**Chapter 1**

**Introduction**

* 1. **Introduction**

A data warehouse is a special type of database.  It is used to store large amounts of data, such as analytics, historical, or customer data, and then build large reports and data mining against it.  It is markedly different from a web-facing or high-transaction database, which typically has many small transactions or pieces of data that are constantly changing, through many 100’s or 1000’s or small user sessions.  These typically execute in speeds on the order of 1/100th of a second, while in data warehouse you have fewer large queries which can take minutes to execute.

Implementing data warehouse could help a company avoid various challenges. In an era of intense competition, it isn’t sufficient to just take decisions alone. It must be taken on time because if you run out of time, you will witness your competitors getting ahead of you in the marathon.

Let’s assume that a super market chain has not implemented a data warehouse and eventually the supermarket finds it very difficult to analyze what products are sold, what is not selling, when does the sale go up, what is the age group of customers who are buying a particular product and several other queries. This is the first step of attracting challenges because a decision has to be made as to whether, a particular product is a hit among 18-25 age group or not. In case it is analyzed that the selling value has subsided, steps have to be taken to analyze the issue surrounding it. For doing such analysis data warehouse helps to a great extent.

OLAP (online analytical processing) is computer processing that enables a user to easily and selectively extract and view data from different points of view. For example, a user can request that data be analyzed to display a spreadsheet showing all of a company's beach ball products sold in Dhaka in the month of July, compare revenue figures with those for the same products in September, and then see a comparison of other product sales in Dhaka in the same time period. To facilitate this kind of analysis, OLAP data is stored in a multidimensional database. Whereas a relational database can be thought of as two-dimensional, a multidimensional database considers each data attribute (such as product, geographic sales region, and time period) as a separate "dimension."

Multidimensional online analytical processing (MOLAP) is a kind of online analytical processing (OLAP) that, like relational online analytical processing (ROLAP), uses a multidimensional data model to analyze data.

Multidimensional online analytical processing (MOLAP) extends OLAP functionality to multidimensional database management systems (MDBMSs). (An MDBMS uses special proprietary techniques to store data in matrix like n- dimensional arrays.) MOLAP’s premise is that multidimensional databases are best suited to manage, store, and analyze multidimensional data. Most of the proprietary techniques used in MDBMSs are derived from engineering fields such as computer- aided design/computer-aided manufacturing (CAD/CAM) and geographic information systems (GlS).

MOLAP provides with the facility of faster query in the data cube. It is also optimal for slicing and dicing operations in the data cube. Using array indexing and using the idea of snowflake schema it can be made faster.

**1.2 Statement of Problem**

MOLAP provides with the facility of faster query for data retrieval. It also provides optimal result for slicing and dicing operations.

There are many techniques for querying in a single MOLAP. But what if user wants summarized information that is related to multiple MOLAPs? What if user wants the very fundamental data rather than the summarized data? Sometimes user may ask for the summarized data or information as well. To answer these questions we have to make query in multiple MOLAPs.

In this thesis, we are trying to propose a new model which is mainly built using the idea of snowflake schema. Here, MOLAP that holds the fundamental data is made by joining 2-D tables using array index reference. Then the summarized MOLAP is made of this fundamental MOLAP. Then a fact table is formed using the data of each summarized MOLAP. We are trying to answer the above mentioned questions using this model.

Our main problem is to implement the proposed model and to find out whether it can answer the above mentioned questions or not. We have to check the efficiency of the model by comparing it with MySQL database.

* 1. **Objectives**

In this thesis, we are trying to propose a new model for faster query processing in multiple MOLAP combined by a fact table. The main objectives of this thesis are:

* To study about Data Warehouse
* To study about Multidimensional Online Analytical Processing(MOLAP)
* To propose a new model for faster query using multiple MOLAPs
* To implement join operation of MOLAP data
* To implement the proposed model
* To determine the time efficiency of the model
  1. **Scope of the thesis**

The term scope defines the area or sphere of work, in short the boundary of work of the use of it. The main task of this thesis is to establish a new model for querying multiple OLAPs in a faster way. This model also facilitates to make query in the individual MOLAP. If one wants fundamental data then he/she has to do query in the MOLAPs which preserve fundamental of basic data. If one wants summarized data then he/she has to do query in the summarized MOLAPs. If one wants most summarized data then he/she has to go for the fact table where most summarized data are preserved.

We are trying to implement the proposed model and compare the performance of the model using MySQL database.

We implement the model using programming language C++. The program is run on Intel® Core™i5 CPU 2.60GHz processor with RAM 4.000 GB in 64 bit windows operation system.

* 1. **Thesis Organization**

We have tried to propose a new model for querying multiple OLAPs in a faster way. In the next chapters, we will discuss about the mechanism and techniques, how it works, the coding implementation details. The overview of the next chapters is mentioned below:

**Chapter 2: Background and Literature review**

In this chapter we have discussed about the background of our work. Such as, about Data Warehouse, Importance of Data Warehouse, On-line Analytical Processing (OLAP), Multidimensional On-line Analytical Processing (MOLAP), Relational On-line Analytical Processing (ROLAP), Hybrid On-line Analytical Processing (HOLAP), Data Cube, different types of Data Cube, Cuboid, different types of Database Schemas such as Star Schema, Snowflake Schema, different types of Join operations in Database.

**Chapter 3: Join Operation implementation in MOLAP data using array index reference**

In this chapter, we will discuss about the whole implementation process that means we will discuss about every step of our methodology. We will discuss about the architecture and structure of our model. We will also discuss each section of our model. We will explain each section of our model briefly.

In this chapter, we will also discuss about how Join operation in implemented and how array index reference is used while implementing the model. We will discuss about how data is summarized and summarized MOLAP is formed. We will also explain how the data is kept in the MOLAPs. The formation of the fact table using the summarized data will also be explained in this chapter. So, this chapter mainly describes how the whole thing works.

**Chapter 4: Experimental Results**

In this chapter, we will discuss about the experimental results of our model. We will discuss about the various output of the queries in our model.

In this chapter, we will show the MySQL results using the same data set which is used in our model. Then we will compare our model with the MySQL output with respect to time. That means, in this section, we will discuss about the time efficiency of our model.

**Chapter 5: Conclusion**

In this chapter, we conduct a conclusion of our thesis work. We have mentioned some limitations of our proposed model in here. Then we have discussed about some recommended future works that we want to develop in future.

**Chapter 2**

**Background and Literature review**

**2.1 Data Warehouse**

Data warehousing provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions. Data warehouse systems are valuable tools in today’s competitive, fast-evolving world. In the last several years, many firms have spent millions of dollars in building enterprise-wide data warehouses. Many people feel that with competition mounting in every industry, data warehousing is the latest must-have marketing weapon – a way to retain customers by learning more about their needs [1].

Data warehouses have been defined in many ways, making it difficult to formulate a rigorous definition. Loosely speaking, a data warehouse refers to a database that is maintained separately from an organization’s operational databases. Data warehouse systems allow for the integration of a variety of application systems. They support information processing by providing a solid platform of consolidated historical data for analysis.

According to William H. Inmon, a leading architect in the construction of data warehouse systems, “A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management’s decision making process.”[1] This short, but comprehensive definition presents the major features of a data warehouse. The four keywords *subject-oriented, integrated, time-variant*, and *nonvolatile*, distinguish data warehouses from other data repository systems, such as relational database systems, transaction processing systems, and file systems. Let’s take a closer look at each of these key features.

* **Subject-oriented:** A data warehouse is organized around major subjects, such as customer, supplier, product, and sales. Rather than concentrating on the day-to-day operations and transaction processing of an organization, a data warehouse focuses on the modeling and analysis of data for decision makers. Hence, data warehouses typically provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process.

* **Integrated:** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and on-line transaction records. Data cleaning and data integration techniques are applied to ensure consistency in naming conventions, encoding structures, attribute measures, and so on.
* **Time-variant:** Data are stored to provide information from a historical perspective (e.g., the past 5–10 years). Every key structure in the data warehouse contains, either implicitly or explicitly, an element of time.
* **Nonvolatile:** A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. Due to this separation, a data warehouse does not require transaction processing, recovery, and concurrency control mechanisms. It usually requires only two operations in data accessing: *initial loading of data* and *access of data*.

In sum, a data warehouse is a semantically consistent data store that serves as a physical implementation of a decision support data model and stores the information on which an enterprise needs to make strategic decisions. A data warehouse is also often viewed as an architecture, constructed by integrating data from multiple heterogeneous sources to support structured and/or ad hoc queries, analytical reporting, and decision making. Because most people are familiar with commercial relational database systems, it is easy to understand what a data warehouse is by comparing these two kinds of systems. The major task of on-line operational database systems is to perform on-line transaction and query processing. These systems are called **on-line transaction processing (OLTP)** systems. They cover most of the day-to-day operations of an organization, such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting. Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse needs of the different users [1].

**2.2 On-line Analytical Processing (OLAP)**

Online Analytical Processing (OLAP) databases facilitate business-intelligence queries. OLAP is a database technology that has been optimized for querying and reporting, instead of processing transactions. The source data for OLAP is Online Transactional Processing (OLTP) databases that are commonly stored in data warehouses. OLAP data is derived from this historical data, and aggregated into structures that permit sophisticated analysis. OLAP data is also organized hierarchically and stored in cubes instead of tables. It is a sophisticated technology that uses multidimensional structures to provide rapid access to data for analysis. This organization makes it easy for a PivotTable report or PivotChart report to display high-level summaries, such as sales totals across an entire country or region, and also display the details for sites where sales are particularly strong or weak.

OLAP databases are designed to speed up the retrieval of data. Because the OLAP server, rather than Microsoft Office Excel, computes the summarized values, less data needs to be sent to Excel when you create or change a report. This approach enables you to work with much larger amounts of source data than you could if the data were organized in a traditional database, where Excel retrieves all of the individual records and then calculates the summarized values.

OLAP databases contain two basic types of data: measures, which are numeric data, the quantities and averages that you use to make informed business decisions, and dimensions, which are the categories that you use to organize these measures. OLAP databases help organize data by many levels of detail, using the same categories that you are familiar with to analyze the data.

**2.3 Multidimensional On-line Analytical Processing (MOLAP)**

MOLAP (multidimensional online analytical processing) is online analytical processing (OLAP) that indexes directly into a multidimensional database. In general, an OLAP application treats data multi-dimensionally; the user is able to view different aspects or facets of data aggregates such as sales by time, geography, and product model. If the data is stored in a relational database, it can be viewed multi-dimensionally, but only by successively accessing and processing a table for each dimension or aspect of a data aggregate. MOLAP processes data that is already stored in a multi-dimensional array in which all possible combinations of data are reflected, each in a cell that can be accessed directly.  The MOLAP storage mode causes the aggregations of the partition and a copy of its source data to be stored in a multidimensional structure in Analysis Services when the partition is processed. This MOLAP structure is highly optimized to maximize query performance. The storage location can be on the computer where the partition is defined or on another computer running Analysis Services. Because a copy of the source data resides in the multidimensional structure, queries can be resolved without accessing the partition's source data. Query response times can be decreased substantially by using aggregations. The data in the partition's MOLAP structure is only as current as the most recent processing of the partition.

Multidimensional On-line Analytical Processing (MOLAP) processed data that is already stored in a multidimensional array in which all possible combinations of data are reflected. It employs multidimensional array as basic data structure [2].

There are two more terms related to this. They are ROLAP (Relational On-line Analytical Processing) and HOLAP (Hybrid On-line Analytical Processing).

The ROLAP storage mode causes the aggregations of the partition to be stored in indexed views in the relational database that was specified in the partition's data source. Unlike the MOLAP storage mode, ROLAP does not cause a copy of the source data to be stored in the Analysis Services data folders. Instead, when results cannot be derived from the query cache, the indexed views in the data source is accessed to answer queries. Query response is generally slower with ROLAP storage than with the MOLAP or HOLAP storage modes. Processing time is also typically slower with ROLAP. However, ROLAP enables users to view data in real time and can save storage space when you are working with large datasets that are infrequently queried, such as purely historical data.

The HOLAP storage mode combines attributes of both MOLAP and ROLAP. Like MOLAP, HOLAP causes the aggregations of the partition to be stored in a multidimensional structure in an SQL Server Analysis Services instance. HOLAP does not cause a copy of the source data to be stored. For queries that access only summary data in the aggregations of a partition, HOLAP is the equivalent of MOLAP. Queries that access source data—for example, if you want to drill down to an atomic cube cell for which there is no aggregation data—must retrieve data from the relational database and will not be as fast as they would be if the source data were stored in the MOLAP structure. With HOLAP storage mode, users will typically experience substantial differences in query times depending upon whether the query can be resolved from cache or aggregations versus from the source data itself.

**2.4 Data Cube**

A data cube refers is a three-dimensional (3D) (or higher) range of values that are generally used to explain the time sequence of an image's data. It is a data abstraction to evaluate aggregated data from a variety of viewpoints. It is also useful for imaging spectroscopy as a spectrally-resolved image is depicted as a 3-D volume. A data cube can also be described as the multidimensional extensions of two-dimensional tables. It can be viewed as a collection of identical 2-D tables stacked upon one another. Data cubes are used to represent data that is too complex to be described by a table of columns and rows. As such, data cubes can go far beyond 3-D to include many more dimensions.

A data cube is generally used to easily interpret data. It is especially useful when representing data together with dimensions as certain measures of business requirements. A cube's every dimension represents certain characteristic of the database, for example, daily, monthly or yearly sales. The data included inside a data cube makes it possible analyze almost all the figures for virtually any or all customers, sales agents, products, and much more. Thus, a data cube can help to establish trends and analyze performance.

**2.5 Snowflake Schema**

The entity-relationship data model is commonly used in the design of relational databases, where a database schema consists of a set of entities and the relationships between them. Such a data model is appropriate for on-line transaction processing. A data warehouse, however, requires a concise, subject-oriented schema that facilitates on-line data analysis.

The most popular data model for a data warehouse is a **multidimensional model**. Such a model can exist in the form of a **star schema**, a **snowflake schema**, or a **fact constellation schema**.

The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in normalized form to reduce redundancies. Such a table is easy to maintain and saves storage space. However, this saving of space is negligible in comparison to the typical magnitude of the fact table. Furthermore, the snowflake structure can reduce the effectiveness of browsing, since more joins will be needed to execute a query. Consequently, the system performance may be adversely impacted. Hence, although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design [1].

In our work, we used the idea of snowflake schema. We have not strictly followed the concrete method of the snowflake schema but have tried to apply the idea in a situation where multiple MOLAPs form a fact table and the summarized data is reserved in the fact table.

**2.6 MOLAP Basic Operations**

There are four types of basic operations [3]:

* Roll-up
* Drill-down
* Slice and dice
* Pivot

**2.7 Join Operation in Database**

Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

SQL Joins are used to relate information in different tables. A Join condition is a part of the SQL query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL [WHERE Clause](http://beginner-sql-tutorial.com/sql-where-clause.htm) of select, update, delete statements [4].

There are mainly four types of join in SQL. They are:

* Inner Join
* Left Join
* Right join
* Full join

**2.8 Related Works**

The main advantage of Main Memory Database (MMDB) is that it is very faster than the secondary memory. Although main memory is very expensive but it gives high performance. Hence the interest of using main memory database is increasing day by day. Powerful CPUs, large RAM encourage us to work with main memory database.

A work has been done for enhancing the performance of data warehouses [5]. There virtual de-normalization has been done using array index reference for Main Memory OLAP. We have got the concept of array index reference from this work.

**Chapter 3**

**Join Operation Implementation in MOLAP Data Using Array Index Reference**

**3.1 Model Description**

For instance, we have shown the whole analysis of our work using a company’s business data.

The company has mainly three sections:

* Selling Section
* Shipping Section
* Production Section

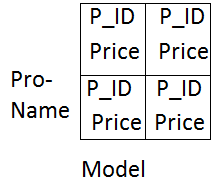
**Selling Section:**

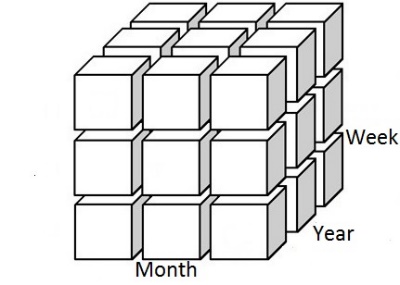
There are two types of MOLAP in the sell section. One is *pre-MOLAP* and the other one is summarized MOLAP. In *pre-MOLAP* data is stored on the basis of three dimensions:

* Week
* Month
* Year

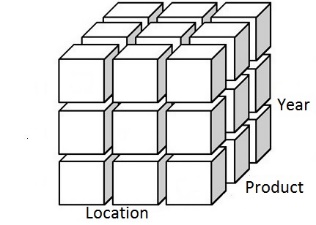
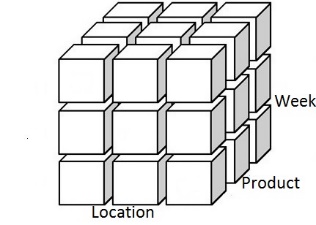
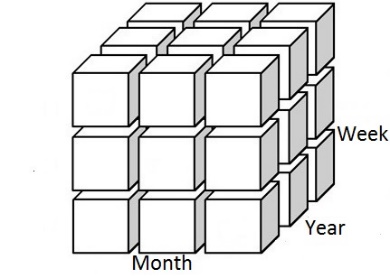
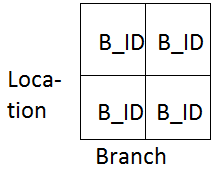
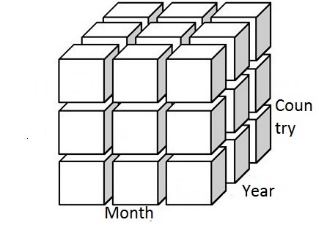
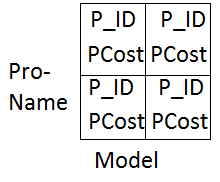
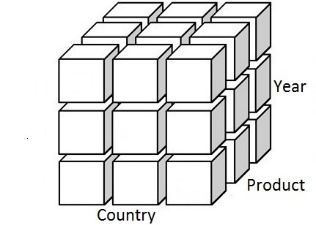
This MOLAP contains the raw data. The main *Selling Section MOLAP* or the summarized data MOLAP for the selling section is constructed from this *pre-MOLAP*. The summarized MOLAP has three dimensions:

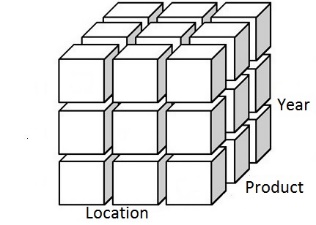
* Production
* Location
* Year

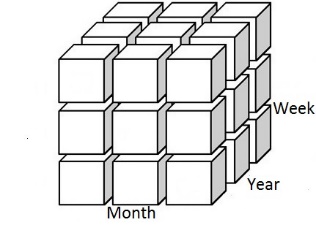
****

****

|  |
| --- |
| Fact Table |

****

****

****

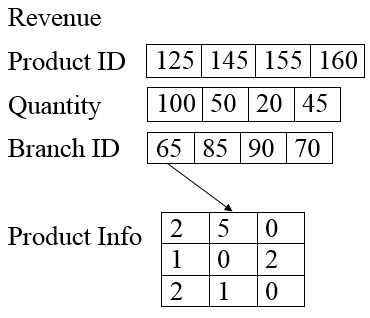
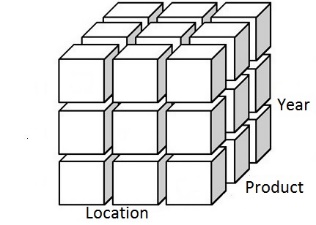
****

Figure 3.2: *Pre-MOLAP* for selling section



Total Quantity

Total Revenue

Figure 3.3: MOLAP for selling section

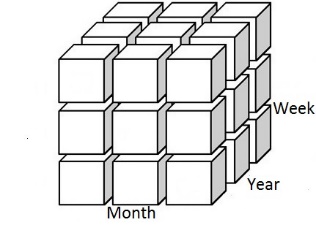
**Shipping Section:**

There are two types of MOLAP in shipping section. The shipping *pre-MOLAP* contains all shipping information on the basis of three dimensions:

* Week
* Month
* Year

The shipping MOLAP has three dimensions. They are:

* Year
* Location
* Product



From Location

Shipping Cost

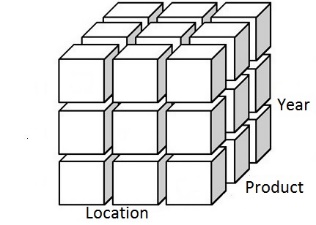
Product ID

Branch ID

Quantity

To Location

Figure 3.4: Shipping *pre\_MOLAP*



Shipping Quantity

Total Shipping Cost

Figure 3.5: Shipping MOLAP

In this Shipping MOLAP one cell contains the total shipping quantity and total shipping cost of a particular product in a particular year in a particular location. Each cell of this MOLAP represent a query value which is very important in respect of time.

**Production Section:**

In production section there are four MOLAPs named Production *pre\_MOLAP*, Available MOLAP*, Production\_cost MOLAP* and Production MOLAP.

The Production *pre\_MOLAP* has three dimensions. They are:

* Country
* Month
* Year

The product quantity which are produced in a particular month, year, country stored in the Production *pre\_MOLAP*.

The Availabe MOLAP has three dimensions like:

* Week
* Product
* Location

Each cell of this MOLAP represents the available quantity in a particular time of a location.

The *Production cost\_MOLAP* has also three dimensions. They are:

* Product Name
* Model Name
* Country

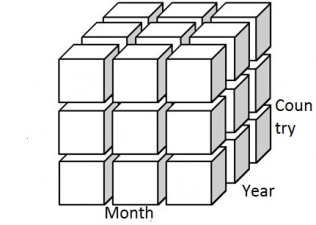
The main Production MOLAP which is constructed by summarizing the production *pre\_MOLAP* has also three dimensions. The name of the dimensions are:

* Country
* Year
* Product

The Production MOLAP contains the production information like number of production, total production cost for each product etc. We can calculate profit, revenue joining Production MOLAP and Selling MOLAP.

The Production *pre\_MOLAP* contains production information on the basis of month where main Production MOLAP contains same information on the basis of year. That means, the Production MOLAP contains summarized information which makes query processing faster than traditional database query.

|  |  |  |  |
| --- | --- | --- | --- |
| 150 | 155 | 135 | 130 |



Product ID

|  |  |  |  |
| --- | --- | --- | --- |
| 30 | 40 | 45 | 50 |

Product Units

|  |  |  |  |
| --- | --- | --- | --- |
| 800 | 900 | 1000 | 1100 |

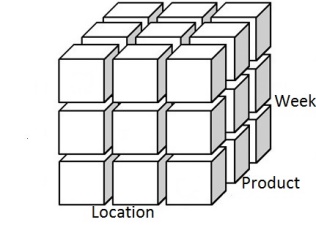
Product Cost

Figure 3.6: Production *pre\_MOLAP*

Production *pre\_MOLAP* holds the raw or fundamental data such as Product ID, Product Units, Product Cost. It gives the total production quantity of a particular product in a particular month of a particular country.

This Production *pre\_MOLAP* is summarized and using this summarized data the main Production MOLAP is formed.

If user wants fundamental data about Product ID, Product Units and Product Cost, query in the Production *pre\_MOLAP* should suffice.



|  |  |  |  |
| --- | --- | --- | --- |
| 140 | 125 | 165 | 155 |

Avail Product ID

|  |  |  |  |
| --- | --- | --- | --- |
| 20 | 30 | 35 | 40 |

Avail Product Unit

|  |  |  |  |
| --- | --- | --- | --- |
| 130 | 125 | 165 | 140 |

Req Product ID

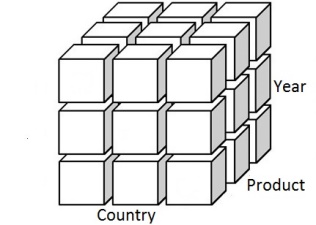
|  |  |  |  |
| --- | --- | --- | --- |
| 15 | 25 | 37 | 43 |

Req Product Unit

Figure 3.7: Available MOLAP

Available MOLAP holds Available Product ID, Available Product Unit, Required Product ID, Required Product Units. These are fundamental or raw data.

The data preserved by this Available MOLAP is used to form the summarized main Production MOLAP. So, the data of the Available MOLAP is summarized and stored in the Production MOLAP.



Total Quantity

Total Production Cost

Figure 3.8: Production MOLAP

Production MOLAP is the summarized MOLAP of the Production Section. It holds Total Quantity and Total Production Cost of a particular product in a particular year in a country.

This Production MOLAP is formed from the two MOLAPs that we have discussed above. They are Production *pre\_MOLAP*, Available MOLAP. Product ID and Product Cost are also mapped from a 2D index reference table named Production Cost table.

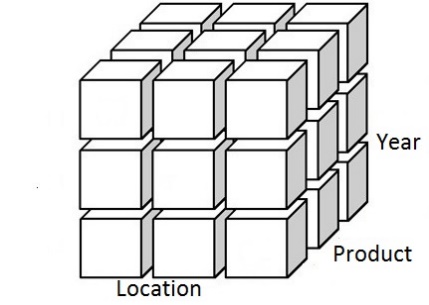
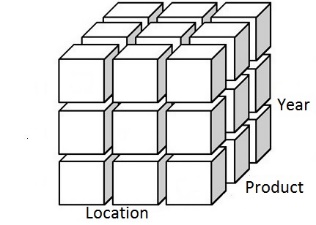
Total Quantity and Total Production Cost of a particular product can be found here. So by joining this MOLAP with the Selling Section summarized MOLAP we can easily get the benefit or loss for a particular product.

**Fact Table**:

This is the most important part of our proposed model. Actually this Fact Table contain

measure values like total sold product quantity, total revenue, total shipping quantity, total shipping cost, total production quantity and total production cost. It is a 2D table and each cell contains the most summarized data extracted from those three main summarized MOLAP.

|  |  |  |
| --- | --- | --- |
| Index | Total Quantity | Total Revenue/Cost |
| 0 |  |  |
| 1 |  |  |
| 2 |  |  |
| 3 | . | . |
| 4 | . | . |
| .. | …… | …….. |
| 27 |  |  |
| 28 |  |  |
| 29 |  |  |
| .. | …… | …….. |
| 54 |  |  |
| 55 |  |  |
| 56 |  |  |
| .. | …… | …….. |

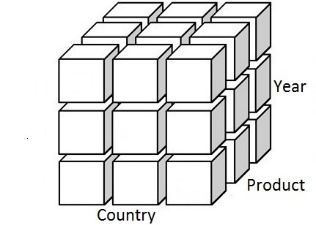
****

Figure 3.9: Fact Table creation from summarized MOLAPs**3.2 Mapping Technique**

We used array index for reference. Each index value maps a particular Location, Product Name, Year etc. Example for mapping month and location is given below:

Month\_Name[0] = January Location\_Name[0] = Dhaka

Month\_Name[1] = February Location\_Name[1] = Khulna

Month\_Name[2] = March Location\_Name[2] = Rajshahi

Month\_Name[3] = April Location\_Name[3] = Chittagong

Month\_Name[4] = May Location\_Name[4] = Sylhet

Month\_Name[5] = June Location\_Name[5] = Barisal

Month\_Name[6] = July Location\_Name[6] = Rangpur

Month\_Name[7] = August Location\_Name[7] = Jessore

Month\_Name[8] = September Location\_Name[8] = Mymensingh

Month\_Name[9] = October Location\_Name[9] = Bogra

Month\_Name[10] = November

Month\_Name[11] = December

The mapping technique works in the following way:

For Selling main MOLAP:

Sell\_MOLAP[0][0][0]

Year[0] = 2007

Location\_Name[0] = Dhaka

Product\_Name[0] = Mobile

For Shipping main MOLAP:

Shipping\_MOLAP[5][2][7]

Year[7] = 2014

Loacation\_Name[2] = Rajshahi

Product\_Name[5] = Fan

For Production main MOLAP:

Production\_MOLAP[4][0][8]

Year[8] = 2015

Country\_Name[0] = Bangladesh

Product\_Name[4] = Motor

**3.3 Implementation of Join Operation**

In traditional querying system of MOLAP, there is no opportunity to join two or more MOLAPs. Only five types of operations named Roll-up, Drill-down, Slice, Cube, Pivot can be performed in MOLAP. But in our proposed model we have tried to implement join operation between multiple MOLAPs so that the querying system becomes convenient and faster.

In typical MOLAP, each cell contains only measure value but no reference. But in our model each cell contains not only the measure value but also a reference. Hence, it is possible to join multiple MOLAPs with respect to different dimensions using that reference value.

A simple Join Operation between Selling *pre\_MOLAP* and Production *pre\_MOLAP* is demonstrated below:

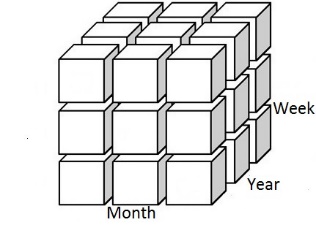


Figure 3.10: Selling *pre\_MOLAP*

Table view for Selling pre\_MOLAP[0][0][0]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product ID | Quantity | Month | Week | Year |
| 125 | 20 | January | Week 1 | 2007 |
| 145 | 25 | January | Week 1 | 2007 |
| 165 | 30 | January | Week 1 | 2007 |
| 135 | 35 | January | Week 1 | 2007 |
| 155 | 25 | January | Week 1 | 2007 |

Table 3.1: Table view for *Selling pre\_MOLAP[0][0][0]*

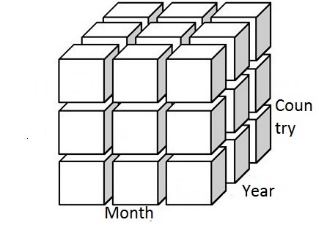


Figure 3.11: *Production pre\_MOLAP*

Table view for Production *pre\_MOLAP[0][0][0]:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product ID | Production Unit | Production Cost($) | Month | Year | Country |
| 125 | 15 | 1000 | January | 2007 | Bangladesh |
| 130 | 30 | 1200 | January | 2007 | Bangladesh |
| 135 | 55 | 1500 | January | 2007 | Bangladesh |
| 140 | 25 | 1700 | January | 2007 | Bangladesh |
| 145 | 45 | 3000 | January | 2007 | Bangladesh |
| 150 | 20 | 2500 | January | 2007 | Bangladesh |
| 155 | 75 | 1300 | January | 2007 | Bangladesh |
| 160 | 60 | 1800 | January | 2007 | Bangladesh |

Table 3.2: Table view for Production *pre\_MOLAP*

Result after Join Operation between the two MOLAPs using Product ID

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Product ID | Quantity | Production Unit | Production  Cost($) | Month | Week | Year | Country |
| 125 | 20 | 15 | 1000 | January | Week 1 | 2007 | Bangladesh |
| 145 | 25 | 45 | 3000 | January | Week 1 | 2007 | Bangladesh |
| 135 | 35 | 55 | 1500 | January | Week 1 | 2007 | Bangladesh |
| 155 | 25 | 75 | 1300 | January | Week 1 | 2007 | Bangladesh |

Table 3.3: Output of Join Operation between *Selling pre\_MOLAP* and *Production pre\_MOLAP*

Another demonstration of Join Operation between shipping *pre\_MOLAP* and *Product Info* *Table* and *Location Branch Table* is shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Model 1 | Model 2 | Model 3 | Model 4 |
| Computer | 125  1000 | 130  2000 | 135  2500 | 170  1700 |
| Mobile | 140  1800 | 145  1900 | 150  2200 | 175  1200 |
| TV | 155  1300 | 160  1400 | 165  2600 | 180  2900 |
| Freeze | 185  1600 | 190  1500 | 195  1100 | 200  2200 |

Table 3.4: Product Info

Each cell of this table contains two values. The first value denotes the *Product ID* and the secondone denotes the *Price* of a particular model of a product.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Branch 1 | Branch 2 | Branch 3 |
| Dhaka | 70 | 80 | 90 |
| Khulna | 75 | 85 | 95 |
| Chittagong | 60 | 65 | 50 |

Table 3.5: Location Info

Table view of *Shipping pre\_MOLAP[1][1][1]:*

|  |  |  |  |
| --- | --- | --- | --- |
| Product ID | From Location ID | To Location ID | Shipping Units |
| 130 | 80 | 50 | 45 |
| 150 | 75 | 60 | 25 |
| 180 | 50 | 90 | 60 |
| 200 | 70 | 95 | 70 |

Table 3.6: Table view of *Shipping pre\_MOLAP*

Output Table after Join Operation:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product ID | Product Name | Model Name | From Location | To Location | Shipping Units |
| 130 | Computer | Model 2 | Dhaka | Chittagong | 45 |
| 150 | Mobile | Model 3 | Khulna | Chittagong | 25 |
| 180 | TV | Model 4 | Chittagong | Dhaka | 60 |
| 200 | Freeze | Model 4 | Dhaka | Khulna | 70 |

Table 3.7: Output of Join Operation between Shipping *pre\_MOLAP* and Reference Tables

**3.4 Selection Operation**

We have implemented basic *Selection* query for MOLAP. The query syntax is partially similar as SQL query syntax. As the fact table contains summarized data like OLTP data, so the query syntax is not completely similar to the SQL query syntax. If we want the total sold product quantity (Computer sold in Dhaka throughout the whole time), then the query syntax will be like:

SELECT quantity of Computer in Dhaka throughout year FROM MOLAP1

Another query example is (total Product quantity sold in Khulna on 2010)

SELECT quantity in Khulna on 2010 throughout product FROM MOLAP1

**Mapping Technique:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | Y | Z | index | X | Z | Y | Index | Y | Z | X | index |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 9 | 1 | 1 | 0 | 18 |
| 1 | 2 | 0 | 1 | 1 | 2 | 0 | 10 | 1 | 2 | 0 | 19 |
| 1 | 3 | 0 | 2 | 1 | 3 | 0 | 11 | 1 | 3 | 0 | 20 |
| 2 | 1 | 0 | 3 | 2 | 1 | 0 | 12 | 2 | 1 | 0 | 21 |
| 2 | 2 | 0 | 4 | 2 | 2 | 0 | 13 | 2 | 2 | 0 | 22 |
| 2 | 3 | 0 | 5 | 2 | 3 | 0 | 14 | 2 | 3 | 0 | 23 |
| 3 | 1 | 0 | 6 | 3 | 1 | 0 | 15 | 3 | 1 | 0 | 24 |
| 3 | 2 | 0 | 7 | 3 | 2 | 0 | 16 | 3 | 2 | 0 | 25 |
| 3 | 3 | 0 | 8 | 3 | 3 | 0 | 17 | 3 | 3 | 0 | 26 |

Table 3.8: Index mapping technique

**Index mapping:**

For first query,

Computer X =1 Dhaka Y=1 Z=0 (As throughout year)

XYZ index

110 0

For second query,

Khulna Y=2 Year Z=3 X=0(As throughout product) XYZ index

023 23

**Chapter 4**

**Experimental Result**

**4.1 Time comparison of Join Operation with MySQL database**

We have compared the required time for performing join operation both in proposed model and MySQL database system. The experimental result is given below:

|  |  |  |
| --- | --- | --- |
| Size of dataset | Time in MySQL (second) | Time in Proposed Model (second) |
| 10000 | 16.25 | 0.4687 |
| 20000 | 17.73 | 0.5156 |
| 50000 | 16.48 | 0.7031 |
| 80000 | 16.64 | 0.8750 |
| 100000 | 16.65 | 1.0156 |
| 130000 | 17.03 | 1.4062 |
| 150000 | 16.84 | 1.3281 |
| 180000 | 17.29 | 1.5156 |
| 200000 | 17.58 | 1.6406 |
| 230000 | 16.73 | 1.8764 |
| 250000 | 16.83 | 2.0669 |
| 280000 | 16.92 | 2.6127 |
| 300000 | 18.47 | 2.5016 |

Table 4.1: Time comparison between MySQL and Proposed Model

Time comparison with MySQL database can’t provide the actual performance of our proposed model. MySQL database stores data in secondary memory where our model works with primary memory and primary memory is many times faster than secondary memory. So time comparison with MySQL database is not so compatible. Experimental result helps us to decide that the performance of the model we proposed is not so bad.

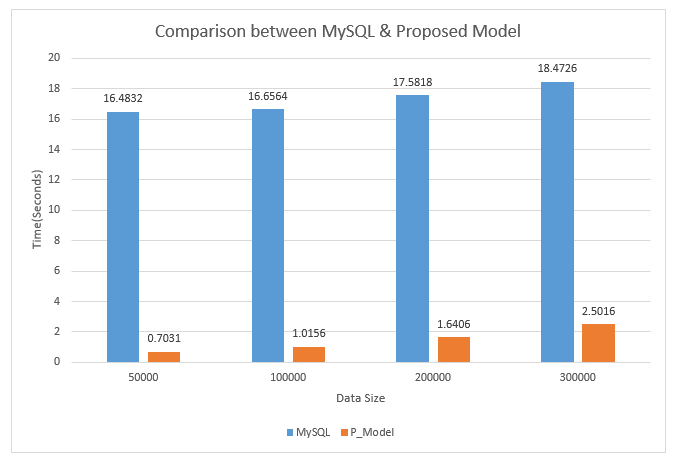


Figure 4.1: Comparison between MySQL and Proposed Model with respect to time

**Chapter 5**

**Conclusion**

**5.1 Summary**

In this thesis, we have proposed a model for performing Join Operation between multiple MOLAPs. We have joined two MOLAPs using array index reference for making the query processing more efficient. Moreover, we have designed the model such a way that query can be made to extract both raw or fundamental data that means less summarized data and summarized data. For this, there are three *pre\_MOLAP*swhich contain raw or fundamental data and three main MOLAPs which contain summarized data. There is a fact table which contains the most summarized data which summarizes the main MOLAPs. Then we have compared the performance of our model with MySQL database.

**5.2 Future Work**

For making the query processing natural, Natural Language Processing (NLP) can be applied. A real life business database can be tested for determining the actual performance of the model. The comparison can be improved by keeping the dataset in the secondary memory as MySQL works with secondary memory.

**5.3 Conclusion**

Data warehousing and Multidimensional On-line Analytical Processing have become a very important issue for storing and handling business data nowadays. It helps in decision making about various facts to a great extent. We have tried to make the query processing in MOLAP more efficient and convenient by proposing our model.

**References**

[1]Jiawei Han and Micheline Kamber, “Data Mining – Concepts and Techniques”, Second Edition, Page 105, 106, 107, 116.

[2]Nilufar Easmin, K. M. Azharul Hasan, Nilufa Parvin, “An Efficient Implementation Scheme for Multidimensional Online Analytical Processing”, 11th International Conference on Computer and Information Technology (ICCIT 2008).

[3] https://www.tutorialspoint.com/dwh/dwh\_olap.htm

[4] http://www.sql-join.com/sql-join-types/

[5] Yansong Zhang, Xuan Zhou, Ying Zhang, Yu Zhang, Mingchuan Su and Shan Wang, “Virtual De-normalization via Array Index Reference for Main Memory OLAP”, IEEE Transactions on Knowledge and Data Engineering, Vol. 28, NO. 4, April 2016.

[6] K. M. Azharul Hasan, T. Tsuji and K. Higuchi, “An efficient implementation for MOLAP basic data structure and its evaluation”, in Proc. Int. Conf. Database Syst. Adv. Appl., Apr 2007, pp. 288-299.

[7].G. M. Hector and S. Kenneth, “Main Memory database systems: An Overview”, IEEE Trans. Knowl. Data Eng., vol. 4, no. 6, pp. 509-516, Dec 1992.

[8] C. Balkesen, J. Teubner, G. Alonso and T. Ozsu, “Main memory hash joins on multicore CPUs: Tuning to the underlying hardware”, Proc. Int. Conf. Data Eng., Apr 2013, pp. 363-373.

[9] S. K. Shin and G. L. Sanders, “De-normalization strategies for data retrieval from data warehouses”, J. Decision Support Syst., Vol. 42, no. 1, pp. 267-282, Oct. 2006

[10] A. Kemper and T. Neumann, “HyPer: A hybrid OLTP & OLAP main memory database system based on virtual memory snapshots”, in Proc. Int. Conf. Data Eng., Apr. 2011, pp. 195-206.

[11] Z. Morteza, P. A. Somnuk, and S. C. Haw, “Optimizing the data warehouse designed by hierarchical denormalizing”, in Proc. Int. Conf. Appl. Comput. Sci., Nov 2008, pp. 131-138.

[12] H. Wang, X. Qin, Y. Zhang, S. Wang and Z. Wang, “LinearDB: A relational approach to make data warehouse scale like mapreduce”, in Proc. 16th Int. Conf. Database Syst. Adv. Appl., Apr 2011, pp. 306-320.

[13] Y. Li and J. M. Patel, “WideTable: An accelerator for analytical data processing”, Proc. VLDB Endowment, vol. 7, no. 10, pp. 907-911, Aug 2014.