عنوان: **آشنایی با Window Function ها در SQL Server بخش دوم**
نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۹/۱۹ ۹:۳۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SQL Server 2012, T-SQL

قبل از مطالعه این بخش لطفا آشنایی با Window Functionها در SQL Server بخش اول را مطالعه نمایید.
در بخش اول ،در مورد Syntax مربوط به Over Clause صحبت کردیم، و برای درک استفاده از Over Clause، مثالهایی را بررسی نمودیم، در این بخش نیز،به تفاوت Row Clause و Range Clause می پردازیم.
مثال: با ایجاد یک Script،عملیات جمع روی یک فیلد خاص، بوسیله Row Clause و Range Clause انجام می‌دهیم. تا تفاوت آنها را درک نماییم.
در ادامه Script زیر را اجرا نمایید:

```
DECLARE @Test TABLE
(
RowID INT IDENTITY,
FName VARCHAR(20),
Salary SMALLINT
);
INSERT INTO @Test (FName, Salary)
VALUES ('George', 800),
('Sam', 950),
('Diane', 1100),
('Nicholas', 1250),
('Samuel', 1250),
('Patricia', 1300),
('Brian', 3000),
('Thomas', 1600),
('Fran', 2450),
('Debbie', 2850),
('Mark', 2975),
('James', 3000),
('Cynthia', 3000),
('Christopher', 5000);

SELECT RowID,FName,Salary,
       SumByRows = SUM(Salary) OVER (ORDER BY Salary ROWS UNBOUNDED PRECEDING),
   SumByRange = SUM(Salary) OVER (ORDER BY Salary RANGE UNBOUNDED PRECEDING)
FROM @Test
ORDER BY RowID;
```

خروجی بصورت زیر خواهد بود:

	RowID	FName	Salary	SumByRows	SumByRange
1	1	George	800	800	800
2	2	Sam	950	1750	1750
3	3	Diane	1100	2850	2850
4	4	Nicholas	1250	4100	5350
5	5	Samuel	1250	5350	5350
6	6	Patricia	1300	6650	6650
7	7	Brian	1500	8150	8150
8	8	Thomas	1600	9750	9750
9	9	Fran	2450	12200	12200
10	10	Debbie	2850	15050	15050
11	11	Mark	2975	18025	18025
12	12	James	3000	21025	24025
13	13	Cynthia	3000	24025	24025
14	14	Christopher	5000	29025	29025

با مشاهده شکل بالا، به وضوح می‌توان تفاوت Row و Range را تشخیص داد. در Script بالا از UNBOUNDED PRECEDING استفاده کردیم ، و مفهوم قالب آن به شرح ذیل می‌باشد:

مقدار فیلد Salary سطر جاری = جمع مقادیر فیلد Salary همه سطرهای ماقبل،سطر جاری + مقدار فیلد Salary سطر جاری

Row Clause بصورت فیزیکی به سطرها می‌نگرد و قالب بیان شده در Script را،روی تمامی سطرها،نسبت به جایگاه آنها در جدول، به ترتیب اعمال می‌نماید.و در شکل نیز قابل مشاهده می باشد، یعنی به چیدمان سطرها در خروجی که بصورت فیزیکی نمایش داده شده است، توجه می کند، و حاصل جمع هر سطر برابر است با حاصل جمع سطرهای ماقبل + سطر جاری

اما Range Clause:به چیدمان فیزیکی سطرها توجه نمی‌کند، بلکه بصورت منطقی به مقدار فیلد Salary سطرها توجه می‌نماید، یعنی مقادیری که در یک محدوده(Range) قرار دارند، حاصل جمع آنها،یکی است.

مقدار فیلد Salary سطر چهار و پنج برابر است با 1250 بنابراین حاصل جمع آنها برابر هم می‌باشد. و بصورت زیر محاسبه می‌شود:

5350= 1250 + 1250 + 1100 + 950 + 800

روش بیان شده، در مورد سطرهای 12 و 13 نیز صادق است.

امیدوارم با مثالهایی که در بخش اول و بخش دوم بررسی نمودیم، روش استفاده از Over Clause را درک کرده باشیم.

Window Functionها را به چهار بخش تقسیم بندی شده اند، که به شرح ذیل می‌باشد:

1- Ranking functions (توابع رتبه بندی)، که بررسی نمودیم.

2- NEXT VALUE FOR ، که در بحث ایجاد Sequence آن را بررسی نمودیم.

3- Aggregate Functions (توابع جمعی)، اکثرا با اینگونه توابع آشنا هستیم.

4- Analytic Functions (توابع تحلیلی) که در بخش بعدی آن را بررسی می‌نماییم.

یکی از منابع بسیار مفید در مورد Window Function ها کتاب Microsoft SQL Server 2012 High-Performance T-SQL Using Window Functions ، می باشد،که بطور کامل به Window Functionها اختصاص دارد و تکنیک‌های بسیار مفیدی را بیان می‌کند.

مطالعه آن به علاقمندان، پیشنهاد می‌گردد.

موفق باشید.

نظرات خوانندگان

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۹/۱۹ ۱۲:۲۱

دوست عزیز ممنون...
من قسمت Range و Row رو که شما توضیح دادی درست متوجه نشدم سرچی که زدم متوجه شدم این قابلیت تقریبا شبیه قسمت WITH TIES تو Select هست .برای مثال اگه بخواهیم 3 شاگرد برتر کلاس رو کوئری بزنیم اگه تو کلاس 3 نفر معدل 18 داشته باشن(با توجه به اینکه یک معدل 20 و 19 داریم) 2 نفر از شاگردها که معدل 18 دارن تو این کوئری نمیان(TOP 3) و...
برداشت منم ازRange اینه که بواسط ی برابر بودن تاریخ‌ها این 2 مقدار به هم گره خوردن و هنگام محاسبه مقدار یکسانی را تولید میکنن.

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۹/۱۹ ۱۴:۲۱

سلام
اگر سئوال شما رو درست متوجه شده باشم،با یک مثال مفهوم Range رو بررسی می‌کنیم:

```
CREATE TABLE #Transactions
(
AccountId INTEGER,
TranDate DATE,
TranAmt NUMERIC(8, 2)
);
INSERT INTO #Transactions
SELECT *
FROM ( VALUES ( 1, '2011-01-15', 50),( 1, '2011-01-17', 500),( 1, '2011-01-17', 500),
  ( 1, '2011-01-16', 500),( 1, '2011-01-24', 75),( 1, '2011-01-26', 125),
  ( 1, '2011-02-28', 500),( 2, '2011-01-01', 500),( 2, '2011-01-15', 50),
              ( 2, '2011-01-22', 25),( 2, '2011-01-23', 125),( 2, '2011-01-26', 200),
              ( 2, '2011-01-29', 250),( 3, '2011-01-01', 500),( 3, '2011-01-15', 50 ),
              ( 3, '2011-01-22', 5000),( 3, '2011-01-25', 550),( 3, '2011-01-27', 95 ),
              ( 3, '2011-01-30', 2500)
) dt (AccountId, TranDate, TranAmt);
```

روی جدول فوق دو نوع Script اجرا می‌کنیم، مثال اول، براساس AccountID جدول را گروه بندی می‌نماییم. سپس هر گروه را براساس تاریخ Sort می‌کنیم، و در هر گروه مقدار Sum آن را بدست می‌آوریم:

```
SELECT
    AccountId,
    TranDate,
  TranAmt,
    Sum(TranAmt) OVER(partition by Accountid ORDER BY TranDate RANGE UNBOUNDED PRECEDING) AS SumAmt
FROM  #Transactions
GO
```

خروجی :

	AccountId	TranDate	TranAmt	SumAmt
1	1	2011-01-15	50.00	50.00
2	1	2011-01-16	500.00	550.00
3	1	2011-01-17	500.00	1550.00
4	1	2011-01-17	500.00	1550.00
5	1	2011-01-24	75.00	1625.00
6	1	2011-01-26	125.00	1750.00
7	1	2011-02-28	500.00	2250.00

مطابق شکل Sort براساس TranDate است، که چهار مقدار 500 در سه بازه تاریخی دیده می‌شود، حال محاسبه جمع هر سطر بصورت زیر است:

سطر دوم با وجود اینکه مقدار آن 500 است و در بازه تاریخی 2011-01-16 قرار دارد: مقدار آن برابر است با 550=50 + 500

سطر سوم و چهارم که در بازه تاریخی 2011-01-17 می‌باشد(به عبارتی در یک محدوده می‌باشند): برابر است با :

500+500+500+50=1550

در اینجا چیزی حذف نشده، حاصل جمع سطر سوم و چهارم ، چون در یک محدوده (Range) می‌باشد، برابر است با حاصل جمع سطرهای ما قبل یعنی سطر اول و دوم (500+50=550) + حاصل جمع تمامی سطرهای آن محدوده(Range)، یعنی سطر سوم و چهارم (500+500=1000)

مثال دیگر، در این حالت Sort روی فیلد TranAMT انجام می‌شود، و جدول همچنان روی فیلد Accountid گروه بندی می‌شود، بنابراین خواهیم داشت:

```
SELECT
    AccountId,
    TranDate,
  TranAmt,
    Sum(TranAmt) OVER(partition by Accountid ORDER BY TranAmt RANGE UNBOUNDED PRECEDING) AS SumAmt
FROM  #Transactions
GO
```

خروجی :

	AccountId	TranDate	TranAmt	SumAmt
1	1	2011-01-15	50.00	50.00
2	1	2011-01-24	75.00	125.00
3	1	2011-01-26	125.00	250.00
4	1	2011-02-28	500.00	2250.00
5	1	2011-01-17	500.00	2250.00
6	1	2011-01-17	500.00	2250.00
7	1	2011-01-16	500.00	2250.00
8	2	2011-01-22	25.00	25.00
9	2	2011-01-15	50.00	75.00
10	2	2011-01-23	125.00	200.00

در شکل، مقدار جمع هیچ سطری حذف نشده است، و Top ی هم در کار نیست.
حال اگر مثال فوق را روی میانگین در نظر بگیرید، باز هم تمام مقادیر، در محاسبه میانگین تاثیر گذار میباشند.

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۹/۱۹ ۱۴:۵۹

ممنون...
« سطرهای فیزیکی » و «سطرهای منطقی» که شما گفتید برا من گیج کننده بود من با مثال Top ی که زدم فرق رو متوجه شدم گفتم عنوان کنم دیگران هم استفاده کنن.البته این پست شما خیلی بهتر موضوع رو توضیح داد.

نویسنده: محمد سلم ابادی
تاریخ: ۱۳۹۱/۱۰/۳۰ ۱۲:۱۱

سلام،
ممنون از مطالب مفیدتون.
آیا دو دستور زیر با هم یکسان هستند یا خیر؟
range unbounded preceding
range between unbounded preceding and current row

و کوئری اولتون باید مساله running total باشه، که به سادگی توسط Over Clause حل شده.
کوئری زیر روشی بوده که قبل از نسخه 2012 برای حل اینگونه مسائل مورد استفاده قرار میگرفته

```
SELECT AccountId,
       TranDate,
       TranAmt,
       D.sumAmt AS older_method
       Sum(TranAmt) OVER(partition by Accountid
                         ORDER BY TranDate
                         RANGE UNBOUNDED PRECEDING) AS SumAmt
FROM  #Transactions AS T1
CROSS APPLY (SELECT SUM(T2.TranAmt)
   FROM #Transactions AS T2
  WHERE T2.AccountId = T1.AccountId
    AND T2.TranDate <= T1.TranDate) AS D(SumAmt)
ORDER BY T1.AccountId, T1.TranDate;
```

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۱۰/۳۰ ۱۳:۲۵

سلام
در جواب سؤال شما باید بگویم هر دو دستور یکی میباشند و هر دو از اولین سطر تا سطر جاری را در نظر می‌گیرند.

نویسنده: zarei
تاریخ: ۱۳۹۲/۰۲/۲۲ ۱۹:۳

سلام
من یه کوئری توسط Over ()Sum() .. نوشتم که تو در تو هست که ترتیب جمع دستور بیرونی برام مهمه .

```
;WITH cteBed ([Counter], id_doc , [Year] ,id_Total , date_duc ,Number_Temp , number_fix , sumbed ,
sumbes , row_no ) AS (
SELECT [Counter], d.id_doc , d.[Year] ,r.id_Total , d.date_duc ,d.Number_Temp ,d.number_fix ,
SUM( r.Mablagh_bed) OVER(PARTITION BY d.[Year] ,r.id_Total , d.Number_Temp) AS sumbed ,
 sumbes= 0,
```

```
ROW_NUMBER() OVER (PARTITION BY d.[Year] ,r.id_Total , d.date_duc , d.Number_Temp , d.number_fix  ORDER
BY  d.date_duc )AS  row_no
FROM tbl_Records r JOIN tbl_Documents d ON d.id_doc = r.id_doc  ) ,

 cteBes ([Counter], id_doc , [Year] ,id_Total , date_duc ,Number_Temp , number_fix , sumbed , sumbes  ,
row_no) AS (
SELECT   [Counter], d.id_doc , d.[Year] ,r.id_Total , d.date_duc ,d.Number_Temp ,d.number_fix , sumbed
= 0 ,
SUM( r.Mablagh_bes ) OVER(PARTITION BY d.[Year] ,r.id_Total , d.Number_Temp ) AS sumbes,
ROW_NUMBER() OVER (PARTITION BY d.[Year] ,r.id_Total ,  d.date_duc ,d.Number_Temp , d.number_fix ORDER
BY  d.date_duc )AS row_no
FROM tbl_Records r JOIN tbl_Documents d ON d.id_doc = r.id_doc )

SELECT [Counter], id_doc , [Year] ,id_Total , date_duc ,Number_Temp , number_fix , sumbed , sumbes ,
amountBed ,amountBes
,SUM(amountBed)OVER(  ORDER BY [Year] ,id_Total , date_duc , number_Temp, number_Fix ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW ) AS bed
,SUM(amountBes)OVER(  ORDER BY [Year] ,id_Total , date_duc , number_Temp, number_Fix ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW ) AS bes
FROM (
SELECT [Counter], id_doc , [Year] ,id_Total , date_duc ,Number_Temp , number_fix , sumbed , sumbes ,
amountBed = CASE WHEN id_Total  LIKE '1%' OR  Id_Total LIKE '2%' OR  Id_Total LIKE '7%' OR  Id_Total
LIKE '8%' THEN (tt.sumbed-tt.sumbes) ELSE 0 END ,
amountBes=CASE WHEN Id_Total LIKE '3%' OR Id_Total LIKE '4%' OR Id_Total LIKE '5%' OR Id_Total LIKE
'6%' OR Id_Total LIKE '9%' THEN (tt.sumbes-tt.sumbed)ELSE 0 END ,
ROW_NUMBER() OVER (PARTITION BY [Year] ,id_Total , date_duc , Number_Temp , number_fix  ORDER BY
date_duc )AS  row_no
FROM (
SELECT * FROM cteBed cb WHERE cb.row_no = 1
UNION ALL
SELECT * FROM cteBes cb WHERE cb.row_no = 1
) AS tt ([Counter], id_doc , [Year] ,id_Total , date_duc ,Number_Temp , number_fix , sumbed ,
sumbes,row_no ) WHERE not(sumbed = 0 AND sumbes = 0)
) AS rr
```

اگه تو یه دستور Select از Row_Number استفاده کرده باشم ، اول خروجی رو بدست میاره بعد خروجی رو بر حسب نوع مرتب سازی مربوط به Row_Number مرتب میکنه ؟ و دیگه اینکه خروجی دستور اول که شامل Row_Number هست بعد از مرتب شدن به همون صورت به دست دستور دوم (یا همون Select بیرونی) میرسه یا باز باید روی اون نیز مرتب سازی انجام بدم ؟ اصلا جای ستونی که مربوط به Row_Number هست اول یا آخر فرق میکنه ؟

اینارو به این خاطر پرسیدم ، چون هر بار داده هام جواب متفاوتی میداد و نتونستم تشخیص بدم . ممنون

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۲/۰۲/۲۵ ۸:۱۴

سلام

جواب سوال اول: در Syntax تابع Row_Number عملیات order by اجباری است، بنابراین عملیات سورت در ابتدا انجام می‌شود و سپس Row_Number (اعداد ترتیبی) روی رکوردها اعمال می‌گردد.

در سایت مایکروسافت به خوبی اشاره شده است که هیچ تضمینی وجود ندارد، خروجی یک Query با استفاده از Row_number در هر بار اجرا، با اجرای قبلی یکی باشد مگر آنکه موارد زیر را رعایت کرده باشید:

-1 مقادیر ستونی که برای قسمت Partition در نظر گرفته اید، منحصربفرد باشد.

-2 مقادیری که برای قسمت Order by در نظر گرفته اید منحصربفرد باشد.

-3 ترکیب مقادیر Partition و Order by نیز مقدار منحصربفردی را ایجاد نماید.

جواب سوال دوم: جای ستون Row_number در زمان نمایش اهمیتی ندارد.

پیشنهاد دوستانه:

-1 تاجایی که امکان دارد از OR در Queryهای خود استفاده ننمایید، باعث افزایش زمان اجرای Query شما می‌شود و هزینه بالایی دارد.

-2 از Like نیز در نوشتن Queryهای خود اجتناب کنید.

برای اطلاعات بیشتر در مورد Row_Number به آدرس زیر مراجعه نمایید: Row_Number()

موفق باشید.

نویسنده: zarei
تاریخ: ۱۳۹۲/۰۲/۲۷ ۱:۳۸

ممنون از راهنماییتون . ولی برای رسیدن به پاسخ راه دیگه ای به ذهنم نرسید (استفاده از OR و Like)
میشه خواهش کنم راه جایگزین رو بهم بگید ؟ اموزشی در خصوص بررسی مفهومی Planهای تولیدی SQL سراغ دارید ؟
دیگه اینکه از کجا میتونم به طور دقیق و مفهومی هزینه استفاده از دستورات SQL رو مثل OR و ... رو بخونم ؟
ممنون

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۲/۰۲/۲۷ ۲۲:۱۹

سلام
مقاله زیر به خوبی طرز استفاده از Execution Plan را آموزش می‌دهد. How to read SQL Server graphical query execution plans

دو کتاب زیر، جهت مطالعه و بهینه سازی در ایجاد Query مفید است: SQL Server 2012 T-SQL Recipes
SQL Server 2012 Query Performance Tuning

موفق باشید.