**بسمه تعالی**



برنامه نویسی بانک اطلاعاتی SQL Server

**نویسنده : محمد حسین فخرآوری**

**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 Function ‌ها می‌توان، ROW‌ های 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.

کلمه "Window" در  Window Function، به مجموعه ROW هایی اشاره می‌کند، که محاسبات و ارزیابی و غیرو... روی آنها اعمال می‌گردد.

* Window Function‌ ها برای ارائه قابلیت‌های خود، از Over Clause استفاده می‌کنند

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 Subquery ، Scalar Function 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.

Rowو 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 FOLLOWINGیا :< 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.

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

* [Ranking functions](https://www.dotnettips.info/Post/968/%D8%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-row_number-rank-dense_rank-ntile) (توابع رتبه بندی)
* [NEXT VALUE FOR](https://www.dotnettips.info/Post/1067/%D9%86%D8%AD%D9%88%D9%87-%D8%A7%DB%8C%D8%AC%D8%A7%D8%AF-sequence-%D9%88-%D8%A7%D8%B3%D8%AA%D9%81%D8%A7%D8%AF%D9%87-%D8%A2%D9%86-%D8%AF%D8%B1-sql-server-2012) ، در بحث ایجاد Sequence
* [Aggregate Functions](http://msdn.microsoft.com/en-us/library/ms173454.aspx) (توابع جمعی)
* Analytic Functions (توابع تحلیلی)

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:

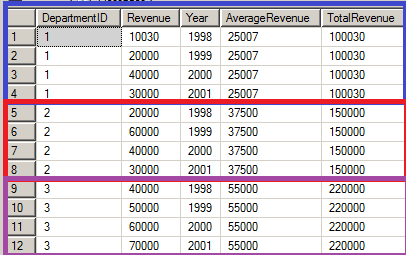
|  |
| --- |
| CREATE TABLE REVENUE ([DepartmentID] int,[Revenue] int,[Year] int);  insert into REVENUE  values (1,10030,1998),(2,20000,1998),(3,40000,1998),  (1,20000,1999),(2,60000,1999),(3,50000,1999),  (1,40000,2000),(2,40000,2000),(3,60000,2000),  (1,30000,2001),(2,30000,2001),(3,70000,2001) |

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:

[](https://www.dotnettips.info/File/Image?name=1-ae00177a0962406a9221d019af74f19a.png)

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 فیلد Revenue انجام دهیم. بطوریکه جمع هر مقدار برابر است با سه مقدار قبلی + مقدار جاری:

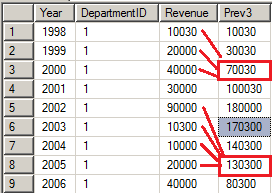
 لطفا رکورد‌های زیر را به جدول فوق درج نمایید:

|  |
| --- |
| 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 :

[](https://www.dotnettips.info/File/Image?name=1-ff53181f47a04b51a794d097bcfcbdc1.png)

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.

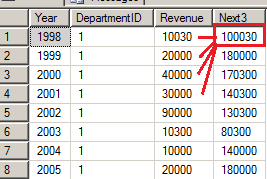
|  |
| --- |
| ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) as Prev3 |

برای شرح چگونگی استفاده ازPRECEDING، 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:

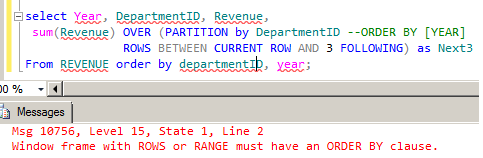
|  |
| --- |
| select Year, DepartmentID, Revenue,  sum(Revenue) OVER (PARTITION by DepartmentID ORDER BY [YEAR]  ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING) as Next3  From REVENUE order by departmentID, year; |

خروجی :

[](https://www.dotnettips.info/File/Image?name=1-96d8fb35c69f498baf557aabf9a9a10a.png)

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.

[](https://www.dotnettips.info/File/Image?name=1-72bfb3c510834535ad90e3b542bed219.png)

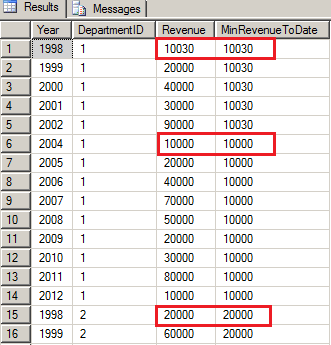
نحوه استفاده از UNBOUNDED PRECEDING ، این امکان در T-SQL Server 2012 افزوده شده است.

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:

[](https://www.dotnettips.info/File/Image?name=1-0eb9467d53a14f05b66549fd0fd08e7c.png)

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 40000،20000 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...

نکنه: 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:

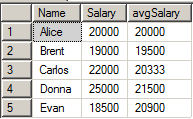
ابتدا Script زیر را اجرا نمایید، که شامل ایجاد یک جدول و درج چند رکورد در آن می‌باشد:

|  |
| --- |
| 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 :

[](https://www.dotnettips.info/File/Image?name=1-1a45c1b8bf7747829e1b120a83e2ba24.png)

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.

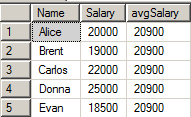
سطر دوم 20000 + 19000 تقسیم بر دو برابر است با 19500

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 :

[](https://www.dotnettips.info/File/Image?name=1-cc25ff761f5642ad80031df78735e00b.png)

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.

**Ranking Function**

گروهی از توابع در اس کیو ال سرور هستند که برای رتبه بندی یا Ranking مورد استفاده قرار میگیرند به این توابع Ranking Functions می گویند. توابع رنکینگ مقداری را بعنوان رتبه(RANK) به هر رکورد موجود در نتایج اختصاص میدهند.

**جدولی جهت تست**

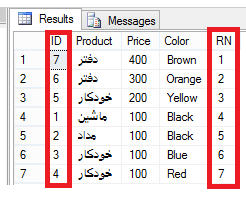
|  |
| --- |
| CREATE TABLE Test (ID INT IDENTITY(1,1), Product NVARCHAR(100), Price INT, Color NVARCHAR(100))  GO  SET IDENTITY\_INSERT [dbo].[Test] ON  INSERT [dbo].[Test] ([ID], [Product], [Price], [Color]) VALUES (1, N'ماشین', 100, N'Black')  INSERT [dbo].[Test] ([ID], [Product], [Price], [Color]) VALUES (2, N'مداد', 100, N'Black')  INSERT [dbo].[Test] ([ID], [Product], [Price], [Color]) VALUES (3, N'خودکار', 100, N'Blue')  INSERT [dbo].[Test] ([ID], [Product], [Price], [Color]) VALUES (4, N'خودکار', 100, N'Red')  INSERT [dbo].[Test] ([ID], [Product], [Price], [Color]) VALUES (5, N'خودکار', 200, N'Yellow')  INSERT [dbo].[Test] ([ID], [Product], [Price], [Color]) VALUES (6, N'دفتر', 300, N'Orange')  INSERT [dbo].[Test] ([ID], [Product], [Price], [Color]) VALUES (7, N'دفتر', 400, N'Brown')  SET IDENTITY\_INSERT [dbo].[Test] OFF |

### **ROW\_NUMBER**

این تابع یک سری اعداد متوالی را برطبق انچه در عبارت بعدی آن ذکر میشود باز میگرداند .

|  |
| --- |
| Select \*, ROW\_NUMBER() OVER (ORDER BY Price DESC) AS RN from Test |
| Select \*, ROW\_NUMBER() OVER (ORDER BY Price ASC) AS RN from Test |

خروجی آن بصورت زیر می‌باشد

اگر به Syntax تابع Row\_Number توجه نماییم، با کلمه Partition مواجه می‌شویم،که جهت گروه بندی استفاده می‌شود، به عبارت دیگر ممکن است شما بخواهید، ابتدا جدول خود را براساس فیلد یا فیلدهایی Group by نمایید و سپس روی آنها Row\_Number را اعمال کنید، که در این حالت از Partition استفاده می‌شود.

|  |
| --- |
| Select \*,ROW\_NUMBER() OVER (PARTITION BY Product ORDER BY Price DESC) AS RN from Test |
| Select \*,ROW\_NUMBER() OVER (PARTITION BY Product ORDER BY Price ASC) AS RN from Test |

### **RANK**

این تابع امتیازی از ۱ تا الی آخر به هر سطر از خروجی دستور SELECT ما می دهد. اگر دو سطر دارای مقادیر برابر باشند در این صورت مقدار تابع Rank برای هر دو سطر برابر خواهد بود

|  |
| --- |
| Select \*,RANK() over (ORDER BY Price ASC) AS RANK from Test |
| Select \*,RANK() over (ORDER BY Price DESC) AS RANK from Test |

در Syntax تابع Rank نیز کلمه Partition هم وجود دارد، که در جهت Group by فیلد یا فیلدهای خاصی استفاده می‌شود، و رتبه بندی نیز در این حالت روی Group by انجام می‌گردد.

برای درک بهتر Query زیر را اجرا نمایی:

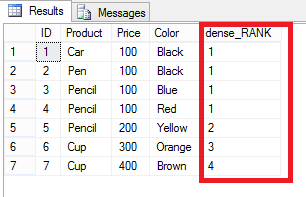
|  |
| --- |
| Select \*,RANK() over (Partition by Product ORDER BY Price ASC) AS RANK from Test |

### **DENSE\_RANK**

عملکرد این تابع دقیقا همانند تابع RANK() می باشد، با این تفاوت که مقادیر مربوط به ستون DENSE\_RANK() به صورت مرتب به سطرها اختصاص داده می شود

|  |
| --- |
| Select \*,dense\_RANK() over (ORDER BY Price ) AS dense\_RANK from Test |

خروجی به صورت زیر می باشد



### **(NTILE(n**

این تابع برای دسته بندی کردن اطلاعات استفاده می شود. تعداد دسته ها به صورت یک پارامتر به این تابع داده می شود.

|  |
| --- |
| Select \* ,NTILE(4) over ( ORDER BY Price desc) from Test |