International Journal of Advanced Research in IT and Engineering

ISSN: 2278-6244

EFFECT OF CUBE ON QUERY PERFORMANCE IN DATA WAREHOUSE

Ashok Kumar Verma*

Abstract: *A way to improve performance when processing large amounts of data in a data warehouse is to build aggregates. A query answered from base-level data can take hours and involve millions of data records and millions of calculations. With precalculated aggregates, the same query can be answered in seconds with just a few records and calculations.*

Building aggregates from dimensional models is a simple approach because each large fact table can provide numerous aggregates with a predictable structure and a predictable relationship to the base fact table. There are three basic ways to build aggregates from the fact table in this type of model. The first approach is to exclude one or more dimensions when summarizing a fact table. This is the easiest type of aggregate to create because the dimensional data doesn't need to be accessed. The aggregate can also provide significant performance advantages over the base fact table.

Keywords: *Data warehouse, aggregate, Percent performance gain, Memory optimization, Size of cube.*

***M. Tech (Software Engineering), Sri Ram Murty College of Engineering and Technology, Bareilly

INTRODUCTION:

As defined by the Father of Data Warehousing William H. Inmon ,"A Data Warehouse is a subject oriented, integrated, non-volatile and time variant collection of data in support of management's decisions". A data warehouse maintains its functions in three layers: Staging, Integration and Access. Staging layer is used to store raw data for use by developers (analysis and support). Integration layer is used to integrate data and to have a level of abstraction from users. Access layer is for getting data out for users.

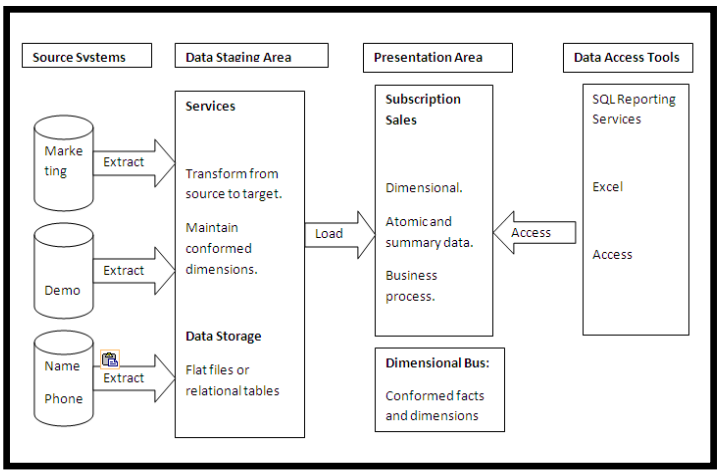
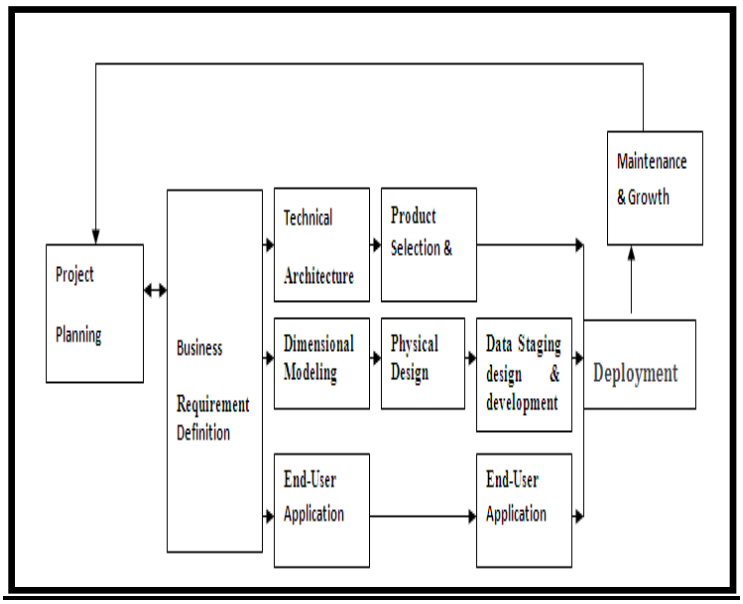


Figure 1.1: Three Layers of Data Warehouse

Data warehousing professionals build and maintain critical warehouse infrastructure to support business and assist business executives in making smart business decisions. Traditional database systems are designed to support typical day-to-day operations via individual user transactions. Such systems are generally called *operational* or *transactional* systems. The data is uploaded from the operational systems. The data may pass through an operational data store for additional operations before it is used in the DW for reporting. A data warehouse *complements* an existing operational system and is therefore designed and subsequently used quite differently. While an operational system is transaction or process-oriented, a data warehouse is *subject-oriented*, geared toward flexible *analytical processing* of high volumes of business data. The main source of the data is cleaned, transformed, catalogued and made available for use by managers and other business professionals for data mining, online analytical processing, market research and decision support. However, the means to retrieve and analyze data, to extract, transform and load data, and to manage the data dictionary are also considered essential components of a data warehousing system. Many references to data warehousing use this broader context. Thus, an expanded

definition for data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.



Kimball-Data Warehouse life cycle diagram

OBJECTIVE OF THIS WORK:

Query performance is one of the most important aspects while designing a Data Warehouse as users will be querying from the Data Warehouse. There is always a tremendous amount of load on the Data Warehouse server as numerous users will be accessing the Data Warehouse simultaneously through various queries. The time it takes for a query response plays a vital role in the functioning of an enterprise and taking strategic decisions. The Data Warehouse should be designed such that the response time of query is minimized and at the same load on the server is reduced.

This work outlines the performance tuning technique applicable to data warehousing environments and illustrates them with the help of a case study of a Insurance Company data warehouse. We will discuss how the creation of cubes can enhance the performance of a query and at the same time it will reduce the load on the Data Warehouse Server. Cubes reduce the size of the data from which the query fetches the result which in turn increases query performance. Various possible cubes have been created for the base cube of Insurance company. Comparisons have been done with the size of the base fact table and cubes. Reduction in the size of the memory and pre-calculated summary in the cubes causes

the query performance to go up significantly. Also the load on the Data Warehouse server is reduced.

The various parameters which have been considered are as below:

I. Number of Records .

II. Percentage Performance gains are taken for each set of data.

III. Memory optimization/requirement of the cubes for each case.

The results have been plotted and compared for various combinations. How they vary with each other has been observed and concluded. The Data Warehouse has been designed using the Microsoft's SQL Analysis Services 2008.

IMPLEMENTATION:

Implementation is perform in the following steps –

Step 1st: In it we perform the implementation and design of "Insurance Company Data Warehouse"

Step 2nd: Ten data sets have been generated for the cube by the Microsoft SQL Analysis Services 2008.

Step 3rd: The data set varies from 10,000 to 2,00,00,000 records.

Step 4th : The cube structure is desined for each data set.

Step 5th : Find the different parameter observation for each Cube.

Step 6th : Comparisons have been done between the size of the base Cube, memory required by the cubes and percentage performance gain obtained through various possible plots of the graphs for different data set and the two groups.

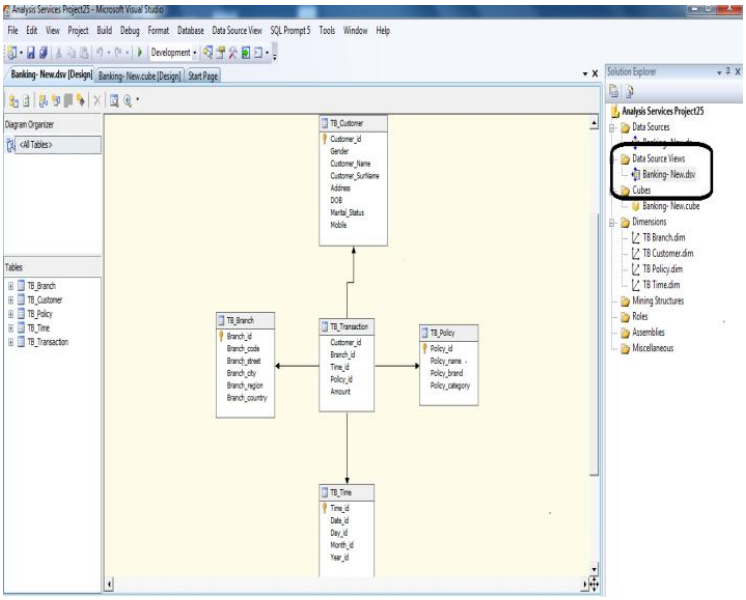
Step 7th: Simulation result has been shown on basis of different parameter -

I. Number of Records in the Cube.

II. Various Percentage Performance gains for each set of data.

III. Memory requirement of the cube for each case

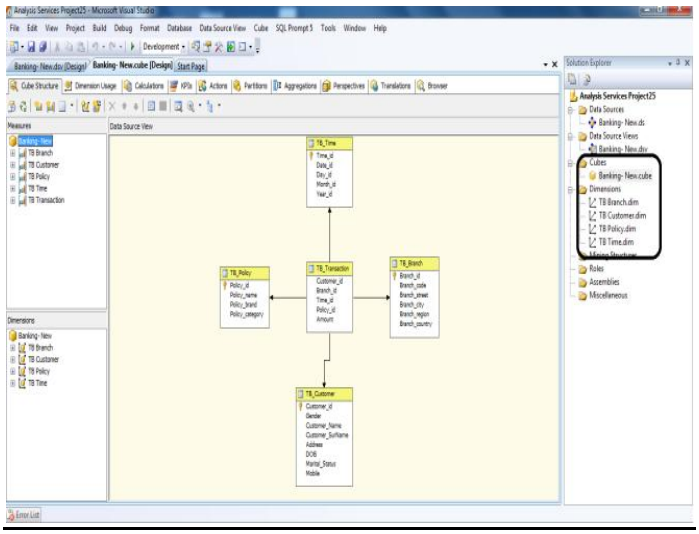
Data Source View of Insurance Company Data Warehouse



Data Source View of Insurance Company Data Warehouse

RESULT:-

A. Cube of Insurance Company Data Warehouse:



B. Comparison of small memory size cubes with memory required for 100% performance gain

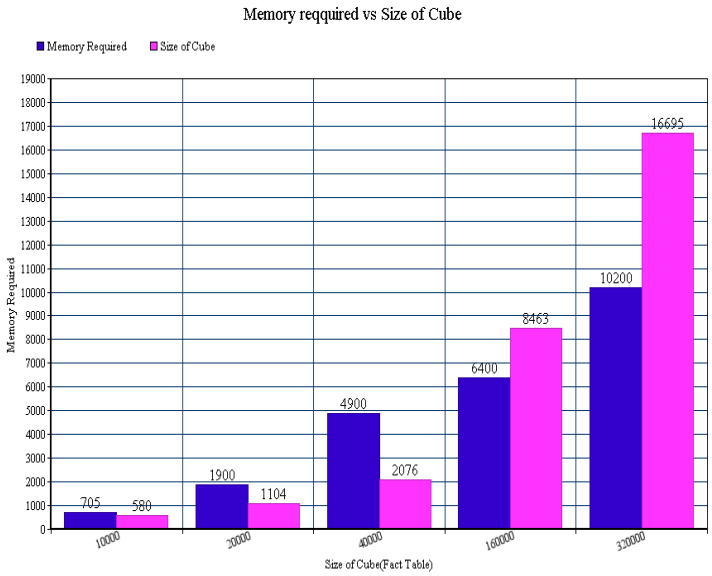


Fig-Comparison of small memory size cubes with memory required (100% performance gain

C. Comparison of large size cubes with memory required for 100% performance gain

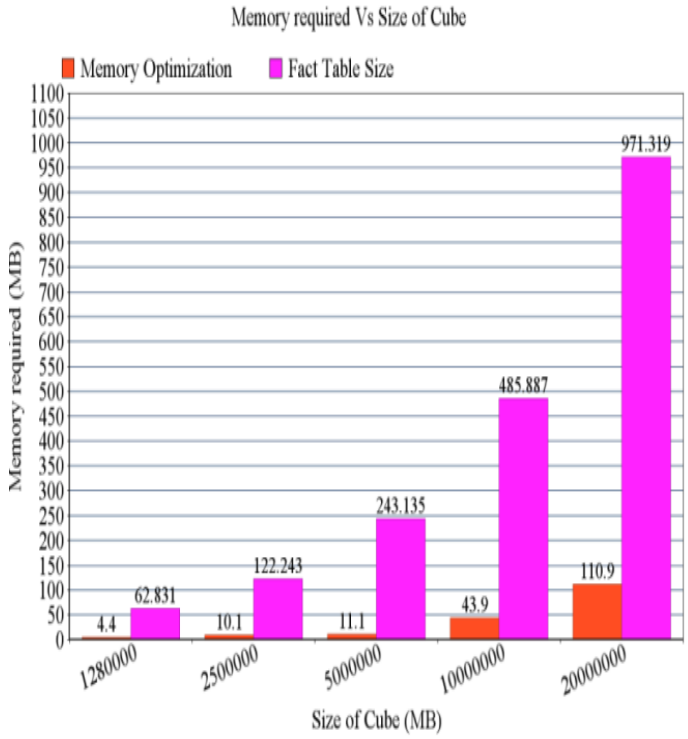


Fig-Comparison of large size cubes with memory required (100% performance gain)

CONCLUSIONS AND FUTURE WORK:

When the data size of the cube is large the memory required for the implementation of cubes to enhance the query performance is lesser than the size of the base cube. This has

been demonstrated and plotted by varying the number of records from 12,80,000 to 20000000 (twenty million) for various performance gains. Therefore if the gain in query performance is the requirement on cubes of large sizes then it is recommended to go for the design and implementation of Aggregates. When the data size of the cube is small (10,000 to 3,20,000) and high performance gain (around 100 percent) is required, then the memory used by the implementation of Aggregates is itself larger than the memory required by the base cube.

REFERENCES

[1] "Building a Data Warehouse" by W.H. Inmon.

[2] Dimension-Join: A New Index for Data Warehouseshttp://www4.wiwiss.fuberlin.de/dblp/resource/record/conf/sbbd/BizarroM010 Opatija, Croatia.

[3] Jane Zhao ," Designing Distributed Data Warehouses and OLAP Systems", Page 254-263.

[4] Joshi.S, Jermaine.C, "Materialized Sample Views for Database," [J] IEEE Transactions on Knowledge and Data Engineering, Volume 20, Issue 3, pp: 337 – 351, March 2008

[5] Ralph Kimball, Laura Reeves, Margy Ross, Warren Thornthwaite "The Data Warehouse Lifecycle: Toolkit Tools and Techniques for Designing, Developing, and Deploying Data Warehouses"

[6] Scott Cameron and Hitachi Consulting" Microsoft SQL Server 2008 Analysis Services ",Chapter 13.

[7] Z.kazi, B.Radulovic, D.Radovanovic and Lj.Kazi," MOLAP Data Warehouse of a Software Products Servicing Call Center", in MIPRO 2010,May 24-28,201