Understanding PostgreSQL Query Costs with EXPLAIN: A Deep Dive

Ivano Natalini
Senior Database Administrator PostgreSQL, MsSqlServer,...

March 8, 2025

When optimizing queries in PostgreSQL, the EXPLAIN command is an indispensable tool. It provides detailed insights into how the database engine executes a query, including the estimated costs of each operation. However, interpreting these costs can be challenging without a clear understanding of how PostgreSQL calculates them. In this article, we'll break down the cost calculation process using a real-world example and explore the factors that influence these estimates.

The Query and Its EXPLAIN Output

Consider the following query:

```
 sql

EXPLAIN

SELECT *

FROM large_test2

WHERE num1 = 4;
```

The EXPLAIN output for this query might look like this:

```
 sql

Bitmap Heap Scan on large_test2 (cost=23.56..176.20 rows=1971 width=24)

 Recheck Cond: (num1 = 4)

 -> Bitmap Index Scan on idx_num1_1 (cost=0.00..23.07 rows=1971 width=0)

 Index Cond: (num1 = 4)
```

Here, PostgreSQL uses a Bitmap Index Scan to find rows matching the condition num1 = 4, followed by a Bitmap Heap Scan to retrieve the actual rows from the table. Let’s break down the costs step by step.

Key Cost Parameters

PostgreSQL uses a cost model to estimate the "expense" of executing a query. This model relies on several configurable parameters:

- seq_page_cost: The cost of reading a page sequentially from disk (default: 1.0).

- cpu_tuple_cost: The cost of processing a single row (default: 0.01).

- cpu_operator_cost: The cost of executing an operator or condition (default: 0.0025).

These parameters allow PostgreSQL to estimate the total cost of a query based on the number of pages read, rows processed, and operations performed.

Step 1: Bitmap Index Scan

The first step in the query plan is the Bitmap Index Scan. PostgreSQL uses the index idx_num1_1 to quickly locate rows where num1 = 4. The cost of this operation is estimated as:

- Index Access Cost: This includes the cost of traversing the index structure to find the relevant rows. In this case, the cost is 23.07.

- Bitmap Creation: PostgreSQL creates a bitmap in memory to represent the matching rows. This cost is included in the 23.07 estimate.

The output shows:

sql

Bitmap Index Scan on idx_num1_1 (cost=0.00..23.07 rows=1971 width=0)

Here, rows=1971 indicates that PostgreSQL estimates 1,971 rows will match the condition.

Step 2: Bitmap Heap Scan

Once the bitmap is created, PostgreSQL performs a Bitmap Heap Scan to retrieve the actual rows from the table. The cost of this operation is calculated as follows:

1. Page Reads:

- PostgreSQL estimates that 128 pages need to be read from the table (`pg_relation_size('large_test2') / 8192.0 = 128`).

- The cost of reading these pages is 128 seq_page_cost = 128 1.0 = 128.

2. Row Processing:

- PostgreSQL processes 1,971 rows, with a cost of 1971 cpu_tuple_cost = 1971 0.01 = 19.71.

3. Rechecking Conditions:

- For each row, PostgreSQL rechecks the condition num1 = 4. This adds a cost of 1971 cpu_operator_cost = 1971 0.0025 = 4.9275.

4. Total Cost:

- Summing these costs gives 128 + 19.71 + 4.9275 = 152.6375.

However, the EXPLAIN output shows a higher cost range (`23.56..176.20`). This discrepancy arises because PostgreSQL includes additional overhead, such as the cost of accessing the index and more complex I/O and CPU estimations.

Key Takeaways

1. Understand the Cost Parameters:

- PostgreSQL's cost model relies on configurable parameters like seq_page_cost, cpu_tuple_cost, and cpu_operator_cost. These parameters influence the estimated cost of a query.

2. Break Down the Query Plan:

- Use the EXPLAIN output to identify the operations performed (e.g., index scans, heap scans) and their associated costs.

3. Validate Estimates:

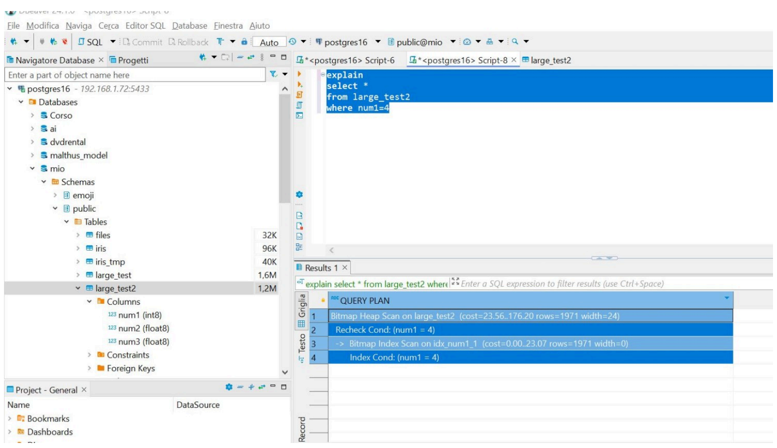
- Compare the estimated costs with your own calculations to ensure they align. If they don't, revisit the assumptions and parameters used.

4. Optimize Based on Costs:

- Use the cost estimates to identify bottlenecks and optimize your queries. For example, adding or refining indexes can reduce the cost of index scans.

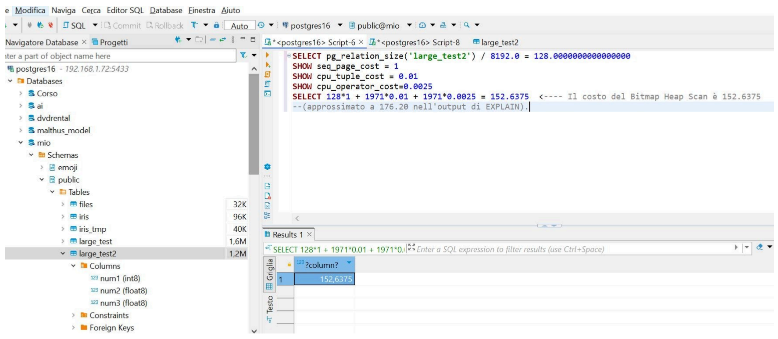
Conclusion

PostgreSQL's EXPLAIN command is a powerful tool for understanding and optimizing query performance. By breaking down the cost calculations and understanding the underlying parameters, you can gain deeper insights into how your queries are executed and identify opportunities for improvement. Whether you're a database administrator or a developer, mastering EXPLAIN is a critical step toward building efficient and scalable applications.



```
explain
select *
from large_test2
where num1=4
```

```
QUERY PLAN
1  Bitmap Heap Scan on large_test2  (cost=23.56..176.20 rows=1971 width=24)
2    Recheck Cond: (num1 = 4)
3    ->  Bitmap Index Scan on idx_num1_1  (cost=0.00..23.07 rows=1971 width=0)
4          Index Cond: (num1 = 4)
```



```
SELECT pg_relation_size('large_test2') / 8192.0 = 128.0000000000000000
SHOW seq_page_cost = 1
SHOW cpu_tuple_cost = 0.01
SHOW cpu_operator_cost=0.0025
SELECT 128*1 + 1971*0.01 + 1971*0.0025 = 152.6375  <---- Il costo del Bitmap Heap Scan è 152.6375
--(approssimato a 176.20 nell'output di EXPLAIN).
```

```
?column?
1   152,6375
```

#PostgreSQL #EXPLAIN #QueryOptimization #DatabasePerformance #CostEstimation, #QueryPlanning #DatabaseTuning #SQL #ExecutionPlan #seq_page_cost #cpu_tuple_cost #cpu_operator_cost #DatabaseAdministration #PerformanceTuning #SQLOptimization

Report this article

Comments

1

Like Comment Share

Add a comment...

No comments, yet.

Be the first to comment.

Start the conversation

Enjoyed this article?

Follow to never miss an update.

Ivano Natalini

Senior Database Administrator PostgreSQL, MsSqlServer, Mysql,Oracle,Cassandra,Mongodb,Redis,Artificial Intelligence ,Neural Network,Deep Learning,Computational Neuroscience,Programming.

Godson, Suresh and 2 others you know followed

Follow

More articles for you

LSTM: A Pillar of Artificial Intelligence

Ivano Natalini

Managing Lists in Redis Nosql Database with RPUSH, LTRIM and LREM

Ivano Natalini

Bayes' Theorem and Updating Priors in Neuroscience

Ivano Natalini