



Vertica

Distributed Query Execution

Vertica Query Execution Basics

- SQL query is written against tables

```
SELECT count(*) FROM fact;
```

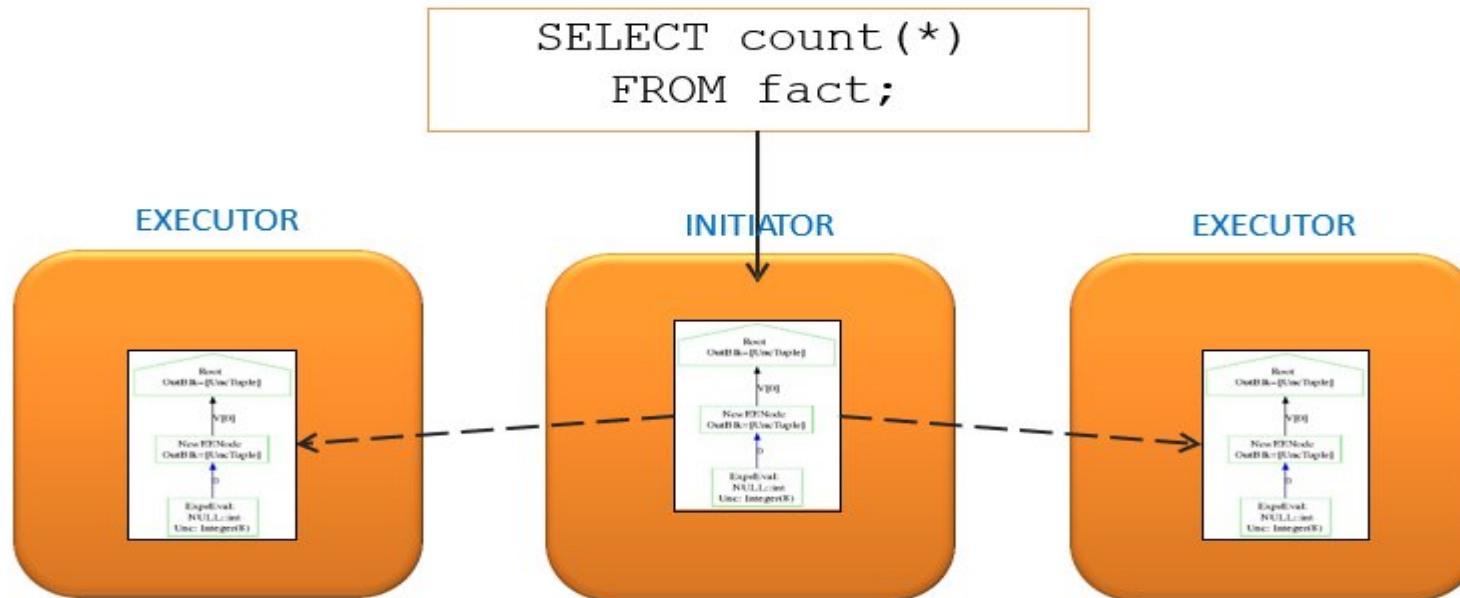
- Vertica translates the query to execute against projections

```
SELECT count(*) FROM fact_p1;
```

- The query Optimizer picks an optimal query plan
 - Picks the best projection(s) for the query
 - Picks the order in which to execute joins, query predicates, etc.
 - The query plan with the lowest cost is selected

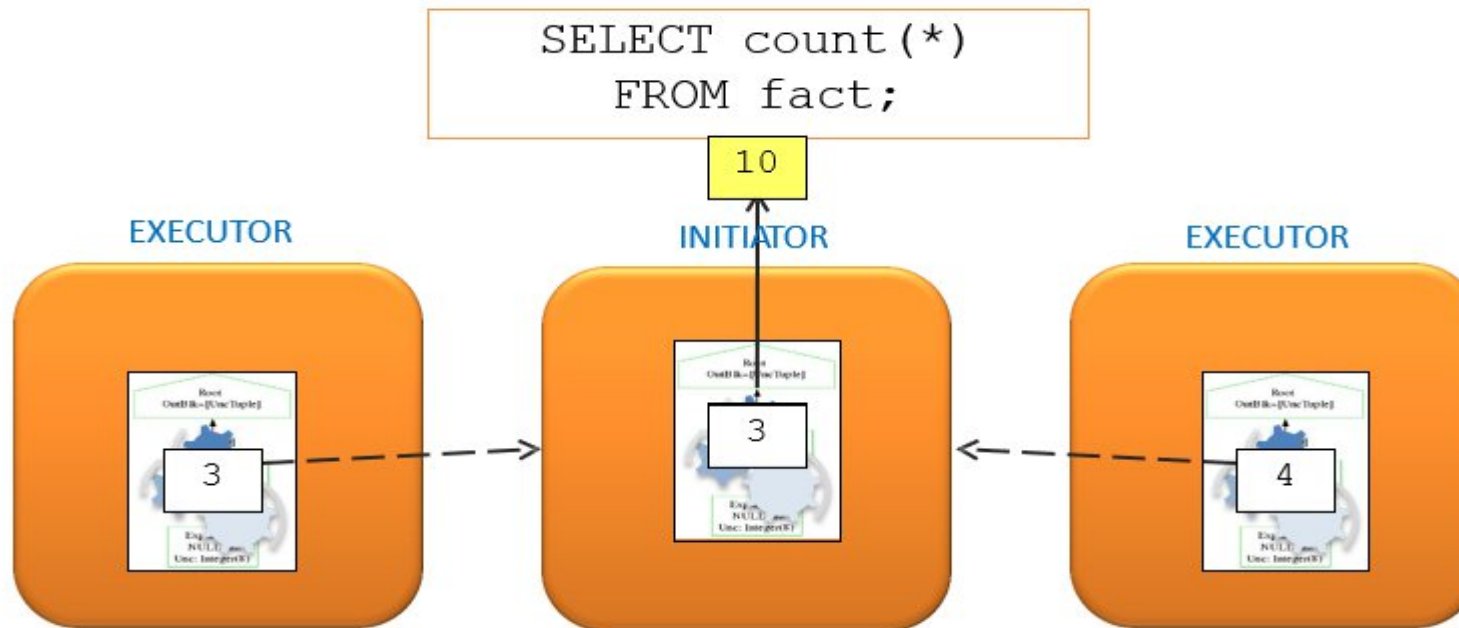
Query Execution Workflow

- Client connects to a node and issues a query
 - Node the client is connected to becomes the *initiator* node
 - Other nodes in the cluster are *executor nodes*
- Initiator node parses the query and picks an execution plan
- Initiator node distributes query plan to executor nodes



Query Execution Workflow

- All nodes execute the query plan locally
- Executor nodes send partial query results back to initiator node
- Initiator node aggregates results from all nodes
- Initiator node returns final result to the user



Query plans in Management Console

```
1 SELECT customer_name, customer_region, AVG(customer_age), COUNT(*) FROM CUSTOMER_DIMENSION
2 WHERE customer_gender = 'Male' AND customer_region IN ('East', 'Midwest')
3 GROUP BY customer_region, customer_name;
```

Execute Query

SELECT customer

Query Results | **Query Plan** | Query Profile

Expand All | Collapse All

+SELECT LIMIT 10K [Cost: 3K, Rows: 10K (NO STATISTICS)] (PATH ID: 0)
Output Only: 10000 tuples
Execute on: Query Initiator

+---> GROUPBY HASH (GLOBAL RESEGMENT GROUPS) (LOCAL RESEGMENT GROUPS) [Cost: 3K, Rows: 10K (NO STATISTICS)] (PATH ID: 1)
Aggregates: sum_float(CUSTOMER_DIMENSION.customer_age), count(CUSTOMER_DIMENSION.customer_age), count(*)
Group By: CUSTOMER_DIMENSION.customer_region, CUSTOMER_DIMENSION.customer_name
Output Only: 10000 tuples
Execute on: All Nodes

+---> STORAGE ACCESS for CUSTOMER_DIMENSION [Cost: 895, Rows: 10K (NO STATISTICS)] (PATH ID: 2)
Projection: public.customer_dimension_b0
Materialize: CUSTOMER_DIMENSION.customer_region, CUSTOMER_DIMENSION.customer_name, CUSTOMER_DIMENSION.customer_age
Filter: (CUSTOMER_DIMENSION.customer_gender = 'Male')
Filter: (CUSTOMER_DIMENSION.customer_region = ANY (ARRAY['East', 'Midwest']))
Execute on: All Nodes

Feedback

Query plans in vsql

Sample query:

```
dbadmin@node1:~  
dbadmin=> explain  
dbadmin-> select customer_name, customer_region, avg(customer_age), count(*)  
dbadmin-> from customer_dimension  
dbadmin-> where customer_gender = 'Male' and customer_region in ('East', 'Midwest')  
dbadmin-> group by customer_region, customer_name;
```

Generated query plan:

```
dbadmin@node1:~  
Access Path:  
+-GROUPBY HASH (GLOBAL RESEGMENT GROUPS) (LOCAL RESEGMENT GROUPS) [Cost: 3K, Rows: 10K (NO STATISTICS)] (PATH ID: 1)  
| Aggregates: sum_float(customer_dimension.customer_age), count(customer_dimension.customer_age), count(*)  
| Group By: customer_dimension.customer_region, customer_dimension.customer_name  
| Execute on: All Nodes  
| +---> STORAGE ACCESS for customer_dimension [Cost: 895, Rows: 10K (NO STATISTICS)] (PATH ID: 2)  
| | Projection: public.customer_dimension_b0  
| | Materialize: customer_dimension.customer_region, customer_dimension.customer_name, customer_dimension.customer_age  
| | Filter: (customer_dimension.customer_gender = 'Male')  
| | Filter: (customer_dimension.customer_region = ANY (ARRAY['East', 'Midwest']))  
| | Execute on: All Nodes
```

Reading an execution plan

The screenshot shows a database interface with a query plan for the query "SELECT customer". The plan is displayed in a tabbed view with "Query Results", "Query Plan", and "Query Profile". The "Query Plan" tab is active, showing a tree of operations. Annotations on the right side of the plan identify key information:

- Display information:** Points to the top of the plan, including the query text, tabs, and expand/collapse controls.
- Aggregate information:** Points to the "GROUPBY HASH" operation, highlighting its aggregates, group by columns, and output.
- Identify the projection, choose the columns, filter for values:** Points to the "STORAGE ACCESS" operation, highlighting its projection, materialized columns, filters, and execution context.

```
SELECT customer
```

Query Results | **Query Plan** | Query Profile | Expand All | Collapse All

+.-SELECT LIMIT 10K [Cost: 5K, Rows: 10K (NO STATISTICS)] (PATH ID: 0)
Output Only: 10000 tuples
Execute on: Query Initiator

+---> GROUPBY HASH (GLOBAL RESEGMENT GROUPS) (LOCAL RESEGMENT GROUPS) [Cost: 5K, Rows: 10K (NO STATISTICS)] (PATH ID: 1)
Aggregates: sum_float(CUSTOMER_DIMENSION.customer_age), count(CUSTOMER_DIMENSION.customer_age), count(*)
Group By: CUSTOMER_DIMENSION.customer_region, CUSTOMER_DIMENSION.customer_name
Output Only: 10000 tuples
Execute on: All Nodes

+---> STORAGE ACCESS for CUSTOMER_DIMENSION [Cost: 319, Rows: 50K (NO STATISTICS)] (PATH ID: 2)
Projection: public.customer_dimension_b0
Materialize: CUSTOMER_DIMENSION.customer_region, CUSTOMER_DIMENSION.customer_name, CUSTOMER_DIMENSION.customer_age
Filter: (CUSTOMER_DIMENSION.customer_gender = 'Male')
Filter: (CUSTOMER_DIMENSION.customer_region = ANY (ARRAY['East', 'Midwest']))
Execute on: All Nodes

Timing a query: Management Console

```
1 SELECT customer_name, customer_region, AVG(customer_age), COUNT(*) FROM CUSTOMER_DIMENSION
2 WHERE customer_gender = 'Male' AND customer_region IN ('East', 'Midwest')
3 GROUP BY customer_region, customer_name;
```

Execute Query

SELECT customer

Query Results | Query Plan | Query Profile

Export Data | Auto-Resize all columns | Search query results

customer_name	customer_region	AVG	COUNT
Steve Q. Garcia	East	35	1
Duncan N. Goldberg	East	22	1
Matt H. Moore	East	60	1
Dean P. Greenwood	East	37	1
David J. Lang	East	29	1
David L. Winkler	East	61	1

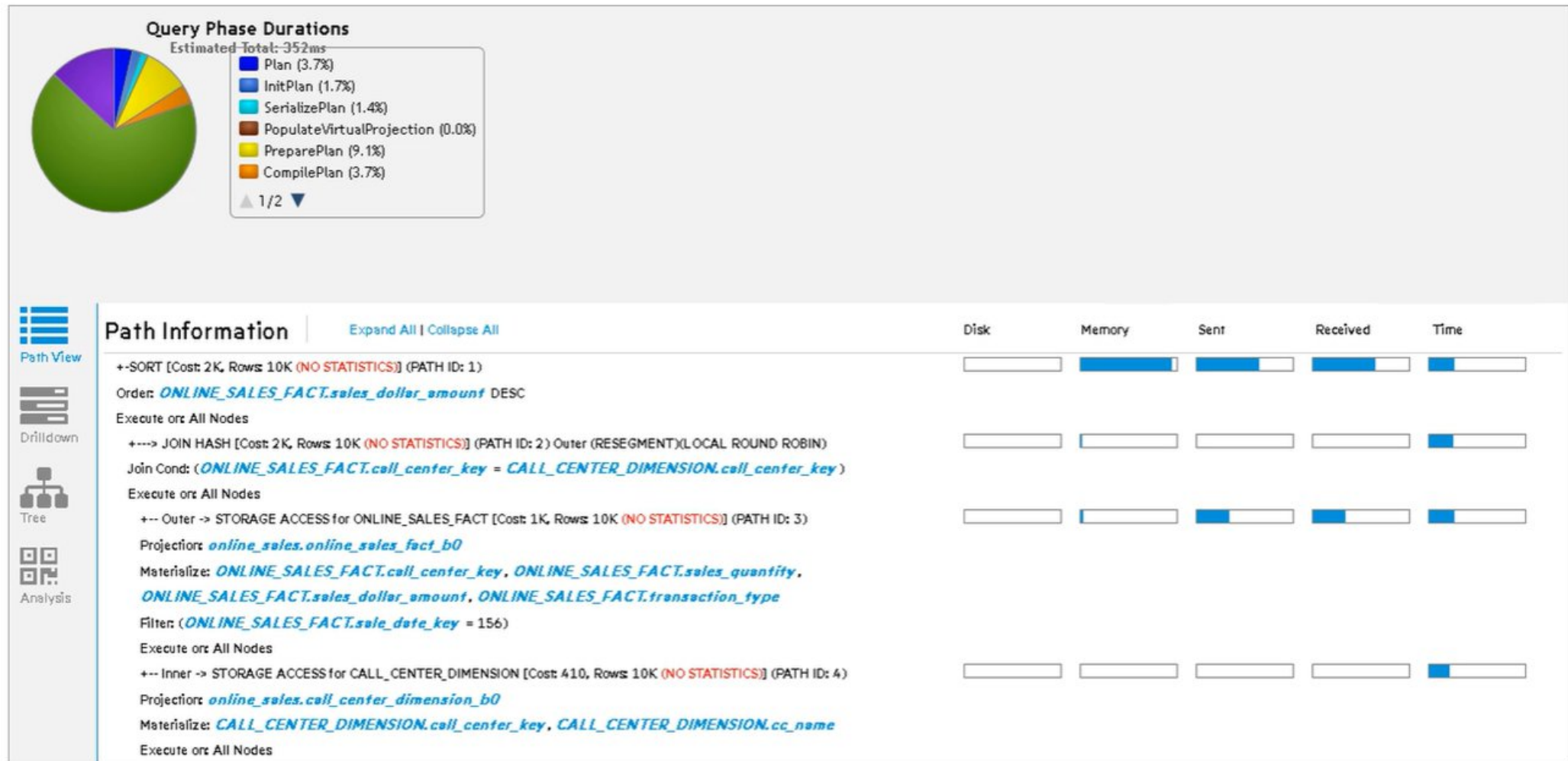
5033 rows | Execution time: 0.235s

Feedback

Timing a query: vsql

```
dbadmin@node1:~  
dbadmin=> \o /dev/null  
dbadmin=> \timing on  
Timing is on.  
dbadmin=> select customer_name, customer_region, avg(customer_age), count(*)  
dbadmin-> from customer_dimension  
dbadmin-> where customer_gender = 'Male' and customer_region in ('East', 'Midwest')  
dbadmin-> group by customer_region, customer_name;  
Time: First fetch (1000 rows): 179.310 ms. All rows formatted: 213.117 ms  
dbadmin=>
```

Query profiling: Management Console



Query profiling: vsql

```
dbadmin@node1:~  
vaotdb=> \o /dev/null  
vaotdb=> profile  
vaotdb-> select customer_name, customer_region, avg(customer_age), count(*)  
vaotdb-> from customer_dimension  
vaotdb-> where customer_gender = 'Male' and customer_region in ('East', 'Midwest')  
vaotdb-> group by customer_region, customer_name;  
NOTICE 4788: Statement is being profiled  
HINT: Select * from v_monitor.execution_engine_profiles where transaction_id=45035996273708151 and statement_id=1;  
NOTICE 3557: Initiator memory for query: [on pool general: 1064230 KB, minimum: 1064230 KB]  
NOTICE 5077: Total memory required by query: [1064230 KB]  
vaotdb=>
```

The background features several bright blue, glowing, curved lines that sweep across the frame from the bottom left towards the top right, creating a sense of motion and energy. The lines vary in thickness and brightness, with some appearing as sharp arcs and others as softer, more diffuse bands.

opentext™

Thank you