

**Project Report**

**Course Name: Data Warehousing & Business Intelligence**

**Semester-Fall, 2022**

**Practical Project**

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

**Prototype for METRO Shopping Store in Pakistan**

**Name**: Arsalan Ahmed

**Roll no**. 19i-1875

**Section**: BS(DS-N)

**Due Date**: 2 Dec, 2022

**Submitted to**: Sir Asif Naeem

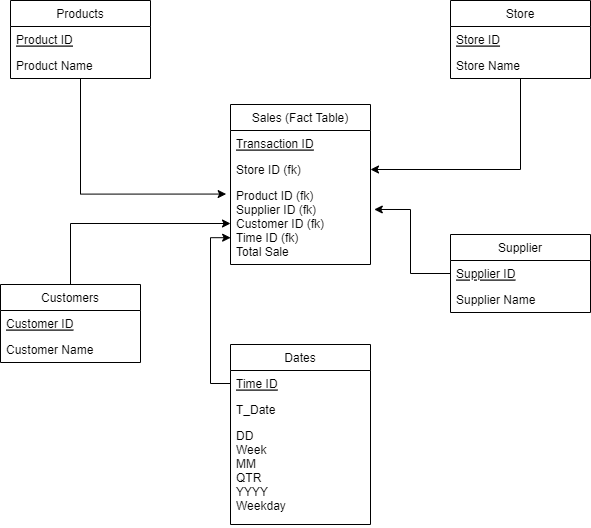
**Project overview:**

Metro is one of the biggest supermarkets and has many branches spreading across Pakistan. It generates tons of transaction data and information. With this data we can create a data warehouse for analyzing their business information. To achieve the analysis goal, we need to build a near-real-time warehouse.

The aim of this project is to create a data warehouse with 10,000 transaction records, 100 products and 50 customers. As the data from the transaction data is incomplete, we need to get some information from master data tables like details of product, supplier and customer. To do that we need to join the master data tables with transaction data and for joining we need to implement the extended Mesh Join algorithm.

**Star Schema:**

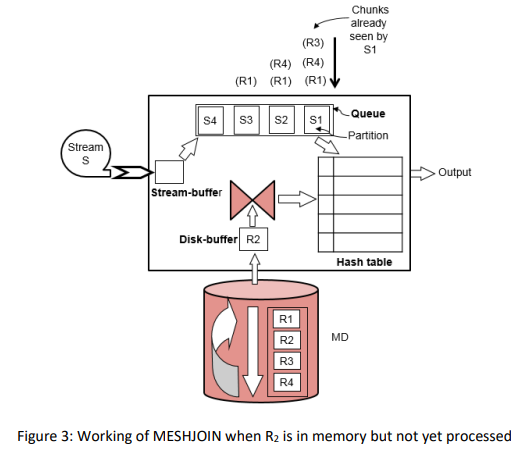
Before doing anything, we need to design a star schema for our warehouse. Following is the start that we are going to use, it includes different dimensions like Dates, Store, Supplier, Customer, Products and a Fact table sales.



**Implementation of Mesh Join algorithm:**

**Steps:**

* First read 50 tuples from transaction data and input it into the hash table which stores these tuples on the basis of product id (meaning key of hash table is product id/customer id) and their joining attributes (Product id,customer id and transaction id) into the queue.
* Load the master data partition. After the last partition the next partition will be the first one.
* Compare the master data partition with the hash table.
* If the master data partition matches with the hash table calculate the required attribute like total sale and store in the transaction tuple.
* Then load that newly transaction tuple into the Data Warehouse but first add the required information (according to the above star schema) into the dimensions table.
* This process is continuously running and new streams from transaction data and new partitions from master data are loaded continuously. But when the queue is full it means that we match all partitions with the hash table.
* Now remove all join attributes value from queue and their corresponding transaction tuple from hash table. This process is repeated continuously until we load all the data into the data warehouse.



**Pseudocode:**

While (true):

Read transaction\_data (50 tuples)

Add to hash table

Add join attribute to queue

Load master data partitions

Compare master data partitions with hash table

If match:

Aggregate and add to DW

If queue. is full ():

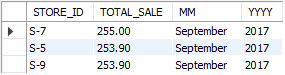
Dequeue ()

Remove corresponding records form hash table

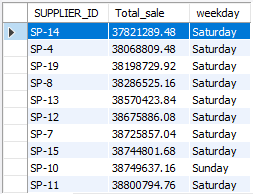
**Note**: in our case we have limited transaction data i.e.,10,000, so we will apply a check in our algorithm i.e., if the transaction limit reaches to 10,000 and completes all comparisons and the queue becomes empty then we break our loop.

**OLAP Queries Results:**

**1. Determine the top 3 store names who generated highest sales in September, 2017.**

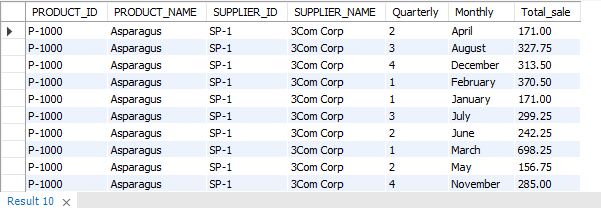
****

**2. Find Top 10 suppliers that generated the most revenue over the weekends. Just Explain how we can forecast the top suppliers for the next weekend?**

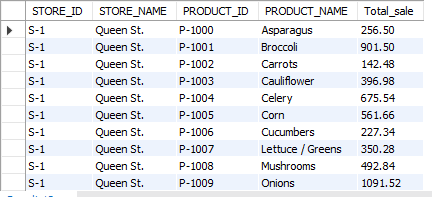
****

We’ll have to check total sales of all suppliers for the weekends for the whole year. On the basis of that we can take average sales on weekends on the basis of suppliers and select the top 10 suppliers with highest average sales.

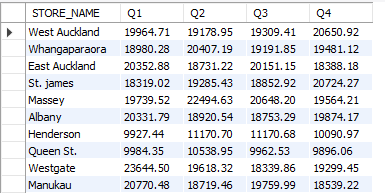
**3. Present total sales of all products supplied by each supplier with respect to quarter and month.**

****

**4. Present total sales of each product sold by each store. The output should be organised store wise and then product wise under each store.**

****

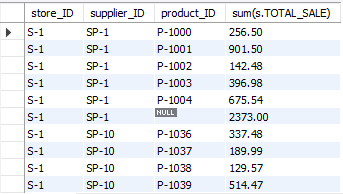
**5. Present the quarterly sales analysis for all stores using drill down query concepts.**

****

**6. Find the 5 most popular products sold over the weekends.**

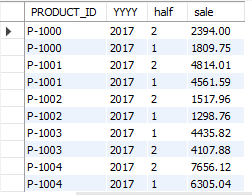
****

**7. Perform ROLLUP operation to store, supplier, and product. Explain your query results in a few lines.**

****

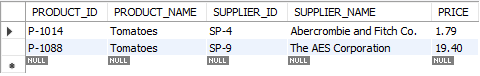
Rollup is used when we have to go from highly detailed data towards less detail. In our case rollup gives the total sale of products by a single supplier in a specific store.

**8. Extract total sales of each product for the first and second half of year 2017 along with its total yearly sales.**

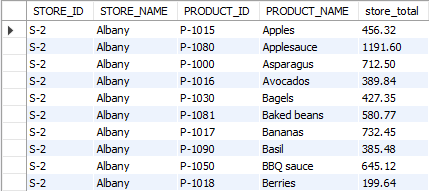
****

**9. Find an anomaly in the data warehouse dataset. Write a query to show that anomaly and explain the anomaly in your project report.**

****

**** Multiple products exist for a single product id. On further checking it was found that tomatoes have multiple productid, supplierid and different prices which causes anomaly.

**10. Create a materialised view with name “STORE\_PRODUCT\_ANALYSIS” that presents store and product wise sales. The results should be ordered by store name and then product name. How does the materialized view help in OLAP query optimisation?**

****

Materialized view helps in scenario when data required is accessed using alot of joins which makes the query expensive. In our scenario,STOREANALYSIS\_MV makes it easier to check store wise product sales without requiring any complex joins.

**Shortcomings in Mesh Join:**

1. Works well with only uniform data. Mesh Join cannot perform as expected when non uniform data is fed to it.
2. Meshjoin keeps all join records in the memory which caused slow query processing in a couple of queries which required multiple joins for analysis.
3. The algo requires different external libraries to work such as for creating hash maps.

**Lessons learned:**

* When reading and writing to the database continuously I get to know that Reading from the database is fast as compared to writing to the database. If we want to implement a real-time ETL, writing to the database will become a big problem.
* The size of dimension tables is small. It helps to improve the performance.
* If we want to change the DBMS, it creates a problem like we need to write SQL queries again.
* This is the first time I interacted with Java and learned alot about it.