Data Warehousing and Business Intelligence Project Report



**Abubakar Siddiq**

21i-2742

DS-N

# Project Overview

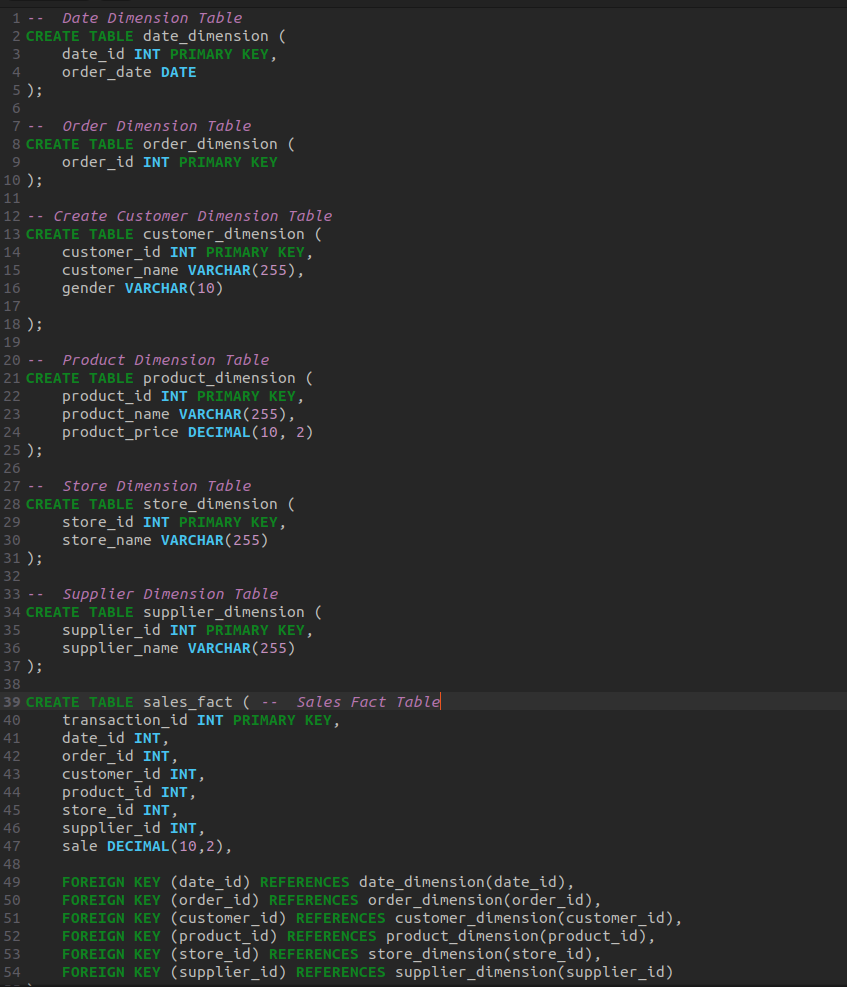
This project focuses on integrating transactional data from the 'transactions' table and master data from the 'master\_data' table into the 'ELECTRONICA-DW' Data Warehouse. The primary goal is to create a cohesive and centralized repository. Using multi-threading techniques and join operations, the project aims to streamline the process for an optimized and resilient data integration framework.

# Schema for DW

I have created a StarSchema with all the foreign keys to my dimension tables in the sales fact table and the aggregated sales fact. While the other dimensions have relevant attributes.

# 

To create this, I ran these sql queries:



# 

# Main Class Structure:

The main class creates the 3 threads used in this project: StreamGeneratorThread, HybridJoinThread and ControllerThread. It also creates the multiHashMap that is used to map the Transactional data into a hash and queue.

It then passes the hashmap to relevant threads. To start the threads I have used the start() method.

# StreamGeneratorThread:

Reads data from the transactions table and Master table and populates a MultiHashMap and data structure. Implements logic to process chunks of transactional data and enqueue join attributes, product ID. The processTransactions() method does this specifically for Transactions, while the processMasterData() method would handle Master data. Unfortunately, I haven’t been able to do this yet.

The run method is used as the actual method when the start() method is called. So in it I have called the relevant methods making sure i use try-catch to catch any errors/exceptions caused.

# HybridJoinThread:

Conducts a hybrid join operation between transactional data and master data. Loads initial data, processes chunks, and performs a join operation. Matches tuples, calculates sales, and loads joined data into the Data Warehouse.

# ControllerThread:

Monitors and regulates the pace of the StreamGeneratorThread based on performance metrics (arrival rate, service rate). Controls the overall workflow and regulates the HybridJoinThread and StreamGeneratorThread.

# 

# Data Structures:

Transactions and MD classes encapsulate transactional and master data attributes, respectively. MultiHashMap manages transactions and their join attributes. DiskBuffer facilitates loading and retrieval of MD tuples during the join operation.

# Key Implementations:

## Multi-threaded Processing:

Utilizes multi-threading to concurrently handle data retrieval, processing, and integration. It does this by using wait() and notify() methods. Unfortunately, this has not been implemented yet.

## Data Retrieval and Processing:

StreamGeneratorThread retrieves transactional data and organizes it into a MultiHashMap. HybridJoinThread performs a hybrid join operation, combining transactional and master data.

## Join Operation and Data Loading:

Matches tuples, calculates sales, and loads joined data into the Data Warehouse.

## ControllerThread Regulation:

Controls the pace of data generation and regulates the data integration process.

# Challenges and Solutions:

## Contradictions within the project:

The project describes the use of streams but says we do not need streams. The purpose of StreamGeneratorThread is basically rendered useless. Instead of making use of it, it becomes a hurdle to manage.

## Concurrency Management:

Ensured synchronized access to shared resources like the MultiHashMap and the DiskBuffer.

## Performance Optimization:

Balanced arrival rates and service rates to ensure efficient data processing without backlogs.

## Error Handling:

Implemented robust exception handling mechanisms for SQL queries and thread interruptions.

# Shortcoming of HybridJoin:

HybridJoin has a very complex implementation, it took me multiple days and reading the project description hundreds of times before I could understand what it actually did. After I did understand, I realized that Hybridjoin has a really high spatial complexity as it uses multi hashmaps, queues, buffers and more. It also has high overhead from disk I/O operations. Hence, making it impractical for large datasets.

# Conclusion and What I Learnt:

The project tries to implement multi-threaded data processing and integration techniques, efficiently handling transactional and master data to populate the Data Warehouse. From this project I learnt how to implement java, I also learnt how to use JDBC to connect my database to java and implementation of Hybridjoin. It also improved my knowledge in threading. Lastly, it helped me to understand a real world application of DW and data science.