



Dữ liệu thông minh Quản lý với SQL

Máy chủ

Phiên: 9

Truy vấn nâng cao và Tham gia

Mục tiêu

- Giải thích về nhóm và tổng hợp dữ liệu • Mô tả các truy vấn phụ • Mô tả các biểu thức bảng • Giải thích về phép nối
- Mô tả các loại phép nối khác nhau
- Giải thích về việc sử dụng các toán tử tập hợp khác nhau để kết hợp dữ liệu • Mô tả các hoạt động xoay vòng và nhóm tập hợp

Giới thiệu

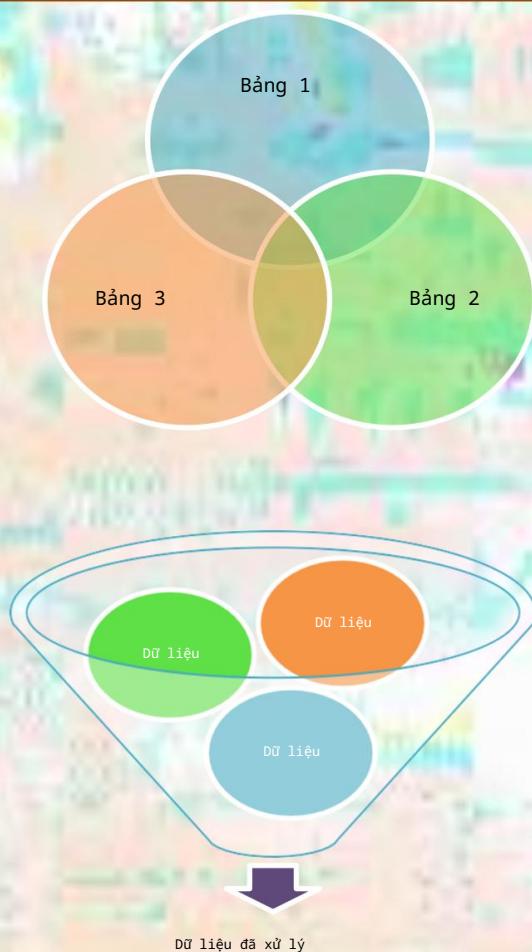
SQL Server 2019 bao gồm một số tính năng truy vấn mạnh mẽ giúp truy xuất dữ liệu hiệu quả và nhanh chóng.

Dữ liệu có thể được nhóm và/hoặc tổng hợp lại với nhau để trình bày thông tin tóm tắt.

Joins giúp kết hợp dữ liệu cột từ hai hoặc nhiều bảng dựa trên mối quan hệ logic giữa các bảng. Các toán tử tập hợp như UNION và INTERSECT kết hợp dữ liệu hàng từ hai hoặc nhiều bảng.

Các toán tử IVOT và UNPIVOT được sử dụng để chuyển đổi hướng dữ liệu từ hướng cột sang hướng hàng và ngược lại.

Mệnh đề con GROUPING SET của mệnh đề GROUP BY giúp chỉ định nhiều nhóm trong một truy vấn duy nhất.



Nhóm dữ liệu 1-5

Mệnh đề này phân vùng tập kết quả thành một hoặc nhiều tập hợp con và mỗi tập hợp con có các giá trị và biểu thức chung.

Theo sau là danh sách các cột, được gọi là các cột được nhóm.
Nó hạn chế số lượng hàng của tập kết quả.

	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

Sử dụng mệnh đề GROUPBY

Nhóm dữ liệu 2-5

NHÓM THEO VỊ TRÍ

Để hạn chế các hàng để nhóm.

Các hàng thỏa mãn điều kiện tìm kiếm được xem xét cho nhóm.

Các hàng không đáp ứng các điều kiện trong mệnh đề WHERE sẽ bị loại bỏ trước khi thực hiện bất kỳ nhóm nào.

	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

Đầu ra của GROUP BY with WHERE

Nhóm dữ liệu 3-5

NHÓM THEO VỚI NULL

Nếu cột nhóm chứa giá trị NULL, hàng đó sẽ trở thành một nhóm riêng biệt trong tập kết quả.

Nếu cột nhóm chứa nhiều hơn một giá trị NULL, các giá trị NULL sẽ được đưa vào một hàng duy nhất.

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

Đầu ra của GROUP BY WITHNULL

Nhóm dữ liệu 4-5

NHÓM THEO VỚI TẤT CẢ

Từ khóa ALL cũng có thể được sử dụng với mệnh đề GROUP BY.

Nó bao gồm tất cả các nhóm mà mệnh đề GROUP BY tạo ra và thậm chí cả những nhóm không đáp ứng các điều kiện tìm kiếm.

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

Đầu ra của GROUP BY with ALL

Nhóm dữ liệu 5-5

NHÓM THEO VỚI CÓ

Mệnh đề HAVING chỉ được sử dụng với câu lệnh SELECT để chỉ định một điều kiện tìm kiếm cho một nhóm. Mệnh đề HAVING hoạt động như một mệnh đề WHERE ở những nơi mà mệnh đề WHERE không thể được sử dụng với các hàm tổng hợp như SUM().

Cú pháp:

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

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

Tóm tắt dữ liệu 1-4

Mệnh đề GROUP BY cũng sử dụng các toán tử như CUBE và ROLLUP để trả về dữ liệu được tóm tắt.

Số cột trong mệnh đề GROUP BY xác định số hàng tóm tắt trong tập kết quả.

Tóm tắt dữ liệu 2-4

KHÔI:

CUBE là hàng MỘT tổng hợp người điều hành cái đó sản xuất
siêu tổng hợp.

Ngoài các hàng thông thường được cung cấp bởi GROUP BY, nó cũng cung cấp tóm tắt các hàng mà mệnh đề GROUP BY tạo ra.

Cú pháp:

CHỌN <tên_cột>. TỪ <tên_bảng> NHÓM THEO
<tên_cột> VỚI CUBE

Ví dụ

```
select student, [subject],
       AVG(mark) as BQ, COUNT(*) as [so lan thi]
    from tbMark
   group by student, [subject] with cube
go
```

Sử dụng GROUP BY với CUBE

	student	subject	BQ	so lan thi
1	SV01	1	46	2
2	SV02	1	90	1
3	SV04	1	60	1
4	NULL	1	60	4
5	SV01	2	55	2
6	SV02	2	40	1
7	NULL	2	50	3
8	SV01	3	35	2
9	SV02	3	25	2
10	SV03	3	60	1
11	NULL	3	36	5
12	SV01	4	55	2
13	SV03	4	1...	1
14	SV04	4	50	1
15	NULL	4	65	4
16	NULL	NULL	52	16
17	SV01	NULL	47	8
18	SV02	NULL	45	4
19	SV03	NULL	80	2
20	SV04	NULL	55	2

Tóm tắt dữ liệu 3-4

CUỘN LÊN:

Nó giới thiệu các hàng tóm tắt vào tập kết quả.

Tạo ra một tập kết quả hiển thị các nhóm được sắp xếp theo thứ tự phân cấp.

Sắp xếp các nhóm từ thấp đến cao.

ROLLUP giả định có một hệ thống phân cấp giữa các cột chiều và chỉ tạo các tập hợp nhóm dựa trên hệ thống phân cấp này.

ROLLUP thường được sử dụng để tạo tổng phụ và tổng cho mục đích báo cáo.

ROLLUP thường được sử dụng để tính tổng hợp của dữ liệu phân cấp như doanh số theo năm quý tháng.

Tóm tắt dữ liệu 4-4

Cú pháp:

CHỌN <tên_cột> . TỪ <tên_bảng> NHÓM THEO
<tên_cột> VỚI ROLLUP

Ví dụ

```
select student, [subject], AVG(mark) as BQ, COUNT(*) as [so lan thi]
  from tbMark
 group by student, [subject] with rollup
 go
```

	student	subject	BQ	so lan thi
1	SV01	1	46	2
2	SV01	2	55	2
3	SV01	3	35	2
4	SV01	4	55	2
5	SV01	NULL	47	8
6	SV02	1	90	1
7	SV02	2	40	1
8	SV02	3	25	2
9	SV02	NULL	45	4
10	SV03	3	60	1
11	SV03	4	1...	1
12	SV03	NULL	80	2
13	SV04	1	60	1
14	SV04	4	50	1
15	SV04	NULL	55	2
16	NULL	NULL	52	16

Hàm tổng hợp 1-3

Các nhà phát triển cần thực hiện phân tích trên hàng, chẳng hạn như đếm hàng, đáp ứng cụ thể tiêu chí hoặc tóm tắt tổng doanh số của tất cả các đơn hàng.

Các hàm tổng hợp cho phép thực hiện được điều đó.

Các hàm tổng hợp bỏ qua các giá trị NULL, ngoại trừ khi sử dụng COUNT(*) .

Các hàm tổng hợp trong danh sách SELECT không tạo ra bí danh cột.

Các hàm tổng hợp trong mệnh đề SELECT hoạt động trên tất cả các hàng được truyền vào giai đoạn SELECT.

Hàm tổng hợp 2-3

Tên hàm	Cú pháp	Sự miêu tả
TRUNG BÌNH	AVG(<biểu thức>)	Tính toán giá trị trung bình của tất cả các giá trị số không phải NULL trong một cột.
ĐẾM hoặc ĐẾM_LỚN	ĐẾM(*) hoặc ĐẾM(<biểu thức>)	Khi sử dụng (*), hàm này sẽ đếm tất cả các hàng, bao gồm cả những hàng có NULL. Hàm này trả về số lượng các hàng không phải NULL cho cột khi một cột được chỉ định là <biểu thức>. Giá trị trả về của hàm COUNT là một int. Giá trị trả về của COUNT_BIG là một big_int.
TỐI ĐA	MAX(<biểu thức>)	Trả về số lớn nhất, ngày/giờ gần nhất hoặc chuỗi xuất hiện gần nhất.
TỐI THIỂU	MIN(<biểu thức>)	Trả về số nhỏ nhất, ngày/giờ sớm nhất hoặc chuỗi xuất hiện đầu tiên.
TỔNG	TỔNG(<biểu thức>)	Tính tổng của tất cả các giá trị số không phải NULL trong một cột.

Hàm tổng hợp 3-3

	AvgUnitPrice	MinQty	MaxDiscount
1	465.0934	1	0.40

Sử dụng AggregateFunctions

	Earliest	Latest
1	2011-05-31 00:00:00.000	2014-06-30 00:00:00.000

Sử dụng hàm tổng hợp với dữ liệu không phải số

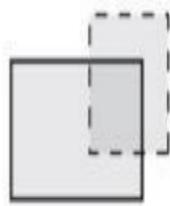
Tổng hợp không gian 1-3

SQL Server cung cấp một số phương pháp giúp tổng hợp hai mục dữ liệu hình học hoặc địa lý riêng biệt.

Method	Description
STUnion	Returns an object that represents the union of a geometry/geography instance with another geometry/geography instance.
STIntersection	Returns an object that represents the points where a geometry/geography instance intersects another geometry/geography instance.
STConvexHull	Returns an object representing the convex hull of a geometry/geography instance. A set of points is called convex if for any two points, the entire segment is contained in the set. The convex hull of a set of points is the smallest convex set containing the set. For any given set of points, there is only one convex hull.

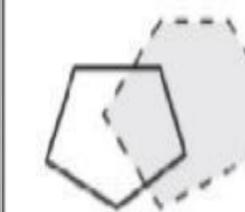
Phương pháp tổng hợp không gian

Tổng hợp không gian 2-3



Combining two polygons using the `STUnion()` method results in a merged polygon.

`STUnion()`



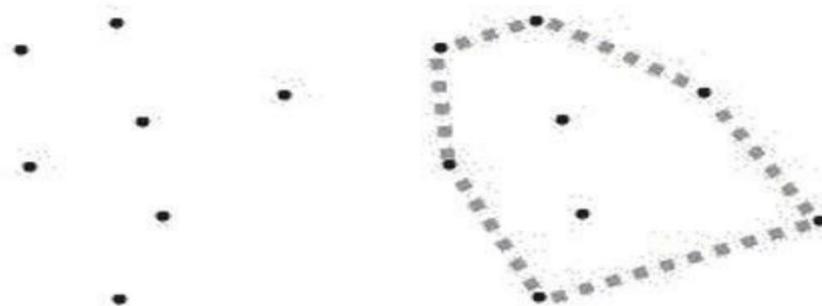
Using `STIntersection()` with two polygons can lead to another polygon.



Giao điểm ST()

A Set of Points

The Convex Hull



`STConvexHull()`

Tổng hợp không gian 3-3

SQLQuery1.sql - INT...(intel\admin (54))*

```

DECLARE @Sq geometry
= geometry::STGeomFromText('POLYGON((15 15, 15 250, 250 250, 250 15, 15 15))', 0),
@Tri geometry
= geometry::STGeomFromText('POLYGON((100 100,200 300,300 100, 100 100))', 0);

SELECT @Sq.STUnion(@Tri)

```

100 %

Results Spatial results Messages

Sử dụng STUnion() với geographyType

SQLQuery1.sql - INT...(intel\admin (54))*

```

DECLARE @Sq geometry
= geometry::STGeomFromText('POLYGON((15 15, 15 250, 250 250, 250 15, 15 15))', 0),
@Tri geometry
= geometry::STGeomFromText('POLYGON((100 100,200 300,300 100, 100 100))', 0);

SELECT @Sq.STIntersection(@Tri)

```

100 %

Results Spatial results Messages

Sử dụng STIntersection() với geographyType

Thêm SpatialAggregate 1-5

Union
Aggregate

Envelope
Aggregate

Collection
Aggregate

Convex Hull
Aggregate

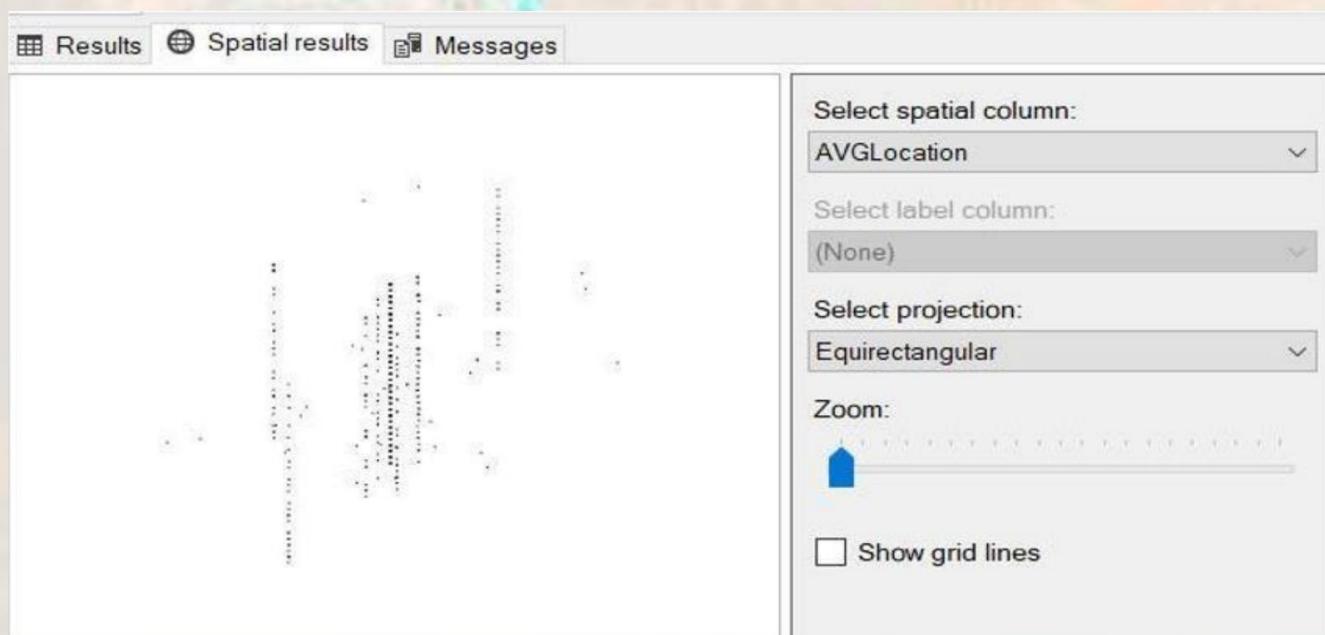
Các tổng hợp này được triển khai như các phương pháp tĩnh, hoạt động cho hoặc kiểu dữ liệu địa lý hoặc hình học.

Thêm SpatialAggregates 2-5

Tổng hợp liên hợp

Thực hiện thao tác hợp nhất trên một tập hợp các đối tượng hình học.

Nó kết hợp nhiều đối tượng không gian thành một đối tượng không gian duy nhất, xóa bỏ ranh giới bên trong, nếu có thể.



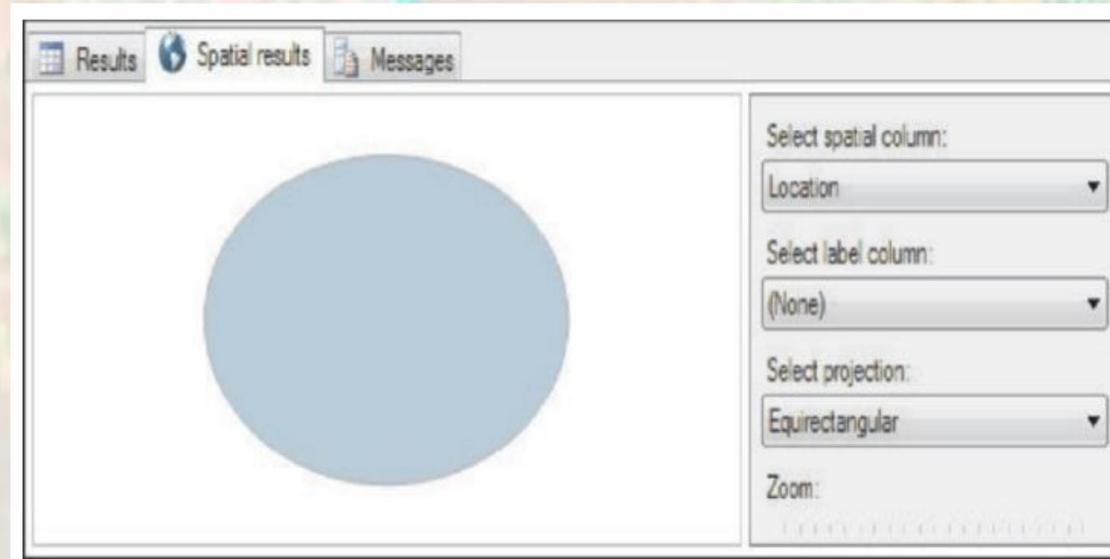
Xem SpatialResults

Thêm SpatialAggregate 3-5

Phong bìTổng hợp

Envelope Aggregate trả về một vùng giới hạn cho một tập hợp hình học hoặc đối tượng địa lý nhất định.

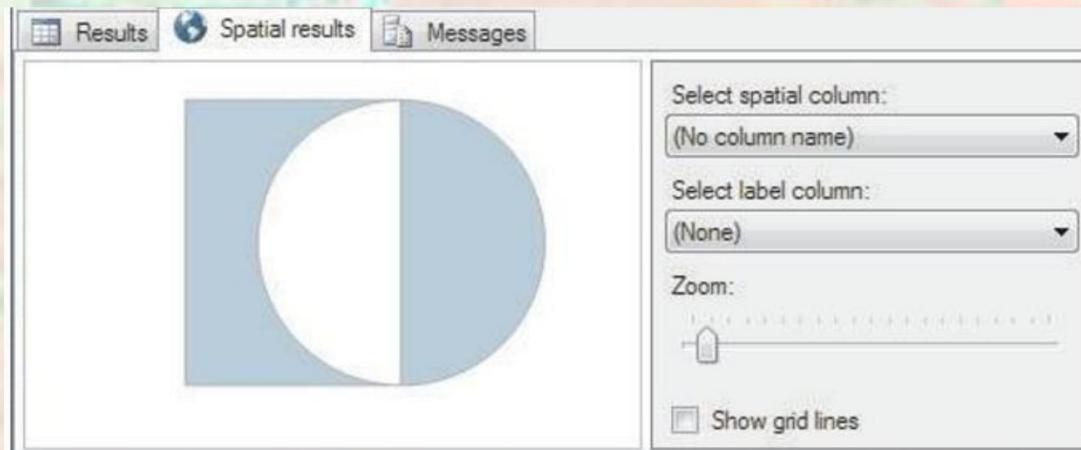
Nó thể hiện những hành vi khác nhau đối với các loại địa lý và hình học.



MoreSpatial Aggregates 4-5

Bộ sưu tập Tổng hợp

Nó trả về một thẻ hiện GeometryCollection/GeographyCollection với một phần hình học/địa lý cho mỗi đối tượng không gian trong tập lựa chọn.

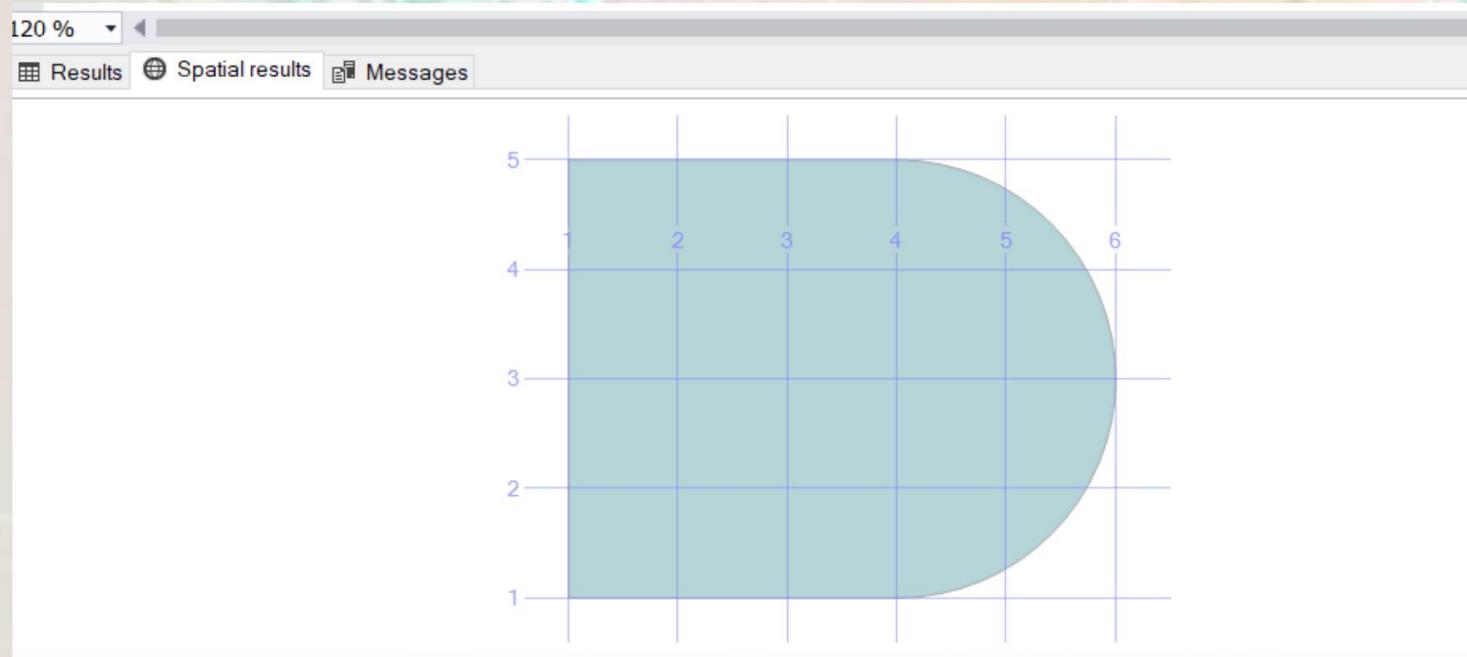


Sử dụng CollectionAggregate

More Spatial Aggregates 5-5

Vở lồi Tổng hợp

Trả về một đa giác lồi bao quanh một hoặc nhiều đối tượng không gian cho một tập hợp các đối tượng hình học/địa lý nhất định.



Sử dụng ConvexHullAggregate

Các truy vấn phụ 1-2

Truy vấn bên ngoài được gọi là truy vấn cha và truy vấn bên trong được gọi là truy vấn phụ.

Mục đích của truy vấn phụ là trả về kết quả cho truy vấn bên ngoài.

Nói cách khác, câu lệnh truy vấn bên trong phải trả về cột hoặc các cột được sử dụng trong tiêu chí của câu lệnh truy vấn bên ngoài.

Cú pháp:

CHỌN <Tên Cột> TỪ <bảng>

NƠI <Tên Cột> =

(CHỌN <Tên Cột> TỪ <Bảng> NƠI <Tên Cột> = <Điều kiện>)

Ví dụ

CHỌN Ngày đến hạn, Ngày giao hàng TỪ SalesOrderHeader
 NƠI OrderDate = (CHỌN MAX(OrderDate) TỪ SalesOrderHeader)

	Results	Messages
	DueDate	ShipDate
1	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
2	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
3	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
4	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
5	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
6	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
7	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
8	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
9	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
10	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000

Các truy vấn phụ 2-2

Dựa trên kết quả trả về của truy vấn bên trong, một truy vấn phụ có thể được phân loại thành truy vấn phụ vô hướng hoặc truy vấn phụ đa giá trị:

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

Các truy vấn con đa giá trị trả về kết quả tương tự như bảng một cột. Ở đây, truy vấn bên ngoài phải được viết để xử lý nhiều kết quả có thể có.

Làm việc với các truy vấn đa giá trị

The ntext, text, and image data types cannot be used in the SELECT list of subqueries.

The SELECT list of a subquery introduced with a comparison operator can have only one expression or column name.

Subqueries that are introduced by a comparison operator not followed by the keyword ANY or ALL cannot include GROUP BY and HAVING clauses.

You cannot use DISTINCT keyword with subqueries that include GROUP BY.

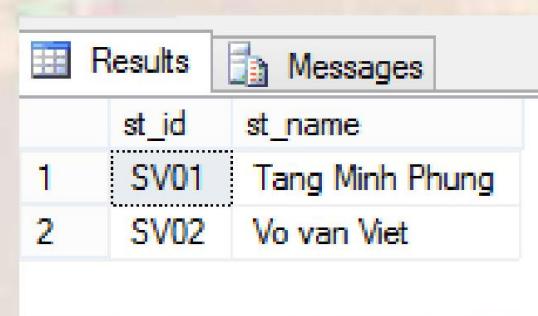
You can specify ORDER BY only when TOP is also specified.

Các truy vấn con lồng nhau

Một truy vấn con được định nghĩa bên trong một truy vấn con khác được gọi là truy vấn con lồng nhau.

Ví dụ:

```
-- list of students have taken HTML5 exam
select st_id, st_name from tbStudent a
where st_id in (select student from tbMark
                 where [subject] in (select sb_id from tbSubject
                                       where sb_name like 'html5')));
go
```



The screenshot shows the 'Results' tab of a SQL query window in SSMS. The results are displayed in a table with two columns: 'st_id' and 'st_name'. There are two rows returned:

	st_id	st_name
1	SV01	Tang Minh Phung
2	SV02	Vo van Viet

Các truy vấn có liên quan

Khi một truy vấn con lấy số từ truy vấn cha của nó, nó được gọi là truy vấn con có tương quan.

Ví dụ:

```
--list of 18-year-old students have passed exam
select st_id, st_name, datediff(yy,dob,getdate()) as [age]
from tbStudent
where st_id in (select student from tbMark where mark>40
and datediff(yy,dob,getdate())=18);
go
```

The screenshot shows the SQL Server Management Studio interface with the 'Results' tab selected. The query output displays a table with three columns: st_id, st_name, and age. There is one row returned, with the value 'SV04' highlighted in the st_id column.

	st_id	st_name	age
1	SV04	Trinh Minh The	18

Đầu ra của các truy vấn tương quan

Tham gia

- Specifying the column from each table to be used for the join. A typical join specifies a foreign key from one table and its associated key in the other table.

- Specifying a logical operator such as =, <> to be used in comparing values from the columns.

	First Name	Last Name	Job Title
1	Ken	Sánchez	Chief Executive Officer
2	Temi	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

Đầu ra củaJoin

Ba loại liên kết:

Inner Joins

Outer Joins

Self-Joins

Tham gia bên trong

Một liên kết bên trong được hình thành khi các bản ghi từ hai bảng được kết hợp chỉ khi các hàng từ cả hai bảng được khớp dựa trên một cột chung

Sau đây là cú pháp của phép nối bên trong:

```
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>
```

Nối ngoài

Outer join là các câu lệnh join trả về tất cả các hàng từ ít nhất một trong các bảng được chỉ định trong FROM mệnh đề, miễn là các hàng đó đáp ứng bất kỳ WHERE hoặc có điều kiện của câu lệnh SELECT.

Hai loại mối nối ngoài thường được sử dụng



Nối ngoài bên trái

Trả về tất cả các bản ghi từ bảng bên trái và chỉ các bản ghi khớp từ bảng bên phải.

Cú pháp:

CHỌN <TênCột1>, <TênCột2>...<TênCộtN>

TỪ Table_A AS **Alias_A** TRÁI OUTER JOIN Table_B AS **Alias_B**

TRÊN Biệt danh_A.<CommonColumn> = Biệt danh_B.<CommonColumn>

Ví dụ:

CHỌN A.CustomerID, B.DueDate, B.ShipDate TỪ Khách hàng A LIỀN

KẾT NGOÀI TRÁI SalesOrderHeader B

TRÊN A.CustomerID = B.CustomerID VÀ NĂM(B.DueDate)<2012;

	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

Nối ngoài phải

Truy xuất tất cả các bản ghi từ bảng thứ hai trong liên kết bất kể có hay không có dữ liệu khớp trong bảng đầu tiên hay không.

Cú pháp:

CHỌN <Danh sách cột>

TỪ Table_A AS **Alias_A** PHẢI OUTER JOIN Table_B AS **Alias_B**

TRÊN Biệt danh_A.<CommonColumn> = Biệt danh_B.<CommonColumn>

Tự tham gia

Tự nối được sử dụng để tìm các bản ghi trong một bảng liên quan đến các bản ghi khác trong cùng một bảng.

Một bảng được liên kết với chính nó bằng phép tự liên kết.

	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

Ví dụ:

```
CHỌN TOP 7 A.fname + ' ' + A.name AS 'Tên nhân viên',
B.fname + ' ' + B.lname AS 'Quản lý'
TỪ
Nhân viên NHƯ MỘT INNER JOIN Nhân viên NHƯ B
TRÊN A.mgr_id = B.emp_id
```

	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

Tuyên bố MERGE

Insert a new row from the source if the row is missing in the target table

Update a target row if a record already exists in the source table

Delete a target row if the row is missing in the source table

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

Biểu thức bảng chung (CTE)

CTE tương tự như một tập kết quả tạm thời được xác định trong phạm vi thực thi của một câu lệnh SELECT, INSERT, UPDATE, DELETE hoặc CREATE VIEW. CTE là một biểu thức được đặt tên được xác định trong truy vấn.

Một CTE bao gồm các tham chiếu đến chính nó được gọi là CTE đệ quy.

CTEs are limited in scope to the execution of the outer query. Hence, when the outer query ends, the lifetime of the CTE will end.

You must define a name for a CTE and also, define unique names for each of the columns referenced in the SELECT clause of the CTE.

It is possible to use inline or external aliases for columns in CTEs.

A single CTE can be referenced multiple times in the same query with one definition.

Biểu thức bảng chung (CTE)

Đoạn mã sau định nghĩa hai CTE bằng cách sử dụng một mệnh đề WITH duy nhất:

```
VỚI CTE_Students AS
(
    Chọn StudentCode, S.Name,C.CityName, St.Status TỪ Student S
    INNER JOIN Thành phố C
    TRÊN S.CityCode = C.CityCode INNER
    JOIN Trạng thái St TRÊN
    S.StatusId = St.StatusId
)
'
StatusRecord -- Đây là CTE thứ hai được định nghĩa
BẰNG
(
    CHỌN Trạng thái, Đếm(Tên) NHƯ Đếm số lượng học sinh
    TỪ CTE_Sinh viên
    NHÓM THEO Trạng thái
)
LỰA CHỌN * TỪ StatusRecord
```

	Status	CountofStudents
1	Failed	2
2	Passed	2

Kết hợp dữ liệu bằng cách sử dụng toán tử SET

SQL Server 2019 cung cấp một số từ khóa nhất định, còn gọi là toán tử, để kết hợp dữ liệu từ nhiều bảng.

Các toán tử này như sau:

CÔNG ĐOÀN

GIAO THOẠI

NGOẠI TRỪ

Toán tử UNION 1-2

Kết quả từ hai câu lệnh truy vấn khác nhau có thể được kết hợp thành một tập kết quả duy nhất bằng toán tử UNION .

Các câu lệnh truy vấn phải có cột tương thích các loại và số lượng cột bằng nhau.

Tên cột có thể khác nhau trong mỗi câu lệnh, nhưng các kiểu dữ liệu phải tương thích.

Theo kiểu dữ liệu tương thích, điều này có nghĩa là có thể chuyển đổi nội dung của một trong các cột sang cột khác.

Sau đây là cú pháp của toán tử UNION.

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

Nếu bạn đưa mệnh đề ALL vào, tất cả các hàng đều được đưa vào tập kết quả, bao gồm cả các bản ghi trùng lặp.

Toán tử UNION 2-2

Ví dụ: Liên minh

Table 1 – CUSTOMERS Table is as follows.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Table 2 – ORDERS Table is as follows.

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

Now, let us join these two tables in our SELECT statement as follows –

```
SQL> SELECT ID, NAME, AMOUNT, DATE
      FROM CUSTOMERS
      LEFT JOIN ORDERS
      ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
UNION
      SELECT ID, NAME, AMOUNT, DATE
      FROM CUSTOMERS
      RIGHT JOIN ORDERS
      ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result –

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL

Toán tử UNION 2-3

Ví dụ: Union All

Table 1 – CUSTOMERS Table is as follows.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Table 2 – ORDERS Table is as follows.

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

Now, let us join these two tables in our SELECT statement as follows –

```
SQL> SELECT ID, NAME, AMOUNT, DATE
   FROM CUSTOMERS
   LEFT JOIN ORDERS
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
UNION ALL
   SELECT ID, NAME, AMOUNT, DATE
   FROM CUSTOMERS
   RIGHT JOIN ORDERS
     ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
2	Khilan	1560	2009-11-20 00:00:00
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

Toán tử INTERSECT

Toán tử INTERSECT được sử dụng với hai truy vấn các câu lệnh trả về một tập hợp các hàng riêng biệt chung cho cả hai câu lệnh truy vấn.

Các quy tắc cơ bản để sử dụng INTERSECT như sau:

Số lượng cột và thứ tự mà chúng được đưa ra phải giống nhau trong cả hai truy vấn.

Kiểu dữ liệu của các cột đang sử dụng phải tương thích.

Cú pháp:

Câu lệnh truy vấn 1

GIAO CẮT

Câu lệnh truy vấn 2

Toán tử INTERSECT

Ví dụ:

Customers Table:

ID	Name	Address	Age	Salary
1	Harsh	Delhi	20	3000
2	Pratik	Mumbai	21	4000
3	Akash	Kolkata	35	5000
4	Varun	Madras	30	2500
5	Souvik	Banaras	25	6000
6	Dhanraj	Siliguri	22	4500
7	Riya	Chennai	19	1500

Orders Table:

Oid	Date	Customer_id	Amount
102	2017-10-08	3	3000
100	2017-10-08	3	1500
101	2017-11-20	2	1560
103	2016-5-20	4	2060

Sample Queries:

```

SELECT ID, NAME, Amount, Date
FROM Customers
LEFT JOIN Orders
ON Customers.ID = Orders.Customer_id
INTERSECT
SELECT ID, NAME, Amount, Date
FROM Customers
RIGHT JOIN Orders
ON Customers.ID = Orders.Customer_id;
    
```

Output:

ID	Name	Amount	Date
3	Akash	3000	2017-10-08
3	Akash	1500	2017-10-08
2	Pratik	1560	2017-11-20
4	Varun	2060	2016-05-20

NGOẠI TRỪ Toán tử

Toán tử EXCEPT trả về tất cả các hàng riêng biệt từ truy vấn được đưa ra ở bên trái toán tử EXCEPT và xóa tất cả các hàng khỏi tập kết quả khớp với các hàng ở bên phải toán tử EXCEPT.

Cú pháp:

```
Câu lệnh truy vấn 1  
NGOẠI TRỪ  
Câu lệnh truy vấn 2
```

Hai quy tắc áp dụng cho toán tử INTERSECT cũng áp dụng cho EXCEPT người điều hành.

Ví dụ:

```
CHỌN ProductId Từ Sản phẩm  
NGOẠI TRỪ  
CHỌN ProductId Từ SalesOrderDetail
```

NGOẠI TRỪ Toán tử

Table 1 – CUSTOMERS Table is as follows.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Table 2 – ORDERS table is as follows.

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

Now, let us join these two tables in our SELECT statement as shown below.

```
SQL> SELECT ID, NAME, AMOUNT, DATE
   FROM CUSTOMERS
   LEFT JOIN ORDERS
   ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
EXCEPT
   SELECT ID, NAME, AMOUNT, DATE
   FROM CUSTOMERS
   RIGHT JOIN ORDERS
   ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result.

ID	NAME	AMOUNT	DATE
1	Ramesh	NULL	NULL
5	Hardik	NULL	NULL
6	Komal	NULL	NULL
7	Muffy	NULL	NULL

Các hoạt động xoay và nhóm tập hợp

Quá trình chuyển đổi dữ liệu từ hướng theo hàng sang hướng theo cột được gọi là xoay trực.

Các toán tử PIVOT và UNPIVOT của SQL

Mã chủ giúp thay đổi hướng dữ liệu từ hướng cột sang hướng hàng và ngược lại
ngược lại.

Toán tử PIVOT

Grouping

In the FROM clause, the input columns must be provided. The PIVOT operator uses those columns to determine which column(s) to use for grouping the data for aggregation.

Spreading

Here, a comma-separated list of values that occur in the source data is provided that will be used as the column headings for the pivoted data.

Aggregation

An aggregation function, such as SUM, to be performed on the grouped rows.

CHỌN TOP 5 SUM(SalesYTD) LÀ TotalSalesYTD, Tên
TỪ Sales.SalesTerritory
NHÓM THEO Tên

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

Toán tử PIVOT

Top 5 doanh số bán hàng trong năm tính đến nay cùng với tên lãnh thổ được nhóm theo tên lãnh thổ được hiển thị.

Truy vấn tương tự được viết lại trong đoạn mã sau bằng cách sử dụng PIVOT để dữ liệu được chuyển đổi từ hướng dựa trên hàng sang hướng dựa trên cột:

```
-- Bảng trực với một hàng và sáu cột CHỌN TOP 5
'TotalSalesYTD' LÀM GrandTotal, [Tây Bắc], [Đông
Bắc], [Trung tâm], [Tây Nam], [Đông Nam]
TỪ
(CHỌN TÊN TOP 5, SalesYTD TỪ
Sales.SalesTerritory ) NHƯ
SourceTable PIVOT

(
SUM(Doanh số năm nay)
CHO Tên IN ([Tây Bắc], [Đông Bắc], [Trung tâm], [Tây Nam], [Đông Nam])
) AS PivotTable;
```

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

Nhà điều hành UNPIVOT

Source columns to be unpivoted

A name for the new column that will display the unpivoted values

A name for the column that will display the names of the unpivoted values

```

SELECT SalesYear, TotalSales FROM
(
    SELECT * FROM
    (
        SELECT YEAR(SOH.OrderDate) AS SalesYear,
               SOH.SubTotal AS TotalSales
        FROM sales.SalesOrderHeader SOH
        JOIN sales.SalesOrderDetail SOD ON SOH.SalesOrderId =
SOD.SalesOrderId
    ) AS Sales PIVOT(SUM(TotalSales) FOR SalesYear IN([2011],
[2012],
[2013],
[2014])) AS PVT
) T UNPIVOT(TotalSales FOR SalesYear IN([2011],
[2012],
[2013],
[2014])) AS upvt;

```

	SalesYear	TotalSales
1	2011	151706131.5475
2	2012	862786335.1754
3	2013	1190722789.0552
4	2014	391255200.8993

Đầu ra cho UNPIVOT

Bản tóm tắt

- Mệnh đề GROUP BY và các hàm tổng hợp cho phép nhóm và/hoặc tổng hợp dữ liệu lại với nhau để trình bày thông tin tóm tắt.
- Các hàm tổng hợp không gian lần đầu tiên được giới thiệu trong SQL Server 2012 và cũng được hỗ trợ trong SQL Server 2019.
- Một truy vấn phụ cho phép tập kết quả của một câu lệnh SELECT được sử dụng như tiêu chí cho câu lệnh SELECT khác.
- Các phép nối giúp bạn kết hợp dữ liệu cột từ hai hoặc nhiều bảng dựa trên mối quan hệ logic giữa các bảng.
- Các toán tử tập hợp như UNION và INTERSECT giúp bạn kết hợp các hàng dữ liệu 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.