DATA WAREHOUSING AND OLAP TECHNOLOGY WITH MULTIDIMENSIONAL DATABASE TECHNOLOGY

M.A.Kalyankar*

Prof. S. J. Alaspurkar**

Abstract

Data warehouses are large repositories that integrate data from several sources in an enterprise for analysis. Online analytical processing (OLAP) systems provide fast answers for queries that aggregate large amounts of detail data to find overall trends. Data mining applications seek to discover knowledge by searching semi automatically for previously unknown patterns and relationships in multidimensional databases. This paper is aimed at outlining some open issues in modeling and design of data warehouses. More precisely, issues regarding conceptual models, logical models, methods for design, interoperability, and design for new architectures and applications are considered.

Keywords: Data warehouses, online analytical processing, multidimensional databases, Data mining, interoperability.

* M.E. (CSE), G.H.R.C.E.M. Amravati .

** M.E. Course Coordinator, G.H.R.C.E.M. Amravati

I. INTRODUCTION

Data warehousing is a collection of decision support technologies, aimed at enabling the knowledge worker to make better and faster decisions. The past three years have seen explosive growth, both in the number of products and services offered and in the adoption of these technologies by industry. According to the META Group, the data warehousing market, including hardware, database software, and tools, is projected to grow from $2 billion in 1995 to $8 billion in 1998. Data warehousing technologies have been successfully deployed in many industries: manufacturing, retail, financial services, transportation, telecommunications, utilities, and healthcare. This paper presents a roadmap of data warehousing technologies, focusing on the special requirements that data warehouses place on database management systems (DBMSs). A data warehouse is a "subject-oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making."1 Typically, the data warehouse is maintained separately from the organization's operational databases. There are many reasons for doing this. The data warehouse supports on-line analytical processing (OLAP), the functional and performance requirements of which are quite different from those of the on-line transaction processing (OLTP) applications traditionally supported by the operational databases. OLTP applications typically automate clerical data processing tasks such as order entry and banking transactions that are the bread-and-butter day-to-day operations of an organization. These tasks are structured and repetitive, and consist of short, atomic, isolated transactions. The transactions require detailed, up-to-date data, and read or update a few (tens of) records accessed typically on their primary keys. Operational databases tend to be hundreds of megabytes to gigabytes in size. Consistency and recoverability of the database are critical, and maximizing transaction throughput is the key performance metric. Consequently, the database is designed to reflect the operational semantics of known applications, and, in particular, to minimize concurrency conflicts.

II ARCHITECTURE AND END-TO-END PROCESS

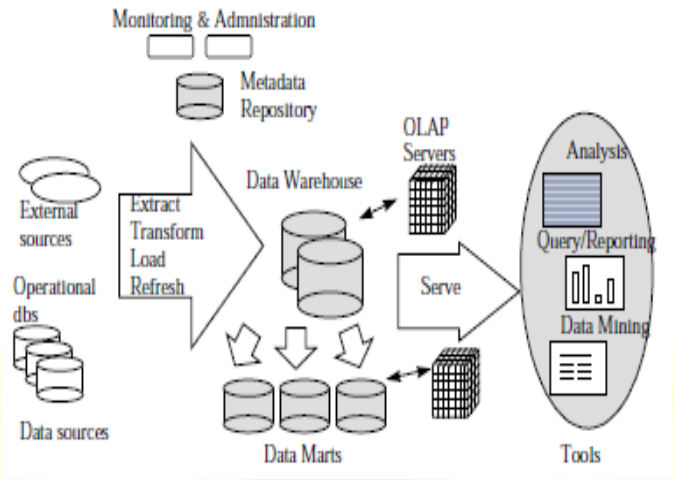

Figure 1. Data Warehousing Architecture

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 purge data from the warehouse, perhaps onto 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.

Designing and rolling out a data warehouse is a complex process, consisting of the following activities.

1. Define the architecture, do capacity planning, and select the storage servers, database and OLAP servers, and Tools.
2. Integrate the servers, storage, and client tools.
3. Design the warehouse schema and views.
4. Define the physical warehouse organization, data placement, partitioning, and access methods.
5. Connect the sources using gateways, ODBC drivers, or other wrappers.
6. Design and implement scripts for data extraction, cleaning, transformation, load, and refresh.
7. Populate the repository with the schema and view definitions, scripts, and other metadata.
8. Design and implement end-user applications.

9. Roll out the warehouse and applications.

III. 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, and Gateway.

Data Cleaning

Since a data warehouse is used for decision making, it is important that the data in the warehouse be correct. However, since large volumes of data from multiple sources are involved, there is a high probability of errors and anomalies in the data.. Therefore, tools that help to detect data anomalies and correct them can have a high payoff. Some examples where data cleaning becomes necessary are: inconsistent field lengths, inconsistent descriptions, inconsistent value assignments, missing entries and violation of integrity constraints.

Load

After extracting, cleaning and transforming, data must be loaded into the warehouse. Additional pre-processing 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.

Refresh

Refreshing a warehouse consists in propagating updates on Source data to correspondingly update the base data and Derived data stored in the warehouse. There are two sets of issues to consider: when to refresh, and how to refresh. Usually, the warehouse is refreshed periodically (e.g., daily or weekly). Only if some OLAP queries need current data (e.g., up to the minute stock quotes), is it necessary to propagate every update. The refresh policy is set by the warehouse administrator, depending on user needs and traffic, and may be different for different sources. Refresh techniques may also depend on the characteristics of the source and the

capabilities of the database servers. Extracting an entire source file or database is usually too expensive, but may be the only choice for legacy data sources. Most contemporary database systems provide replication servers that support incremental techniques for propagating updates from a primary database to one or more replicas. Such replication servers can be used to incrementally refresh a warehouse when the sources change.

IV. CONCEPTUAL MODEL AND FRONT END TOOLS

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. 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

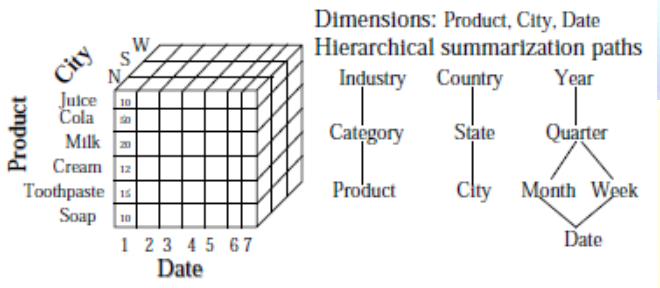

Figure 2. Multidimensional data

Front End Tools

The multidimensional data model grew out of the view of business data popularized by PC spreadsheet programs that were extensively used by business analysts. The spreadsheet is still the most compelling front-end application for OLAP. The challenge in supporting a query environment for OLAP can be crudely summarized as that of supporting spreadsheet operations

efficiently over large multi-gigabyte databases. Indeed, the Essbase product of Arbor Corporation uses Microsoft Excel as the front-end tool for its multidimensional engine. We shall briefly discuss some of the popular operations that are supported by the multidimensional spreadsheet applications. One such operation is pivoting. Consider the multidimensional schema of Figure 2 represented in a spreadsheet where each row corresponds to a sale. Let there be one column for each dimension and an extra column that represents the amount of sale. The simplest view of pivoting is that it selects two dimensions that are used to aggregate a measure, e.g., sales in the above example. The aggregated values are often displayed in a grid where each value in the (x,y) coordinate corresponds to the aggregated value of the measure when the first dimension has the value x and the second dimension has the value y. Thus, in our example, if the selected dimensions are city and year, then the x-axis may represent all values of city and the y-axis may represent the years. The point (x,y) will represent the aggregated sales for city x in the year y. Thus, what were values in the original spreadsheets have now become row and column headers in the pivoted spreadsheet. Other operators related to pivoting are rollup or drill-down. Rollup corresponds to taking the current data object and doing a further group-by on one of the dimensions. Thus, it is possible to roll-up the sales data, perhaps already aggregated on city, additionally by product. The drill-down operation is the converse of rollup. Slice_and_dice corresponds to reducing the dimensionality of the data, i.e., taking a projection of the data on a subset of dimensions for selected values of the other dimensions. For example, we can slice_and_dice sales data for a specific product to create a table that consists of the dimensions city and the day of sale. The other popular operators include ranking (sorting), selections and defining computed attributes. Although the multidimensional spreadsheet has attracted a lot of interest since it empowers the end user to analyse business data, this has not replaced traditional analysis by means of a managed query environment. These environments use stored procedures and predefined complex queries to provide packaged analysis tools. Such tools often make it possible for the end-user to query in terms of domain-specific business data. These applications often use raw data access tools and optimize the access patterns depending on the back end database server. In addition, there are query environments (e.g., Microsoft Access) that help build ad hoc SQL queries by “pointing-and-clicking”. Finally, there are a variety of data mining tools that are often used as front end tools to data warehouses.

V. IMPLEMENTATIONS

Multidimensional database implementations take two basic forms:

1. Multidimensional online analytical processing stores data on disks in specialized multidimensional structures. MOLAP systems typically include provisions for handling sparse arrays and apply advanced indexing and hashing to locate the data when performing queries.

2.Relational OLAP (ROLAP) systems5 use relational database technology for storing data, and they also employ specialized index structures, such as bit-mapped indices, to achieve good query performance. MOLAP systems generally provide more space efficient storage as well as faster query response times.

In a multidimensional database, measures take on different

values for various dimension combinations. The "Achieving Fast Query Response Time" side bar outlines some of the techniques used to accomplish this. ROLAP systems typically scale better in the number of facts they can store (although some MOLAP tools are now becoming just as scalable), are more flexible with respect to cube redefinitions, and provide better support for frequent updates. The virtues of the two approaches are combined in the hybrid OLAP approach, which uses MOLAP technology to store higher-level summary data and ROLAP systems to store the detail data. ROLAP implementations typically employ star or snowflake schemas,5 both of which store data in fact tables and dimension tables. A fact table holds one row for each fact in the cube. It has a column for each measure, containing the measure value for the particular fact, as well as a column for each dimension that contains a foreign key referencing a dimension table for the particular dimension. Star and snowflake schemas differ in how they handle dimensions, and choosing between them largely depends on the desired properties of the system being developed. As Figure 3 shows, a star schema has one table for each dimension. The dimension table contains a key column, one column for each dimension level containing textual descriptions of that level's values, and one column for each level property in the dimension. The star schema's fact table holds the sales price for one particular sale and its related dimension values. It has a foreign key column for each of the three dimensions: product, location, and time. The dimension tables have corresponding key columns and one column for each dimension level—for example, LocID, City, and Country. No column is necessary for the T level, which will always hold the same value. The dimension table's key column is typically a dummy integer key

without any semantics. This prevents misuse of keys, offers better storage use, and provides more support for dimension updates than information- bearing keys from the source systems. Redundancy will occur in higher-level data. For example, because May 2001 has 31 day values, the year value "2001" will be repeated 30 times. Because dimensions typically only take up one to five percent of a cube's total required storage, however, redundancy is not a storage problem. Also, the central handling of dimension updates ensures consistency. Thus, using de-normalized dimension tables, which support a simpler formulation of better-performing queries, is often beneficial. Snowflake schemas contain one table for each dimension level to avoid redundancy, which may be advantageous in some situations. The dimension tables each contain a key, a column holding textual descriptions of the level values, and possibly columns for level properties.

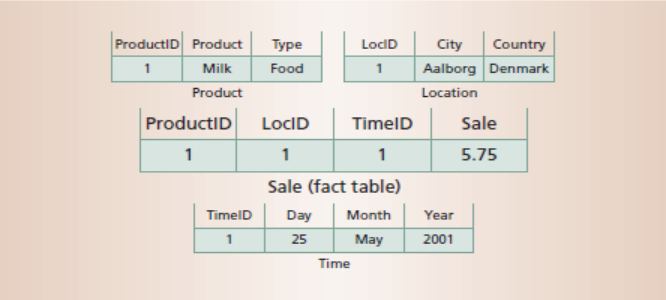

Figure 3. Star schema for sample sales cube. Information from all levels in a dimension is stored in one dimension table—for example, product names and product types are both stored in the Product table.

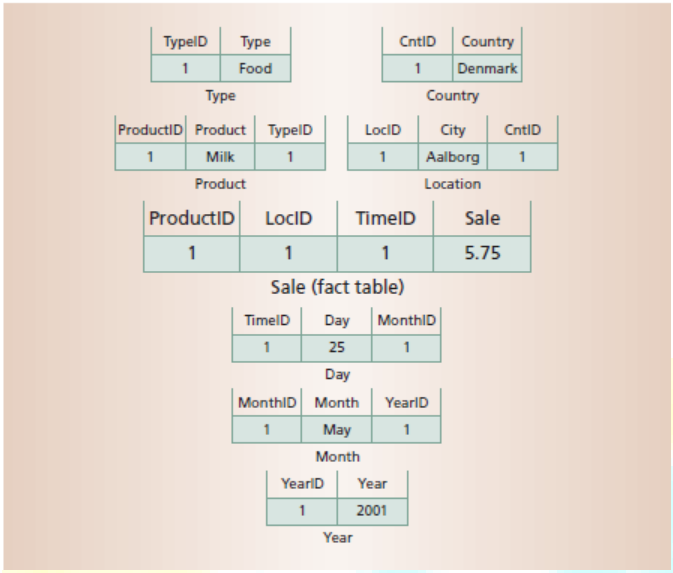

Figure 4. Snowflake schema for sample sales cube. Information from different levels in a dimension is stored in different tables—for example, product names and product types are stored in the Product and Type tables, respectively.

Complex Multidimensional Data

Traditional multidimensional data models and implementation techniques assume that all facts map directly to the lowest-level dimension values and only to one value in each dimension, and dimension hierarchies are balanced trees. When these assumptions fail, however, standard models and systems do not adequately support the desired applications. Complex multidimensional data is especially problematic because it is not summarizable— higher-level aggregate results cannot be derived from lower-level aggregate results. Queries on lower-level results will provide the wrong results or pre-computing, storing, and subsequently reusing lower-level results to compute higher-level results is no longer possible. Aggregates must instead be calculated directly from base data, which considerably increases computational costs. Summarizability requires distributive aggregate functions and dimension hierarchy values. Informally, a dimension hierarchy is strict if no dimension value has more than one direct parent, onto if the

Achieving Fast Query Response Time

The most essential performance-enhancing techniques in multidimensional databases are pre-computation and its more specialized cousin, pre-aggregation, which enable response times to queries involving potentially huge amounts of data to be fast enough to allow interactive data analysis. Computing and storing, or materializing, a product's total sale by country and month is one application of pre-aggregation. This enables fast answers to queries that ask for the total sales—for example, by month alone, by country alone, or by quarter and country in combination. These answers can be derived entirely from the pre-computed results without needing to access bulks of data in the data warehouse. The latest versions of commercial relational database products, as well as dedicated multidimensional systems, offer query optimization based on pre-computed aggregates and automatic maintenance of stored aggregates during updating of base data. Full pre-aggregation—materializing all combinations of aggregates— is infeasible because it takes too much storage and initial computation time. Instead, modern OLAP systems adopt the practical pre-aggregation approach of materializing only select combinations of aggregates and then reusing these to efficiently compute other aggregates. Reusing aggregates requires a well-behaved multidimensional data structure. hierarchy is balanced, and covering if no containment path skips a level. Intuitively, this means that dimension hierarchies must be balanced trees.

VI. DESIGN FOR NEW ARCHITECTURESAND APPLICATIONS

Advanced architectures for business intelligence are emerging to support new kinds of applications, possibly involving new and more complex data types. The modeling and design techniques devised so far are mainly targeted towards traditional business applications, and aimed at managing simple alphanumerical data. Thus, it appears inevitable that more general, broader techniques will have to be devised. In this section we discuss the impact of some of the new applications and architectures on modeling and design; other related topics that we do not address here due to space constraints are active DWs and DWs for the life sciences.

1. Spatial data warehousing

Spatial DWs are characterized by a strong emphasis on

spatial data, coming in the form of spatial dimensions or

spatial measures. Several works, like [1,8], show the advantages of using Geographic Information Systems (GIS) characteristics in the analysis of multidimensional data in specific

domains. Other works, like [5, 8], implemented more general systems mixing GIS and OLAP. While all existing conceptual models support basic modeling of a spatial dimension (e.g., most business DWs include a geographic hierarchy built on customers), location data are usually represented in an alphanumeric format. Conversely, picking a more expressive and intuitive representation for these data would reveal patterns that are difficult to discover otherwise. Preliminary approaches to conceptual modeling for spatial DWs are proposed in [7, 4], where multidimensional models are extended with spatial dimensions, spatial hierarchies, and spatial measures. Also topological relationships and operators such as intersect and inside as well as user-defined aggregate functions are included to augment the expressivity of these models. From the point of view of logical modeling, the main issue raised by spatial warehousing is how to seamlessly integrate the classical ROLAP and MOLAP solutions (e.g., the star schema) with the specialized data structures used in GISs while preserving high-level performance. In this line, [5] investigates the definition of mappings between the geographical dimension of an OLAP tool and a GIS. Finally, as concerns design methods, adequate solutions for properly moving from conceptual to logical schemata in presence of spatial information must be devised.

2 Web warehousing

Web warehouses are DWs that collect Web data. The characteristics of the Web raise new difficulties, mainly due to the semi-structured nature of data, to the lack of control over the sources, and to the frequency of changes on them. The main challenges in this field are how to integrate heterogeneous web sources and how to automate the process of conceptual design when some or most data sources reside on the Web. Some attempts have been made in this direction, mainly aimed at building a conceptual schema from XML data [3, 8]. In other approaches, like [9], the design of the Web warehouse is driven by frequent user queries and by data quality. Importantly, the development of the Semantic Web opens new exciting possibilities since knowledge is represented according to formal ontology capable of expressing semantic relationships, which will allow more powerful methods for conceptual design and for data integration to be devised.

3 Real-time data warehousing and BPM

As DW systems provide an integrated view of an enterprise, they represent an ideal starting point to build a platform for business process monitoring (BPM). However, performing BPM on top of a DW has a deep impact on design and modeling, since BPM requires extended architectures that

may include components not present on standard DW architectures and may be fed by non-standard types of data (such as data streams). In particular, the fact that BPM implies real-time requirements leads to rethinking ETL components, making the ETL design techniques devised so far questionable. In addition, achieving satisfactory performance for continuous monitoring queries will require more sophisticated logical models for storing data cubes. Arising design issues are summarized in [2]:

1. Right-time design. While strict real-time will not actually be needed for most applications, data processing must take place in so-called right-time, meaning that information must be ready and complete not later than required by the decision-making process. Thus, a relevant problem for the designer is to understand what is the right-time for the specific business domain.

2.KPI and rule design. BPM architectures typically include dashboards for viewing key performance indicators (KPIs) and inference engines for managing business rules aimed at giving the decision maker an accurate and timely picture of the business. Hence, suitable techniques for modeling and designing KPIs and business rules, capable of establishing a conceptual connection with the related business goals and of coping with quickly changing requirements, will be necessary.

3.Process design. In BPM a leading role is played by processes. Hence, BPM design also requires to understand business processes and their relationships in order to find out the relevant KPIs and rules, and to determine where the data to compute them can be found.

4. Distributed data warehousing

As in distributed databases, in distributed data warehousing a new phase needs to be added to the design method: the one for designing the distribution, from both the architectural and the physical points of view. During architectural design, general decisions will be taken about which distribution paradigm (P2P, federation, grid) better suits the requirements, how to deploy the DW on the infrastructure, which communication protocols to use, etc. For example, [2] makes the case for a P2P infrastructure for warehousing XML resources, whereas reports how DW systems can be deployed on a grid. On the other hand, the physical point of view mainly addresses how to fragment the DW and how to allocate fragments on the different sites in order to maximize local references to data and to take advantage of the intrinsic parallelism arising from distribution, thus optimizing the overall performance. Though some approaches to fragmentation of DWs have been tempted [1, 11], they are mainly aimed at exploiting local parallelism or at designing ad hoc

view fragments for a given workload. Indeed, distribution is particularly useful in contexts where new data marts are often added, typically because of company mergers or acquisitions. In this case, the most relevant issue is related to integration of heterogeneous data.

VII.CONCLUSION

In this paper we have discussed open issues related to modeling and design of DWs. It is apparent that, though these topics have been investigated for about a decade, several important challenges still arise. Furthermore, ad hoc techniques are required for dealing with the emerging applications of data warehousing and with advanced architectures for business intelligence. Besides, the need for real-time data processing raises original issues that were not addressed within traditional periodically-refreshed DWs. Thus, overall, we believe that research on DW modeling and design is far from being dead, partly because more sophisticated techniques are needed for solving known problems, partly because of the new problems raised during the adaptation of DWS to the peculiar requirements of today's business.

REFERENCES

[1] M. Scotch and B. Parmano. SOVAT: Spatial OLAP visualization and analysis tool. In Proc. HICSS, 2005.

[2] M. Golfarelli, S. Rizzi, and I. Cella. Beyond data warehousing: What's next in business intelligence? In Proc. DOLAP, pages 1–6, 2004.

[3] M. R. Jensen, T. H. Møller, and T. B. Pedersen. Converting XML DTDs to UML diagrams for conceptual data integration. Data & Knowledge

Engineering, 44(3):323–346, 2003.

[4] S. Bimonte, A. Tchounikine, and M. Miquel. Towards a spatial multidimensional model. In Proc. DOLAP, pages 39 46, 2005.

[5] E. Pourabbas. Cooperation with geographic databases. In M. Rafanelli, editor, Multidimensional databases: Problems and solutions, pages 393–432. Idea Group, 2003.

[7] E. Malinowski and E. Zim´anyi. Representing spatiality in a conceptual multidimensional model. In ACM Int. Work. on GIS, pages 12–22, 2004.

[8] J. Park and C. Hwang. A design and practical use of spatial data warehouse. In Proc. IEEE Int. Geoscience and Remote Sensing Symp., pages 726– 729, 2005.

[9] J. Zhang, T. W. Ling, R. Bruckner, and A. M. Tjoa. Building XML data warehouse based on frequent patterns in user queries. In Proc. DaWaK, pages 99–108, 2003.

[10] D. Munneke, K. Wahlstrom, and M. Mohania. Fragmentation of multidimensional databases. In Proc. Australasian Database Conference, pages 153–164, 1999.

[11] M. Golfarelli, V. Maniezzo, and S. Rizzi. Materialization of fragmented views in multidimensional databases. Data & Knowledge Engineering, 49(3):325–351, 2004.