**Data Warehousing Project**

**Building and Analyzing a Near-Real-Time Data Warehouse**

**Prototype for METRO Shopping Store in Pakistan**

1. **Project Overview**

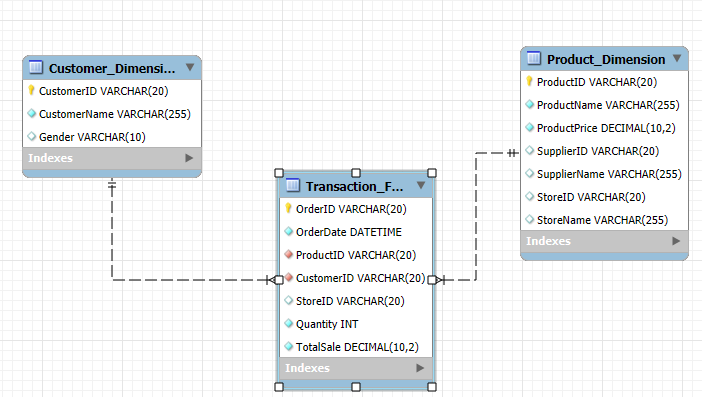
The project focuses on creating and querying a Data Warehouse (DW) optimized for efficient data analysis. The key objectives were to:

1. Design a schema for a Data Warehouse tailored for a retail business.
2. Implement OLAP (Online Analytical Processing) queries to extract insights such as revenue trends, sales performance, and product pair analysis.
3. Integrate and evaluate the **MESHJOIN** algorithm for real-time query processing of large datasets.
4. Identify shortcomings of the MESHJOIN algorithm and reflect on lessons learned.

The project simulated a retail business scenario with large transaction logs, product details, and supplier data. A combination of OLAP queries and the MESHJOIN algorithm was used to provide insightful reports.

1. **Data Warehouse Schema**

The DW schema follows the **Star Schema** design.



This schema ensures normalization of data, facilitating efficient storage and retrieval.

1. **MESHJOIN Algorithm**

The **MESHJOIN** algorithm is a join operator designed to handle real-time, large-scale data streams.

**Algorithm Overview**

MESHJOIN divides a large dataset into two parts:

1. **Static relation**: Pre-loaded into memory in small chunks.
2. **Streaming relation**: Incrementally joins with chunks of the static relation.

**Steps:**

1. Partition the static relation into manageable chunks.
2. Load the chunks into memory.
3. Continuously process the stream in batches.
4. Perform incremental joins between the loaded chunks and the current stream.

**Advantages:**

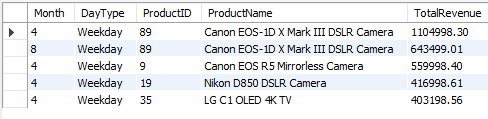
* Suitable for large datasets with real-time data streams.
* Minimizes memory usage compared to conventional joins.

**Implementation:**

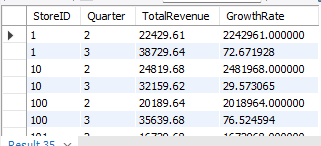
The algorithm was implemented for OLAP queries where a product or transaction stream was joined with a static dimension table (e.g., Product Dimension).

1. **OLAP Query Outputs**

* Query 1: Get Top Revenue Generating Products By Day And Month



* Query 2: Get Store Revenue Growth Rate Quarterly 2019 (there was no data for 2017)



* Query 3: Get Detailed Supplier Sales Contribution

A screenshot of a computer

Description automatically generated

* Query 4: Get Seasonal Sales Contribution

A screenshot of a computer

Description automatically generated

* Query 5: Get Supplier Revenue Volatility

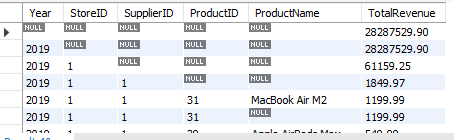
A screenshot of a computer

Description automatically generated

* Query 6: Get Top Product Pairs



* Query 7: Get Revenue by Product and Supplier



* Query 8: Get Product Revenue

A screenshot of a computer

Description automatically generated

* Query 9: Get Sales Spikes

A screenshot of a data

Description automatically generated

* Query 10: Create Store Quarterly Sales View



1. **Shortcomings of MESHJOIN**

* **Latency with Small Batches**:
  + MESHJOIN's performance drops when the streaming relation arrives in small batches, leading to frequent context switching and inefficient use of memory.
* **Static Relation Dependency**:
  + The algorithm requires the static relation to be pre-partitioned. Any changes to this data require a complete reload, which is inefficient.
* **Complexity in Scaling**:
  + While designed for large datasets, MESHJOIN becomes inefficient when the static relation grows beyond the manageable chunk size, leading to increased I/O operations.

1. **Lessons Learned**

* **Importance of Schema Design**: A well-designed schema like the star schema significantly improves query performance.
* **Algorithm Suitability**: Choosing the right algorithm for specific tasks is crucial. While MESHJOIN is efficient for specific use cases, its limitations must be understood before implementation.
* **OLAP Utility**: OLAP queries provide invaluable insights for decision-making in business environments.

1. **Conclusion**

This project provided practical experience in designing and querying a Data Warehouse. Implementing the MESHJOIN algorithm highlighted both its strengths and areas for improvement in handling real-time data streams. The lessons learned will guide future efforts in optimizing large-scale data processing systems.

**Prepared by**: *Abdul Rehman 21i-1780*  
**Date**: *24th November 2024*