

## **Set Operations and Built In Functions**



Pivotal® Greenplum  
Database

Pivotal.

© 2013 Pivotal Software, Inc. All rights reserved.

1

## Agenda

- Set Based Operations
- Grouping, Cubes and Rollup
- Built In Functions
- References And Review

Pivotal

Hello, this is Marshall Presser. In this section, we'll be talking about how you can use some important functions that are part of the Pivotal Greenplum Database to help solve analytic business problems. Some come with the database itself and others are easily installed.

First, we'll consider set based operations and grouping operators.  
Then we'll look at some built in functions and do a brief demo

Next we'll look at the machine learning functions that come with Madlib, a package of very useful functions that make doing Data Science in GPDB a lot easier

We'll also look at PostGIS, the PostgreSQL geospatial database functions and have another brief demo.

.

If you're used to OLTP operations in a SQL database, you probably think more in terms of inserts, deletes, updates, foreign key constraints, triggers and other such operations.

While GPDB has triggers, update statements and the like, its main use is for analytics in which much or all of the data needs to be examined, not just a single row or a few rows.

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

3

## Set Operations

Greenplum supports the following set operations as part of a `SELECT` statement:

- `UNION` – Returns a combination of rows from multiple `SELECT` statements with no repeating rows
- `UNION ALL` – Returns a combination of rows from multiple `SELECT` statements with repeating rows
- `INTERSECT` – Returns rows that appear in all answer sets
- `EXCEPT/MINUS` – Returns rows from the first answer set and excludes those from the second

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

4

Set operators:

- Manipulate the results sets of two or more queries by combining the results of individual queries into a single results set.
- Do not perform row level filtering.

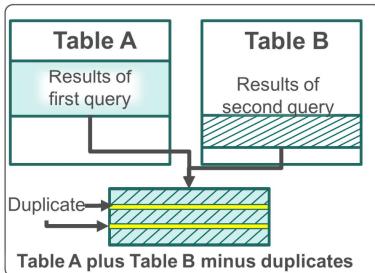
Set operations supported by Greenplum are:

- `INTERSECT` which returns rows that appear in all answer sets generated by individual `SELECT` statements.
- `EXCEPT` returns all rows from the first `SELECT` except for those which also selected by the second `SELECT`. This operation is the same as the `MINUS` operation.
- `UNION` combines the results of two or more `SELECT` statements. There will be no repeating rows.
- `UNION ALL` combines all the results of two or more `SELECT` statements. There may be repeating rows.

## Set Operations – UNION

UNION:

- Combines rows from the first query with rows from the second query
- Removes duplicates or repeating rows



```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
  JOIN dimensions.customer c  
    ON c.customerid = t.customerid  
 WHERE t.transdate BETWEEN  
      '2008-01-01' AND '2008-05-17'
```

UNION

```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
  JOIN dimensions.customer c  
    ON c.customerid = t.customerid  
 WHERE t.transdate BETWEEN  
      '2008-01-01' AND '2008-01-21'
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

5

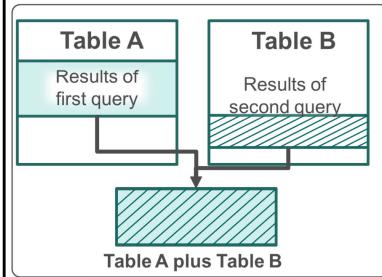
A union operation combines the results of the SELECT statement from the first table with the results from the query on the second table. The result set does not contain any repeating rows.

Since you may have very large data sets, doing a union or union all without some sort of aggregation can generate a result set with billions of rows. It's unlikely that you mean to do this. That's why our examples show the unions with aggregation operators.

## Set Operations – UNION ALL

UNION ALL:

- Combines rows from the first query with rows from the second query
- Does not remove duplicate rows



```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
 JOIN dimensions.customer c  
    ON c.customerid = t.customerid  
 WHERE t.transdate BETWEEN  
      '2008-01-01' AND '2008-05-17'
```

UNION ALL

```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
 JOIN dimensions.customer c  
    ON c.customerid = t.customerid  
 WHERE t.transdate BETWEEN  
      '2008-01-01' AND '2008-01-21'
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

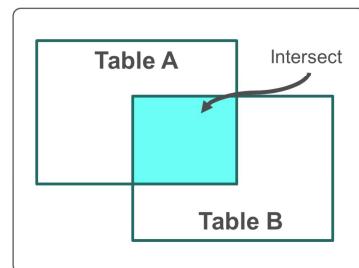
6

The UNION ALL set operation is like the UNION operation but it does not remove duplicate or repeating rows.

## Set Operations – INTERSECT

INTERSECT:

- Returns only the rows that appear in both SQL queries
- Removes duplicate rows



```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
  JOIN dimensions.customer c  
    ON c.customerid = t.customerid
```

**INTERSECT**

```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
  JOIN dimensions.customer c  
    ON c.customerid = t.customerid  
 WHERE t.transdate BETWEEN  
      '2008-01-01' AND '2008-01-21'
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

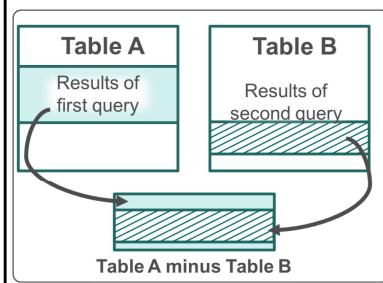
7

A set operation takes the results of two queries and returns only the results that appear in both result sets. Duplicate rows are removed from the final set returned.

## Set Operations – EXCEPT

EXCEPT:

- Returns all rows from the first SELECT statement
- Omits all rows that appear in the second SELECT statement



```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
  JOIN dimensions.customer c  
    ON c.customerid = t.customerid
```

EXCEPT

```
SELECT t.transid  
      c.custname  
  FROM facts.transaction t  
  JOIN dimensions.customer c  
    ON c.customerid = t.customerid  
 WHERE t.transdate BETWEEN  
   '2008-01-01' AND '2008-01-21'
```

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

8

The EXCEPT set operation takes the distinct rows of the first query and returns all of the rows that do not appear in the result set of the second query.

## What Is OLAP?

Online analytic processing:

- Is an approach to quickly provide answers to multi-dimensional queries
- Uses window functions that allow access to multiple rows in a single table scan
- Is enhanced with OLAP grouping extensions which are similar to GROUP BY but much more flexible

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

9

Part of broader category of BI, online analytic process is an approach to answer multi-dimensional queries. Databases configured for OLAP use a multidimensional data model that allows for complex analytical and ad-hoc queries.

Greenplum supports OLAP with window functions that provide access to multiple rows in a single table scan and grouping extensions, which provide flexibility when grouping result sets using the GROUP BY clause.

## Greenplum SQL OLAP Grouping Extensions

Greenplum supports the following grouping extensions:

- Standard GROUP BY
- ROLLUP
- GROUPING SETS
- CUBE
- grouping(column [, . . . ]) function

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

10

Greenplum introduced support for extensions to the standard GROUP BY clause, which is fully supported. These clauses can simplify the expression of complex groupings:

- **ROLLUP** – This extension provides hierarchical grouping.
- **CUBE** – Complete cross-tabular grouping, or all possible grouping combinations, is provided with this extension.
- **GROUPING SETS** – Generalized grouping is provided with the GROUPING SETS clause.
- **grouping function** – This clause helps identify super-aggregated rows from regular grouped rows.
- **group\_id function** – This clause is used to identify duplicate rows in grouped output.

## Standard GROUP BY Example

GROUP BY:

- Groups results together based on one or more columns specified
- Is used with aggregate statements

The following example summarizes product sales by vendor:

```
SELECT pn, vn, sum(prc*qty)
FROM sale
GROUP BY pn, vn
ORDER BY 1,2,3;
```

pn	vn	sum
100	20	0
100	40	2640000
200	10	0
200	40	0
300	30	0
400	50	0
500	30	120
600	30	60
700	40	1
800	40	1
(10 rows)		

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

11

In the examples we'll show, the base table contains a row for each sale in which where the columns are: pn, the product number, vn = the vendor number, prc= the price of the product for that transaction and qty, the quantity of the product sold.

The standard GROUP BY clause groups results together based on one or more columns specified. It is used in conjunction with aggregate statements, such as SUM, MIN, or MAX. This helps to make the resulting data set much more readable to a user.

The slide shows an example of a standard GROUP BY clause used to summarize product sales by vendor.

## Standard GROUP BY Example with UNION ALL

This example extends the previous example with the requirement that sub-totals and a grand total are added:

```
SELECT pn, vn, sum(prc*qty)
FROM sale
GROUP BY pn, vn
UNION ALL
SELECT pn, null, sum(prc*qty)
FROM sale
GROUP BY pn
UNION ALL
SELECT null, null,
sum(prc*qty)
FROM SALE
ORDER BY 1,2,3;
```

pn	vn	sum
100	20	0
100	40	2640000
<b>100</b>		<b>2640000</b>
200	10	0
200	40	0
<b>200</b>		<b>0</b>
300	30	0
<b>300</b>		<b>0</b>
400	50	0
<b>400</b>		<b>0</b>
500	30	120
<b>500</b>		<b>120</b>
600	30	60
<b>600</b>		<b>60</b>
700	40	1
<b>700</b>		<b>1</b>
800	40	1
<b>800</b>		<b>1</b>
		<b>2640182</b>

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

12

In this follow-on example, the requirements for the query have been extended to include sub-totals and a grand total. You would need to use a UNION ALL to continue the grouping and provide for the additional requirements.

## ROLLUP Example

The following example meets the requirement where the sub-total and grand totals are to be included:

```
SELECT pn, vn, sum(prc*qty)
FROM sale
GROUP BY ROLLUP(pn, vn)
ORDER BY 1,2,3;
```

pn	vn	sum
100	20	0
100	40	2640000
100		2640000
200	10	0
200	40	0
200		0
300	30	0
300		0
400	50	0
400		0
500	30	120
500		120
600	30	60
600		60
700	40	1
700		1
800	40	1
800		1
		2640182

(19 rows)

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

13

This slide meets the requirements provided in the previous slide, but uses the ROLLUP grouping extension. ROLLUP allows you to perform hierarchical grouping and helps to reduce the code.

## GROUPING SETS Example

The following examples shows how to achieve the same results with the GROUPING SETS clause:

```
SELECT pn, vn, sum(prc*qty)
FROM sale
GROUP BY GROUPING SETS
( (pn, vn), (pn), () )
ORDER BY 1,2,3;
```

Subtotals for each vendor

Summarize product sales by vendor

Grand total

pn	vn	sum
100	20	0
100	40	2640000
100		2640000
200	10	0
200	40	0
200		0
300	30	0
300		0
400	50	0
400		0
500	30	120
500		120
600	30	60
600		60
700	40	1
700		1
800	40	1
800		1
		2640182

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

14

The GROUPING SETS extension allows you to specify grouping sets. If you use the GROUPING SETS clause to meet the earlier requirements so that it produced the same output as ROLLUP, it would use the following groups:

- (pn, vn) – This grouping summarizes product sales by vendor.
- (pn) – This grouping provides subtotal sales for each vendor.
- () – This grouping provides the grand total for all sales for all vendors.

## CUBE Example

CUBE creates subtotals for all possible combinations of grouping columns.

The following example

```
SELECT pn, vn, sum(prc*qty)
FROM sale
GROUP BY CUBE(pn, vn)
ORDER BY 1,2,3;
```

is the same as

```
SELECT pn, vn, sum(prc*qty)
FROM sale
GROUP BY GROUPING SETS
    ( (pn, vn), (pn),
      (vn), () )
ORDER BY 1,2,3;
```

pn	vn	sum
100	20	0
100	40	2640000
100		2640000
200	10	0
200	40	0
200		0
300	30	0
300		0
400	50	0
400		0
500	30	120
500		120
600	30	60
600		60
700	40	1
700		1
800	40	1
800		1
	10	0
	20	0
	30	180
	40	2640002
	50	0
		2640182

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

15

A CUBE grouping creates subtotals for all of the possible combinations of the given list of grouping columns, or expressions. In terms of multidimensional analysis, CUBE generates all the subtotals that could be calculated for a data cube with the specified dimensions.

In the example shown on the slide, the additional grouping set of (vn) - subtotaling the sales by vendor, is included as part of the cube.

Note that  $n$  elements of a CUBE translate to  $2n$  grouping sets. Consider using CUBE in any situation requiring cross-tabular reports. CUBE is typically most suitable in queries that use columns from multiple dimensions rather than columns representing different levels of a single dimension. For instance, a commonly requested cross-tabulation might need subtotals for all the combinations of month, state, and product.

## GROUPING Function Example

Grouping distinguishes NULL from summary markers.

store	customer	product	price		SELECT * FROM dsales_null;
s2	c1	p1	90		
s2	c1	p2	50		
s2		p1	44		
s1	c2	p2	70		
s1	c3	p1	40		
(5 rows)					

store	customer	product	sum	grouping
s1	c2	p2	70	0
s1	c2		70	0
s1	c3	p1	40	0
s1	c3		40	0
s1			110	1
s2	c1	p1	90	0
s2	c1	p2	50	0
s2	c1		140	0
s2		p1	44	0
s2			44	0
s2			184	1
(12 rows)				1

© 2015 Pivotal Software, Inc. All rights reserved.

Pivotal.

16

When you use grouping extensions to calculate summary rows, such as sub-totals and grand totals, the output can become confusing if the data in a grouping column contains NULL values. It is hard to tell if a row is supposed to be a subtotal row or a regular row containing a NULL.

In the example shown on the slide, one of the rows shown where the customer field is NULL. Without the grouping id, you could misinterpret the sum of 44 as a subtotal row for store 2.

The GROUPING function returns a result for each output row, where:

- 1 represents a summary row.
- 0 represents grouped rows.

## Built In Functions

The Greenplum Database has a variety of PostgreSQL built in functions.

- Aggregate functions – min, max, count, avg
- Math functions – abs, trig function, floor, round, exp, random
- String functions – trim, pad, split, regexp, reverse, len
- Date Functions – current\_date, current\_time
- Pattern Matching – like, POSIX regexp
- And more.

[http://gpdb.docs.pivotal.io/4370/admin\\_guide/query/topics/functions-operators.html](http://gpdb.docs.pivotal.io/4370/admin_guide/query/topics/functions-operators.html)

Pivotal

© 2015 Pivotal Software, Inc. All rights reserved.

17

There are also many functions that do operations on individual rows and columns in addition to the aggregate functions like min, max, avg, and count.

It's important to remember that the GPDB does these in parallel across all the segments simultaneously, making these operations, like pretty much everything else much faster in GPDB than in a single process database.

## References

- All the Pivotal Greenplum Database documentation is available online
  - Reference Guide
  - Admin Guide
  - Client Tools
  - Load Tools
  - Connectivity Tools
  - Many more
- Pivotal Greenplum Database documentation:
  - <http://gpdb.docs.pivotal.io/gpdb-438.html>

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

18

# Thank You

Pivotal.

© 2015 Pivotal Software, Inc. All rights reserved.

19

# Pivotal

A NEW PLATFORM FOR A NEW ERA