Vertica7.2_Vmart_T-SQL_Queries_Yuan_Wang

Tuesday, August 30, 2016 2:39 PM

Introducing the VMart Example Database

Vertica ships with a sample multi-schema database called the VMart Example Database, which represents a database that might be used by a large supermarket (VMart) to access information about its products, customers, employees, and online and physical stores. Using this example, you can create, run, optimize, and test a multi-schema database.
The VMart database contains the following schema:

- public (automatically created in any newly created Vertica database)
- store
- online_Sales

The complete graph of these three schema is available on *Appendix: VMart Example Database Schema*.

```
localhost login: yuan
Password:
Last login: Tue Aug 30 14:41:12 on tty1
[yuan@localhost ~]$ su - dbadmin
Password:
Last login: Mon Aug 29 16:54:01 EDT 2016 on tty1
[dbadmin@localhost ~]$ vsql
Welcome to vsql, the Vertica Analytic Database interactive terminal.

Type:  \h or \? for help with vsql commands
       \g or terminate with semicolon to execute query
       \q to quit

dbadmin=> select * from databases;
    database_id    | database_name |     owner_id      | owner_name |         start_time         |                    compliance_message
   | export_subnet | load_balance_policy
------------------+---------------+-------------------+------------+----------------------------+-----------------------------------------------------------
---+---------------+--------------------
 45035996273704976 | VMart         | 45035996273704962 | dbadmin    | 2016-08-29 12:10:34.036495-04 | The database is in compliance with respect to raw data siz
e. |             0 | none
(1 row)
```

```
dbadmin=> select table_name from tables where table_schema='online_sales';
      table_name
-----------------------
 online_page_dimension
 call_center_dimension
 online_sales_fact
(3 rows)

dbadmin=> select table_name from tables where table_schema='store';
    table_name
-------------------
 store_dimension
 store_sales_fact
 store_orders_fact
(3 rows)

dbadmin=> select table_name from tables where table_schema='public';
     table_name
--------------------
 customer_dimension
 product_dimension
 promotion_dimension
 date_dimension
 vendor_dimension
 employee_dimension
 shipping_dimension
 warehouse_dimension
 inventory_fact
(9 rows)
```

Transcat-SQL Queries
I have used different T-SQL aggreate and analytic functions to explore this database to get the information I am interested in.

Query 1:
The following query lists name and annual income of male and female customer who has the highest annual income.

```
[dbadmin@localhost ~]$ less /mnt/hgfs/VMShare/Max_Annual.sql
SELECT customer_name, annual_income
FROM public.customer_dimension
WHERE (customer_gender, annual_income) IN (
  SELECT customer_gender, MAX(annual_income)
  FROM public.customer_dimension
  GROUP BY customer_gender);
(END)
```

Output:

```
dbadmin=> \i '/mnt/hgfs/VMShare/Max_Annual.sql'
  customer_name   | annual_income
------------------+---------------
 James M. McNulty |        999979
 Emily G. Vogel   |        999998
(2 rows)
```

Query 2:
The following query calculates the average rent cost of all stores in each state of U.S. and rank it from the highest to lowest.

```
[dbadmin@localhost ~]$ less /mnt/hgfs/VMShare/rank_store.sql
SELECT store_state, round(avg(monthly_rent_cost),2) as state_avg_rent,
RANK() OVER(
  ORDER BY avg(monthly_rent_cost) desc) AS RANK
  FROM store.store_dimension
  GROUP BY store_state;
(END)
```

Output:

```
dbadmin=> \i /mnt/hgfs/VMShare/rank_store.sql
 store_state | state_avg_rent | RANK
-------------+----------------+------
 NJ          |       10042.2 |    1
 VA          |        9806.5 |    2
 MD          |          9721 |    3
 WI          |       8645.67 |    4
 NH          |        8605.5 |    5
 IA          |          8406 |    6
 SD          |       8100.75 |    7
 CO          |       7244.92 |    8
 MS          |          7205 |    9
 DC          |       7077.33 |   10
 CT          |       6509.82 |   11
 MA          |        6476.1 |   12
 IN          |        6462.8 |   13
 FL          |       6387.91 |   14
 IL          |       6358.64 |   15
 WA          |          6341 |   16
 KS          |        6192.6 |   17
 GA          |       6129.22 |   18
 PA          |       5858.11 |   19
 TN          |       5750.83 |   20
 MI          |       5548.68 |   21
 CA          |       5509.66 |   22
 TX          |       5475.33 |   23
 LA          |        5004.5 |   24
 NV          |          4956 |   25
 NC          |          4875 |   26
 NY          |          4573 |   27
 UT          |          4505 |   28
 AZ          |       3805.25 |   29
 OR          |          2488 |   30
 SC          |          2332 |   31
(31 rows)
```

Query 3
The following query calculates the minimum, maximum, average employee annual salary and number of employees of all the stores in the state 'MA'

```
[dbadmin@localhost ~]$ less /mnt/hgfs/VMShare/count.sql
SELECT DISTINCT s.store_name
       , MIN(annual_salary) OVER (PARTITION BY s.store_name) AS MinSalary
       , MAX(annual_salary) OVER (PARTITION BY s.store_name) AS MaxSalary
       , round(AVG(annual_salary) OVER (PARTITION BY s.store_name),1) AS AvgSalary
       ,COUNT(ed.employee_key) OVER (PARTITION BY s.store_name) AS EmployeesPerStore
FROM store.store_dimension AS s
JOIN store.store_orders_fact as sof
     ON s.store_key = sof.store_key
JOIN public.employee_dimension AS ed
     ON sof.employee_key = ed.employee_key
WHERE s.store_state = 'MA'
ORDER BY AvgSalary DESC;
(END)
```

Output:

```
dbadmin=> \i /mnt/hgfs/VMShare/count.sql
 store_name | MinSalary | MaxSalary | AvgSalary | EmployeesPerStore
------------+-----------+-----------+-----------+-------------------
 Store138   |      1202 |    199553 |   55765.3 |              1236
 Store12    |      1202 |    199992 |   54901.1 |              1248
 Store58    |      1202 |    992363 |   54164.3 |              1144
 Store210   |      1204 |    199766 |   54033.8 |              1208
 Store36    |      1200 |    198786 |   53005.5 |              1198
 Store244   |      1202 |    199903 |   52693.1 |              1143
 Store75    |      1202 |    199225 |   52379.2 |              1180
 Store57    |      1200 |    199955 |   52083.4 |              1206
 Store157   |      1202 |    199744 |   51935.8 |              1255
 Store237   |      1202 |    199282 |   51119.8 |              1167
(10 rows)
```

Query 4:
The following query finds the profit for that calendar month and returns a running/cumulative average

```
[dbadmin@localhost ~]$ less /mnt/hgfs/VMShare/avg_analytic.sql
SELECT calendar_month_number_in_year, SUM(product_price-product_cost) AS profit,
   ROUND(AVG(SUM(product_price-product_cost))
    OVER (ORDER BY calendar_month_number_in_year),2) AS moving_profit
   FROM product_dimension AS p
   JOIN inventory_fact AS i ON p.product_key = i.product_key
   JOIN date_dimension AS d ON d.date_key = i.date_key
   GROUP BY calendar_month_number_in_year;
(END)
```

Output:

```
dbadmin=> \i /mnt/hgfs/VMShare/avg_analytic.sql
 calendar_month_number_in_year |  profit  | moving_profit
-------------------------------+----------+---------------
                             1 | 12022340 |      12022340
                             2 |  9858008 |      10940174
                             3 | 11496053 |      11125467
                             4 | 11534402 |   11227700.75
                             5 | 11933574 |    11368875.4
                             6 | 11331899 |   11362712.67
                             7 | 10813047 |      11284189
                             8 | 12485303 |   11434328.25
                             9 | 10912531 |   11376350.78
                            10 | 11883274 |    11427043.1
                            11 | 10618792 |   11353565.73
                            12 | 12453314 |   11445211.42
(12 rows)
```