Pragmatic Works

A Window into Your Data

Using SQL Window Functions

PASS SQL saturday

#453 | MINNESOTA 2015

Welcome to PASS SQL Saturday #435

- Thank you Sponsors!
  - Please visit the sponsors during the vendor break from 2:45 – 3:15 and enter their end-of-day raffles
- Event After Party
  - Dave and Buster's in Southdale Center. 3rd floor by Macy's starting at 6:15
- Want More Free Training?
  - PassMN meets the 3rd Tuesday of every month. https://mnssug.org/

Pragmatic Works

A Window into Your Data

Using SQL Window Functions

PASS SQL saturday

#453 | MINNESOTA 2015

Pragmatic Works

Steve Hughes
Solutions Architect
shughes@pragmaticworks.com
www.dataonwheels.com
@dataonwheels

DATA on WHEELS

Agenda

SQL Window Functions Overview

SQL Server 2008 Functions

SQL Server 2012 Functions

Wrap Up

What Are SQL Window Functions & Why Should We Care?

Introduced in 2005, expanded in 2012

Used for grouping data for use at the row level

Allow for solving complex business questions

- % of customer's total sales
- Running total
- Next and previous row values

Does not require GROUP BY or subselect queries

Structure of the OVER Clause

OVER – used to define the window over which the function will be applied. The default window is the entire set.

- PARTITION BY – groups the data by one or more columns
- ORDER BY – sorts the data in the partition
- ROWS or RANGE – further reduces the size of the window based on proximity from the current row

```
OVER([PARTITION BY <<field list>>]
[ORDER BY <<field list>>]
[ROWS/RANGE <<set of rows>> ])
```

SQL 2008 Functionality

Ranking window functions

```
OVER([PARTITION BY value_expression, … [n]]
<ORDER BY_Clause>)
```

- ROW_NUMBER
- DENSE_RANK
- RANK
- NTILE

Aggregate window functions

```
OVER([PARTITION BY value_expression, … [n]])
```

- Most aggregate function, e.g. SUM, COUNT, AVG
  - Does not support GROUPING or GROUPING_BY aggregations
- Does not support ORDER BY

SQL 2008 Window Illustrated

OrderID	OrderDate	OrderAmt
1	3/1/2012	$10.00
2	3/1/2012	$11.00
3	3/2/2012	$10.00
4	3/2/2012	$15.00
5	3/2/2012	$17.00
6	3/3/2012	$12.00
7	3/4/2012	$10.00
8	3/4/2012	$18.00
9	3/4/2012	$12.00

Aggregation Window – DailyTotal:
SUM(OrderAmt) OVER (PARTITION BY OrderDate)

PARTITION BY OrderDate

OrderID	DailyTotal	DailyRank
3	$42.00	3
4	$42.00	2
5	$42.00	1

Ranking Window – DailyRank:
RANK() OVER (PARTITION BY OrderDate
ORDER BY OrderAmt DESC)

SQL 2008 SQL Comparison: SubSelect vs Window Function

```
select
(select sum(OrderAmt) from CTEOrders o where
o.OrderDate = CTEOrders.OrderDate)
as DailyOrderAmt
from CTEOrders

select SUM(OrderAmt) OVER (PARTITION BY OrderDate)
from CTEOrders
order by OrderID
```

Why Not Group By?

```
select OrderDate ,sum(OrderAmt) as DailyOrderAmt
from CTEOrders
group by OrderDate
```

This returns the amount and is the subselect logic

```
select OrderDate
       ,OrderID
       ,OrderAmt
       ,sum(OrderAmt) as DailyOrderAmt
from CTEOrders
group by OrderDate
       ,OrderID
       ,OrderAmt
```

This returns the details, but the sum is not the correct

One Twist on Group By

```
select sum(OrderAmt)
, sum(OrderAmt) over() as TotalOrderAmt
from CTEOrders
group by CustomerName
```

This returns an error because the first expression is an aggregate, but not the second

Also, OVER() is evaluated over the full set

```
select sum(OrderAmt)
, sum(sum(OrderAmt)) over() as TotalOrderAmt
from CTEOrders
group by CustomerName
```

This returns the aggregate of the aggregates and does not return an error

Thanks to Itzik Ben-Gan for this tip

SQL 2008 Demos

Group By and SubSelect

Ranking Windows

Aggregate Windows

Twisted Group By

SQL 2012 Functionality

Significantly expanded in this release

- Greater ANSI compliance
- Matches Oracle 11g

ORDER BY clause supported by all function types, not just ranking functions

ROWS and RANGE clauses added to OVER clause

- This allows for running totals and other hard to resolve problems

New analytic functions including:

- Position values (e.g. LAG, LAST_VALUE)
- Percentile values and rank (e.g. PCT_RANK, PERCENTILE_DISC)

SQL 2012 Window Illustrated (1)

2008 Functionality

OrderID	OrderDate	OrderAmt
1	3/1/2012	$10.00
2	3/1/2012	$11.00
3	3/2/2012	$10.00
4	3/2/2012	$15.00
5	3/2/2012	$17.00
6	3/3/2012	$12.00
7	3/4/2012	$10.00
8	3/4/2012	$18.00
9	3/4/2012	$12.00

Aggregation Window – DailyTotal:

SUM(OrderAmt) OVER (PARTITION BY OrderDate)

PARTITION BY OrderDate

OrderID	DailyTotal	DailyRank
3	$42.00	3
4	$42.00	2
5	$42.00	1

Ranking Window – DailyRank:

RANK() OVER (PARTITION BY OrderDate
ORDER BY OrderAmt DESC)

SQL 2012 Window Illustrated (2)

Aggregate with Order By Functionality

Aggregate Window – Running Total by Customer:
SUM(OrderAmt) OVER (PARTITION BY CustomerName **ORDER BY** OrderDate, OrderID)

OrdID	OrdDt	OrdAmt	Cust
1	3/1/2012	$10.00	Joe
2	3/1/2012	$11.00	Sam
3	3/2/2012	$10.00	Beth
4	3/2/2012	$15.00	Joe
5	3/2/2012	$17.00	Sam
6	3/3/2012	$12.00	Joe
7	3/4/2012	$10.00	Beth
8	3/4/2012	$18.00	Sam
9	3/4/2012	$12.00	Joe

PARTITION BY Cust

Cust	OrdDt	OrdID	RunTot
Sam	3/1/2012	2	$11.00
Sam	3/2/2012	5	$28.00
Sam	3/4/2012	8	$46.00

SQL 2012 Window Illustrated (2)

Aggregate with Order By Functionality

```
select OrderID
,OrderDate
,OrderAmt
,CustomerName
,SUM(OrderAmt)
     OVER (PARTITION BY CustomerName
                ORDER BY OrderDate, OrderID)
as RunningByCustomer
from CTEOrders
```

SQL 2012 Window Illustr

Aggregate with Rows Functionality

Aggregate Rows Window – Next 2 Orders by Customer:

SUM(OrderAmt) OVER (PARTITION BY CustomerName ORDER BY OrderDate, OrderID ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING)

OrdID	OrdDt	OrdAmt	Cust
1	3/1/2012	$10.00	Joe
2	3/1/2012	$11.00	Sam
3	3/2/2012	$10.00	Beth
4	3/2/2012	$15.00	Joe
5	3/2/2012	$17.00	Sam
6	3/3/2012	$12.00	Joe
7	3/4/2012	$10.00	Beth
8	3/4/2012	$18.00	Sam
9	3/4/2012	$12.00	Joe

PARTITION BY Cust

Cust	OrdDt	OrdID	NextTwo
Joe	3/1/2012	1	$27.00
Joe	3/2/2012	4	$24.00
Joe	3/3/2012	6	$12.00
Joe	3/4/2012	9	NULL

SQL 2012 Window Illustrated (3)

Aggregate with Rows Functionality

```
select OrderID
,OrderDate
,OrderAmt
,CustomerName
,SUM(OrderAmt)
      OVER (PARTITION BY CustomerName
                  ORDER BY OrderDate, OrderID
                  ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING)
as NextTwoAmts
from CTEOrders
```

ROWS clause sets the window based on physical proximity to current row.

SQL 2012 Window Illustr

Aggregate with Range Functionality

OrdID	OrdDt	OrdAmt	Cust
1	3/1/2012	$10.00	Joe
2	3/1/2012	$11.00	Sam
3	3/2/2012	$10.00	Beth
4	3/2/2012	$15.00	Joe
5	3/2/2012	$17.00	Sam
6	3/3/2012	$12.00	Joe
7	3/4/2012	$10.00	Beth
8	3/4/2012	$18.00	Sam
9	3/4/2012	$12.00	Joe

PARTITION BY Cust

Aggregate Range Window – Running Total by Customer:

```
SUM(OrderAmt)
OVER (PARTITION BY CustomerName
ORDER BY OrderDate, OrderID RANGE
BETWEEN UNBOUNDED PRECEDING and
CURRENT ROW)
```

Cust	OrdDt	OrdID	Running
Joe	3/1/2012	1	$10.00
Joe	3/2/2012	4	$25.00
Joe	3/3/2012	6	$37.00
Joe	3/4/2012	9	$49.00

SQL 2012 Window Illustrated (4)

Aggregate with Range Functionality

```
select OrderID
,OrderDate
,OrderAmt
,CustomerName
,SUM(OrderAmt)
      OVER (PARTITION BY CustomerName
                  ORDER BY OrderDate, OrderID
                  RANGE BETWEEN UNBOUNDED PRECEDING
                        AND CURRENT ROW) as RunningTotalAmt
from CTEOrders
```

The results are the same as without the RANGE clause. This RANGE clause represents the default window when the ORDER BY is specified, but RANGE is not.

More Detail on ROWS and RANGE

- UNBOUNDED key word is supported by both ROWS and RANGE
- ROWS function only supports the numbered FOLLOWING or PRECEDING
  - e.g. 1 FOLLOWING will only work with ROWS
- CURRENT ROW can be specified alone with different results
  - ROWS – only the current row is included in the partition, thus no aggregation
  - RANGE – honors the partition and order definitions to create a range based on the current row's definition

SQL 2012 Window Illustrated (5)

Analytic Functionality

OrdID	OrdDt	OrdAmt	Cust
1	3/1/2012	$10.00	Joe
2	3/1/2012	$11.00	Sam
3	3/2/2012	$10.00	Beth
4	3/2/2012	$15.00	Joe
5	3/2/2012	$17.00	Sam
6	3/3/2012	$12.00	Joe
7	3/4/2012	$10.00	Beth
8	3/4/2012	$18.00	Sam
9	3/4/2012	$12.00	Joe

Previous Row – Previous Order Amount:

LAG(OrderAmt)
OVER (ORDER BY OrderDate, OrderID)

Sample Rows

Cust	Prev Amt	Next Amt
Beth	$11.00	$10.00
Joe	$10.00	$12.00
Sam	$15.00	$18.00

Next Row – Next Customer Order Amount:

LEAD(OrderAmt) OVER (PARTITION BY CustomerName
ORDER BY OrderDate, OrderID)

SQL 2012 Window Illustrated (5)

Analytic Functionality

```
select OrderID
,OrderDate
,OrderAmt
,CustomerName
,LAG(OrderAmt)
      OVER (ORDER BY OrderDate, OrderID)
      as PrevOrderAmt
,LEAD(OrderAmt)
      OVER (PARTITION BY CustomerName
                  ORDER BY OrderDate, OrderID)
      as NextOrderAmtForCustomer
from CTEOrders
```

SQL 2012 Demos

Ranking Windows

Aggregate Windows

New SQL 2012 Window Functionality

- Aggregate with Order
- Aggregate with ROW
- Aggregate with RANGE
- Analytic Functions

Wrap Up – Final Thoughts

- OVER() – defaults to the entire set
- Each refinement to the OVER clause changes the set of data that the window function is applied to, even ORDER BY
- Both the PARTITION BY and ORDER BY clauses support multiple columns
- Now supported in Azure SQL Database

Wrap Up

More Information

MSDN

- OVER Clause: http://msdn.microsoft.com/en-us/library/ms189461(v=SQL.110).aspx
- Analytic Functions: http://msdn.microsoft.com/en-us/library/hh213234(v=sql.110).aspx

Supporting Series on DataOnWheels

- http://dataonwheels.wordpress.com/category/t-sql/

Questions?

Wrap Up Part 2

- Remember to fill out your online evaluations for the event and any sessions you have attended. They will be online until 10/17/15.

http://www.sqlsaturday.com/453/eventeval.aspx
http://www.sqlsaturday.com/453/sessionevaluation.aspx