Name: Razi Haider

Roll #: 19I-1762

Section: BS(DS)-N

**DATA WAREHOUSE AND BUSINESS INTELLIGENCE**

**Semester Project**

**Report**

* **Reading Database (DB) data**

Initially I ran the DB SQL scripted provided and populated the DB tables. Afterwards, I connected my Java Eclipse IDE with MySQL DB using SQL connectors. This allowed me to access the DB tables.



To execute my queries to read the data from the DB, I used “Statement”. The results of the query (if any) were in a variable of “ResultSet”.



I used a loop to extract data from the DB tables and stored them in variables. Further code will be based on this extracted data.

Text

Description automatically generated

* **MeshJoin implementation**

Even though the data given to us is not in a stream, we are still assuming that it is in streams, so we must use MeshJoin to combine masterdata and transactions data. In this MeshJoin, I iterated 10 masterdata tuples against 50 transaction data. SQL “limits” were used to make sure of it. Limits were incremented after each iteration. Transaction data was stored in a MultiValueMap (Apache) which showed multiple tuples against each Product\_ID as required.



A queue consisting of list of transaction data was used to determine which of the mappings need to be removed from the MultiValueMap and when to do it.



Now, to the joining part. When I am extracting the masterdata in a loop, like the one for transactions data, I create another loop (nested) which matches the data of map and masterdata based on Product\_IDs.



* **DDL**

In this step, for the creation of dimension tables and fact table, I created a separate DDL SQL file and a function in my Java code. Java function is more convenient because all we need to do is run the code and it will first delete the columns if they already exist, and then create all the dimension tables and fact table. I you want, you comment the function and simply run the SQL script.

Text, letter

Description automatically generated

* **Inserting data to Data Warehouse (DW)**

Here I am only inserting data in fact and dimension tables. The fact table will consist of all the joined data (redundancy is not an issue in DW) with the sum (SALES\_MONEY) and foreign keys from dimension tables. The dimension tables, however, will not have redundant data and reach table will have its own key, whether primary or composite primary.

The data is only inserted in the dimension tables if it already does not exist.

Graphical user interface, text

Description automatically generated

* **Problems faced**

There was a problem while inserting supplier names to the DB because one name consisted of “ ’ “. To deal with it I had to replace all the “ ‘ “ in the column. I only replaced it in this column because all the other ones were fine.

* **Draw backs of MeshJoin**

One draw-back is that MeshJoin assumes that the data we have is not skewed, which is not realistic. The data distribution in real life is skewed in most of the scenarios. If we are getting frequent items, then MeshJoin does not use memory to store the most repeated items. This takes a toll on I/O operations cost (According to the research).

The other is that with every iteration, there are some transactions that are not joined. Since we are getting data in stream, we cannot join all transactions, but we can minimize the number of the ones left behind by adjusting the buffer size. This will reduce the number of iterations required (according to my own understanding).

**STAR SCHEMA**

Diagram

Description automatically generated