Analysis of Data Warehousing and OLAP Technology

Rutuja Chavan [#1]
Computer Department, Pune University, India
Pratibha College of Commerce and Computer Studies
[1]rutuja.shinde81@gmail.com

Rupali Nehe[#2]
Computer Department, Pune University, India
Pratibha College of Commerce and Computer Studies
[2]rups.nehe@gmail.com

Abstract **- Data warehousing and on-line analytical processing (OLAP) helps business executives to organize, analyze, and use their data for decision making. A data warehouse serves as a sole part of a plan-execute-assess "closed-loop" feedback system for the enterprise management. Data warehouses are widely used in the fields of Financial services, banking services, Consumer goods, Retail sectors, Controlled manufacturing. Many commercial products and services are now available, and all of the principal database management system vendors now have offerings in these areas. Decision support places some rather different requirements on database technology compared to traditional on-line transaction processing applications. This paper provides an analysis of data warehousing and OLAP technologies, with an emphasis on their new requirements. We describe data warehouse architecture, back end tools for extracting, cleaning and loading data into a data warehouse; types of OLAP server, multidimensional data models typical of OLAP; front end client tools for querying and data analysis; server extensions for efficient query processing; and tools for metadata management and for managing the warehouse.**

Keywords— Data warehouse, OLAP, ETL, Server, Data Mart.

I. INTRODUCTION

Data warehousing is a collection of decision support technologies, aimed at enabling the knowledge worker (executive, manager, analyst) to make better and faster decisions. OLAP (or *Online Analytical Processing*) has been growing in popularity due to the increase in data volumes and the recognition of the business value of analytics. Until the mid-nineties, performing OLAP analysis was an extremely costly process mainly restricted to larger organizations. The major OLAP vendor are Hyperion, Cognos, Business Objects, Micro Strategy. The cost per seat were in the range of $1500 to $5000 per annum. The setting up of the environment to perform OLAP analysis would also require substantial investments in time and monetary resources. This has changed as the major database vendor have started to incorporate OLAP modules within their database offering - Microsoft SQL Server 2000 with Analysis Services, Oracle with Express and Darwin, and IBM with DB2. Data warehousing technologies have been successfully deployed in many industries: manufacturing (for order shipment and customer support), retail (for user profiling and inventory management), financial services (for claims analysis, risk analysis, credit card analysis, and fraud detection), transportation (for fleet management), telecommunications (for call analysis and fraud detection), utilities (for power usage analysis), and healthcare (for outcomes analysis This paper presents a roadmap of data warehousing technologies, focusing on the special requirements that data warehouses place on database management systems (DBMSs).The term "Data Warehouse" was first coined by Bill Inmon in 1990. According to Inmon, a data warehouse is a subject oriented, integrated, time-variant, and non-volatile collection of data. This data helps analysts to take informed decisions in an organization A data warehouses provides us generalized and consolidated data in multidimensional view. Along with generalized and consolidated view of data, a data warehouses also provides us Online Analytical Processing (OLAP) tools. These tools help us in interactive and effective analysis of data in a multidimensional space. This analysis results in data generalization and data mining.

Data mining functions such as association, clustering, classification, prediction can be integrated with OLAP operations to enhance the interactive mining of knowledge at multiple level of abstraction. That's why data warehouse has now become an important platform for data analysis and online analytical processing. OLAP, *Online Analytical Processing*, are being used aggressively by organizations to discover valuable business trends from data marts and data warehouses. OLAP provides a historical view of data, although useful when used by itself, OLAP analysis becomes truly powerful when combined with predictive analysis from Data mining. warehouse is typically modeled multidimensionally. Information processing, analytical processing, and data mining are the three types of data warehouse applications that are discussed below:

Information Processing - A data warehouse allows to process the data stored in it. The data can be processed by means of querying, basic statistical analysis, reporting using crosstabs, tables, charts, or graphs.

Analytical Processing - A data warehouse supports analytical processing of the information stored in it. The data can be analyzed by means of basic OLAP operations, including slice-and-dice, drill down, drill up, and pivoting.

Data Mining - Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction. These mining results can be presented using the visualization tools.

Given that operational databases are finely tuned to support known OLTP workloads, trying to execute complex OLAP queries against the operational databases would result in unacceptable performance. Furthermore, decision support requires data that might be missing from the operational databases; for instance, understanding trends or making predictions requires historical data, whereas operational databases stores only current data. Decision support usually requires consolidating data from many heterogeneous sources: these might include external sources such as stock market feeds, in addition to several operational databases.

The different sources might contain data of varying quality, or use inconsistent representations, codes and formats, which have to be reconciled. Finally, supporting the multidimensional data models and operations typical of OLAP requires special data organization, access methods, and implementation methods, not generally provided by commercial DBMSs targeted for OLTP. It is for all these reasons that data warehouses are implemented separately from operational databases. Data warehouses might be implemented on standard or extended relational DBMSs, called Relational OLAP (ROLAP) servers. These servers assume that data is stored in relational databases, and they support extensions to SQL and special access and implementation methods to efficiently implement the multidimensional data model and operations. In contrast, multidimensional OLAP (MOLAP) servers are servers that directly store multidimensional data in special data structures (e.g., arrays) and implement the OLAP operations over these special data structures.

There is more to building and maintaining a data warehouse than selecting an OLAP server and defining a schema and some complex queries for the warehouse. Different architectural alternatives exist. Many organizations want to implement an integrated enterprise warehouse that collects information about all subjects (e.g., customers, products, sales, assets, personnel) spanning the whole organization.

However, building an enterprise warehouse is a long and complex process, requiring extensive business modeling, and may take many years to succeed. Some organizations are settling for data marts instead, which are departmental subsets focused on selected subjects (e.g., a marketing data mart may include customer, product, and sales information).

These data marts enable faster roll out, since they do not require enterprise-wide consensus, but they may lead to complex integration problems in the long run, if a complete business model is not developed.

II. Operations of OLAP Server

Since OLAP servers are based on multidimensional view of data, we will discuss OLAP operations in multidimensional data. Here is the list of OLAP operations:

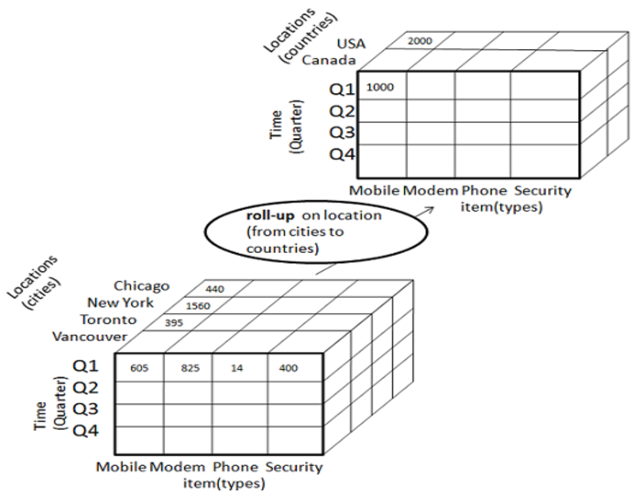
- Roll-up
- Drill-down
- Slice and dice
- Pivot (rotate)

Roll-up

Roll-up performs aggregation on a data cube in any of the following ways:

➢ By climbing up a concept hierarchy for a dimension
➢ By dimension reduction

The following diagram illustrates how roll-up works:



Roll-up is performed by climbing up a concept hierarchy for the dimension location. Initially the concept hierarchy was "street < city < province < country".

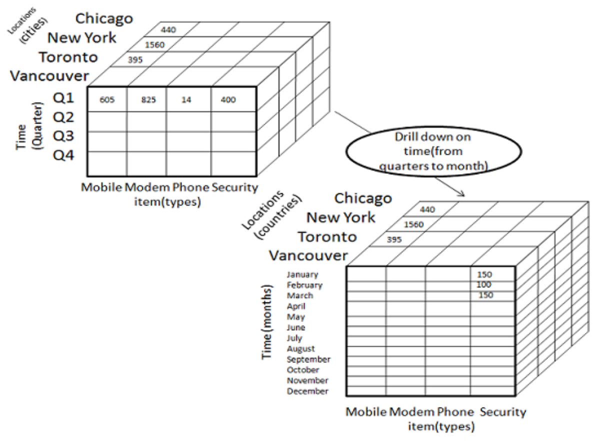
On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country. The data is grouped into cities rather than countries. When roll-up is performed, one or more dimensions from the data cube are removed.

Drill-down

Drill-down is the reverse operation of roll-up. It is performed by either of the following ways:

➢ By stepping down a concept hierarchy for a dimension
➢ By introducing a new dimension.

The following diagram illustrates how drill-down works:



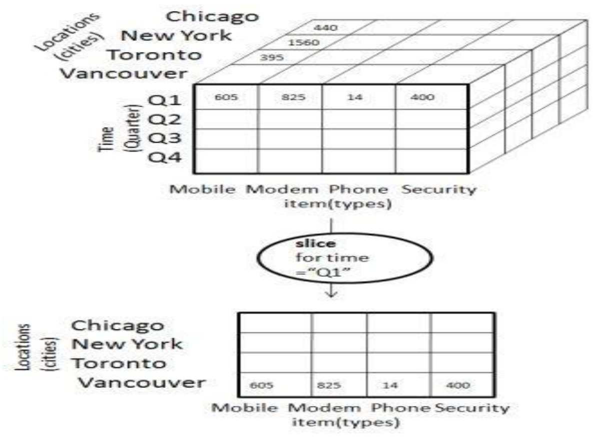
Drill-down is performed by stepping down a concept hierarchy for the dimension time.

- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.

- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.

Slice

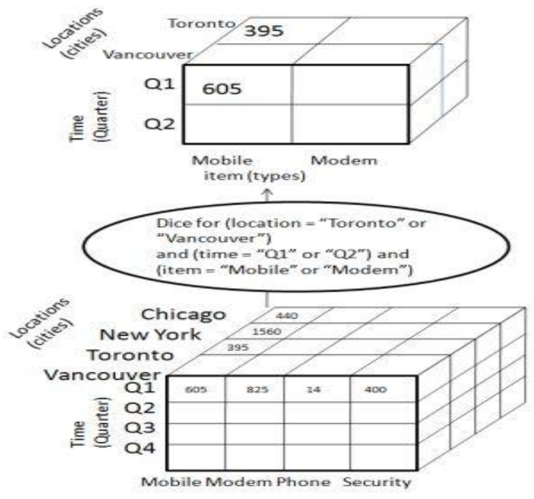
The slice operation selects one particular dimension from a given cube and provides a new sub-cube. Consider the following diagram that shows how slice works.



- Here Slice is performed for the dimension "time" using the criterion time = "Q1".
- It will form a new sub-cube by selecting one or more dimensions.

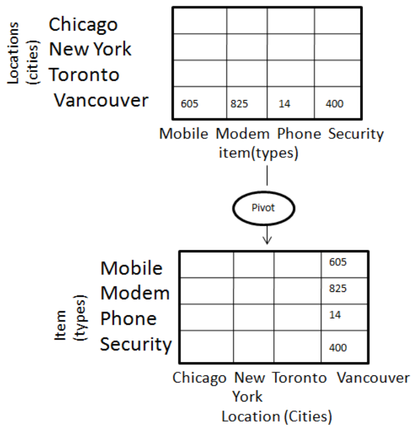
Dice

Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following diagram that shows the dice operation.



Pivot

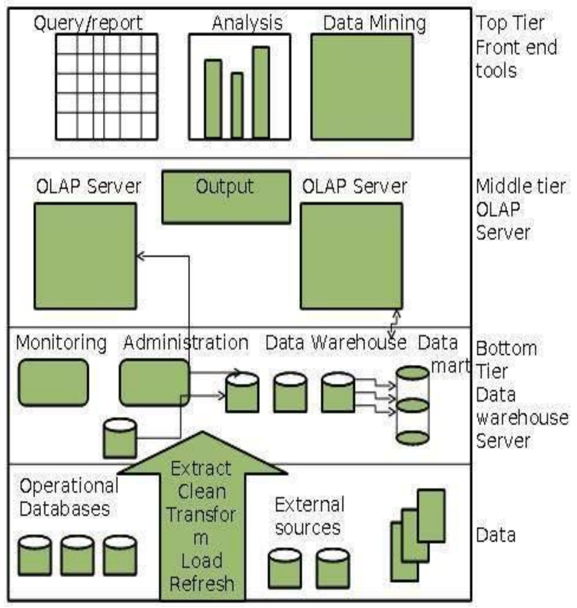
The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data. Consider the following diagram that shows the pivot operation.



In this the item and location axes in 2-D slice are rotated.

III. Architecture of Data Warehouse

The following diagram depicts the three-tier architecture of data warehouse:



It includes tools for extracting data from multiple operational databases and external sources; for cleaning, transforming and integrating this data; for loading data into the data warehouse; and for periodically refreshing the warehouse to reflect updates at the sources and to update data from the warehouse, perhaps on to slower archival storage.

In addition to the main warehouse, there may be several departmental data marts. Data in the warehouse and data marts is stored and managed by one or more warehouse servers, which present multidimensional views of data to a variety of front end tools: query tools, report writers, analysis tools, and data mining tools. Finally, there is a repository for storing and managing metadata, and tools for monitoring and administering the warehousing system.

The warehouse may be distributed for load balancing, scalability, and higher availability. In such a distributed architecture, the metadata repository is usually replicated with each fragment of the warehouse, and the entire warehouse is administered centrally. An alternative architecture, implemented for expediency when it may be too expensive to construct a single logically integrated enterprise warehouse, is a federation of warehouses or data marts, each with its own repository and decentralized administration.

IV. Back End Tools and Utilities

Data warehousing systems use a variety of data extraction and cleaning tools, and load and refresh utilities for populating warehouses. Data extraction from "foreign" sources is usually implemented via gateways and standard interfaces (such as Information Builders EDA/SQL, ODBC, Oracle Open Connect, Sybase Enterprise Connect, and Informix Enterprise Gateway).

1. Data Cleaning

Data cleansing, data cleaning, or data scrubbing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting.

In the business world, incorrect data can be costly. Many companies use customer information databases that record data like contact information, addresses, and preferences. For instance, if the addresses are inconsistent, the company will suffer the cost of resending mail or even losing customers. The profession of forensic accounting and fraud investigating uses data cleansing in preparing its data and is typically done before data is sent to a data warehouse for further investigation. There are packages available so you can cleanse/wash address data while you enter it into your system. This is normally done via an application programming interface (API).

2. Transformation

Data transformations are often the most complex and, in terms of processing time, the most costly part of the extraction, transformation, and loading (ETL) process. They can range from simple data conversions to extremely complex data scrubbing techniques. Many, if not all, data transformations can occur within an Oracle database, although transformations are often implemented outside of the database (for example, on flat files) as well.

The ETL transformation element is responsible for data validation, data accuracy, data type conversion, and business rule application. It is the most complicated of the ETL elements. It may appear to be more efficient to perform some transformations as the data is being extracted (inline transformation); however, an ETL system that uses inline transformations during extraction is less robust and flexible than one that confines transformations to the transformation element. Transformations performed in the OLTP system impose a performance burden on the OLTP database. They also split the transformation logic between two ETL elements and add maintenance complexity when the ETL logic changes.

Tools used in the transformation element vary. Some data validation and data accuracy checking can be accomplished with straightforward Transact-SQL code. More complicated transformations can be implemented using DTS packages. The application of complex business rules often requires the development of sophisticated custom applications in various programming languages. You can use DTS packages to encapsulate multi-step transformations into a single task.

3. Load

After extracting, cleaning and transforming, data must be loaded into the warehouse. Additional preprocessing may still be required: checking integrity constraints; sorting; summarization, aggregation and other computation to build the derived tables stored in the warehouse; building indices and other access paths; and partitioning to multiple target storage areas. Typically, batch load utilities are used for this purpose. In addition to populating the warehouse, a load utility must allow the system administrator to monitor status, to cancel, suspend and resume a load, and to restart after failure with no loss of data integrity.

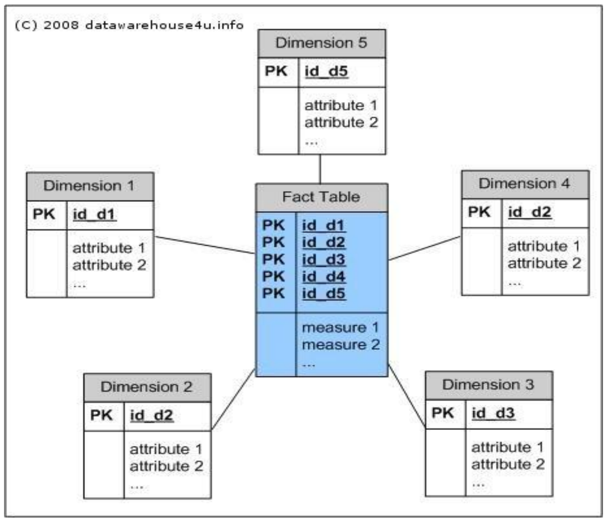
Data warehouses are usually updated periodically rather than continuously, and large numbers of records are often loaded to multiple tables in a single data load. The data warehouse is often taken offline during update operations so that data can be loaded faster and SQL Server 2000 Analysis Services can update OLAP cubes to incorporate the new data. BULK INSERT, bcp, and the Bulk Copy API are the best tools for data loading operations. The design of the loading element should focus on efficiency and performance to minimize the data warehouse offline time

V. Conceptual Model

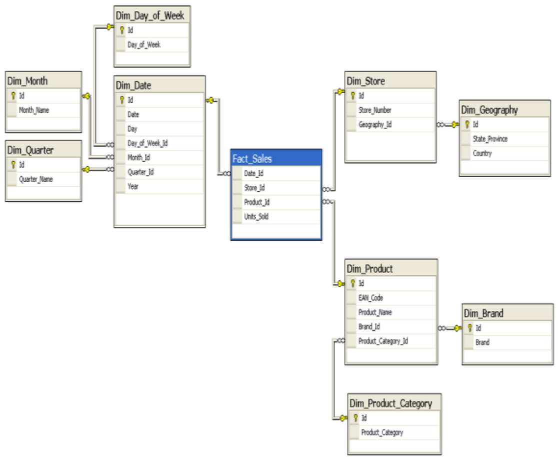
A popular conceptual model that influences the front-end tools, database design, and the query engines for OLAP is the multidimensional view of data in the warehouse. In a multidimensional data model, there is a set of numeric measures that are the objects of analysis. Examples of such measures are sales, budget, revenue, inventory, ROI (return on investment). Each of the numeric measures depends on a set of dimensions, which provide the context for the measure. For example, the dimensions associated with a sale amount can be the city, product name, and the date when the sale was made. The dimensions together are assumed to uniquely determine the measure. Thus, the multidimensional data views a measure as a value in the multidimensional space of dimensions. Each dimension is described by a set of attributes. For example, the Product dimension may consist of four attributes: the category and the industry of the product, year of its introduction, and the average profit margin. For example, the soda Surge belongs to the category beverage and the food industry, was introduced in 1996, and may have an average profit margin of 80%. The attributes of a dimension may be related via a hierarchy of relationships. In the above example, the product name is related to its category and the industry attribute through such a hierarchical relationship.

VI. *Database Design Methodology*

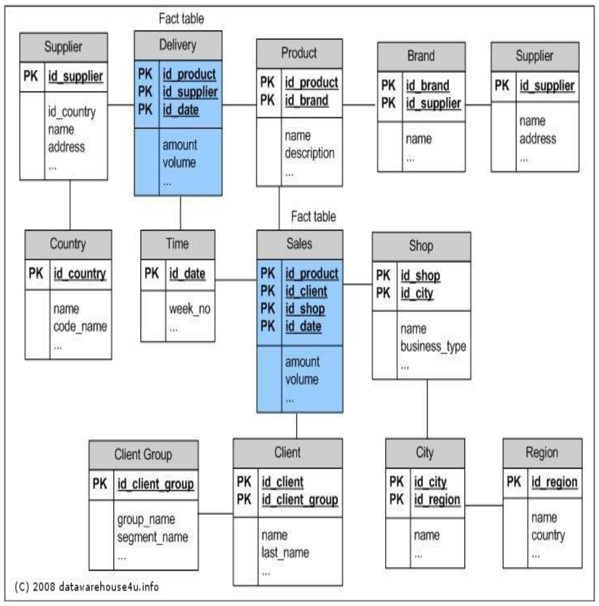
Entity Relationship diagrams and normalization techniques are popularly used for database design in OLTP environments. However, he database designs recommended by ER diagrams are inappropriate for decision support systems where efficiency in querying and in loading data (including incremental loads) are important. Most data warehouses use a star schema to represent the multidimensional data model. The database consists of a single fact table and a single table for each dimension. Each tuple in the fact table consists of a pointer (foreign key – often uses a generated key for efficiency) to each of the dimensions that provide its multidimensional coordinates, and stores the numeric measures for those coordinates. Each dimension table consists of columns that correspond to attributes of the dimension. Figure below shows an example of a star schema.



Star schemas do not explicitly provide support for attribute hierarchies. Snowflake schemas provide a refinement of star schemas where the dimensional hierarchy is explicitly represented by normalizing the dimension tables, as shown in Figure . This leads to dvantages in maintaining the dimension tables. However, the denormalized structure of the dimensional tables in star schemas may be more appropriate for browsing the dimensions.



Fact constellations are examples of more complex structures in which multiple fact tables share dimensional tables as shown in figure below



In addition to the fact and dimension tables, data warehouses store selected summary tables containing pre-aggregated data. In the simplest cases, the pre-aggregated data corresponds to aggregating the fact table on one or more selected dimensions.

VII. *Warehouse Servers*

Data warehouses may contain large volumes of data. To answer queries efficiently, therefore, requires highly efficient access methods and query processing techniques. Several issues arise. First, data warehouses use redundant structures such as indices and materialized views. Choosing which indices to build and which views to materialize is an important physical design problem. The next challenge is to effectively use the existing indices and materialized views to answer queries. Optimization of complex queries is another important problem. Also, while for data-selective queries, efficient index scans may be very effective, data-intensive queries need the use of sequential scans. Thus, improving the efficiency of scans is important. Finally, parallelism needs to be exploited to reduce query response times. In this short paper, it is not possible to elaborate on each of these issues. Therefore, we will only briefly touch upon the highlights.

1. Index Structures and their Usage

A number of query processing techniques that exploit indices are useful. For instance, the selectivity of multiple conditions can be exploited through index intersection. Other useful index operations are union of indexes. These index operations can be used to significantly reduce and in many cases eliminate the need to access the base tables. Warehouse servers can use bit map indices, which support efficient index operations (e.g., union, intersection). Consider a leaf page in an index structure corresponding to a domain value d. Such a leaf page traditionally contains a list of the record ids (RIDs) of records that contain the value d. However, bit map indices use an alternative representation of the above RID list as a bit vector that has one bit for each record, which is set when the domain value for that record is d. In a sense, the bit map index is not a new index structure, but simply an alternative representation of the RID list. The popularity of the bit map index is due to the fact that the bit vector representation of the RID lists can speed up index intersection, union, join, and aggregation11. For example, if we have a query of the form column1 = d & column2 = d', then we can identify the qualifying records by taking the AND of the two bit vectors. While such representations can be very useful for low cardinality domains (e.g., gender), they can also be effective for higher cardinality domains through compression of bitmaps (e.g., run length encoding). Bitmap indices were originally used in Model 204, but many products support them today (e.g., Sybase IQ). An interesting question is to decide on which attributes to index. In general, this is really a question that must be answered by the physical database design process.

In addition to indices on single tables, the specialized nature of star schemas makes join indices especially attractive for decision support. While traditionally indices map the value in a column to a list of rows with that value, a join index maintains the relationships between a foreign key with its matching primary keys. In the context of a star schema, a join index can relate the values of one or more attributes of a dimension table to matching rows in the fact table. Multikey join indices can represent precomputed n-way joins.

2. Materialized Views and their Usage

Many queries over data warehouses require summary data, and, therefore, use aggregates. Hence, in addition to indices, materializing summary data can help to accelerate many common queries. For example, in an investment environment, a large majority of the queries may be based on the performance of the most recent quarter and the current fiscal year. Having summary data on these parameters can significantly speed up query processing.

A simple, but extremely useful, strategy for using a materialized view is to use selection on the materialized view, or rollup on the materialized view by grouping and aggregating on additional columns. For example, assume that a materialized view contains the total sales by quarter for each product. This materialized view can be used to answer a query that requests the total sales of Levi's jeans for the year by first applying the selection and then rolling up from quarters to years. It should be emphasized that the ability to do roll-up from a partially aggregated result, relies on algebraic properties of the aggregating functions In general, there may be several candidate materialized views that can be used to answer a query. If a view V has the same set of dimensions as Q, if the selection clause in Q implies the selection clause in V, and if the group-by columns in V are a subset of the group-by columns in Q, then view V can act as a generator of Q. Given a set of materialized views M, a query Q, we can define a set of minimal generators M' for Q (i.e., smallest set of generators such that all other generators generate some member of M'). There can be multiple minimal generators for a query. For example, given a query that asks for total sales of clothing in Washington State, the following two views are both generators: (a) total sales by each state for each product (b) total sales by each city for each category. The notion of minimal generators can be used by the optimizer to narrow the search for the appropriate materialized view to use. On the commercial side, HP Intelligent Warehouse pioneered the use of the minimal generators to answer queries. While we have defined the notion of a generator in a restricted way, the general problem of optimizing queries in the presence of multiple materialized views is more complex. In the special case of Select-Project-Join queries, there has been some work in this area.

3. Data Partitioning Strategy

Partitioning is done to enhance performance and facilitate easy management of data. Partitioning also helps in balancing the various requirements of the system. It optimizes the hardware performance and simplifies the management of data warehouse by partitioning each fact table into multiple separate partitions.

a. Horizontal Partitioning

There are various ways in which a fact table can be partitioned. In horizontal partitioning, we have to keep in mind the requirements for manageability of the data warehouse.

- **Partitioning by Time into Equal Segments**

In this partitioning strategy, the fact table is partitioned on the basis of time period. Here each time period represents a significant retention period within the business. For example, if the user queries for month to date data then it is appropriate to partition the data into monthly segments. We can reuse the partitioned tables by removing the data in them.

- **Partition by Time into Different-sized Segments**

This kind of partition is done where the aged data is accessed infrequently. It is implemented as a set of small partitions for relatively current data, larger partition for inactive data.

- **Partition on a Different Dimension**

The fact table can also be partitioned on the basis of dimensions other than time such as product group, region, supplier, or any other dimension. Let's have an example.

Suppose a market function has been structured into distinct regional departments like on a state by state basis. If each region wants to query on information captured within its region, it would prove to be more effective to partition the fact table into regional partitions. This will cause the queries to speed up because it does not require to scan information that is not relevant.

- **Partition by Size of Table**

When there are no clear basis for partitioning the fact table on any dimension, then we should partition the fact table on the basis of their size. We can set the predetermined size as a critical point. When the table exceeds the predetermined size, a new table partition is created.

b. Round Robin Partitions

In the round robin technique, when a new partition is needed, the old one is archived. It uses metadata to allow user access tool to refer to the correct table partition.

c. Vertical Partition

Vertical partitioning splits the data vertically. Vertical partitioning can be performed in the following two ways:

- **Normalization**

Normalization is the standard relational method of database organization. In this method, the rows are collapsed into a single row, hence it reduce space.

- **Row Splitting**

Row splitting tends to leave a one-to-one map between partitions. The motive of row splitting is to speed up the access to large table by reducing its size.

VIII. *Data Warehouse Future Aspects*

As we have seen that the size of the open database has grown approximately double its magnitude in the last few years, it shows the significant value that it contains.

As the size of the databases grow, the estimates of what constitutes a very large database continues to grow.

The hardware and software that are available today do not allow to keep a large amount of data online. For example, a Telco call record requires 10TB of data to be kept online, which is just a size of one month's record. If it requires to keep records of sales, marketing customer, employees, etc., then the size will be more than 100 TB.

The record contains textual information and some multimedia data. Multimedia data cannot be easily manipulated as text data. Searching the multimedia data is not an easy task, whereas textual information can be retrieved by the relational software available today.

Apart from size planning, it is complex to build and run data warehouse systems that are ever increasing in size. As the number of users increases, the size of the data warehouse also increases. These users will also require accessing the system. With the growth of the Internet, there is a requirement of users to access data online.

Hence the future shape of data warehouse will be very different from what is being created today.

IX. *Conclusion*

Understanding the importance of having the right data warehouse and good data warehouse design are key to long term success.

In this paper we focus on various database model in warehouse, tool and techniques for managing data in warehouse. The future aspects of data warehouse.

References

1. Inmon, W.H., Building the Data Warehouse. John Wiley, 1992.
2. http://www.olapcouncil.org
3. Codd, E.F., S.B. Codd, C.T. Salley, "Providing OLAP (On-Line Analytical Processing) to User Analyst: An IT Mandate." AvailablefromArborSoftware'swebsite
4. http://www.arborsoft.com/OLAP.html.
5. http://pwp.starnetinc.com/larryg/articles.html
6. Kimball, R. The Data Warehouse Toolkit. John Wiley, 1996.
7. Barclay, T., R. Barnes, J. Gray, P. Sundaresan, "Loading Databases using Dataflow Parallelism." SIGMOD Record, Vol. 23, No. 4, Dec.1994.
8. Blakeley, J.A., N. Coburn, P. Larson. "Updating Derived Relations: Detecting Irrelevant and Autonomously Computable Updates." ACM TODS, Vol.4, No. 3, 1989.
9. Gupta, A., I.S. Mumick, "Maintenance of Materialized Views: Problems, Techniques, and Applications." Data Eng. Bulletin, Vol. 18, No. 2, June 1995.
10. Zhuge, Y., H. Garcia-Molina, J. Hammer, J. Widom, "View Maintenance in a Warehousing Environment, Proc. Of SIGMOD Conf., 1995.
11. Roussopoulos, N., et al., "The Maryland ADMS Project: Views R Us." Data Eng. Bulletin, Vol. 18, No.2, June 1995.
12. O'Neil P., Quass D. "Improved Query Performance with Variant Indices", To appear in Proc. of SIGMOD Conf., 1997.
13. O'Neil P., Graefe G. "Multi-Table Joins through Bitmapped Join Indices" SIGMOD Record, Sep 1995.
14. Harinarayan V., Rajaraman A., Ullman J.D. " Implementing Data Cubes Efficiently" Proc. of SIGMOD Conf., 1996.
15. Chaudhuri S., Krishnamurthy R., Potamianos S., Shim K. "Optimizing Queries with Materialized Views" Intl. Conference on Data Engineering, 1995.
16. Levy A., Mendelzon A., Sagiv Y. "Answering Queries Using Views" Proc. of PODS, 1995.
17. Yang H.Z., Larson P.A. "Query Transformations for PSJ Queries", Proc. of VLDB, 1987.
18. Kim W. "On Optimizing a SQL-like Nested Query" ACM TODS, Sep 1982.
19. Ganski,R., Wong H.K.T., "Optimization of Nested SQL Queries Revisited " Proc. of SIGMOD Conf., 1987.
20. Dayal, U., "Of Nests and Trees: A Unified Approach to Processing Queries that Contain Nested Subqueries, Aggregates and Quantifiers" Proc. VLDB Conf., 1987.
21. Murlaikrishna, "Improved Unnesting Algorithms for Join Aggregate SQL Queries" Proc. VLDB Conf., 1992.
22. Seshadri P., Pirahesh H., Leung T. "Complex Query Decorrelation" Intl. Conference on Data Engineering, 1996.
23. Mumick I.S., Pirahesh H. "Implementation of Magic Sets in Starburst" Proc.of SIGMOD Conf., 1994.
24. Chaudhuri S., Shim K. "Optimizing Queries with Aggregate Views", Proc. of EDBT, 1996.
25. Chaudhuri S., Shim K. "Including Group By in Query Optimization", Proc. of VLDB, 1994.
26. Yan P., Larson P.A. "Eager Aggregation and Lazy Aggregation", Proc. of VLDB, 1995
27. Gupta A., Harinarayan V., Quass D. "Aggregate-Query Processing in Data Warehouse Environments", Proc. of VLDB, 1995.
28. Chaudhuri S., Shim K. "An Overview of Cost-based Optimization of Queries with Aggregates" IEEE Data Enginering Bulletin, Sep 1995.
29. Dewitt D.J., Gray J. "Parallel Database Systems: The Future of High Performance Database Systems" CACM, June 1992.
30. Gray J. et.al. "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab and Sub Totals" Data Mining and Knowledge Discovery Journal, Vol 1, No 1, 1997.
31. Agrawal S. et.al. "On the Computation of Multidimensional Aggregates" Proc. of VLDB Conf., 1996.
32. Kimball R., Strehlo., "Why decision support fails and how to fix it", reprinted in SIGMOD Record, 24(3), 1995.
33. Chatziantoniou D., Ross K. "Querying Multiple Features in Relational Databases" Proc. of VLDB Conf., 1996.
34. Widom, J. "Research Problems in Data Warehousing." Proc. 4th Intl. CIKM Conf., 1995.
35. Wu, M-C., A.P. Buchmann. "Research Issues in Data Warehousing." Submitted for publication.