عنوان: **آشنایی با Window Function ها در SQL Server بخش سوم**
نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۹/۲۴ ۱۹:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, SQL Server 2012, T-SQL

آشنایی با Window Functionها در SQL Server بخش اول
آشنایی با Window Functionها در SQL Server بخش دوم

در این بخش به دو Function از Analytic Functionها (توابع تحلیلی)، یعنی Lead Function و LAG Function می پردازیم.
قبل از اینکه به توابع ذکرشده بپردازیم، باید عرض کنم، شرح عملکرد اینگونه توابع کمی مشکل می‌باشد، بنابراین با ذکر مثال و توضیح آنها،سعی می‌کنیم،قابلیت هریک را بررسی و درک نماییم.

Lead Function:

این فانکشن در SQL Server 2012 ارائه شده است، و امکان دسترسی، به Dataهای سطر بعدی نسبت به سطر جاری را در نتیجه یک پرس و جو (Query)، ارائه می‌دهد. بدون آنکه از Self-join استفاده نمایید،
Syntax تابع فوق بصورت زیر است:

```
LEAD ( scalar_expression [ ,offset ] , [ default ] )
    OVER ( [ partition_by_clause ] order_by_clause )
```

شرح Syntax:
Scalar_expression: در Scalar_expression، نام یک فیلد یا ستون درج می‌شود، و مقدار برگشتی فیلد مورد نظر، به مقدار تعیین شده offset نیز بستگی دارد. خروجی Scalar_expression فقط یک مقدار است.
offset: منظور از Offset در این Syntax همانند عملکرد Offset در Syntax مربوط به Over می‌باشد. یعنی هر عددی برای offset در نظر گرفته شود، بیانگر نقطه آغازین سطر بعدی یا قبلی نسبت به سطر جاری است. به بیان دیگر، عدد تعیین شده در Offset به Sql server می‌فهماند چه تعداد سطر را در محاسبه در نظر نگیرد.
Default: زمانی که برای Offset مقداری را تعیین می‌نمایید، SQL Server به تعداد تعیین شده در Offset، سطرها را در نظر نمی‌گیرد، بنابراین مقدار خروجی Scalar_expression بطور پیش فرض Null در نظر گرفته می‌شود، چنانچه بخواهید، مقداری غیر از Null درج نمایید، می‌توانید مقدار دلخواه را در قسمت Default وارد کنید.
(OVER ([partition_by_clause] order_by_clause : در بخش اول بطور کامل توضیح داده شده است.

برای درک بهتر Lead Function چند مثال را بررسی می‌نماییم:
ابتدا Script زیر را اجرا می‌نماییم، که شامل ایجاد یک جدول و درج 18 رکورد در آن:

```
Create Table TestLead_LAG
(SalesOrderID int not null,
 SalesOrderDetailID int not null ,
 OrderQty smallint not null);
 GO
Insert Into TestLead_LAG
       Values (43662,49,1),(43662,50,3),(43662,51,1),
          (43663,52,1),(43664,53,1),(43664,54,1),
  (43667,77,3),(43667,78,1),(43667,79,1),
  (43667,80,1),(43668,81,3),(43669,110,1),
  (43670,111,1),(43670,112,2),(43670,113,2),
  (43670,114,1),(43671,115,1),(43671,116,2)
```

مثال:قصد داریم در هر سطر مقدار بعدی فیلد SalesOrderDetailID در فیلد دیگری به نام LeadValue نمایش دهیم، بنابراین Script زیر را ایجاد می‌کنیم:

```
SELECT s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty,
LEAD(SalesOrderDetailID) OVER (ORDER BY SalesOrderDetailID) LeadValue
FROM TestLead_LAG s
WHERE SalesOrderID IN (43670, 43669, 43667, 43663)
```

```
ORDER BY s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty
```

خروجی بصورت زیر خواهد بود:

	SalesOrderID	SalesOrderDetailID	OrderQty	LeadValue
1	43663	52	1	77
2	43667	77	3	78
3	43667	78	1	79
4	43667	79	1	80
5	43667	80	1	110
6	43669	110	1	111
7	43670	111	1	112
8	43670	112	2	113
9	43670	113	2	114
10	43670	114	1	NULL

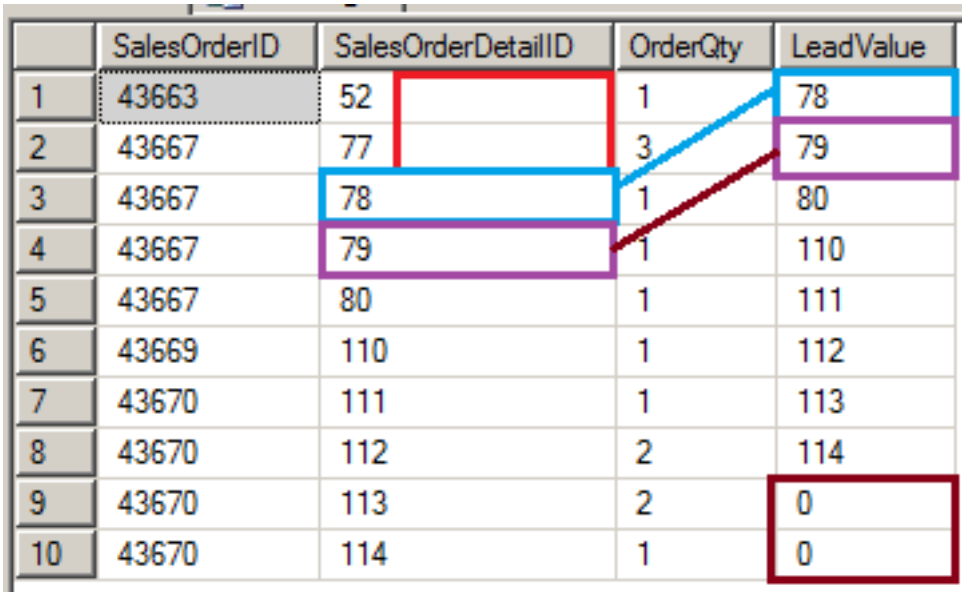
مطابق شکل، براحتی واضح است، که در هر سطر مقدار بعدی فیلد SalesOrderDetailID در فیلد LeadValue درج و نمایش داده می‌شود. فقط در سطر 10، چون مقدار بعدی برای فیلد SalesOrderDetailID وجود ندارد، SQL Server مقدار فیلد LeadValue را، Null در نظر می‌گیرد.

در این مثال فقط از آرگومان Scalar_expression، استفاده کردیم، و Offset و Default را مقدار دهی ننمودیم، بنابراین SQL Server بطور پیش فرض هیچ سطری را حذف نمی‌کند و مقدار Default را Null در نظر می‌گیرد.

مثال دوم: قصد داریم در هر سطر مقدار دو سطر بعدی فیلد SalesOrderDetailID را در فیلد LeadValue نمایش دهیم، و در صورت وجود نداشتن مقدار فیلد SalesOrderDetailID، مقدار پیش فرض صفر ،در فیلد LeadValue قرار دهیم،بنابراین Script آن بصورت زیر خواهد شد:

```
SELECT s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty,
LEAD(SalesOrderDetailID,2,0) OVER (ORDER BY SalesOrderDetailID) LeadValue
FROM TestLead_LAG s
WHERE SalesOrderID IN (43670, 43669, 43667, 43663)
ORDER BY s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty
```

خروجی:



	SalesOrderID	SalesOrderDetailID	OrderQty	LeadValue
1	43663	52	1	78
2	43667	77	3	79
3	43667	78	1	80
4	43667	79	1	110
5	43667	80	1	111
6	43669	110	1	112
7	43670	111	1	113
8	43670	112	2	114
9	43670	113	2	0
10	43670	114	1	0

در صورت مسئله بیان کرده بودیم، در هر سطر،مقدار فیلد SalesOrderDetailID دو سطر بعدی، را نمایش دهیم، بنابراین مقداری که برای Offset در نظر می‌گیریم، برابر دو خواهد بود، سپس گفته بودیم، چنانچه در هر سطر مقدار فیلد SalesOrderDetailID وجود نداشت،بجای مقدار پیش فرض Null،از مقدار صفر استفاده شود، بنابراین به Default مقدار صفر را نسبت دادیم.

```
LEAD(SalesOrderDetailID,2,0)
```

در شکل، مطابق صورت مسئله، مقدار فیلد LeadValue سطر اول برابر است با 78،
به بیان ساده‌تر برای بدست آوردن مقدار فیلد LaedValue هر سطر، می‌بایست هر سطر را به علاوه 2 (Offset) نماییم، تا سطر بعدی بدست آید، سپس مقدار SalesOrderDetailID را در فیلد LeadValue قرار می‌دهیم.
به سطر 9 و 10 توجه نمایید، که مقدار فیلد LeadValue آنها برابر با صفر است، واضح است، سطر 10 + 2 برابر است با 12(10+2=12)، چنین سطری در خروجی نداریم، بنابراین بطور پیش فرض مقدار LeadVaule توسط Sql Server برابر Null در نظر گرفته می‌شود، اما نمی‌خواستیم، که این مقدار Null باشد، بنابراین به آرگومان Default مقدار صفر را نسبت دادیم، تا SQL Server ، به جای استفاده از Null، مقدار در نظر گرفته شده صفر را استفاده نماید.
اگر چنین فانکشنی وجود نداشت، برای شبیه سازی آن می‌بایست از Join روی خود جدول استفاده می‌نمودیم، و یکسری محاسابت دیگر، که کار را سخت می‌نمود، مثال دوم را با Script زیر می‌توان شبیه سازی نمود:

```
WITH cteLead
AS
(
SELECT SalesOrderID,SalesOrderDetailID,OrderQty,
       ROW_NUMBER() OVER (ORDER BY SalesOrderDetailID) AS sn
FROM TestLead_LAG
WHERE
SalesOrderID IN (43670, 43669, 43667, 43663)
)
SELECT m.SalesOrderID, m.SalesOrderDetailID, m.OrderQty,
       case  when sLead.SalesOrderDetailID is null Then 0 Else sLead.SalesOrderDetailID END as
leadvalue
FROM cteLead AS m
LEFT OUTER JOIN cteLead AS sLead ON sLead.sn = m.sn+2
ORDER BY m.SalesOrderID, m.SalesOrderDetailID, m.OrderQty
```

جدول موقتی ایجاد نمودیم، که ROW_Number را در آن اضافه کردیم، سپس جدول ایجاد شده را با خود Join کردیم، و گفتیم، که مقدار فیلدLeadValue هر سطر برابر است با مقدار فیلد SalesOrderDetailID دو سطر بعد از آن. و با Case نیز مقدار پیش فرض را صفر در نظر گرفتیم.

LAG Function:

این فانکشن نیز در SQL Server 2012 ارائه شده است، و امکان دسترسی، به Dataهای سطر قبلی نسبت به سطر جاری را در نتیجه یک پرس و جو (Query)، ارائه می‌دهد. بدون آنکه از Self-join استفاده نمایید،

Syntax آن شبیه به فانکشن Lead میباشد و بصورت زیر است:

```
LAG (scalar_expression [,offset] [,default])
    OVER ( [ partition_by_clause ] order_by_clause )
```

Syntax مربوط به فانکشن LAG را شرح نمی‌دهم، بدلیل آنکه شبیه به فانکشن Lead می‌باشد، فقط تفاوت آن در Offset است، Offset در فانکشن LAG روی سطرهای ماقبل سطر جاری اعمال می‌گردد.

مثال دوم را برای حالت LAG Function شبیه سازی می‌نماییم:

```
SELECT s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty,
LAG(SalesOrderDetailID,2,0) OVER (ORDER BY SalesOrderDetailID) LAGValue
FROM TestLead_LAG s
WHERE SalesOrderID IN (43670, 43669, 43667, 43663)
ORDER BY s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty
go
```

خروجی :

	SalesOrderID	SalesOrderDetailID	OrderQty	LAGValue
1	43663	52	1	0
2	43667	77	3	0
3	43667	78	1	52
4	43667	79	1	77
5	43667	80	1	78
6	43669	110	1	79
7	43670	111	1	80
8	43670	112	2	110
9	43670	113	2	111
10	43670	114	1	112

همانطور که گفتیم، LAG Function عکس LEAD Function میباشد. یعنی مقدار فیلد LAGValue سطر جاری برابر است با مقدار SalesOrderDetailID دو سطر ما قبل خود.

مقدار فیلد LAGValue دو سطر اول و دوم نیز برابر صفر است، چون دو سطر ماقبل آنها وجود ندارد، و مقدار صفر نیز بدلیل این است که Default را برابر صفر در نظر گرفته بودیم.

مثال: در این مثال از Laed Function و LAG Function بطور همزمان استفاده می‌کنیم، با این تفاوت، که از گروه بندی نیز استفاده شده است:

Script زیر را اجرا نمایید:

```
SELECT s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty,
       Lead(SalesOrderDetailID) OVER (PARTITION BY SalesOrderID ORDER BY SalesOrderDetailID) LeadValue,
       LAG(SalesOrderDetailID) OVER (PARTITION BY SalesOrderID ORDER BY SalesOrderDetailID) LAGValue
FROM TestLead_LAG s
WHERE SalesOrderID IN (43670, 43669, 43667, 43663)
ORDER BY s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty
go
```

خروجی:

	SalesOrderID	SalesOrderDetailID	OrderQty	LeadValue	LAGValue
1	43663	52	1	NULL	NULL
2	43667	77	3	78	NULL
3	43667	78	1	79	77
4	43667	79	1	80	78
5	43667	80	1	NULL	79
6	43669	110	1	NULL	NULL
7	43670	111	1	112	NULL
8	43670	112	2	113	111
9	43670	113	2	114	112
10	43670	114	1	NULL	113

با بررسی هایی که در مثالهای قبل نمودیم،خروجی زیر را می‌توان براحتی تشخیص داد، و توضیح بیشتری نمی‌دهم.
موفق باشید.

نظرات خوانندگان

نویسنده: محمد
تاریخ: ۱۳۹۱/۱۰/۲۸ ۱۸:۵۳

سلام،
این توابع واقعا کار رو آسون کردن، ما رو از بکارگیری چندین بار self join بی نیاز کردن.
بطور نمونه اگه بخواهیم مقدار SalesOrderDetailID سطر قبلی، دو سطرقبلی، سطربعدی و دو سطر بعدی را بدست بیاریم در نسخه 2008 ساده‌ترین و مناسب‌ترین کوئری این هست:

```
WITH cteLead
AS
(
SELECT SalesOrderID,SalesOrderDetailID,OrderQty,
       ROW_NUMBER() OVER (PARTITION BY SalesOrderID
                          ORDER BY SalesOrderDetailID) AS sn
FROM TestLead_LAG
WHERE
SalesOrderID IN (43670, 43669, 43667, 43663)
)
SELECT m.SalesOrderID, m.SalesOrderDetailID, m.OrderQty,
       COALESCE(sLead1.SalesOrderDetailID, 0) as leadvalue1,
       COALESCE(sLead2.SalesOrderDetailID, 0) as leadvalue2,
       COALESCE(sLag1.SalesOrderDetailID, 0) as lagvalue2,
       COALESCE(sLag2.SalesOrderDetailID, 0) as lagvalue2
FROM cteLead AS m
LEFT OUTER JOIN cteLead AS sLead1
ON m.sn = sLead1.sn - 1
AND m.SalesOrderID = sLead1.SalesOrderID
LEFT OUTER JOIN cteLead AS sLead2
ON m.sn = sLead2.sn - 2
AND m.SalesOrderID = sLead2.SalesOrderID
LEFT OUTER JOIN cteLead AS sLag1
ON m.sn = sLag1.sn + 1
AND m.SalesOrderID = sLag1.SalesOrderID
LEFT OUTER JOIN cteLead AS sLag2
ON m.sn = sLag2.sn + 2
AND m.SalesOrderID = sLag2.SalesOrderID
ORDER BY m.SalesOrderID, m.SalesOrderDetailID, m.OrderQty;
```

در حالی که با دو تابعی که شما در اینجا پوشش دادین میشه کوئری فوق را فوق العاده ساده‌تر نمود:

```
SELECT s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty,
       Lead(SalesOrderDetailID, 1, 0) OVER (PARTITION BY SalesOrderID ORDER BY SalesOrderDetailID)
LeadValue1,
       LAG(SalesOrderDetailID, 1, 0) OVER (PARTITION BY SalesOrderID ORDER BY SalesOrderDetailID)
LAGValue1,
       Lead(SalesOrderDetailID, 2, 0) OVER (PARTITION BY SalesOrderID ORDER BY SalesOrderDetailID)
LeadValue2,
       LAG(SalesOrderDetailID, 2, 0) OVER (PARTITION BY SalesOrderID ORDER BY SalesOrderDetailID)
LAGValue2,
FROM TestLead_LAG s
WHERE SalesOrderID IN (43670, 43669, 43667, 43663)
ORDER BY s.SalesOrderID,s.SalesOrderDetailID,s.OrderQty
```

نویسنده: حسین
تاریخ: ۱۳۹۱/۱۰/۳۰ ۱۴:۵۸

جالب بود مرسی