```

Concepts

Ví dụ, để lấy và hiển thị số lượng khách hàng theo năm cho các đơn hàng hiện diện trong bảng Sales.SalesOrderHeader, đoạn mã này sẽ như được ra trong Code Snippet 28.

Code Snippet 28:

```
WITH CTE_OrderYear
AS
(
SELECT YEAR(OrderDate) AS OrderYear, CustomerID
FROM Sales.SalesOrderHeader
)
SELECT OrderYear, COUNT(DISTINCT CustomerID) AS CustomerCount
FROM CTE_OrderYear
GROUP BY OrderYear;
```

Ở đây, **CTE_OrderYear** được chỉ ra làm tên CTE. Các từ khóa WITH...AS bắt đầu định nghĩa CTE. Sau đó, CTE được sử dụng trong câu lệnh SELECT để lấy và hiển thị các kết quả mong muốn.

Hình 9.29 trình bày kết quả.

	OrderYear	CustomerCount
1	2007	9864
2	2008	11844
3	2005	1216
4	2006	3094

Hình 9.29: Output of CTE

Các hướng dẫn sau đây cần phải được ghi nhớ trong khi định nghĩa các CTE :

- CTE được giới hạn trong phạm vi đến việc thực hiện truy vấn bên ngoài. Do đó, khi truy vấn bên ngoài kết thúc, tuổi thọ của CTE sẽ kết thúc.
- Bạn cần phải định nghĩa tên cho CTE và ngoài ra, định nghĩa tên duy nhất cho mỗi cột đã tham chiếu trong mệnh đề SELECT của CTE.
- Có thể sử dụng các bí danh nội tuyến hoặc bên ngoài cho các cột trong CTE.
- Một CTE đơn lẻ có thể được tham chiếu nhiều lần trong cùng một truy vấn với một định nghĩa.

Concepts

Nhiều CTE cũng có thể được định nghĩa trong cùng một mệnh đề WITH. Ví dụ, hãy xem xét Code Snippet 29. Nó định nghĩa hai CTE sử dụng mệnh đề WITH duy nhất. Đoạn mã này giả định rằng ba bảng có tên là Student, City, và Status được tạo ra.

Code Snippet 29:

```
WITH CTE_Students

AS
(
 Select StudentCode, S.Name,C.CityName, St.Status
         FROM Student S
         INNER JOIN City C
         ON S.CityCode = C.CityCode
INNER JOIN Status St
         ON S.StatusId = St.StatusId)
,
StatusRecord -- This is the second CTE being defined
AS
(
  SELECT Status, COUNT(Name) AS CountofStudents
  FROM CTE_Students
  GROUP BY  Status
  )
SELECT * FROM StatusRecord
```

Giả sử một số bản ghi được chèn vào trong tất cả ba bảng, kết quả có thể như được trình bày trong Hình 9.30.

	Status	CountofStudents
1	Failed	2
2	Passed	2

Hình 9.30: Using Multiple CTE with Single WITH

9.9 Kết hợp dữ liệu bằng cách sử dụng các toán tử SET

SQL Server 2012 cung cấp một số từ khóa nhất định, còn được gọi là các toán tử, để kết hợp dữ liệu từ nhiều bảng. Những toán tử này như sau :

- **UNION**
- **INTERSECT**
- **EXCEPT**

Concepts

9.9.1 Toán tử UNION

Kết quả từ hai câu lệnh truy vấn khác nhau có thể được kết hợp thành một tập kết quả duy nhất sử dụng toán tử UNION. Các câu lệnh truy vấn phải có các loại cột tương thích và số các cột bằng nhau. Tên cột có thể khác nhau trong từng câu lệnh nhưng các loại dữ liệu phải là loại tương thích. Bằng các loại dữ liệu tương thích, nó có nghĩa là sẽ có thể chuyển đổi nội dụng của một trong các cột vào trong cột khác. Ví dụ, nếu một trong những câu lệnh truy vấn có kiểu dữ liệu int và câu lệnh truy vấn kia có kiểu dữ liệu tiền, chúng tương thích và phép hợp có thể xảy ra giữa chúng, vì dữ liệu int có thể được chuyển đổi thành dữ liệu tiền.

Sau đây là cú pháp của toán tử UNION.

Syntax:

```
Query_Statement1
UNION [ALL]
Query_Statement2
```

trong đó,
Query_Statement1 và Query_Statement2 là các câu lệnh SELECT.

Code Snippet 30 trình bày toán tử UNION.

Code Snippet 30:

```
SELECT Product.ProductId FROM Production.Product
UNION
SELECT ProductId FROM Sales.SalesOrderDetail
```

Điều ày sẽ liệt kê tất cả các id sản phẩm của cả hai bảng khớp với nhau. Nếu bạn đưa vào mệnh đề ALL, tất cả các hàng được đưa vào trong tập kết quả bao gồm cả các bản ghi trùng lặp.

Code Snippet 31 trình bày toán tử UNION ALL operator.

Code Snippet 31:

```
SELECT Product.ProductId FROM Production.Product
UNION ALL
SELECT ProductId FROM Sales.SalesOrderDetail
```

Theo mặc định, toán tử UNION loại bỏ các bản ghi trùng lặp từ tập kết quả. Tuy nhiên, nếu bạn sử dụng mệnh đề ALL với toán tử UNION, sau đó tất cả các hàng được trả về. Ngoài UNION, những toán tử khác được sử dụng để kết hợp dữ liệu từ nhiều bảng là INTERSECT và EXCEPT.

9.9.2 Toán tử INTERSECT

Xem xét lại hai bảng Product và SalesOrderDetail hiện diện trong AdventureWorks2012.

Giả sử bạn muốn chỉ hiển thị những hàng là chung cho cả hai bảng. Để làm điều này, bạn sẽ cần phải sử dụng toán tử có tên là INTERSECT. Toán tử INTERSECT được sử dụng với hai câu lệnh truy vấn để trả về một tập riêng biệt các hàng chung cho cả hai câu lệnh truy vấn. Sau đây là cú pháp của toán tử.

Cú pháp:

```
Query_statement1
  INTERSECT
Query_statement2
```

trong đó,

Query_Statement1 và Query_Statement2 là các câu lệnh SELECT.

Code Snippet 32 trình bày toán tử INTERSECT.

Code Snippet 32:

```
SELECT Product.ProductId FROM Production.Product
INTERSECT
SELECT ProductId FROM Sales.SalesOrderDetail
```

Những quy tắc cơ bản cho việc sử dụng INTERSECT như sau:

- Số các cột và thứ tự theo đó chúng được đưa ra phải giống nhau trong cả hai truy vấn.
- The data types of the columns being used must be compatible.

Kết quả của giao điểm của các bảng Production.Product and Sales.SalesOrderDetail sẽ chỉ có những id sản phẩm có các bản ghi so khớp trong bảng Production.Product. Ở các doanh nghiệp lớn, có số lượng lớn dữ liệu được lưu trữ trong các cơ sở dữ liệu. Thay bằng lưu toàn bộ dữ liệu trong một bảng đơn, nó có thể được trải ra trên một số bảng mà có liên quan lẫn nhau. Khi dữ liệu được lưu trong các bảng như vậy, phải có một số phương tiện để kết hợp và gọi ra dữ liệu từ các bảng này. Việc sử dụng SQL Server, có một số cách để kết hợp dữ liệu từ nhiều bảng. Các phần sau đây khám phá các cách này một cách chi tiết.

9.9.3 Toán tử EXCEPT

Toán tử EXCEPT trả về tất cả các hàng riêng biệt từ truy vấn đã cho ở bên trái của toán tử EXCEPT và loại bỏ tất cả các hàng khỏi tập kết quả so khớp các hàng ở bên phải của toán tử EXCEPT.

Concepts

Sau đây là cú pháp của toán tử EXCEPT.

Cú pháp:

```
Query_statement1
 EXCEPT
Query_statement2
```

trong đó,

Query_Statement1 và Query_Statement2 là các câu lệnh SELECT.

Hai quy tắc áp dụng cho toán tử INTERSECT còn được áp dụng cho toán tử EXCEPT.

Code Snippet 33 trình bày toán tử EXCEPT.

Code Snippet 33:

```
SELECT Product.ProductId FROM Production.Product
EXCEPT
SELECT ProductId FROM Sales.SalesOrderDetail
```

Nếu thứ tự của hai bảng trong ví dụ này được hoán đổi, chỉ có những hàng được trả về từ bảng Production.Product mà không khớp với các hàng hiện diện trong Sales.SalesOrderDetail.

Như vậy, trong các điều kiện đơn giản, toán tử EXCEPT chọn tất cả các bản ghi từ bảng đầu tiên ngoại trừ những bản ghi khớp với bảng thứ hai. Do đó, khi bạn đang sử dụng toán tử EXCEPT, thứ tự của hai bảng trong những truy vấn này rất quan trọng. Trong khi đó, với toán tử INTERSECT, không quan trọng bảng nào được chỉ ra đầu tiên.

9.10 Xoay vòng và nhóm các phép tính tập hợp

Xem xét kịch bản khi dữ liệu cần phải được hiển thị theo một hướng khác so với nó được lưu trữ, theo quan điểm bố trí hàng và cột. Quá trình chuyển đổi dữ liệu từ một hướng dựa trên hàng thành một hướng dựa trên cột được gọi là xoay vòng. Các toán tử PIVOT và UNPIVOT của SQL Server giúp thay đổi hướng của dữ liệu từ hướng cột sang hướng hàng và ngược lại. Điều này được thực hiện bằng cách hợp nhất các giá trị hiện diện trong một cột vào danh sách các giá trị riêng biệt và sau đó trình chiếu danh sách đó ở dạng đầu đề cột.

Concepts

9.10.1 Toán tử PIVOT

Sau đây là cú pháp ngắn gọn cho PIVOT.

Cú pháp:

```
SELECT <non-pivoted column>,
    [first pivoted column] AS <column name>,
    [second pivoted column] AS <column name>,
    ...
    [last pivoted column] AS <column name>
FROM
    (<SELECT query that produces the data>)
    AS <alias for the source query>
PIVOT
(
    <aggregation function>(<column being aggregated>)
FOR
[<column that contains the values that will become column headers>]
    IN ( [first pivoted column], [second pivoted column],
    ... [last pivoted column])
) AS <alias for the pivot table>
<optional ORDER BY clause>;
```

trong đó,

table_source: là một bảng hoặc biểu thức bảng.

aggregate_function: là một hàm tổng hợp do người dùng định nghĩa hoặc dựng sẵn nhận một hoặc nhiều giá trị đầu vào.

value_column: là cột giá trị của toán tử PIVOT.

pivot_column: là cột xoay vòng của toán tử PIVOT. Cột này phải thuộc một loại có thể ngụ ý hay rõ ràng được chuyển đổi sang nvarchar().

IN (column_list): là các giá trị trong pivot_column sẽ trở thành tên cột của bảng đầu ra. Danh sách không phải bao gồm bất kỳ tên cột nào đã tồn tại trong table_source đầu vào đang được xoay vòng.

table_alias: là tên bí danh của bảng đầu ra.

Concepts

Đầu ra của phần này sẽ là một bảng chứa tất cả các cột của table_source ngoại trừ pivot_column và value_column. Những cột này của table_source, ngoại trừ pivot_column và value_column, được gọi là các cột gộp nhóm của toán tử xoay vòng.

Nói một cách đơn giản, để sử dụng toán tử PIVOT, bạn cần phải cung cấp ba phần tử cho toán tử này:

- **Gộp nhóm:** Trong mệnh đề FROM, những cột đầu vào phải được cung cấp. Toán tử PIVOT sử dụng những cột đó để xác định cột nào để sử dụng cho gộp nhóm dữ liệu cho tập hợp.
- **Lan rộng**: Ở đây, danh sách tách biệt bằng dấu phẩy của các giá trị xảy ra trong dữ liệu nguồn được cung cấp sẽ được sử dụng làm các đầu đề cột cho dữ liệu xoay vòng.
- **Tổng hợp**: Hàm tổng hợp, như là SUM, để được thực hiện trên những hàng đã phân nhóm.

Xem xét một ví dụ để hiểu toán tử PIVOT. Đoạn mã 34 được trình bày không có toán tử PIVOT và chứng minh sự tổng hợp GROUP BY đơn giản. Bởi số lượng bản ghi sẽ là rất lớn, tập kết quả được giới hạn tới 5 bằng cách chỉ ra TOP 5.

Code Snippet 34:

```
SELECT TOP 5 SUM(SalesYTD) AS TotalSalesYTD, Name
FROM Sales.SalesTerritory
GROUP BY Name
```

Hình 9.31 trình bày kết quả.

	TotalSalesYTD	Name
1	7887186.7882	Northwest
2	2402176.8476	Northeast
3	3072175.118	Central
4	10510853.8739	Southwest
5	2538667.2515	Southeast

Hình 9.31: Grouping without PIVOT

Top 5 doanh số năm cho đến nay cùng với tên lãnh thổ nhóm theo tên lãnh thổ được hiển thị.
Bây giờ, cùng một truy vấn được viết lại trong Code Snippet 35 sử dụng PIVOT để dữ liệu được chuyển từ một hướng dựa trên hàng cho hướng dựa trên cột.

Code Snippet 35:

```
-- Pivot table with one row and six columns
SELECT TOP 5 'TotalSalesYTD' AS GrandTotal,
[Northwest], [Northeast], [Central], [Southwest], [Southeast]
```

Concepts

```
FROM
(SELECT TOP 5 Name, SalesYTD
 FROM Sales.SalesTerritory
) AS SourceTable
PIVOT
(
SUM(SalesYTD)
FOR Name IN ([Northwest], [Northeast], [Central], [Southwest], [Southeast])
) AS PivotTable;
```

Hình 9.32 trình bầy kết quả.

	GrandTotal	Northwest	Northeast	Central	Southwest	Southeast
1	TotalSalesYTD	7887186.7882	2402176.8476	3072175.118	10510853.8739	2538667.2515

Hình 9.32: Grouping with PIVOT

Như được trình bày trong hình 9.32, dữ liệu được chuyển đổi và tên lãnh thổ bây giờ được xem là các cột thay vì các hàng. Điều này cải thiện khả năng đọc. Một thách thức lớn trong việc viết các truy vấn sử dụng PIVOT là sự cần thiết để cung cấp một danh sách cố định của các phần tử lan rộng cho toán tử PIVOT, chẳng hạn như tên lãnh thổ cụ thể được đưa ra trong Đoạn mã 35. Nó sẽ không có tính khả thi hoặc thực tế để thực hiện điều này cho số lượng lớn các phần tử lây lan. Để khắc phục điều này, các nhà phát triển có thể sử dụng SQL động. SQL động cung cấp phương tiện để xây dựng một chuỗi ký tự được truyền cho SQL Server, được phiên dịch là một lệnh, và sau đó, thực hiện.

9.10.2 Toán tử UNPIVOT

UNPIVOT thực hiện hầu hết hoạt động ngược lại của PIVOT, bằng cách xoay cột vào hàng. Bỏ xoay vòng không khôi phục lại dữ liệu gốc. Dữ liệu mức chi tiết đã bị mất trong quá trình xử lý tổng hợp trong xoay vòng gốc. UNPIVOT không có khả năng phân bổ các giá trị để trả về các giá trị chi tiết ban đầu. Thay vì chuyển các hàng thành cột, bỏ xoay vòng dẫn đến các cột được chuyển đổi thành các hàng. SQL Server cung cấp toán tử bảng UNPIVOT để trả về hiển thị dạng bảng hướng hàng từ dữ liệu đã xoay vòng.

Khi bỏ xoay vòng dữ liệu, một hoặc nhiều cột được định nghĩa là nguồn để được chuyển đổi thành hàng. Dữ liệu trong những cột đó được kéo dài ra, hoặc phân chia, thành một hoặc nhiều hàng mới, tùy thuộc vào bao nhiêu cột đang được bỏ xoay vòng.

Để sử dụng toán tử UNPIVOT, bạn cần phải cung cấp ba yếu tố như sau:

- Các cột nguồn để được bỏ xoay vòng
- Tên cho cột mới sẽ hiển thị những giá trị đã bỏ xoay vòng
- Tên cho cột sẽ hiển thị tên của các những trị đã bỏ xoay vòng

Concepts

Xem xét kịch bản trước đây. Code Snippet 36 trình bày đoạn mã để chuyển đổi bảng xoay vòng tạm thời thành bảng lâu dài để cùng một bảng có thể được sử dụng để trình bày các phép tính UNPIVOT.

Code Snippet 36:

```
-- Pivot table with one row and six columns
SELECT TOP 5 'TotalSalesYTD' AS GrandTotal,
[Northwest], [Northeast], [Central], [Southwest], [Southeast]
FROM
(SELECT TOP 5 Name, SalesYTD
 FROM Sales.SalesTerritory
) AS SourceTable
PIVOT
(
SUM(SalesYTD)
FOR Name IN ([Northwest], [Northeast], Central], [Southwest], [Southeast])
) AS PivotTable;
```

Code Snippet 37 trình bày toán tử UNPIVOT.

Code Snippet 37:

```
SELECT Name, SalesYTD FROM
 (SELECT GrandTotal, Northwest, Northeast, Central, Southwest, Southeast FROM
TotalTable) P
UNPIVOT
 (SalesYTD FOR Name IN
  (Northwest, Northeast, Central, Southwest, Southeast)
)AS unpvt;
```

Kết quả sẽ giống như trong Hình 9.32.

9.10.3 GROUPING SETS

Toán tử GROUPING SETS hỗ trợ tổng hợp của nhiều gộp nhóm cột và tổng cộng tùy chọn. Xem xét trường hợp bạn cần một báo cáo có nhóm nhiều cột của một bảng. Hơn nữa, bạn muốn tập hợp các cột lại. Trong các phiên bản trước đây của SQL Server, bạn phải viết một số mệnh đề GROUP BY riêng biệt tiếp theo là các mệnh đề UNION để đạt được điều này. Đầu tiên được giới thiệu trong SQL Server 2008, toán tử GROUPING
SETS cho phép bạn nhóm lại với nhau nhiều gộp nhóm các cột tiếp theo là hàng tổng số tùy chọn, ký hiệu bằng dấu ngoặc đơn (). Sẽ hiệu quả hơn khi sử dụng các toán tử GROUPING SETS thay vì nhiều GROUP BY với các mệnh đề UNION bởi vì cái sau thêm các chi phí xử lý lên máy chủ cơ sở dữ liệu.

Concepts

Sau đây là cú pháp của toán tử GROUPING SETS.

Cú pháp:

```
GROUP BY
GROUPING SETS ( <grouping set list> )
```

trong đó,

danh sách tập hợp gộp nhóm: bao gồm một hoặc nhiều cột, cách nhau bằng dấu phẩy. Cặp dấu ngoặc đơn (), không có bất kỳ tên cột nào biểu thị tổng cộng.

Code Snippet 38 trình bày toán tử GROUPING SETS. Giả định rằng một bảng **Students** được tạo ra với các trường có tên tương ứng là **Id**, **Name**, và **Marks**.

Code Snippet 38:

```
SELECT Id, Name, AVG(Marks) Marks
FROM Students
GROUP BY
GROUPING SETS
(
(Id, Name, Marks),
(Id),
()
)
```

Hình 9.33 trình bày kết quả của đoạn mã.

	Id	Name	Marks
1	91	Sasha Goldsmith	78
2	91	NULL	78
3	92	Karen Hues	55
4	92	NULL	55
5	93	William Pinter	67
6	93	NULL	67
7	94	Yuri Gogol	89
8	94	NULL	89
9	NULL	NULL	72

Hình 9.33:GROUPING SETS

Ở đây, đoạn mã này sử dụng GROUPING SETS để hiển thị điểm trung bình cho mỗi học sinh. Các giá trị NULL trong **Name** chỉ ra điểm trung bình cho mỗi học sinh. Giá trị NULL trong cả cột **Id** và **Name** chỉ ra tổng cộng.

9.11 Kiểm tra tiến độ của bạn

1. Câu lệnh nào sau đây có thể được sử dụng với các truy vấn con để trả về một cột và nhiều hàng?

(A)	ANY	(C)	IN
(B)	ALL	(D)	=

2. Toán tử_____được sử dụng để chỉ hiển thị những hàng là chung cho cả hai bảng.

(A)	INTERSECT	(C)	UNION
(B)	EXCEPT	(D)	UNION WITH ALL

3. Truy vấn con bao gồm truy vấn con khác bên trong nó được gọi là_____

(A)	join	(C)	correlated subquery
(B)	nested subquery	(D)	parent subquery

4. _____ được hình thành khi các bản ghi từ hai bảng được kết hợp chỉ khi các hàng của cả hai bảng được so khớp dựa trên một cột chung.

(A)	Inner join	(C)	Self-join
(B)	Left Outer join	(D)	Right Outer join

5. _____ trả về tất cả các hàng từ ít nhất một trong các bảng trong mệnh đề FROM của câu lệnh SELECT, miễn là những hàng đó đáp ứng bất kỳ điều kiện WHERE hoặc HAVING nào của câu lệnh SELECT.

(A)	Inner join	(C)	Self-join
(B)	Outer join	(D)	Sub queries

Concepts

6. Xem xét bạn có hai bảng là **Products** và **Orders** đã được tập kết. Dựa trên các đơn đặt hàng mới, bạn muốn cập nhật **Quantity** trong bảng **Products**. Bạn viết đoạn mã sau đây:

```
MERGE Products AS T

USING Orders AS S

ON   S.ProductID   =   T.Product ID

_____

_____

_____
```

Đoạn mã nào sau đây, khi đưa vào khoảng trống, sẽ cho phép bạn đạt được điều này?

(A)	**WHEN MATCHED THEN UPDATE SET T .Quantity = S .Quantity**	(C)	**WHEN NOT MATCHED THEN UPDATE SET T . Quantity = S . Quantity**
(B)	**WHEN MATCHED THEN UPDATE SET Quantity = Quantity**	(D)	**WHEN MATCHED THEN UPDATE SET S .Quantity = T . Quantity**

7. Điều nào sau đây sẽ là kết quả của đoạn mã đã cho?

```
SELECT ProdId, Year,
SUM(Purchase) AS TotalPurchase
FROM Products
GROUP BY GROUPING SETS((ProdId),())
```

(A)	Lỗi cú pháp	(C)	Sẽ hiển thị tổng dữ liệu đặt mua theo năm ngoại trừ số tổng cộng
(B)	Sẽ thực thi nhưng sẽ không tạo ra kết quả nào cả	(D)	Sẽ hiển thị tổng dữ liệu mua theo năm với số tổng cộng

9.11.1 Đáp án

1.	C
2.	B
3.	B
4.	B
5.	B
6.	A
7.	A

Tóm tắt

- Mệnh đề GROUP BY và các hàm tổng hợp đã cho phép để gộp nhóm và/hoặc tổng hợp dữ liệu với nhau để trình bày thông tin tóm tắt.
- Các hàm tổng hợp không gian mới được giới thiệu trong SQL Server 2012.
- Truy vấn con cho phép tập kết quả của một câu lệnh SELECT được sử dụng làm tiêu chí cho câu lệnh SELECT khác.
- Phép nối giúp bạn kết hợp dữ liệu cột từ hai cột hoặc nhiều bảng dựa trên mối quan hệ logic giữa những bảng này.
- Các toán tử tập hợp như UNION và INTERSECT giúp bạn kết hợp dữ liệu hàng từ hai hoặc nhiều bảng.
- Các toán tử PIVOT và UNPIVOT giúp thay đổi hướng của dữ liệu từ hướng cột sang hướng hàng và ngược lại.
- Mệnh đề con GROUPING SET của mệnh đề GROUP BY giúp chỉ ra nhiều nhóm trong một truy vấn đơn lẻ.

Try It Yourself

1. Viết một truy vấn để hiển thị tên nhân viên và các phòng ban của họ từ cơ sở dữ liệu AdventureWorks2012..

2. Sử dụng các bảng Sales.SalesPerson and Sales.SalesTerritory, lấy các ID của tất cả những người bán hàng hoạt động ở Canada.

3. Sử dụng các bảng Sales.SalesPerson và Sales.SalesTerritory, lấy các ID của tất cả những người bán hàng hoạt động ở Northwest hoặc Northeast.

4. So sánh các giá trị tiền thưởng của nhân viên bán hàng trong bảng Sales.SalesPerson để tìm ra những người bán hàng kiếm tiền thưởng nhiều hơn. Hiển thị SalesPersonID và các giá trị tiền thưởng theo thứ tự giảm dần. (Gợi ý: Sử dụng phép tự nối và ORDER BY ...DESC).

5. Lấy tất cả các giá trị của SalesPersonID từ bảng Sales.SalesPerson, nhưng bỏ qua những giá trị đó, có hiện diện trong bảng Sales.Store. (Gợi ý: Sử dụng toán tử EXCEPT).

6. Kết hợp tất cả các SalesPersonIDs của các bảng Sales.SalesPerson và Sales.Store.

7. Lấy tất cả các ID người bán hàng và ID lãnh thổ từ bảng Sales.SalesPerson bất kể là họ có các bản ghi khớp với các bản ghi trong bảng Sales.SalesTerritory hay không. (Gợi ý: Sử dụng phép nối ngoài bên trái)..

8. Lấy một tập riêng biệt các ID lãnh thổ hiện diện trong cả hai bảng Sales.SalesPerson và Sales.SalesTerritory. (Gợi ý: Sử dụng toán tử INTERSECT).