A red and white logo with a bird

Description automatically generated

Wroclaw University of Technology

and

Telecommunications

**Data Warehouses Report 4**

Members:

Khalid Muzaffar(269553)

Mert Nuri Kodzhaaslan(260312)

**Performance Analysis**

**Query 1**

* **Execution Plan Highlights**:
  + Key Operations: 4 Nested loops are present for joining tables with a relatively small number of rows, and hash matches are used where a larger set of data is involved.

A screen shot of a computer

Description automatically generated

A white rectangular object with a black line

Description automatically generated

* + Costly Operations: Clustered Index Scan on **SalesOrderDetail** ( 23%)and a Key Lookup on **SalesOrderHeader(53%)** are the most resource-intensive operations in the plan.

A screenshot of a computer

Description automatically generated A screen shot of a computer

Description automatically generated

* + Index Usage: An Index Seek on **SalesOrderHeader** suggests good index utilization, while the Clustered Index Scan indicates potential for optimization.

A blue and white text on a white background

Description automatically generated

* **Performance Metrics**:
  + Execution Time: elapsed time = 366 ms.
  + IO Statistics: highest was clustered index scan 0.918681

A screenshot of a computer screen

Description automatically generated

**Query 2**

* **Execution Plan Highlights**:
  + Key Operations: Nested loops for joining smaller result sets and hash matches for larger, more complex aggregations and joins.
  + Costly Operations: The most expensive operation appears to be a Clustered Index Scan on the **SalesOrderDetail** table, which typically indicates a full table scan.

A screenshot of a computer screen

Description automatically generated

* + Index Usage: The execution plan shows a Clustered Index Seek on the **Person** table, indicating efficient use of indexes there. However, the Clustered Index Scan on **SalesOrderDetail** might benefit from further index optimization.
* **Performance Metrics**:
  + Execution Time: Total execution time: 00:00:00.518
  + IO Statistics: highest was the two clustered index scans

**Query 3**

* **Execution Plan Highlights**:
  + Key Operations: Nested loops and hash matches are the primary operations, with hash matches particularly involved in aggregation, which is typical for grouping and summarizing data.
  + Costly Operations: A significant portion of the cost is associated with a Clustered Index Scan on the **SalesOrderDetail** table, implying a full read of this table.
  + Index Usage: The execution plan includes a Clustered Index Seek on the **Employee** table, which is optimal. However, there's also an Index Scan on the **SalesPerson** table, indicating a complete read-through, which may be less efficient.
* **Performance Metrics**:
  + Execution Time: 00:00:00.355
  + IO Statistics: highest was the two clustered index scans

Brief Analysis:

* Query 1 (No Window Functions): This query had a straightforward execution plan with direct aggregations. However, it included a costly Clustered Index Scan on the SalesOrderDetail table.
* Query 2 (Using Window Functions): It displayed multiple Hash Match operations which can be resource-intensive but are efficient for handling large datasets. The use of window functions adds complexity and can increase CPU usage, particularly if the dataset is large.
* Query 3 (Using CTE): The execution plan indicated the use of a CTE with subsequent nested loop joins, which are efficient for smaller datasets. The plan also involved a Clustered Index Scan, suggesting a full table read.

**Conclusion:**

Taking into account the presence of Index Scans and Hash Matches:

* If the dataset is large: Query 2 might perform better despite its complexity because window functions and hash matches are optimized for large sets of data. However, it may benefit from indexing optimizations to transform scans into seeks, reducing IO and potentially improving execution time.
* For smaller datasets: Query 3 could be advantageous as the nested loop joins and the pre-aggregation logic within the CTE would be very efficient. Additionally, ensuring the CTE does not generate a large intermediate result set would be crucial.
* Overall, optimizing the Clustered Index Scans present in both Query 1 and Query 3 to Index Seeks could yield significant performance benefits. Furthermore, while Hash Matches are powerful, they should be monitored for CPU usage, and where possible, data should be pre-filtered or indexed to reduce the cost of these operations.

In essence, the choice of query should be guided by the size of the data and the specific workload. Proper indexing strategies will enhance the performance of all queries, but the benefits will be most noticeable for the queries with Index Scans that can be converted to Seeks.