# **DATA WAREHOUSE**

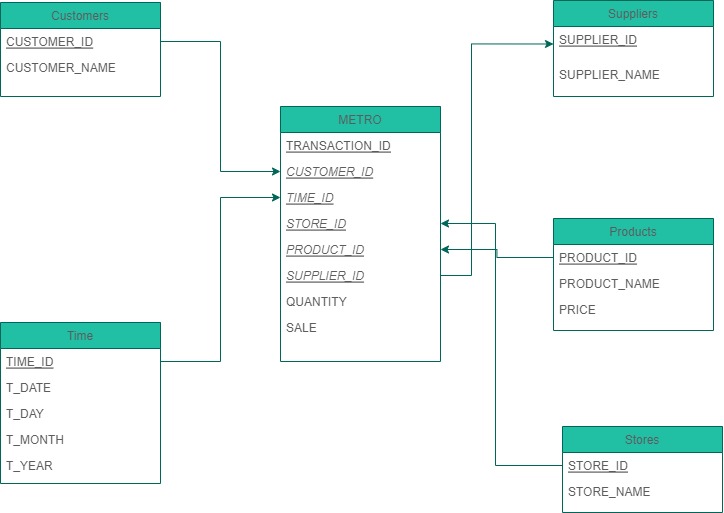
# PROJECT REPORT

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

### **Project Overview:**

This project is the implementation of near real-time data warehouse for Metro Superstore data. To create the warehouse, data is first extracted from the database, transformed and then stored in the data warehouse where it can be analyzed. To do this, transactional data and master data is first extracted from the database in the form of partitions and then MESH JOIN is implemented to enrich the data by joining transactional data with the master data and doing calculations so that data can be converted into the form useful for analysis. When the all the data is successfully stored in the data warehouse, OLAP queries can be used to do analysis.

### **Star- Schema for Metro Data Warehouse:**



### **MESHJOIN Algorithm:**

Mesh join is an algorithm which is used to enrich transactional data by joining it with master data and to transform it into the form useful for analysis. MESHJOIN is implemented in the following steps:

1. Extract data from database in the form of partitions. (I have partitioned transactional data into 12 partitions, 1 for each month and master data into 5 partitions).
2. Each partition of transactional data is compared with all the partitions of the master data. For this, a HASHMAP is created for transactional data and its keys are matched with the tuples of the master data. Whenever a tuple is matched, it is inserted into the HASHMAP.
3. When all the tuples are matched for a partition of transactional data, the HASHMAP for that partition is sent into a thread so that the data can be inserted into data warehouse. Thread is created for the purpose of multiprocessing so that when data is sent for insertion, meanwhile next partition can be compared and joined with master data.
4. Loop iterates till all the partitions of transactional data are compared and sent into the warehouse.

#### Algorithm:

For( i = 0 to number of partitions of transactional data):

Map<Key, Values> MultiValueMap = new Hashmap // Key and Values are classes

Partition[i] = Extract data from transactions table partition[i]

Store Partition[i] data in MultiValueMap where TRANSACTION\_ID , CUSTOMER\_ID ,

PRODUCT\_ID as key and STORE\_ID, STORE\_NAME , TIME\_ID , T\_DATE , QUANTITY as values ;

For ( j = 0 to number of partitions of master data):

Resultset Customer = Extract data from customers table partition[j]

Resultset Product = Extract data from product table partition[j]

For ( iterate MultiValueMap keys ):

While ( iterate resultsets ):

If( tuples from resultsets match with hashmap keys)

Insert into hashmap;

Thread InsertIntoDw(MultiValueMap);

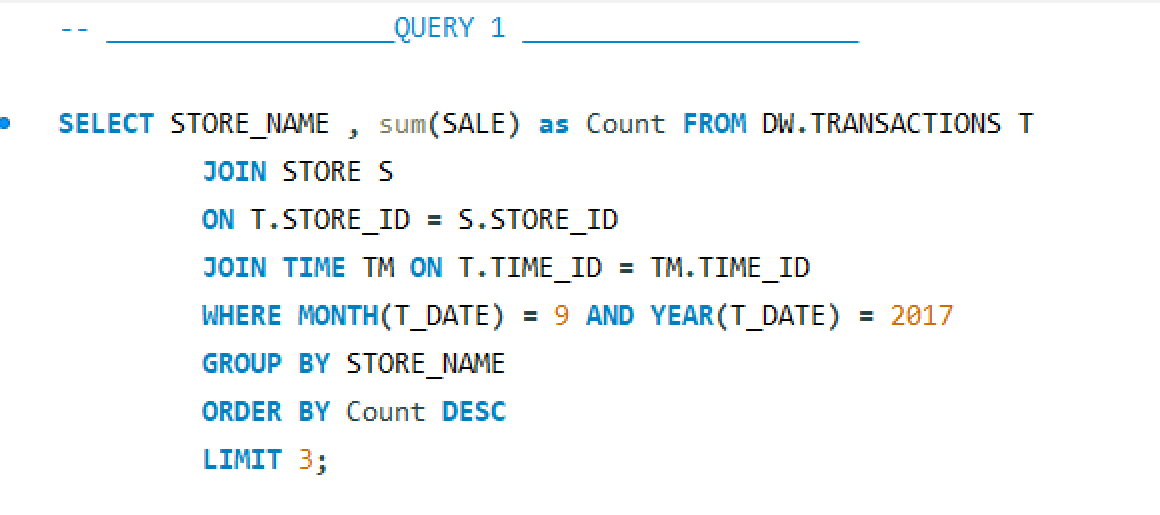
#### Shortcomings of MESHJOIN:

1. MESHJOIN is not ideal for retail data as it is not normally distributed and, some products are sold more than the others. So, if frequently read tuples from master data are stored for fast access, then the algorithm can be designed to work faster for retail data.
2. While comparison with the master data, there is a chance that tuples from transactional data don’t match with any tuple from the master data partition.
3. While insertion into the dimension tables, data is again and again checked if it is already entered or not. This takes extra processing power.

### **OLAP Queries:**

#### QUESTION 1:

##### QUERY:



##### OUTPUT:

Graphical user interface, text, application

Description automatically generated

#### QUESTION 2:

##### QUERY:

Graphical user interface, text, application

Description automatically generated

##### OUTPUT:

Table

Description automatically generated

These are the top 10 suppliers who generate the most revenue over the weekends so we can forecast that the top suppliers for next weekend could be one of them.

#### QUESTION 3:

##### QUERY:

Graphical user interface, text, application, email

Description automatically generated

##### OUTPUT:

Graphical user interface, text, application

Description automatically generated

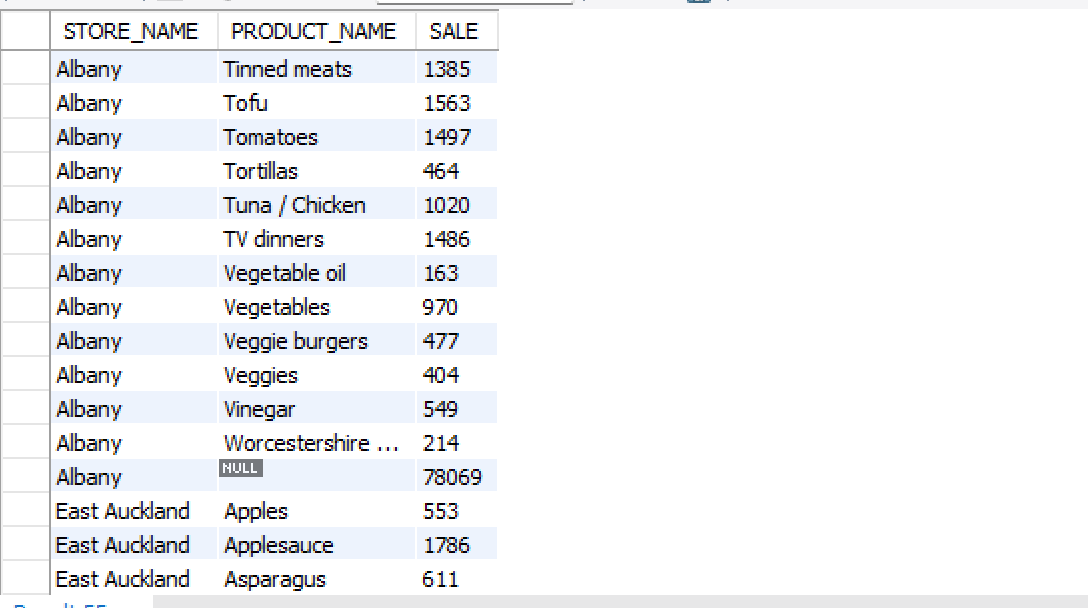
#### QUESTION 4:

##### QUERY:

Graphical user interface, text, application

Description automatically generated

##### OUTPUT:



#### QUESTION 5:

##### QUERY:

Graphical user interface, text, application, email

Description automatically generated

##### OUTPUT:

Table

Description automatically generated

#### QUESTION 6:

##### QUERY:

Graphical user interface, text, application

Description automatically generated

##### OUTPUT:

Graphical user interface, application

Description automatically generated

#### QUESTION 7:

##### QUERY:

Graphical user interface, text, application, email

Description automatically generated

##### OUTPUT:

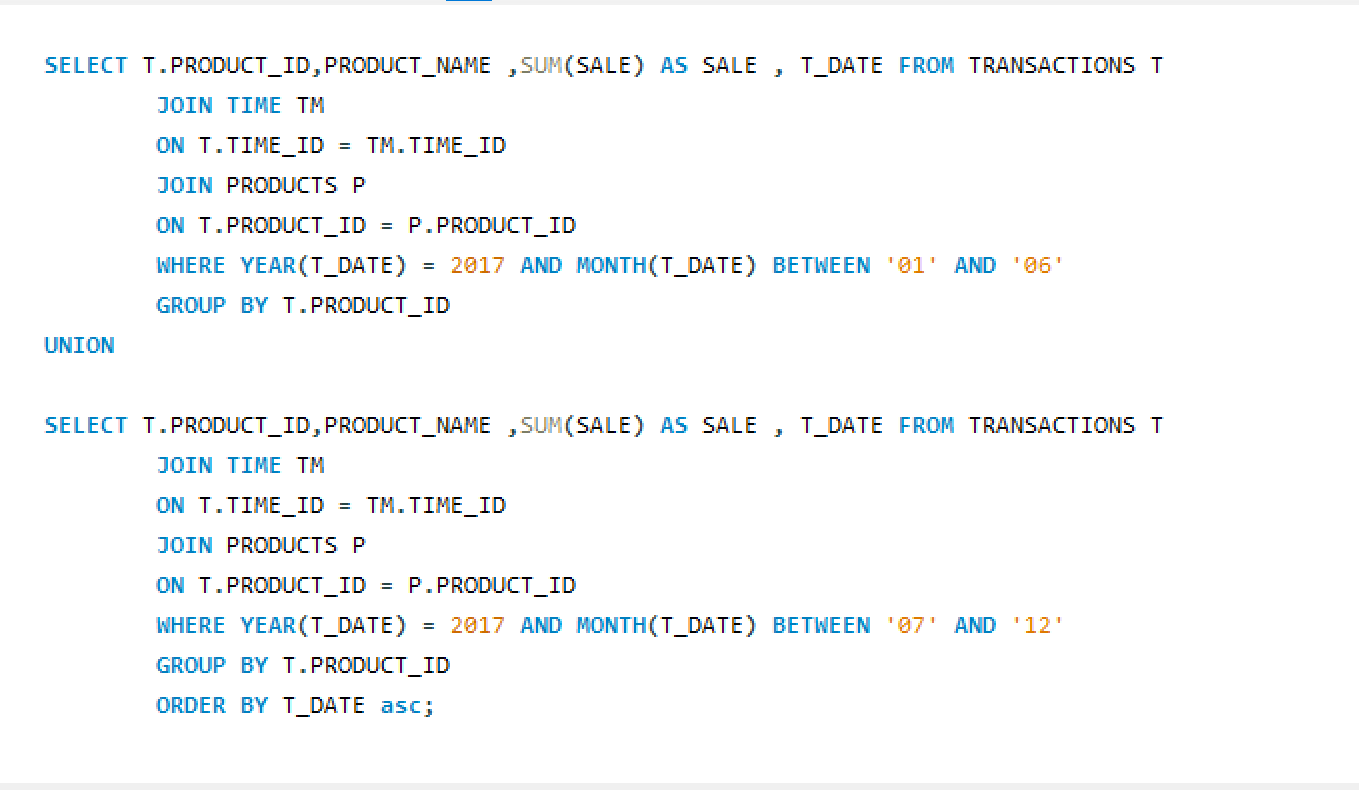
Table

Description automatically generated

By using roll up , we get the sum of each product sold along with the sum of all products sold by each suppliers and sum of all the products sold in each store.

#### QUESTION 8:

##### QUERY:



##### OUTPUT:

Table

Description automatically generated

#### QUESTION 9:

##### QUERY:

Graphical user interface, text, application

Description automatically generated

##### OUTPUT:

Graphical user interface, application, Word

Description automatically generated

The anomaly in the data warehouse dataset is that two products have same name but different product id and different price

#### QUESTION 10:

##### QUERY:

Graphical user interface, text, application

Description automatically generated

##### OUTPUT:

Table

Description automatically generated

Materialized view can improve query performance if you frequently require pre aggregated data, filtered data or joins with large and small tables.

### **What have you learned from this project?**

I have learned the following things from this project:

1. How to implement near real-time data warehouse
2. How data is extracted from the database, transformed into the form useful for analysis and loaded into the data warehouse
3. What kind of problems occur while implementation of mesh join
4. How to efficiently handle large amount of data and why is it important to manage huge amount of data properly
5. How to write OLAP queries to analyze data from the data warehouse
6. Importance of multiprocessing and implementation of theads