عنوان: **آشنایی با Window Function ها در SQL Server بخش اول**
نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۹/۱۷ ۲۳:۲۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SQL Server 2012, T-SQL

Window Functionها برای اولین بار در نسخه SQL Server 2005 ارائه گردیدند، و در ورژن‌های جدیدتر SQL Server، به تعداد این فانکشنها افزوده شده است.

تعریف Window Function :

معمولا از این نوع فانکشنها روی مجموعه ای از ROWهای یک جدول، در جهت اعمال عملیاتهای محاسباتی ،ارزیابی داده ها، رتبه بندی و غیرو... استفاده می‌گردد، به بیان ساده‌تر بوسیله Window Functionها می‌توان، ROWهای یک جدول را گروه بندی نمود. و روی گروه‌ها از توابع جمعی (Aggregate Functions) استفاده کرد. این نوع فانکشنها از قابلیت و انعطاف پذیری زیادی برخوردار می‌باشند، و بوسیله آنها می‌توان نتایج (خروجی) بسیار مفیدی از Query ها، بدست آورد، معمولا از این نوع فانکشنها در Data Mining (داده کاوی) و گزارشگیری‌ها استفاده می‌گردد. و آگاهی و روش استفاده از Window Functionها برای برنامه نویسان و DBA ها، می‌تواند بسیار مفید باشد.
مفهوم Window Function مطابق استاندارد ISO و ANSI می‌باشد، و دیتابیس هایی همچون Oracle،DB2،Sybase از آن پشتیبانی می‌نمایند.برای اطلاعات بیشتر می‌توانید به سایت‌های زیر مراجعه کنید: SQL:2003
و SQL:2008

کلمه "Window" در Window Function، به مجموعه ROW هایی اشاره می‌کند، که محاسبات و ارزیابی و غیرو... روی آنها اعمال می‌گردد.
Window Functionها برای ارائه قابلیت‌های خود، از Over Clause استفاده می‌کنند. اگر مقاله آشنایی با Row_Number،Rank،Dense_Rank،NTILE را مطالعه کرده باشید، می‌توان هریک از آنها را یک Window Function دانست.
برای شروع، به بررسی Over Clause می‌پردازیم، و Syntax آن به شرح ذیل می‌باشد:

```
OVER (
       [ <PARTITION BY clause> ]
       [ <ORDER BY clause> ]
       [ <ROW or RANGE clause> ]
      )

<PARTITION BY clause> ::=
PARTITION BY value_expression , ... [ n ]

<ORDER BY clause> ::=
ORDER BY order_by_expression
    [ COLLATE collation_name ]
    [ ASC | DESC ]
    [ ,...n ]

<ROW or RANGE clause> ::=
{ ROWS | RANGE } <window frame extent>

<window frame extent> ::=
{   <window frame preceding>
  | <window frame between>
}

<window frame between> ::=
  BETWEEN <window frame bound> AND <window frame bound>

<window frame bound> ::=
{   <window frame preceding>
  | <window frame following>
}

<window frame preceding> ::=
{
    UNBOUNDED PRECEDING
  | <unsigned_value_specification> PRECEDING
  | CURRENT ROW
}
```

```
<window frame following> ::=
{
    UNBOUNDED FOLLOWING
  | <unsigned_value_specification> FOLLOWING
  | CURRENT ROW
}

<unsigned value specification> ::=
{  <unsigned integer literal> }
```

OVER دارای سه آرگومان اختیاری است که هر کدام را به تفصیل بررسی می‌کنیم:

1- PARTITION BY clause : بوسیله این پارامتر می‌توانیم Rowهای یک جدول را گروه بندی نماییم. این پارامتر یک value_expression می پذیرد. یک Value_expression می‌تواند نام یک ستون ، یک Scalar Function ، Scalar Subquery و غیرو باشد.

2- ORDER BY clause : از نامش مشخص است و برای Sort استفاده می‌شود، و ویژگی‌های Order By در آن اعمال می‌گردد. به جز Offset.

3- ROW or RANGE clause :این پارامتر بیشتر برای محدود نمودن Row در یک Partition (گروه) مورد استفاده قرار می‌گیرد، به عنوان مثال نقطه شروع و پایان را می‌توان بوسیله پارامتر فوق تعیین نمود.

Row و Range نسبت به هم یک تفاوت عمده دارند،و آن این است که، اگر از ROW Clause استفاده نمایید، ارتباط ROWهای قبلی یا بعدی، نسبت به Row جاری،بصورت فیزیکی (physical association) سنجیده می‌شود، بطوریکه با استفاده از Range Clause ارتباط سطرهای قبلی و بعدی، نسبت به سطر جاری بصورت منطقی (logical association) در نظر گرفته می‌شود. ممکن است درک این مطلب کمی سخت باشد، در ادامه با مثالهایی که بررسی می‌نماییم، براحتی تفاوت این دو را متوجه می‌شوید.

Row یا Range در قالب‌های متفاوتی مقدار می‌پذیرند، که هر کدام را بررسی می‌کنیم:

UNBOUNDED PRECEDING : بیانگر اولین سطر Partition می‌باشد. UNBOUNDED PRECEDING فقط نقطه شروع را مشخص می‌نماید.

UNBOUNDED FOLLOWING : بیانگر آخرین سطر Partition می‌باشد. UNBOUNDED FOLLOWING فقط نقطه پایانی را مشخص می‌نماید.

CURRENT ROW : اولین سطر جاری یا آخرین سطر جاری را مشخص می‌نماید.

n PRECEDING یا <unsigned value specification> PRECEDING : تعداد سطرهای قبل از سطر جاری را تعیین می‌کند، n یا <unsigned value specification> تعداد سطرهای قبل از سطر جاری را تعیین می‌نماید. از n PRECEDING نمی توان برای Range استفاده نمود.

n FOLLOWING یا <unsigned value specification> FOLLOWING : تعداد سطرهای بعد از سطر جاری را تعیین می‌کند، n یا<unsigned value specification> تعداد سطر های بعد از سطر جاری را تعیین می‌نماید. از n FOLLOWING نمی توان برای Range استفاده نمود.

BETWEEN <window frame bound > AND <window frame bound> : از چارچوب فوق برای Range و Row می‌توان استفاده نمود، و نقطه آغازین و نقطه پایانی توسط قالب فوق تعیین می‌گردد. نکته قابل توجه آن است که نقطه پایانی نمی‌تواند، کوچکتر از نقطه آغازین گردد.

در ادامه برای درک هرچه بیشتر تعاریف بیان شده، چندین مثال می‌زنیم و هر کدام را بررسی می‌نماییم:

در ابتدا Script زیر را اجرا نمایید، که شامل جدولی به نام Revenue (سود،درآمد) و درج چند درکورد در آن:

```
CREATE TABLE REVENUE
(
[DepartmentID] int,
[Revenue] int,
[Year] int
);

insert into REVENUE
values (1,10030,1998),(2,20000,1998),(3,40000,1998),
 (1,20000,1999),(2,60000,1999),(3,50000,1999),
 (1,40000,2000),(2,40000,2000),(3,60000,2000),
 (1,30000,2001),(2,30000,2001),(3,70000,2001)
```

مثال اول : می‌خواهیم براساس فیلد DepartmentID جدول Revenue را Partition بندی نماییم و از توابع جمعی AVG و SUM روی فیلد درآمد(Revenue) استفاده کنیم.

ابتدا Script زیر را اجرا می‌کنیم:

```
select *,
 avg(Revenue) OVER (PARTITION by DepartmentID) as AverageRevenue,
 sum(Revenue) OVER (PARTITION by DepartmentID) as TotalRevenue
from REVENUE
order by departmentID, year;
```

خروجی بصورت زیر خواهد بود:

	DepartmentID	Revenue	Year	AverageRevenue	TotalRevenue
1	1	10030	1998	25007	100030
2	1	20000	1999	25007	100030
3	1	40000	2000	25007	100030
4	1	30000	2001	25007	100030
5	2	20000	1998	37500	150000
6	2	60000	1999	37500	150000
7	2	40000	2000	37500	150000
8	2	30000	2001	37500	150000
9	3	40000	1998	55000	220000
10	3	50000	1999	55000	220000
11	3	60000	2000	55000	220000
12	3	70000	2001	55000	220000

مطابق شکل، جدول براساس فیلد DepartmentID به سه Partition تقسیم شده است، و عملیات میانگین و جمع روی فیلد Revenue انجام شده است و عملیات Sort روی هرگروه بطور مستقل انجام گرفته است. چنین کاری را نمی‌توانستیم بوسیله Group By انجام دهیم.

مثال دوم : نحوه استفاده از ROWS PRECEDING،در این مثال قصد داریم عملیات جمع را روی فیلدRevenue انجام دهیم. بطوریکه جمع هر مقدار برابر است با سه مقدار قبلی + مقدار جاری:

لطفا رکوردهای زیر را به جدول فوق درج نمایید:

```
insert into REVENUE
 values(1,90000,2002),(2,20000,2002),(3,80000,2002),
 (1,10300,2003),(2,1000,2003), (3,90000,2003),
 (1,10000,2004),(2,10000,2004),(3,10000,2004),
 (1,20000,2005),(2,20000,2005),(3,20000,2005),
 (1,40000,2006),(2,30000,2006),(3,30000,2006),
 (1,70000,2007),(2,40000,2007),(3,40000,2007),
 (1,50000,2008),(2,50000,2008),(3,50000,2008),
 (1,20000,2009),(2,60000,2009),(3,60000,2009),
 (1,30000,2010),(2,70000,2010),(3,70000,2010),
 (1,80000,2011),(2,80000,2011),(3,80000,2011),
 (1,10000,2012),(2,90000,2012),(3,90000,2012)
```

سپس Script زیر را اجرا می‌نماییم:

```
select Year, DepartmentID, Revenue,
sum(Revenue) OVER (PARTITION by DepartmentID ORDER BY [YEAR]
             ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) as Prev3
From REVENUE order by departmentID, year;
```

خروجی :

	Year	DepartmentID	Revenue	Prev3
1	1998	1	10030	10030
2	1999	1	20000	30030
3	2000	1	40000	70030
4	2001	1	30000	100030
5	2002	1	90000	180000
6	2003	1	10300	170300
7	2004	1	10000	140300
8	2005	1	20000	130300
9	2006	1	40000	80300

در Script بالا، جدول را براساس فیلد DepartmentID گروه بندی می‌کنیم، که سه گروه ایجاد می‌شود، هر گروه را بطور مستقل، روی فیلد Year بصورت صعودی مرتب می‌نماییم. حال برای آنکه بتوانیم سیاست جمع، روی فیلد Revenue، را پیاده سازی نماییم ، قطعه کد زیر را در Script بالا اضافه کردیم.

```
ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) as Prev3
```

برای شرح چگونگی استفاده از PRECEDING،فقط به شرح گروه اول بسنده می‌کنیم. مقدار جمع فیلد Revenue سطر اول، که قبل از آن سطری وجود ندارد، برابر است با مقدار خود، یعنی 10030، مقدار جمع فیلد Revenue سطر دوم برابر است با حاصل جمع مقدار فیلدRevenue سطر اول و دوم ، یعنی 30030 . این روند تا سطر چهار ادامه دارد، اما برای بدست آوردن مقدار جمع فیلدRevenue سطر پنجم، مقدار جمع فیلد Revenue سطر دوم،سوم،چهارم و پنجم در نظر گرفته می‌شود، و مقدار فیلدRevenue سطر اول در حاصل جمع در نظر گرفته نمی‌شود،بنابراین مقدار جمع فیلد Revenue سطر پنجم برابر است با 180000. در صورت مسئله گفته بودیم، مقدار جمع فیلد Revenue هر سطر جاری برابر است با حاصل جمع مقدارسطر جاری و مقادیر سه سطر ماقبل خود.

مثال سوم: نحوه استفاده از ROWS FOLLOWING ، این مثال عکس مثال دوم است، یعنی حاصل جمع مقدار فیلد Revenue هر سطر برابر است با حاصل جمع سطر جاری با سه سطر بعد از خود. بنابراین Script زیر را اجرا نمایید:

```
select Year, DepartmentID, Revenue,
 sum(Revenue) OVER (PARTITION by DepartmentID ORDER BY [YEAR]
              ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING) as Next3
From REVENUE order by departmentID, year;
```

خروجی :

	Year	DepartmentID	Revenue	Next3
1	1998	1	10030	100030
2	1999	1	20000	180000
3	2000	1	40000	170300
4	2001	1	30000	140300
5	2002	1	90000	130300
6	2003	1	10300	80300
7	2004	1	10000	140000
8	2005	1	20000	180000

مطابق شکل مقدار جمع فیلد اول برابراست با حاصل جمع مقدار سطر جاری و سه سطر بعد از آن.

نکته ای که در مثالهای دوم و سوم،می بایست به آن توجه نمود، این است که در زمان استفاده از Row یا Range ، استفاده از Order by در Partition الزامی است، در غیر این صورت با خطا مواجه می‌شوید.

```
select Year, DepartmentID, Revenue,
  sum(Revenue) OVER (PARTITION by DepartmentID --ORDER BY [YEAR]
                     ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING) as Next3
 From REVENUE order by departmentID, year;
```

Messages

```
Msg 10756, Level 15, State 1, Line 2
Window frame with ROWS or RANGE must have an ORDER BY clause.
```

نحوه استفاده از UNBOUNDED PRECEDING ، این امکان در T-SQL Server 2012 افزوده شده است.

مثال چهار: در این مثال می‌خواهیم کمترین سود بدست آمده در چند سال را بدست آوریم:

ابتدا Script زیر را اجرا نمایید:

```
select Year, DepartmentID, Revenue,
       min(Revenue) OVER (PARTITION by DepartmentID ORDER BY [YEAR]
                          ROWS UNBOUNDED PRECEDING) as MinRevenueToDate
From REVENUE order by departmentID, year;
```

خروجی:

Results | Messages

	Year	DepartmentID	Revenue	MinRevenueToDate
1	1998	1	10030	10030
2	1999	1	20000	10030
3	2000	1	40000	10030
4	2001	1	30000	10030
5	2002	1	90000	10030
6	2004	1	10000	10000
7	2005	1	20000	10000
8	2006	1	40000	10000
9	2007	1	70000	10000
10	2008	1	50000	10000
11	2009	1	20000	10000
12	2010	1	30000	10000
13	2011	1	80000	10000
14	2012	1	10000	10000
15	1998	2	20000	20000
16	1999	2	60000	20000

طبق تعریف UNBOUNDED PRECEDING اولین سطر هر Partition را مشخص می‌نماید، و چون از PRECEDING استفاده کرده ایم، بنابراین مقایسه همیشه بین سطر جاری و سطرهای قبل از آن انجام می‌پذیرد. بنابراین خواهیم داشت، کمترین مقدار فیلد Revenue در سطر اول، برابر با مقدار خود می‌باشد، چون هیچ سطری ماقبل از آن وجود ندارد. در سطر دوم مقایسه کمترین مقدار، بین 20000 و 10030 انجام می‌گیرد، که برابر است با 10030، در سطر سوم، مقایسه بین مقادیر سطر اول،دوم و سطر سوم صورت می‌گیرد، یعنی کمترین مقدار بین 20000،40000 و 10030، بنابراین کمترین مقدار سطر سوم برابر است با 10030.
به بیان ساده‌تر برای بدست آوردن کمترین مقدار هر سطر، مقدار سطر جاری با مقادیر همه سطرهای ماقبل خود مقایسه می‌گردد.
برای بدست آوردن کمترین مقدار در سطر ششم، مقایسه بین مقادیر سطرهای اول،دوم،سوم،چهارم،پنجم و ششم صورت می‌گیرد که عدد 10000 بدست می‌آید و الی آخر...
نکته: اگر در Over Clause شرط Order by را اعمال نماییم، اما از Row یا Range استفاده نکنیم، SQL Server بصورت پیش فرض از قالب زیر استفاده می‌نماید:

```
RANGE UNBOUNDED PRECEDING AND CURRENT ROW
```

برای روشن‌تر شدن مطلب فوق مثالی می‌زنیم:
ابتدا Script زیر را اجرا نمایید، که شامل ایجاد یک جدول و درج چند رکورد در آن می‌باشد:

```
CREATE TABLE Employees (
    EmployeeId INT IDENTITY PRIMARY KEY,
    Name VARCHAR(50),
    HireDate DATE NOT NULL,
    Salary INT NOT NULL
)
GO
INSERT INTO Employees (Name, HireDate, Salary)
VALUES
```

```
    ('Alice', '2011-01-01', 20000),
    ('Brent', '2011-01-15', 19000),
    ('Carlos', '2011-02-01', 22000),
    ('Donna', '2011-03-01', 25000),
    ('Evan', '2011-04-01', 18500)
GO
```

سپس Script زیر را اجرا نمایید:

```
SELECT
    Name,
    Salary,
    AVG(Salary) OVER(ORDER BY HireDate) AS avgSalary
FROM Employees
GO
```

خروجی :

	Name	Salary	avgSalary
1	Alice	20000	20000
2	Brent	19000	19500
3	Carlos	22000	20333
4	Donna	25000	21500
5	Evan	18500	20900

حال اگر Script زیر را نیز اجرا نمایید، خروجی آن مطابق شکل بالا خواهد بود:

```
SELECT
    Name,
    Salary,
    AVG(Salary) OVER(ORDER BY HireDate
                  RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS avgSalary
FROM Employees
GO
```

توضیح درباره Script بالا، در این روش برای بدست آوردن میانگین هر سطر، مقدار سطر جاری با مقادیر سطرهای ماقبل خود جمع و تقسیم بر تعداد سطر می‌شود.

سطر دوم 20000 + 19000 تقسیم بر دو برابر است با 19500

میانگین سطر پنجم، حاصل جمع فیلد Salary همه مقادیر سطرها تقسیم بر 5

*** اگر بخواهید بوسیله Over Clause ، میانگین همه سطرها یکسان باشد می‌توانید از Script زیر استفاده نمایید:

```
SELECT
    Name,
    Salary,
    AVG(Salary) OVER(ORDER BY HireDate
                        RANGE
                        BETWEEN UNBOUNDED PRECEDING
                        AND UNBOUNDED FOLLOWING
                    ) AS avgSalary
FROM Employees
GO
```

خروجی :

	Name	Salary	avgSalary
1	Alice	20000	20900
2	Brent	19000	20900
3	Carlos	22000	20900
4	Donna	25000	20900
5	Evan	18500	20900

منظور از ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING یعنی در محاسبه میانگین برای هر سطر تمامی مقادیر سطرهای دیگر در نظر گرفته شود.

پایان بخش اول

امیدوارم مفید واقع شده باشد.

نظرات خوانندگان

نویسنده: ناصر
تاریخ: ۱۳۹۱/۰۹/۱۸ ۰:۲۳

بسیار عالی. فقط خواستم تشکری کرده باشم :)

نویسنده: reza
تاریخ: ۱۳۹۱/۰۹/۱۸ ۰:۳۳

SQLQuery2.sql - (l...Er-PC\na3Er (51))* | SQLQuery1.sql - (l...Er-PC\na3Er (54))* | NA3ER-PC.learn - dbo.REVENUE

```
select Year, DepartmentID, Revenue,
 sum(Revenue) OVER (PARTITION by DepartmentID ORDER BY [YEAR]
              ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) as Prev3
From REVENUE order by departmentID, year;
```

Messages

```
Msg 102, Level 15, State 1, Line 3
Incorrect syntax near 'ROWS'.
```

علت این خطا چیست ؟ آیا این دستور مال 2012 هستش ؟ جون من تو 2008 تست میکنم

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۹/۱۸ ۶:۸

سلام

Syntax ارائه شده Over Clause در SQL Server 2008 شامل دو پارامتر است :

```
Ranking Window Functions
< OVER_CLAUSE > :: =
    OVER ( [ PARTITION BY value_expression , ... [ n ] ]
           <ORDER BY_Clause> )

Aggregate Window Functions
< OVER_CLAUSE > :: =
    OVER ( [ PARTITION BY value_expression , ... [ n ] ] )
```

در SQL Server 2012 پارامتر <ROW or RANGE clause> به Syntax فوق افزوده شد. بنابراین، Query هایی که از Row یا Range

استفاده کرده اند، در SQL Server 2008 با خطا مواجه می‌شوند.

نویسنده: معتمدی
تاریخ: ۱۳۹۱/۰۹/۱۸ ۱۲:۵۹

با سلام

آیا استفاده از این queryها برای محیط‌های transactional هم مناسب است؟ یا بیشتر در databaseهای آماری جهت تهیه گزارشات کاربرد دارد؟

از نظر زمان و هزینه‌ی اجرا و نیز تخصیص منابع سرور به آنها می پرسم.

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۹/۱۸ ۱۴:۵۷

سلام
سرعت و قابلیت اجرایی Over Clause به مراتب از Group by بهتر است. بطوریکه اگر یک عملیات یکسان را،بطور جداگانه،هم با Over Clause و هم با Group By انجام دهید. و در Execution Plan مشاهده نمایید، تفاوت را حس خواهید نمود. سایت زیر یک مثال ساده در این رابطه قرار داده است: Windowing functions - Underappreciated

در مورد اینکه برای محیط های Transactional مناسب است یا نه، عوامل زیادی در آن دخیل است و بسته به حجم داده ای مورد انتظار شما در خروجی دارد،بطور مثال اگر بخواهید یک گزارش 400 صفحه ای ایجاد نمایید، بطور حتم در چنین محیط هایی هیچ Scriptی مناسب، نیست، اما بطور قاطع می‌توان گفت که Window Functionها از کارایی بسیار خوبی برخوردار هستند، و انجام عملیات‌های پیچیده محاسباتی را برای ما آسانتر نموده اند.

نویسنده: فاطمه
تاریخ: ۱۳۹۱/۱۰/۲۳ ۱۷:۲۲

بسیار عالی بود
با تشکر

نویسنده: zarei
تاریخ: ۱۳۹۲/۰۲/۲۱ ۲۲:۱۴

سلام
ممنون از آموزش مفیدتون . سوال من اینه که اگه بخواهیم رکوردهای تکراری حذف بشن باید چیکار کنیم ؟ مثلا من میخام مجموع مبلغ بدهکار برای یک کد کل و در یک سند را در یک ردیف و همین مورد برای مجموع مبلغ بستانکار را نیز در یک رکورد یا ردیف دیگر بدهد . در صورتی که اگر از توابع Over ()Sum استفاده کنیم به ازای هر کد کل درآن سند یک رکورد در خروجی داریم (چه بدهکار و چه بستانکار)

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۲/۰۲/۲۲ ۸:۱۹

سلام
اگر سئوال شما را درست متوجه شده باشم، فکر می‌کنم، می‌بایست از Pivot استفاده نمایید،مقاله زیر می‌تواند در درک Pivot به شما کمک نماید. Pivot

موفق باشید.

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۳/۰۲/۲۱ ۹:۰

با سلام
آیا امکان استفاده از scaler functionها هم هست ؟

ممنون

نویسنده: فرهاد
تاریخ: ۱۳۹۳/۰۲/۲۲ ۱۵:۱۶

سلام
معمولا از توابع Aggregate Functions می‌توان در Window Function استفاده نمود: ^