****

**PROJECT REPORT**

METRO CASH AND CARRY

*Building and Analysing a Near-Real-Time Data Warehouse Prototype for METRO Shopping Store in Pakistan*

TABIDAH USMANI

22I-2070

DS-C

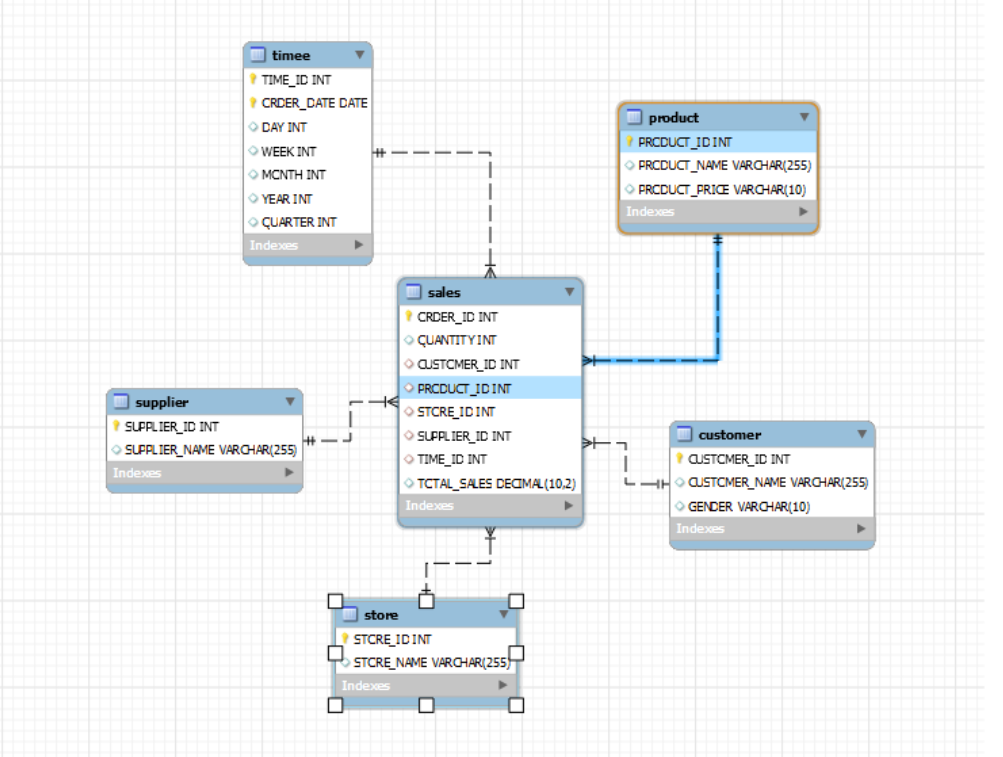
**PROJECT OVERVIEW**

The project involves designing, implementing, and analyzing a near-real-time Data Warehouse (DW) prototype for METRO, a major shopping store chain in Pakistan. This DW will enable real-time analysis of customer shopping behaviors, optimizing strategies like promotions.

Key highlights include:

1. Objective: Build a near-real-time DW using near-real-time ETL (Extraction, Transformation, Loading) processes. Transactions from data sources will be enriched and loaded into DW.
2. Algorithm: Implement an extended version of the MESHJOIN algorithm to integrate streaming transactional data with master data in the transformation phase.
3. Data Structure: Create a star schema with facts and dimensions based on sales scenarios for efficient data organization and analysis.
4. Implementation Tools: Use Java with Eclipse IDE for the ETL process and SQL for defining schema and running OLAP queries.

**SCHEMA FOR DATAWAREHOUSE**

****

**MESHJOIN ALGORITHM**

It is an algorithm used in data integration and real time ETL processes. It is designed to join streaming data that is transactions data over here with master data(customer and products) efficiently, even if master data is large enough to fit in memory.

Instructions are as follows:

1. The master data is split into fixed size partition and a partition is loaded into the disk buffer during each iteration.
2. The incoming transactions are read in segments and store in memory as queue. Each transaction in the queue joins with all the partitions of the master data before being removed from the queue.
3. In every iteration load a partition of the master data into the disk buffer and match each transaction in the queue with current master data. If a match is found generate an enriched transaction with all the other details provided.
4. When new partition are loaded, older transactions in the queue continue to join with newly loaded partitions. This ensures all transactions are joined with the full master data.
5. Once the transaction has joined, it is successful and removed from the queue and loaded into datawarehouse
6. Repeating all this until transactions are processed and all master data partitions are used.

**SHORTCOMINGS IN MESHJOIN**

1. When the amount of transactional data increases, the queue holding it grows large which leads to increased memory usage and processing delays.
2. Transactions in the queue has to wait for all partitions of the master data to processed cyclically which can cause delays for newer transactions if master data is large.

**WHAT DID YOU LEARN FROM THE PROJECT**

I explored the challenges and solutions involved in near real time ETL processes by implementing meshjoin algorithm and how large datasets can be handled through partitions to improve system performance.