

Part 3

Query Processing & Optimization Techniques

Introduction

In the realm of database management, the increasing complexity of data structures and the demand for efficient data retrieval have necessitated the evolution of robust query processing and optimization techniques. This is particularly crucial in distributed database systems, where data is distributed across multiple nodes or servers. Query processing involves the transformation of high-level query languages into executable operations, while optimization aims to enhance the performance of these operations for faster and more efficient data retrieval.

In a distributed database system, data is distributed across multiple nodes or servers, allowing for parallel processing and improved scalability. However, this distribution introduces challenges in terms of query processing and optimization, as the system must efficiently coordinate and retrieve data from various locations.

In the project, we use the schema created in part 1 and initially add data of size that can show distinguishable results. We added realistic data using Python libraries.

Query Processing

- **Parsing and Translation:** The first step in query processing involves parsing and translating high-level queries into an internal representation that can be executed by the database system.
- **Query Rewrite:** Some distributed systems utilize query rewriting to enhance efficiency. This involves transforming a query into an equivalent, more optimized form.
- **Query Execution:** The translated query is executed across the distributed nodes, and the results are gathered and presented to the user.

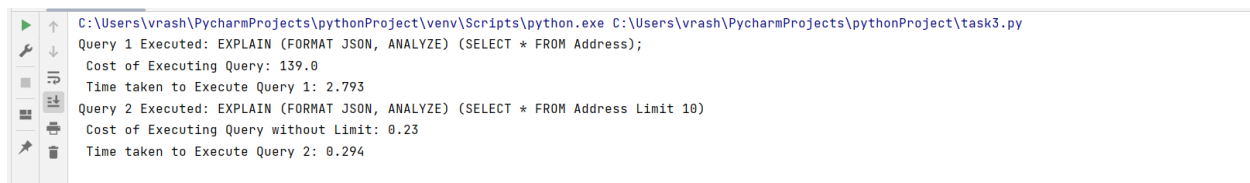
In our project, PostgreSQL was employed to handle these processes, while Python served as the interface for connecting with PostgreSQL to showcase query optimization.

Optimization Techniques Used in Project

Query Rewriting: Optimization often involves rewriting queries to minimize data transfer between nodes, reduce the need for expensive operations, and enhance overall efficiency.

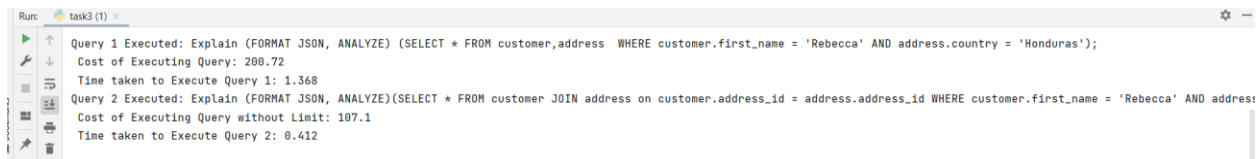
In our project, we take several scenarios of rewriting query strategies tailored for PostgreSQL, aiming to enhance the overall efficiency of our database operations.

- **Scenario 1:** Using a Limit in the query for pagination according to the necessary scenario helps optimize the query. Below are the results of cost and execution for query with limit and query without limit. As we clearly see from the results that query with limit has been executed with fewer cost and less execution time. However, the Limit can only be used in according to the selective scenarios.



```
C:\Users\vrash\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\vrash\PycharmProjects\pythonProject\task3.py
Query 1 Executed: EXPLAIN (FORMAT JSON, ANALYZE) (SELECT * FROM Address);
Cost of Executing Query: 139.0
Time taken to Execute Query 1: 2.793
Query 2 Executed: EXPLAIN (FORMAT JSON, ANALYZE) (SELECT * FROM Address Limit 10)
Cost of Executing Query without Limit: 0.23
Time taken to Execute Query 2: 0.294
```

- **Scenario 2:** Using a Join to combine the tables in the query for efficient retrieval of data. Below are the results of cost and execution for query with Join and query without Join. As we clearly see from the below results that query with Join has been executed with fewer cost and less execution time.



```
Run: task3 (1)
Query 1 Executed: Explain (FORMAT JSON, ANALYZE) (SELECT * FROM customer,address WHERE customer.first_name = 'Rebecca' AND address.country = 'Honduras');
Cost of Executing Query: 200.72
Time taken to Execute Query 1: 1.368
Query 2 Executed: Explain (FORMAT JSON, ANALYZE)(SELECT * FROM customer JOIN address on customer.address_id = address.address_id WHERE customer.first_name = 'Rebecca' AND address
Cost of Executing Query without Limit: 107.1
Time taken to Execute Query 2: 0.412
```

Distribution Indexing

Distribution indexing is a crucial aspect of database management that plays a pivotal role in optimizing query performance and enhancing overall system efficiency. It involves the strategic placement of index structures across distributed data sets to expedite query processing and retrieval of information. This section delves into the concept of distribution indexing, its significance, and the impact it has on database operations.

In this project, we have created several indexes for the tables

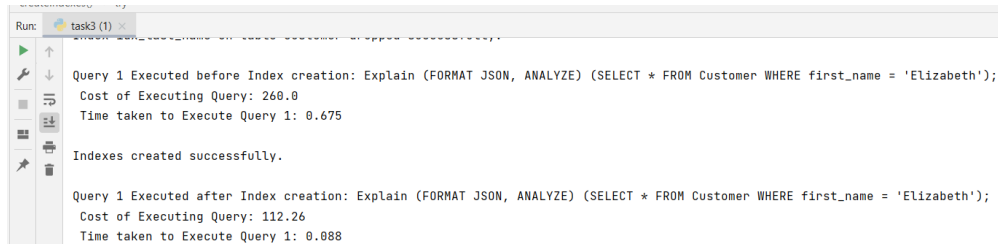
- Address on column country and city,
- Categories on column category_name, created_time, modified_time.
- Inventory on column quantity
- Contact Details on column mobile_num, email.

- Customer on column first_name, last_name.

Scenario 3: Optimize queries by utilizing indexes effectively, ensuring a balance between selectivity and efficiency. Below is the screenshot of the results of executing the query before and after indexing. It is clear that indexing optimizes query retrieval.

```
; Query 1 Executed before Index creation: Explain (FORMAT JSON, ANALYZE) (SELECT * FROM Address WHERE country = 'USA');
Cost of Executing Query: 154.0
Time taken to Execute Query 1: 1.117

Indexes created successfully.
Query 1 Executed after Index creation: Explain (FORMAT JSON, ANALYZE) (SELECT * FROM Address WHERE country = 'USA');
Cost of Executing Query: 52.75
Time taken to Execute Query 1: 0.052
```



The screenshot shows a database query execution interface with a 'Run' button and a 'task3 (1)' tab. The interface displays the results of a query execution, including the query text, the cost of execution, and the time taken to execute the query. The query is: `Query 1 Executed before Index creation: Explain (FORMAT JSON, ANALYZE) (SELECT * FROM Customer WHERE first_name = 'Elizabeth');`. The cost of execution is 260.0 and the time taken to execute the query is 0.675. After indexing, the cost of execution is 112.26 and the time taken to execute the query is 0.088. The interface also shows a message: 'Indexes created successfully.'

```
Run: task3 (1)
Query 1 Executed before Index creation: Explain (FORMAT JSON, ANALYZE) (SELECT * FROM Customer WHERE first_name = 'Elizabeth');
Cost of Executing Query: 260.0
Time taken to Execute Query 1: 0.675

Indexes created successfully.

Query 1 Executed after Index creation: Explain (FORMAT JSON, ANALYZE) (SELECT * FROM Customer WHERE first_name = 'Elizabeth');
Cost of Executing Query: 112.26
Time taken to Execute Query 1: 0.088
```