Contents

CHAPTER 1

Data Warehousing: An Introduction

1.1 CHARACTERISTICS OF A DATA WAREHOUSE

A *data warehouse* is a repository of subjectively selected and adapted operational data, which can successfully answer any ad hoc, complex, statistical or analytical queries. It is situated at the centre of a decision support system (DSS) of an organization (or corporation) and contains integrated historical data, both summarized and detailed information—common to the entire organization (Hammergren, 1997). Data warehousing technology is becoming essential for effective business intelligence, business strategy formulation and implementation in a globally competitive environment, wherein larger and larger amounts of data (doubling every 18 months) are required to be processed faster and faster (in a few seconds) for comprehension of its real meaning and impact. Data warehousing enables easy organization and maintenance of large data in addition to fast retrieval and analysis in the manner and depth required from time to time.

Data can be classified into three categories: reference and transaction data, derived data, and denormalized data. *Reference and transaction data* originates from operational (source) systems and is normally kept in a conventional database system. At a predetermined periodicity, it will be loaded for refreshing the data warehouse. If it is purged from the source then it is archived into the data warehouse. But once put in the data warehouse, it cannot be modified. *Derived data*, on the other hand, is only derived from the reference and transaction data, based on certain rules or computations. It can always be derived again on fresh or modified basis of derivation. *Denormalized data*, which is the basis for online analytical processing (OLAP) tools, is prepared periodically but is directly based on detailed reference (transaction) data.

1.2 DATA MARTS

From a data warehouse, data flows to various departments for their customized DSS usage. These individual departmental components are called *data marts*. In

other words, a data mart is a body of DSS data for a department that has an architectural foundation of a data warehouse. Data mart is a subset of a data warehouse and is much more popular than data warehouse. What is the reason? Why does every department or part of an organization prefer to have a data mart instead of becoming part of the data warehouse? As a central data warehouse keeps growing, it becomes more and more complex or even possibly unwieldy. Gradually, the data becomes harder to customize. When the data warehouse is small, the analyst (of DSS) can easily customize, summarize and analyse to draw analytical results relatively easily. But he can do that no more once the data becomes very large as it calls for large amount of time, which he may not be able to afford. The cost of processing the data also increases as the volume of data increases. Further, the software that is available to access or analyse the large quantity of data may not be as easy or elegant as the software that will be able to processs small amounts of data, as in the data mart. As a result, the analyst finds the data mart extremely useful for fast and easy analysis, as also the users of the results of such analysis. Since the data flows into the data mart (from the data warehouse), the department which owns it can easily customize the data. It can easily summarize, sort, select or structure without any global considerations or any necessity to meet the requirements of another department. Similarly, in the case of historical data, a department can choose to limit the historical data to its own department. The processing load or overhead is also very limited in this case. Whereas, the resource requirements are very restricted. Therefore, data marts are very much preferred by both users and analysts. There are other organizational, technological and economic reasons as well, why the data mart is so attractive. It is only because a data mart is a natural outgrowth of a data warehouse.

The source of data that flows into the data mart is of current level detail. The detailed data is customized, sorted or summarized as it is placed in the data mart. In addition, the data mart can be fed by data from external sources.

It is important to note that all the issues related to data mart (as discussed below) are equally relevant for a data warehouse, since basically a data warehouse is only a collection of data marts.

1.2.1 Types of Data Marts

Data marts can be classified into two groups: *multidimensional* (*MDDB OLAP* or *MOLAP*) and *relational OLAP* (ROLAP). In a multidimensional (MDDB) data mart, the numeric data, which is basically multidimensional in its original nature, can be sliced and diced in a free fashion, i.e. free from data modelling constraints of a DBMS (as RDBMS). As an example, let us consider the data of prices of commodities. This data may be stored just as basic tables in RDBMS, even though the data inherently may be multidimensional, i.e. commodity-wise, year-wise, region-wise, etc. The multidimensionality of the data is lost or not considered at

all in an RDBMS which handles it in terms of tabular form of data. Whereas, on the other hand, in an MDDB situation, the data can be so viewed as to fully harness the natural multidimensionality. Queries can be asked based on this multi-dimensionality of data. Thus, the query processing and analysis will be much more powerful than what could be achieved in a conventional RDBMS environment. Only specialized DBMSs or engines can therefore support MDDB model. On the other hand, *ROLAP* or *relational OLAP* data marts may contain both text and numeric data and are supported by RDBMS. These data marts are used for general purpose DSS analysis with many indices which support for star schema. ROLAP data marts can be analysed either on ad hoc basis (ad hoc queries) or on preconceived queries. The processing in ROLAP is predictable in terms of summary and detailed data which will be available in the ROLAP data marts (Inmon, 1996). Hybrid approach is also possible (as discussed in Chapter 2).

1.2.2 Loading a Data Mart

The data mart is loaded with data from a data warehouse by means of a *load program*. The chief considerations for a load program are: frequency and schedule; total or partial refreshment (or addition); customization of data from the warehouse (selection); re-sequencing and merging of data; aggregation of data; summarization; efficiency (loading time optimization); integrity of data; data relationships and integrity of data domains; and producing meta data for describing the loading process.

1.2.3 Metadata for a Data Mart

Metadata (or data about data) describes the details about the data in a data warehouse or in a data mart. Such a description may be in terms of the contents and source of data that flows into the data warehouse or data mart. Following are the components of metadata for a given data warehouse or data mart:

1. Description of sources of the data.
2. Description of customization that may have taken place as the data passes from data warehouse into data mart.
3. Descriptive information about data mart, its tables, attributes and relationships, etc.
4. Definitions of all types.

The metadata of a data mart is created and updated from the load programs that move data from data warehouse to data mart. The linkages and relationships between metadata of a data warehouse and metadata of data mart have to be well established or well understood by the manager or analyst using the metadata.

This description is essential to establish drill down capability between the two environments. With this linkage, the manager using data mart metadata can

easily find the heritage of the data in the data warehouse. Further, the DSS analyst also requires to understand how calculations are made and how data is selected for the data mart environment. The metadata pertaining to the individual data mart should also be available to the end-user for effective usage.

1.2.4 Data Model for a Data Mart

A formal data model is required to be built for a large data mart which may also have some processing involved. This processing can be repetitive or predictable. No data model is necessary for ordinary or simple small data marts with no processing. Such a data model should be compatible with the DBMS which handles the data mart. For example, in the case of multidimensional DBMS (or MDDB), no separate formal data model can be built, as the multidimensional data model itself is a necessity for that DBMS. Data marts which do not have to be compliant to a particular DBMS can be modelled in terms of a formal data model which will take care of both summary and detailed levels of data in the particular context of the application for a given department.

1.2.5 Maintenance of a Data Mart

Periodic maintenance of a data mart means loading, refreshing and purging the data in it. Refreshing the data is performed in regular cycles as per the nature of the frequency of data update. For example, rainfall data in a data mart may be refreshed (added) on a daily basis whereas daily prices of commodities may be refreshed on a weekly basis. Similarly, census data may be refreshed on a decennial basis. Thus, refreshing may be daily, weekly, monthly, quarterly, yearly or decennially depending on the particular nature and frequency of the data availability.

As regards to purging of data in a data mart, the data mart is read periodically and some (old) data is selected for purging or removing. The data to be removed may be purged, archived or condensed. The criteria for purging depends on date, time and periodicity, based on any criterion decided by the application requirements.

1.2.6 Nature of Data in a Data Mart

The data in a data mart can be of detailed level, summary level, ad hoc data, preprocessed or prepared data. As a rule, the bulk of a largely populated data mart contains a lot of ad hoc summary (aggregations) data and also a lot of prepared detailed data.

1.2.7 Software Components for a Data Mart

The software that can be found with a data mart includes: DBMS, access and analysis software, software for automatic creation of data mart, purging and

archival software, and metadata management software, etc. The *access and analysis* software, which is the most unique for a data warehousing application, allows the end-user to search through the data mart to find and compute or derive the data required by the user. This software also performs elegant or impressive presentation of the data so derived to the user.

The DBMS, for a data mart can be RDBMS or multidimensional DBMS (MDDB), as already discussed.

1.2.8 Tables in the Data Mart

A data mart may include: summary tables, detailed tables, reference tables, historical tables, analytical (spreadsheet) tables, etc.

As the data grows, the backlog back-up may be kept as a 'library', which can also serve the purpose of a quick reference for a survey of available data, as and when required.

The data is kept in the tables in the form of star joins on normalized tables. Star joins are required to be created when a predictable pattern of usage is required on a significant amount of data. Otherwise, when the data is not large, relational tables are adequate.

1.3 OTHER ASPECTS OF DATA MART

1.3.1 External Data

Let us examine some of the issues relating to usage of external data in a data mart. First, the external data, if required to be used by more than one data mart, shall be best placed in the data warehouse itself and then subsequently be moved to the data marts required. This will avoid redundancy and duplication in procurement of external data (otherwise, several data marts may procure the same data). Secondly, when a particular external data is acquired, its 'pedigree' or descriptive details in addition to the data, are also required to be stored. These details include the source of external data; size and data of acquisition of the external data; editing and filtering criteria applied to the external data; etc.

1.3.2 Reference Data

The reference data tables, stored in addition to basic data in the data mart help and enable the end-users of the data mart to relate the data to its expanded version. In doing so, the end user can operate the data in 'short hand', if desired. The reference data is typically copied over from the data warehouse, although, it is rarely possible that the data mart itself will store and manage its own reference tables. When reference tables are stored and managed in DSS environment, there is a need to manage their contents over time. The time management is a complex task which may be best done by the data warehouse itself (instead of the data mart).

1.3.3 Performance Issues

The performance considerations in DSS environment are very much different from those in OLAP environment. The response time issues are totally different—there is no need for real time or online response as is required in the case of OLTP systems. The response time requirements are relaxable ranging from 1 minute to 24 hours. Especially, in the context of data warehousing, the performance issues are relaxed as there is an abundance of data to be dealt with along with a lot of exploration of data. It is not the same situation as in a data mart environment, where there is very limited data with clear definition of requirements. Therefore, reasonable performance objectives may be set and attained. The star join is one way in which the performance requirements may be achieved. The expectations of performance in multidimensional (MDDB) environment are again different. There can be good performance from MDDB if it is not overloaded.

To conclude, good performance can be achieved in a data mart environment by: extensive use of indexes, using star joins, limiting the volume of the data, creating arrays of data, creating profile records (i.e. aggregate records) and creating pre-joined tables; etc.

1.3.4 Monitoring Requirements for a Data Mart

Periodic monitoring is required on data mart behaviour. Monitoring mainly relates data usage to data content tracking. Data usage tracking has queries on the data mart such as:

What data is being accessed?
Which users are active?
What is the quantity of data accessed?
What are the usage timings?

The data content tracking includes the following queries:

What are the actual contents of the data mart?
What is the best way of accessing?
Is there any bad data in the data mart?
How and how much of the data mart is growing?

Monitoring will become essential and important only when the size of the data in the data mart grows significantly.

1.3.5 Security in a Data Mart

When there is secretive information in the data mart it needs to be secured well. Typically, secretive information includes financial information, medical records and human resources information, etc. The data mart administrator should make

necessary security arrangements such as: firewalls; log on/off security; application-based security; DBMS security ('view' based security); encryption and decryption. The information cost of security depends on its exclusiveness.

CONCLUSION

A data mart is a powerful and natural extension of a data warehouse to a specific functional or departmental usage. The data warehouse provides granular data and various data marts interpret and structure the granular data to suit their needs.

Data marts can be of two types—MDDB and ROLAP—each has different characteristics. The data model for a data mart is same as that of a DBMS or can be different, as required. Metadata is an internal part of a data mart environment. Metadata enables different data marts to achieve a degree of cohesiveness. It also allows the end-user to effectively access the data in a data mart. The software for a data mart includes: a DBMS, access and analysis software, metadata management software, purge and archival software, system management, etc. The data structures formed in a data mart include star joins and normalized data as per the relevant data model.

CHAPTER 2 Online Analytical Processing

2.1 INTRODUCTION

Online Analytical Processing (OLAP) systems, contrary to the regular, conventional online transaction processing (OLTP) systems, are capable of analysing online a large number of past transactions or large number of data records (ranging from mega bytes to giga bytes and tera bytes) and summarize them on the fly. This type of data is usually multidimensional in nature. This multidimensionality is the key driver for OLAP technology, which happens to be central to data warehousing.

Any multidimensional data as spreadsheet cannot be processed by conventional SQL type DBMS. For a complex real-world problem, the data is usually multidimensional in nature. Even though one can manage to put such data in a conventional relational database in normalized tables, the semantics of multidimensionality will be lost and any processing of such data in the conventional SQL will not be capable of handling it effectively. As such, multidimensional query on such a database will explode into a large number of complex SQL statements each of which may involve full table scan, multiple joins, aggregation, sorting and also large temporary table space for storing temporary results.

Finally, the end-result may consume large computing resources in terms of disk space, memory, CPU time, which may not be available and even if they are available the query may take very long time. For example, a conventional DBMS may not be able to handle three months' moving average or net present value calculations—these situations call for extensions to ANSI SQL, a near non-feasible requirement.

In addition to response time and other resources, OLAP is a continuously iterative process and preferably interactive one. Drilling down from summary aggregate levels to lower level details may be required to be done on an ad hoc basis by the user. Such drilling down may lead the user to detect certain patterns in the data. The user may put forward yet another OLAP query based on these patterns.

This process makes it impossible to handle or tackle for a conventional database system.

2.2 OLTP AND OLAP SYSTEMS

Conventional OLTP database applications are developed to meet the day-to-day database transactional requirements and operational data retrieval needs of the entire user community. On the other hand, the data warehousing based on OLAP tools are developed to meet the information exploration and historical trend analysis requirements of the management or executive user communities. The conventional regular database transactions or OLTP transactions are short, high volume, provide concurrent and online update, insert, delete in addition to retrieval queries and other procedures, processing or reporting. These transactions in batch mode may be ad hoc, online or pre-planned. On the other hand, OLAP transactions are long (infrequent or occasional updates or refreshing the data warehouse), but are more efficient in processing a number of ad hoc queries. Information in a data warehouse frequently comes from different operational source systems (which are usually conventional OLTP database systems) and is interpreted, filtered, wrapped, summarized and organized in an integrated manner, making it more suitable for trend analysis and decision support data retrieval. Table 2.1 shows a comparison between OLTP and OLAP systems.

Table 2.1 Comparison between OLTP and OLAP

OLTP	OLAP
1. Only current data available (old data is replaced by current data by updating)	Both current and historic data available (current is appended to historic data)
2. Short transactions (single granularity or more)	Long database transactions
3. Online update/insert/delete transactions	Batch update/insert/delete transactions
4. High volume of transactions in a given period	Low volume transactions, periodic refreshing
5. Concurrency control and transaction recovery	No concurrent transactions and therefore no recovery upon failures required
6. Largely online ad hoc queries, requiring low level of indexing	Largely pre-determined queries requiring high level of indexing

A very popular and early approach for achieving analytical processing is 'star schema' or 'collection model'. This approach is based on the common denomination of the user requirements. In this model, a small number of user-referenced or joint data sets are generated from the detailed data sets. This involves denormalization of data followed by integration as shown in the three-tier architecture in Fig. 2.1.

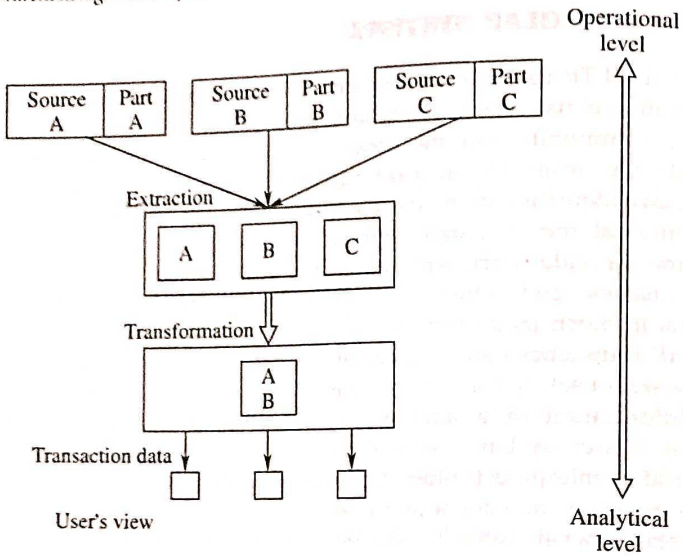


Fig. 2.1 Three-tier architecture of data warehousing environment.

As shown in this architecture, the data extracted from individual data sets is accumulated into a data warehouse after transformation. For example, let us consider the conventional relational database of Part, Customers, Orders and Regions. All these tables are required to be denormalized and stored in a star or collection table that is more in line with user's understanding of the data as shown in Fig. 2.2.

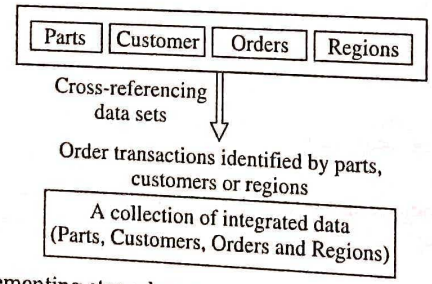


Fig. 2.2 Implementing star schema approach through pre-joining of tables.

2.3 DATA MODELLING—STAR SCHEMA FOR MULTIDIMENSIONAL VIEW

Conventional database modelling (as in semantic and object-oriented database systems) deals with modelling database entities as objects, object hierarchies, etc.

in addition to behavioural modelling in terms of methods (or procedures) of objects.

In the data warehousing environment, the requirements of data modelling are quite different, as the users' expectations from the integrated data warehouse have to be considered. Since the data warehouse comprises a central repository of data collected from divergent sources, the integrated user's view of the data warehouse will form the basis for any data modelling of the data warehouse. In other words, the data collected from different source systems and integrated together in the data warehouse cannot be used, unless it is modelled and organized based on the end-users' perspective of the data. The better the understanding of the user's perspective, the more will be the effectiveness of the data warehousing application. Therefore, the outcome of the understanding of the users' requirements and perspectives should be thoroughly captured in the data model for the data warehouse.

In this context the intuitive perceptions of the users of the relationships of individual data entities in the data warehouse have to be understood by the data model designer for the data warehouse. Such a modelling exercise may lead to a 'star schema' wherein the individual tables or data sets (beaming in from different sources and coverging into a data warehouse) will be denormalized and integrated so as to be presented as per the end-users' requirements. The star schema so developed will effectively handle data navigation difficulties and performance issues of highly normalized data models. In this context the users' perception of data in terms of 'multidimensional' view has to be well understood. 'Dimension' refers to those categories by which the analyst would like to organize, aggregate or view the data. For example, if we take data on prices of commodities (see Fig. 2.3), the users may like to view it from the angle of grouping by time dimension (day, week, month, year) or by region (town, district, state), each of which is of different dimension. Similarly, in the sales analysis application (see Fig. 2.4), the multiple dimensions of sales information will be in terms of market requirements, periods, products and regions. In RDBMS environment, the 'fact' data and 'dimension' data can be described as separate tables.

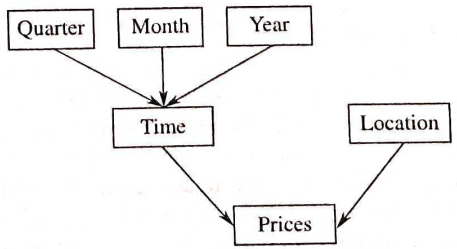


Fig. 2.3 Star schema—prices of commodities.

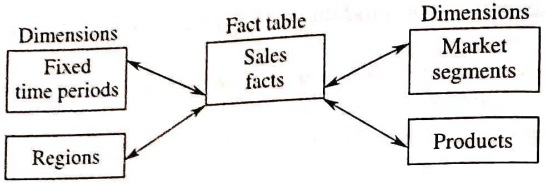


Fig. 2.4 Star schema—sales analysis applications.

The star schema provides a multidimensional view. Thus, the chief model for a multidimensional data warehouse will be the star schema.

Business data analysis or, for that matter, data analysis of any application will require analysis in multiple dimensions—this is not possible with conventional RDBMS or SQL queries (it may be possible with great difficulty, inefficiency and redundancy in certain limited cases).

From the data organization angle or DBA's perspective, a *star schema* is a relational schema. A star schema is organized around a central table called *fact* table, which contains raw numeric items that represent relevant business facts. These facts are additive and are accessible via dimensions. Since the fact tables are presummarized and aggregated along business dimensions, they tend to be very large.

Smaller tables are dimensional tables for the dimensions of data as are already familiar to the users of the facts data. These dimensional tables contain a non-compound primary key and are heavily indexed. Dimensional tables represent majority of data elements. These tables are joined to the fact tables using foreign key references. After the star schema is created and loaded, the dimensional queries (described earlier) can easily be answered. Such multidimensional processing can be made for a variety of dimensions—time periods, regions (geographic or demographic), product ranges, etc. Such multidimensional data as represented by star schema will be very common in all possible industrial sectors such as banking, sales, finance, insurance, manufacturing and also in government. We shall be surveying later the case studies for some sectors.

Let us examine the performance issues of star schema either in a conventional RDBMS environment or in other environments. It may be noted that conventional RDBMS design methodology is oriented for OLTP applications and not for OLAP requirements. Therefore, using multidimensional OLAP, star schema will not be efficient in RDBMS. In other words, if the star schema is hosted on a conventional RDBMS, the performance will be poor. Star schema is defined only for denormalized data, i.e. independent of the relational model. For object-oriented data models, the star schema definition may be required to be made on a novel basis and this is an open area for research. Such research can attempt effectively capturing the semantics of object-oriented and semantic database models into the data warehousing domain.

2.4 DATA MODELLING—MULTIFACT STAR SCHEMA OR SNOW FLAKE SCHEMA

As a data warehouse grows in complexity, the diversity of subjects covered grows which adds more perspectives to access the data. In such situations, the star schema will be inadequate. This can be enhanced by adding additional dimensions which increases the scope of attributes in the star schema fact table. But even further larger growth leads to breakdown of the star schema due to its over complexity and size. Thus, a better technique called *multifact* star schema or *snow flake* schema can be utilized. The goal of this schema is to provide aggregation at different levels of hierarchies in a given dimension. This goal is achieved by normalizing those hierarchical dimensions into more detailed data sets to facilitate the aggregation of fact data. It is possible to model the data warehouse into separate groups, where each group addresses specific performance and trend analysis objective of a specific user. Each group of fact data can be modelled using a separate star schema.

2.5 OLAP TOOLS

2.5.1 Categories of OLAP Tools

As discussed in Chapter 1, OLAP tools can be broadly classified into two categories: MOLAP tools and ROLAP tools. MOLAP tools presuppose the data to be present in a multidimensional database (MDDB). In other words, data which has basically multidimensional nature, if loaded into a multidimensional database, can be utilized by MOLAP tools for analysis. On the other hand, a typical relational database application (without MDDB) can be processed by ROLAP (or relational OLAP) tools. However, there also exist hybrid approaches which integrate both MOLAP and ROLAP techniques and they are usually called multirelational database systems. All these tools basically implement 'star schema' or 'snowflake schema', already discussed (Mattison, 1996).

Applications, as already discussed, are being wide in range—both in business and in government. In business, the typical applications range from sales analysis, marketing campaigning, sales forecasting and capacity planning. Similarly, in the government applications, we can cite examples as commodity price monitoring, analysis and forecasting, plan formulation, analysis and forecasting, agricultural production analysis and forecasting, based on rainfall analysis, etc. The scope of application of these tools both in government and in business is almost unlimited, and the case studies presented at the end of the book illustrate this range of applications. The spread of market between MOLAP and ROLAP sectors is as shown in Fig. 2.5.

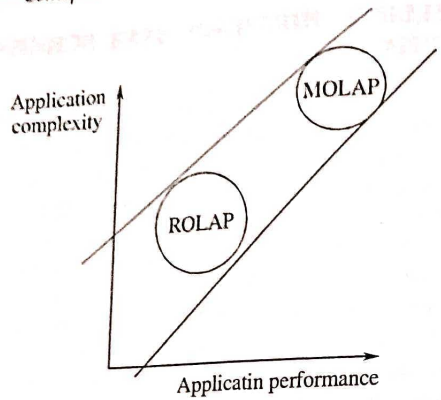


Fig. 2.5 OLAP comparison. (The area of circles indicate the data size.)

MOLAP

MOLAP-based products organize, navigate and analyse data typically in an aggregated form. They require tight coupling with the applications and they depend on a multidimensional database (MDDB) system. Efficient implementations store the data in a way similar to the form in which it is utilized by using improved storage techniques so as to minimize storage. Many efficient techniques are used as sparse data storage management on disk so as to improve the response time. Some OLAP tools, as Pilot products (Software Analysis Server) introduce 'time' also as an additional dimension for analysis, thereby enabling time 'series' analysis. Some products as Oracle Express Server introduce strong analytical capabilities into the database itself.

Applications requiring iterative and comprehensive time series analysis of trends are well suited for MOLAP technology (e.g. financial analysis and budgeting). Examples include Arbor Software's Essbase, Oracle's Express Server, Pilot Software's Lightship Server, Sinper's TM/1, Planning Science's Gentium and Kenan Technology's Multiway.

Some of the problems faced by users are related to maintaining support to multiple subject areas in an RDBMS. As shown in Fig. 2.6, these problems can be solved by some vendors by maintaining access from MOLAP tools to detailed data in an RDBMS.

This can be very useful for organizations with performance-sensitive multidimensional analysis requirements and that have built or are in the process of building a data warehouse architecture that contains multiple subject areas. An example would be the creation of sales data measured by several dimensions (e.g. product and sales region) to be stored and maintained in a persistent structure. This structure would be provided to reduce the application overhead

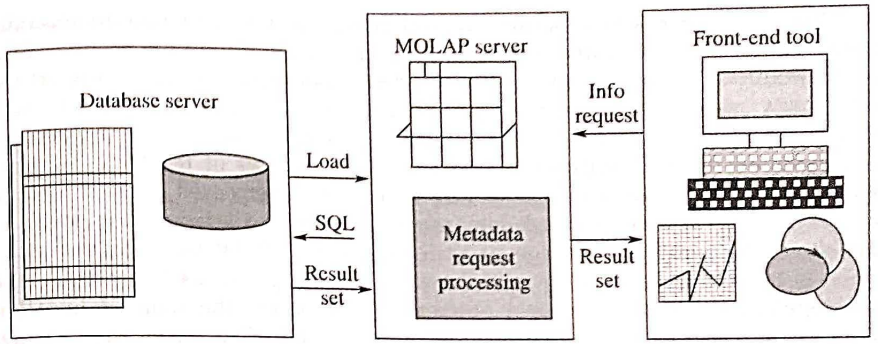


Fig. 2.6 MOLAP architecture.

of performing calculations and building aggregations during application initialization. These structures can be automatically refreshed at predetermined intervals established by an administrator.

ROLAP

Relational OLAP (or *ROLAP*) is the latest and fastest growing OLAP technology segment in the market. Many vendors have entered the fray in this direction (e.g. Sagent Technology and Microstrategy). By supporting a dictionary layer of meta-data, the RDBMS products have bypassed any requirement for creating a static, multidimensional data structure (as was required in the case of MOLAP) as shown in Fig. 2.7.

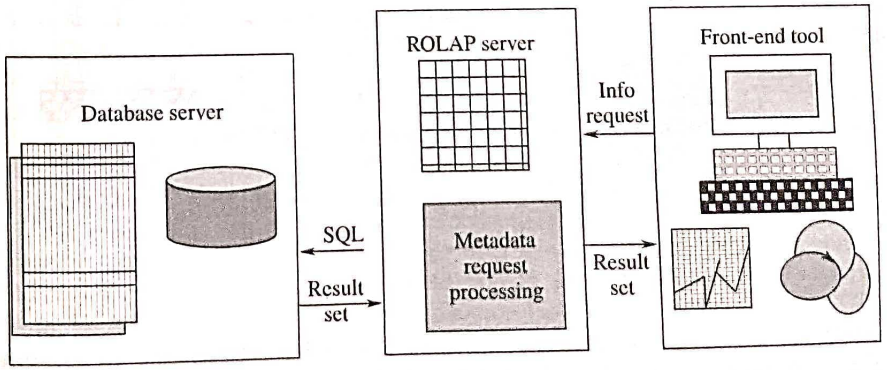


Fig. 2.7 ROLAP architecture.

This approach enables multiple multidimensional views of two-dimensional relational tables to be created, avoiding structuring data around the desired view. Some products in this segment have supported strong SQL engines to support the complexity of multidimensional analysis. This includes creating multiple SQL statements to handle user requests, being 'RDBMS aware' and also being capable of generating the SQL statements based on the optimizer of the DBMS engine. While flexibility is the attractive feature of ROLAP, there exist products which require the use of denormalized database designs (as star schema). However, of late there is a noticeable change or realignment in ROLAP technology. Firstly, there is a shift towards pure middle-ware technology so as to simplify the development of multidimensional applications. Secondly, the sharp delineation between ROLAP and other approaches as hybrid-OLAP is fast disappearing. Thus vendors of ROLAP tools and RDBMS products are now eager to provide multidimensional persistent structures with facilities to assist in the administrations of these structures. Notable among vendors of such products are Microstrategy (DSS Agent/DSS Server), Platinum/Prodea Software (Beacon), Information Advantage (AxSys), Informix/Stanford Technology Group (Metacube) and SyBASE (HighGate Project). (Of late Informix has been acquired by IBM.)

2.5.2 Managed Query Environment (MQE)

Recent trend of OLAP is to enable capability for users to perform limited analysis directly against RDBMS products or by bringing in an intermediate limited MOLAP server (see Fig. 2.8).

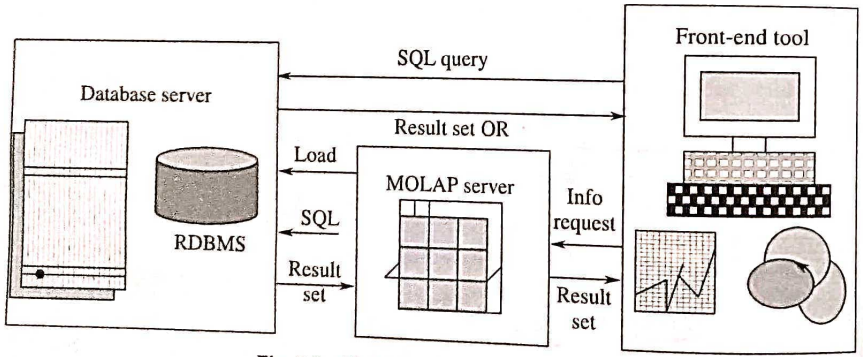


Fig. 2.8 Hybrid/MQE architecture.

Some vendors' products (e.g. Andyne's Pablo) have been able to provide ad hoc query as 'data cube' and 'slice and dice' analysis capabilities. This is achieved by first developing a query to select data from the DBMS, which then

delivers the requested data to the desktop system where it is placed into a data cube. This data cube can be locally stored in the desktop and also manipulated there so as to reduce the overhead required to create the structure each time the query is executed. Once the data is placed in the data cube the user can locally perform multidimensional analysis and also slice, dice and pivot operations on it. In another approach, these tools can work with MOLAP servers, the data from RDBMS can first go to MOLAP server and then to the desktop.

The ease of operation, administration and installation of such products makes them particularly attractive to users who are familiar with simple RDBMS usage and environment. This approach provides sophisticated analysis capabilities to such users without significant costs involved in other more complex products in the market.

With all the ease of installation and administration that accompanies the desktop OLAP products, most of these tools require the data cube to be built and maintained on the desktop or a separate server along with metadata definitions that assist users in retrieving the correct set of data that makes up the data cube. This method causes data redundancy and strain to most network infrastructures that support many users. Although, this mechanism allows for the flexibility of each user to build a customize data cube, the lack of data consistency among users, and the relatively small amount of data that can be efficiently maintained are significant challenges facing all administrators of these tools.

2.6 STATE OF THE MARKET

2.6.1 Overview of the State of the International Market

Basically OLAP tools provide a more intuitive and analytical way to view corporate or organizational data. These tools aggregate data along common subjects of business or dimensions and then let users navigate through the hierarchies and dimensions with the click of a mouse. Users can drill down, across, or up across levels in each dimension or pivot and swap out dimensions to change their view of the data. All this can be achieved by various OLAP tools in the international market. Such free manipulation of data provides insight into data, usually not possible by any other means (Kimball, 1996).

Some tools, such as Arbor Software Corp.'s Essbase and Oracle's Express, preaggregate data into special multidimensional databases. Other tools work directly against relational data and aggregate data on the fly, such as Microstrategy Inc.'s DSS Agent or Information Advantage Inc.'s DecisionSuite. Some tools process OLAP data on the desktop instead of a server. Desktop OLAP tools include Cognos' PowerPlay, Brio Technology Inc.'s Brioquery, Planning Sciences Inc.'s Gentium, and Andyne's Pablo. Many of the differences between OLAP tools are fading. Vendors are re-architecting their products to give the

users greater control over the trade-off between flexibility and performance that is inherent in OLAP tools. Many vendors have rewritten their products in Java.

Eventually conventional database vendors may become the largest OLAP providers. The leading database vendors incorporated OLAP functionality in their database kernels. Oracle, Informix and also Microsoft have taken steps towards this end by acquiring OLAP vendors (IRI Software, Stanford Technology Group and Panorama, respectively). As a result, the OLAP capabilities of their respective DBMS products are varying in their scope. Subsequently, IBM acquired Informix along with all its products.

Red Brick System's Red Brick Ware tools for multidimensional analysis of corporate data, PowerPlay, can be characterized as an MQE tool that can leverage corporate investment in the relational database technology to provide multidimensional access to enterprise data. PowerPlay also provides robustness, scalability and administrative control.

2.6.2 Cognos PowerPlay

Cognos PowerPlay is an open OLAP solution that can interface and interoperate with a wide variety of third-party software tools, databases and applications. The analytical data used by PowerPlay is stored in multidimensional data sets called *PowerCubes*. Cognos' client/server architecture allows for the PowerCubes to be stored on the Cognos universal client or on a server. PowerPlay offers a single universal client for OLAP servers that support PowerCubes locally situated on the LAN or (optionally) inside popular relational databases. In addition to the fast installation and deployment capabilities, PowerPlay provides a high level of usability with a familiar Windows interface, high performance, scalability, and relatively low cost of ownership. Specifically, starting with version 5, Cognos PowerPlay client offers:

- Support for enterprise-size data sets (PowerCubes) of 20+ million records, 100,000 categories and 100 measures
- A drill-through capability for queries from Cognos Impromptu Powerful 3-D charting capabilities with background and rotation control for advanced users
- Scatter charts that let users show data across two measures, allowing easy comparison of budget to actual values
- Linked displays that give users multiple views of the same data in report
- Full support for OLE2 Automation, as both a client and a server
- Formatting features for financial reports: brackets for negative numbers, single and double underlining, and reverse sign for expenses
- Faster and easier ranking of data

- A 'home' button that automatically resets the dimension line to the top level
- Unlimited undo levels and customizable toolbars
- An enhanced PowerPlay portfolio that lets users build graphical, interactive, EIS-type briefing books from PowerPlay reports; Impromptu reports; word processing, spreadsheet, or presentation documents; or any other documents or reports
- A 32-bit architecture for Windows NT, Windows 95/98
- Access to third-party OLAP tools including direct native access to Arbor's Essbase and Oracle's Express multidimensional databases
- PowerCube creation and access within existing relational databases such as Oracle, SyBASE, or Microsoft SQL server right inside the data warehouse
- PowerCube creation scheduled for off-peak processing, or sequential to other processes
- Advanced security control by dimension, category and measure on the client, the server or both
- Remote analysis where users pull subsets of information from the server down to the client
- Complete integration with relational database security and data management features
- An open API through OLE automation, allowing both server and client-based PowerCubes to be accessed by Visual Basic applications, spreadsheets, and other third-party tools and applications.

As mentioned earlier, PowerPlay's capabilities include drill-to-detail using Impromptu. Also, cubes can be built using data from multiple sources. For the administrators who are responsible for creating multidimensional cubes, new capabilities allow them to populate these PowerCubes inside popular relational databases, and to do the processing off the desktop and on UNIX servers. To provide a robust administration capabilities, Cognos offers a companion tool—PowerPlay administrator, which is available in database and server editions.

In PowerPlay administrator database edition, the administrator would continue to model the cube and run the population of the cube process (called *transform*) on the client platform. The advantage is that data from multiple sources can now be used to generate a PointerCube for the client, and the actual PowerCube can be inside a relational database. This means that existing database management tools and the database administrator can be used to manage the business data, and a single delivery mechanism can be employed for both application and OLAP processing. A sophisticated security model is provided which in effect creates a 'master' cube to service a variety of users. This is defined and controlled through an authenticator, also included with PowerPlay.

The administrator server of PowerPlay lets users process the population of the cube on a UNIX platform. An administrator uses client transformer to create a model, and moves it to the UNIX server using the supplied software component called *Powergrid*. The server transformer, once triggered, will create the PowerCube, and the only prerequisite is that all data sources be accessible. Once completed, the resulting PowerCube (or PointerCube if the multidimensional database is placed inside an RDBMS) is copied or transferred to the client platform for subsequent PowerPlay analysis by the user. The authenticator can be used to establish user classes and access security, and can also be used to redirect cube access since all database passwords and locations can be known to the authenticator.

PowerPlay can be used effectively for generation of reports on any multidimensional cube generated by other tools as Oracle Express, Plato, Visual DW of DB2.

PowerPlay supports clients on Windows 3.1, 95, 98 and NT. Administrator database and server editions execute on HP/UX, IBM AIX, and Sun Solaris, and support PowerCubes in Oracle, Sybase SQL server, and Microsoft SQL server.

Latest Cognos PowerPlay release version 6 has Web-enabled features so as to present the reports on the Web in 3-tier undirected use.

2.6.3 IBI Focus Fusion

Focus Fusion from Information Builders Inc. (IBI) is a multidimensional database technology for OLAP and data warehousing. It is designed to address business applications that require multidimensional analysis of detail product data. Focus Fusion complements Cactus and EDA/SQL middleware software to provide a multifaceted data warehouse solution.

Focus Fusion combines a parallel-enabled, high-performance, multi-dimensional database engine with the administrative, copy management and access tools, necessary for a data warehouse solution. Designed specifically for deployment of business intelligence applications in data warehouse environments, Fusion provides:

- Fast query and reporting. Fusion's advanced indexing, parallel query, and roll-up facilities provide high performance for reports, queries and analyses, with the scalability users need to complete data warehouse solutions
- Comprehensive, graphics-based administration facilities that make Fusion database applications easy to build and deploy
- Integrated copy management facilities, which schedule automatic data refresh from any source into Fusion
- A complete portfolio of tightly integrated business intelligence applications that span reporting, query, decision support and EIS needs

- Open access via industry-standard protocols like ANSI SQL, ODBC and HTTP via EDA/SQL, so that Fusion works with a wide variety of desktop tools, including World Wide Web browsers
- Three-tiered reporting architecture for high performance
- Scalability of OLAP applications from the department to the enterprise
- Access to precalculated summaries (roll-up) combined with dynamic detail data manipulation capabilities
- Capability to perform intelligent application partitioning without disrupting users
- Interoperability with the leading EIS, DSS and OLAP tools
- Support for parallel computing environment
- Seamless integration with more than 60 different database engines on more than 35 platforms, including Oracle, Sybase, SAP, Hogon, Microsoft SQL server, DB2, IMS and VSAM.

Focus Fusion's proprietary OverLAP technology allows Fusion to serve as an OLAP front-end or shared cache for relational and legacy databases, effectively providing a virtual warehousing environment for the analysis of corporate data. This can simplify warehouse management and lower overall costs by potentially reducing the need to copy infrequently accessed detail data to the warehouse for a possible drill-down.

Focus Fusion is a modular tool that supports flexible configurations for diverse needs, and includes the following components:

- *Fusion/DBserver*. High-performance, client/server, parallel-enabled, scalable multidimensional DBMS. Fusion/DBserver runs on both UNIX and NT and connects transparently to all enterprise data that EDA/SQL can access (more than 60 different database engines on over 35 platforms). Fusion/DBserver also provides stored procedure and remote procedure call (RPC) facilities.
- *Fusion/Administrator*. Comprehensive GUI-based (Windows) administration utility that provides visual schema definition and bulk load of Fusion databases, multidimensional index definition and build, and rollup definition and creation. Additionally, Fusion/Administrator automates migration of FOCUS databases to Fusion.
- *Fusion/PDQ*. Parallel data query for Fusion/DBserver exploits symmetric multiprocessor (SMP) hardware for fast query execution and parallel loads.
- *EDA/Link*. Fusion's client component supports standard APIs, including ODBC and ANSI SQL. EDA/Link provides access to Fusion from any desktop (Windows 95/NT, UNIX, Macintosh, OS/2) or host system (UNIX, MVS, AS400, VMS, etc.) over TCP/IP and many other network topologies (via EDA Hub servers) to integrate Fusion's products into enterprise processes.

- *EDA/WebLink.* Fusion's open browser client that supports Netscape, Mosaic, Internet Explorer, and all other standard HTML browsers. It works with Information Builders' HTML generator to facilitate Web-based warehouse publishing applications.
- *Enterprise copy manager for fusion.* Fully automated assembly, transformation, summarization, and load of enterprise data from any source(s) into Fusion on scheduled basis. It consists of Enterprise Copy Server, Enterprise Copy Client (the graphical Windows-based interface), and Enterprise Source Server (the remote gateway access to source data).
- *EDA gateways.* Remote data access gateways that provide transparent, live drill through capabilities from Fusion/DBserver to production databases.

2.6.4 Pilot Software

Pilot Software offers the Pilot Decision Support of tools from a high speed multidimensional database (MOLAP), data warehouse integration (ROLAP), data mining, and a diverse set of customizable business applications targetted at after sales and marketing professionals. The following products are at the core of Pilot Software's offering:

- *Pilot analysis server.* A full-function multidimensional database with high-speed consolidation, graphical user interface (Pilot Model Builder), and expert-level interface. The latest version includes relational integration of the multidimensional model with relational data stores, thus allowing the user the choice between high-speed access of a multidimensional database or on-the-fly (ROLAP) access of detail data stored directly in the data warehouse or data mart.
- *Pilot link.* A database connectivity tool that includes ODBC connectivity and high-speed connectivity via specialized drivers to the most popular relational database platforms. A graphical user interface allows the user seamless and easy access to a wide variety of distributed databases.
- *Pilot designer.* An application design environment specifically created to enable rapid development of OLAP applications.
- *Pilot desktop.* A collection of applications that allow the end-user easy navigation and visualization of the multidimensional database.
- *Pilot sales and marketing analysis library.* A collection of sophisticated applications designed for the sales and marketing business end-user (including 80/20 Pareto analysis, time-based ranking, BCG quadrant analysis, trendline and statistical forecasting). The applications can be modified and tailored to meet individual needs for particular customers.
- *Pilot discovery server.* A predictive data mining tool that embeds directly into the relational database and does not require the user to copy or transform the data. The data mining results are stored with metadata into the data warehouse as a predictive segmentation and are embedded into the graphical user interface called *Pilot Discovery Server Launch*, which helps building data mining models.
- *Pilot internet publisher.* A tool that easily allows users to access their Pilot multidimensional database via browsers on the Internet or Intranets.

Some of the distinguishing features of Pilot's product offering include the overall complete solution from powerful OLAP engine and data mining engine to their customizable business applications. Within their OLAP offering, some of the key features of Pilot's products are as under:

- *Time intelligence.* The Pilot Analysis Server has a number of features to support time as a special dimension. Among these are the ability to process data on the fly to convert the native periodicity (e.g. the data collected weekly) to the periodicity preferred by the customer viewing the data (e.g. view the data monthly). This feature is accomplished via special optimized structures within the multidimensional database.
- *Embedded data mining.* The Pilot Analysis Server is the first product to integrate predictive data mining (as it is described in Chapter 3) with the multidimensional database model. This allows the user to benefit not only from the predictive power of data mining but also from the descriptive and analytical power of multidimensional navigation.
- *Multidimensional database compression.* In addition to the compression of sparsity,* Pilot Analysis Server also has special code for compressing data values over time. A new feature called a 'dynamic dimension' allows some dimensions to be calculated on the fly when they are attributes of an existing dimension. This allows the database to be much smaller and still provide fast access. Dynamic variables which are also calculated on the fly are also available to further decrease the total size of the database and thus decrease the time for consolidation of the database.
- *Relational integration.* Pilot allows for a seamless integration of both MOLAP and ROLAP to provide the user with either the speed of MOLAP or the more space-efficient ROLAP. The users interface with the system by defining the multidimensional model *or view that they prefer*, and the system self-optimizes the queries.

2.6.5 Arbor Essbase Web

Essbase is one of the most ambitious of the early Web products. It includes not

*The removal of cells in the multidimensional database, which have no value.

only OLAP manipulations, such as drill up down and across, pivot, slice and dice; and fixed dynamic reporting but also data entry, including full multi-user concurrent write capabilities—a feature that differentiates it from the others.

Arbor sells Essbase only as a server, it does not have a client package. The Web product does not replace administrative and development modules; only user can access it for query and update.

2.6.6 Information Advantage Web OLAP

Information Advantage uses a server-centric message architecture, which is composed of a powerful analytical engine that generates SQL. It also pulls data from relational databases, manipulates the results and transfers the results to a client.

Since all the intelligence of the product is in the server implementing Web OLAP to provide a Web-based client is straightforward, the architecture of Information Advantage's Web product is similar to that of Essbase with a Web gateway between the Web server and the analytical engine. Although in this case, data store and analytical engine are separate, Essbase is both data store and analytical engine.

2.6.7 Microstrategy DSS Web

Microstrategy's DSS Agent was originally a Windows-only tool, but now it is also OLAP tool to have a Web-access product. DSS along with DSS server relational OLAP server—DSS architect data modelling tool, and DSS executive design tool for building Executive Information Systems—generates SQL dynamically and relies on the relational database server to perform complex analysis, rather than creating a 'cube' like most of the other tools. By inserting a Web gateway between the Web server and the DSS server engine, Microstrategy was able to replace the interactive DSS agent front-end with a Web browser, which passes requests to the DSS server's API.

2.6.8 Brio Technology

Brio offers a suite of products called *Brio.web.warehouse*. This suite implements several of the approaches listed above for deploying decision support OLAP applications on the Web. The key to Brio's strategy are new server components called *Brio.quickview* and *Brio.insight* which can off load processing from the clients and thus enable users to access Brio reports via Web browsers. On the client side, Brio uses plug-ins to give users viewing and manipulation capabilities.

For these and other reasons, the Web is a perfect medium for decision support. The general features of the Web-enabled data access are as under:

- The *first-generation Web sites* used a static distribution model, in which clients access static HTML pages via Web browsers. In this model, the decision support reports are stored as HTML documents and are delivered to the users on request. Clearly, this model has some serious deficiencies, including inability to provide Web clients with dynamic updates and interactive analytical capabilities such as *drill-down*.
- The *second-generation Web sites* support interactive and dynamic database by utilizing a multi-tiered architecture in which a Web client submits a query in the form of HTML-encoded request to a Web server, which in turn transforms the request for structured data into a Common Gateway Interface (CGI) script, or a script written to a proprietary Web-server API (i.e. Netscape server API, or NSAPI). The gateway submits SQL queries to the database, receives the results, translates them into HTML and sends the pages to the requester. Requests for the unstructured data (e.g. images, other HTML documents, etc.) can be sent directly to the unstructured data store, pre-calculated MOLAP storage or directly in the relational data store depending on the usage pattern and the time/space trade-off preferences of the end-user.

2.7 OLAP TOOLS AND THE INTERNET

The Web technologies indeed are the natural choice for adaptation for data warehousing (and vice versa). The internet is virtually a free (or a very low cost) resource which provides a universal connectivity within and between companies and also individuals.

Some of the web facilities are as under:

- The Web eases complex administrative tasks of managing distributed environments.
- The Web allows companies to store and manage both data and applications on servers that can be centrally managed and updated, thus eliminating problems with software and data accuracy.
- The emerging third-generation Web sites replace HTML gateways with Web-based application servers. These servers can download Java applets or ActiveX applications that execute on clients, or interact with corresponding applets running on servers–servlets. The third-generation Web servers provide users with all the capabilities of existing decision–support applications without requiring them to load any client software except a Web browser.

Not surprisingly, vendors of decision support applications, especially query, reporting and OLAP tools, are rapidly converting their tools to work on the Web.

Vendor approaches for deploying tools on the Web include the following:

- *HTML publishing.* This approach involves transforming an output of a query into the HTML page that can be downloaded into browser. This approach does not support interactive access to data or reports.
- *Helper applications.* In this approach, a tool is configured as a helper application that resides within a browser. This is the case of a 'fat' (or thick) client, in which, once the data is downloaded, users can take advantage of all capabilities of the tool to analyse data. However, maintaining these helper applications becomes another task for system administrators.
- *Plug-ins.* A variation on the previous approach, plug-ins are helper applications that are downloaded from the server prior to their initial use. Since the plug-ins are downloaded from the server, their normal administration and installation tasks are significantly reduced. However, typically, plug-ins are browser-specific and may not run on all platforms or with all browsers. Also, as browsers get updated, these plug-ins may have to be upgraded as well, creating additional administration workload.
- *Server-centric components.* In this approach, the vendor rebuilds a desktop tool as a server component, or creates a new server component that can be integrated with the Web via a Web gateway (e.g. CGI or NSAPI scripts).
- *Java and ActiveX applications.* This approach is for a vendor to redevelop all or portions of its tool in Java or ActiveX. The result is a true 'thin' client model. Despite some disadvantages, this approach appears to be one of the most promising and flexible.

The following are some of the Web-based tools:

- *Arbor Essbase Web.* This tool offers features as drilling up, down, across; slice and dice and dynamic reporting, all for OLAP. It also offers data entry, including full multi-user concurrent write capabilities.

  Arbor Essbase is only a server product, no client package exists, thus protecting its own desktop (stand-alone) client version market. The Web product does not replace administrative and development modules but it replaces only user access for query and update.

- *Information advantage Web OLAP.* This product uses a server centric messaging architecture, composed of a powerful analytic engine which generates SQL for the retrieval from relational database, manipulate the results and transfer the results into a client.

  Since it is all server-centric capability, implementing Web OLAP to provide web-based client is easy and simple. This architecture is similar to the one of Essbase with a Web gateway between Web server and analytic engine, even though the data store and the engine are separate (as against Essbase which is both a data store and an analytic engine).

- *Microstrategy DSS Web.* The flagship product DSS Agent form Microstrategy, originally a Window tool, is now available on all platforms (as NT) and also on the Web. DSS Agent, along with the complement product—DSS server relational OLAP server, DSS Architect data-modelling tool, the DSS Executive design tool for building executive information system (EIS), generates SQL automatically and dynamically. It depends on the RDBMS server for performing complex analysis, instead of creating a multidimensional 'Cube'. Then it is offering ROLAP (relational OLAP) and not MOLAP (multidimensional OLAP). By inserting a Web gateway between the Web server and DSS server engine, Microstrategy is replacing the interactive DSS Agent front-end with a web browser which passes through it in terms of requests to DSS server API.
- *Brio technology.* Brio released 'brio.web.warehouse' suite of products in the end of 1998. Most of the Web OLAP approaches described earlier are available in this suite. A new server component called 'brio.query.server' works in tandem with Brio Enterprise Web-clients 'brio.quickview' and 'brio.insight' and enable Web access by browsers.

2.8 OLAP TOOLS IN THE OPEN SOURCE DOMAIN

Several OLAP tools are now (2008) available in the open source domain. Many companies are offering free products as open source, downloadable from the web. They have their commercial interests in support activities of such free products. Prominent among such open source OLAP tools are the following companies/products:

1. Pentaho (http://www.pentaho.com)
2. Mondrian (http://mondrian/sourceforge.net)
3. Eclipse (http://www.eclipse.org/birt)
4. Decision Studio professional
5. Jasper (http://www.jaspersoft.com)

2.8.1 Pentaho

Pentaho Reporting allows the user organizations to easily access, format and distribute the information to employees, customers, partners and the top management. Pentaho Reporting provides access to information from a variety of data sources, viz. relational, XML and OLAP. It delivers output information in all popular formats such as Adobe PDF, HTML, Microsoft Excel, RTF (Rich Text Format) or simple text. The formatting features support summarization features

such as totals, sub-totals, create groupings and highlight exceptions. Stand-alone desktop reporting tools, web-based reporting or BI (Business Intelligence) reporting including analysis, dashboards, are all supported by Pentaho Reporting. Through it information can be securely delivered through corporate portals or via e-mails or even through customized applications with good scaling up features.

Pentaho can be connected to a variety of data sources with a range of connectivity such as JDBC 2.0 and can connect to all popular proprietary and open source DBMSs as data sources including Oracle, DB2, MS SQL Server, My SQL, PostgreSQL, Enterprise DB, etc. Pentaho reporting supports following processes:

1. Multiple data sources of different types within one report. Reports can be designed using GUI with full control (with drag and drop) on data access, lay out grouping, calculations, charting and formatting.
2. Centralized metadata layer which allows the creation of one report to satisfy multiple end user needs. Ad hoc reports can be easily created by the end users interactively by using business rules. The centralized maintenance of data connectivity, security, formatting is possible. Changes in meta data are automatically inherited while producing the reports (eliminating manual maintenance of reports). AJAX-based interactive web interface is provided for self-report creation by business users.
3. Spreadsheet services (with any database supporting JDBC 2.0 with any application server supporting JVM 1.4).
4. Dashboards services to provide a one-shot-top-view of the performance at individual, department or enterprise level, by delivering key metrics in attractive visual interfaces. The dashboard can be integrated with BI suite for data mining purposes.
5. Data integration which permits sourcing information from a variety of data sources—it supports Extraction, Transformation and Loading (ETL) (using a simple metadata-driven approach). The GUI provides simple interface with drag and drop design. The methodology adopted enables the open source approach and ensures that the proprietary methods are not required to be adopted in ETL functionality. Pentaho Reporting also supports a wide variety of data mining algorithms for BI.

CONCLUSION

All OLAP vendors position their products to be fully Web compliant. The scope of functionality and a particular style of implementation will become a market differentiator for these tools.

Server-centric tools, such as Information Advantage and Prodea Beacon from Platinum Technology, and the multidimensional OLAP tools—Arbor's Essbase, Oracle Express, Planning Sciences' Gentia, Kenan Technologies Acumate ES, Holistic Systems' Holos, and Pilot Software's Pilot Server—appear to be in an excellent position to take advantage of their architectures to provide easy access from a Web browser. Similarly, client-centric tools such as Business Objects, Software AG's Esperant, Cognos' PowerPlay or Brio Technologies' Brio suite of products, are making available robust and full-featured Web-enabled versions of their products (see discussion of Brio Technology in Section 2.6.8). The tasks become easier as both Java and ActiveX are available for the Web deployment. Open source tools as Pentaho are also available.

CHAPTER 3

Data Mining

3.1 INTRODUCTION

Data mining is the process of extracting previously unknown, valid and actionable information from large data (as transaction data or database or data warehouse) and then using the information so derived to make crucial business and strategic decisions. Data mining may lead to knowledge discovery in the data mined upon (Piatetsky and Frawley, 1991).

The mined information may be in terms of any associations or any other relations between the data items in the data. But it is essential that such associations or relationships identified by mining process be verifiable (i.e. valid), actionable (i.e. can be causing some action), and, of course, be previously unknown (see Fig. 3.1) (Adriaans and Dolf Zantiage, 1996).

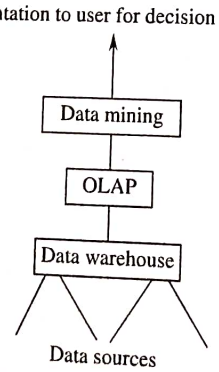


Fig. 3.1 The data mining hierarchy in business decision support.

3.2 FROM DATA WAREHOUSING TO DATA MINING

A data warehouse, being a subject-oriented, integrated and time-variant collection of data, aims at supporting business decisions, whereas data mining

process is a natural and logical continuation to data warehousing. An organization that may have already invested in a data warehouse would naturally like to reap the benefits of a data warehouse by going ahead for data mining process. It is however possible that an organization can take up data mining even without really having built a data warehouse by using either direct transactional data records or a single database of transactions of data on which mining is required to be done. Therefore, as it is clear, a data warehouse will always be the right foundation for taking up data mining process as indicated in Fig. 3.1.

OLAP mining or Online Analytical Mining (OLAM) integrates OLAP with data mining and mining for knowledge discovery in multidimensional data. The high quality of the data, information processing infrastructure in the data warehouse, OLAP analysis itself and online selection of data mining functions—all these are the reasons for OLAM.

3.3 STEPS OF DATA MINING

Step 1: Deciding business objectives for data mining

The entire data mining process is driven by business objectives, which are to be decided in advance. These business objectives may depend on the inferences or conclusions that will be drawn by data mining process in terms of answers to business objectives as questions.

The minimum requirement is prescribed as a business problem or opportunity along with some sponsorship by the organization. This means the organization has identified a problem for data mining and also committed to take some actions based on the conclusions drawn by the data mining process.

The precise definition of the business needs is the first step in this direction. This can be achieved only through the joint effort of the business analyst with domain knowledge and the data analyst who can translate the objectives identified by the analyst into a well-defined data mining problem.

This process also clearly defines the objectives of the data mining and the expectations out of such an exercise.

Step 2: Data preparation

After the business objectives clearly defined, the data is prepared for the mining process. The data preparation process comprises the following types:

- Data selection (identification and extraction of data),
- Data pre-processing (data sampling and quality assurance),
- Data transformation (data conversion into an analytical model).

These are now briefly dicussed.

Data selection. Before the mining process begins, the data analysis process has to be completed. For this purpose, the sources of data have to be identified first. After identifying the sources, the data has to be extracted and prepared for mining. Clearly the data selection will depend on the business objectives.

Along with the data selected the metadata also has to be acquired for understanding the meaning of data. The metadata, as in the case of data warehouse or data mart, should contain all the details as descriptions of data types, initial values, range of values, sources, etc.

For data mining, the 'active' variables have to be selected. The *active variables* are those which will be participating in the mining process. In addition, some data mining algorithms require supplementary variables to be identified; they are helpful in visualization and explanation of results.

The shelf life of data also becomes an important consideration. The *shelf life* is the period after which the data will lose its attraction or usefulness.

The data mining algorithms to be adopted may also be decided by the analyst at this stage, as they determine the required formats and other details of the data being prepared.

Data pre-processing. Data pre-processing is the most crucial step as the operational data is normally never captured and prepared for data mining purpose. Mostly, the data is captured from several inconsistent, poorly documented operational systems. (Capturing data from point of sale (PoS) is an exception.) Thus, data pre-processing requires substantial efforts in purifying and organizing the data. This step ensures that the selected data available for mining is in good quality.

The data pre-processing step begins with a general review of the structure of the data and quality assurance. This is performed using sampling and visualization techniques.

The characteristics of data to be viewed depend on the nature of data, i.e. whether the data is categorical or quantitative.

For categorical variables, the visualization can be in terms of histograms, pie charts, etc. For quantitative variables, the visualization will be in terms of maxima, minima, mean, median, mode, standard deviation, etc.

By utilizing all these methods it is possible to determine the presence of invalid and skewed data which may be incorrect. Spurious data or noise can be identified by quantitative techniques as minima and maxima analysis or by various scatter distribution parameters.

Scatter plots, base plots can help in identifying and clearing the data of spurious and noisy elements by finding outlandish or exceptional data (which may be incorrect).

Missing values of data pose a real problem. This problem can be encountered by various methods—dropping the observations with missing data values or replacing a missing value with a likely value, predictable by reliable techniques, etc.

Data transformation. In this step, an analytical data model is produced by transforming the pre-processed data. The analytical data model is an informational data model, representing a consolidated, integrated and time-dependent restructuring of the data selected and pre-processed from various operational sources. The nature, content and scope of this data model determine the validity and usefulness of the data mining process. For example, if the customer spending patterns are to be analysed, the analyst may decide whether such an analysis is to be performed at the departmental level or individual item level.

Evidently the analytical data model determines the nature of problems that data mining process may solve.

After the informational model is built it is further refined to suit the format requirements of a particular data mining algorithm. This refinement may vary from simple data format changes to more complex derivations of new parameters, for example, the age from the date of birth.

In customer relationship management (CRM) applications, the household profile may be required to be built based on the basic data of the customer.

Statistical data reduction is another business requirement. This means several variables will be reduced to few variables by redefining a group of variables into one representative variable.

If neural networks are used for data mining the acceptable input values are restricted between 0 and 1. This calls for redefining the output values into this range. Continuous data can be described in terms of class intervals.

Step 3: Data mining

This step comprises applying the selected data mining algorithms to the data which is already preprocessed. This step, interactively combined with the next step (i.e. analysis of results), is to be performed by data analyst (associated by data management analyst, as needed from time to time for data definition and structure). This step involves the actual mining process by the analyst who is pre-equipped with pre-processed data from previous steps, the related metadata and also some insight into content of data based on the foregoing analysis (i.e. the next step).

The actual mining process may result in some database segmentation or a cyclic predictive process, depending on the data mining algorithms used.

If a neural network-based learning algorithm is to be used, the neural network may be repeatedly trained or retrained on sample data before testing on real database. Several mining algorithms may be used in sequence. For example,

database partitioning is required to be done before predictive modelling can be done on each partition, thus requiring two algorithms (Widrow and Leht, 1994).

Rule induction algorithms may be used for characterizing different consumer behaviours.

We shall examine more mining algorithms in greater detail in the next section.

Step 4: Analysis of results

As already indicated, data mining and analysis of results go hand in hand, interactively and iteratively. The analysis of results is the most important step in the entire cycle of data mining. Using visualization aids and tools, analysis of data can be done by a highly skilled data analyst with the help of a business analyst.

Analysis of results may or may not yield a clear 'yes' or 'no' to a given hypothesis as in the case of statistical hypothesis testing. The quest is to find some patterns or trend unknown earlier, not even hypothesized. Herein lies the difference in the approaches of data mining and statistical analysis. For example, after database partitioning the analyst will be able to find a pattern or classification between the partitions themselves. This itself is a useful information tip for business decisions. On the other hand, predictive models with hypothesis can also be tested using data mining techniques by testing for the accuracy of the model. Drawing conclusions on the data in the form of 'if–then' rules is another approach of analysis.

Developing association rules is an important aspect of data mining. Such associations require high degree of confidence levels; setting confidence levels in determining associations is also critical in determining the outcome. If high confidence levels are set then only known associations are inferred. Only appropriate confidence levels are to be set to identify possibly new (unknown) associations.

Step 5: Assimilation of knowledge

This is the final step which joins with the initial objective of deciding the business objectives. This step comprises action steps to be taken to implement the knowledge gained by data mining. In other words, knowledge discovered in the data is to be utilized by implementing it in business cycle. Evidently, the business analyst becomes the most important player in this step.

The business analyst identifies the stages for implementation for assimilating business knowledge (gained by mining process) into actual real-life business processes for availing the benefits of data mining exercise performed. For example, a newly discovered set of association rules may trigger a new marketing campaign to sell new product association groups identified. Similarly, a new

predictive model discovered by data mining may lead to a fresh direct mailing compaign to identified customers with new discounts.

Assimilation of knowledge discovered by data mining may also result in some technical effects. For example, the newly identified association rules may be integrated with the existing data mining models for more effective analysis and mining in future. The new knowledge may be integrated into existing technical infrastructure. New predictive models or association rules may be integrated with the existing database procedures, application software, etc. Similarly, existing database structures may be enhanced with new data structures. Data preparation steps would have already presented an analytical view of the existing data flow procedures and systems.

The final consummating benefit will be a recognition or a new focus on importance of data as an asset of the organizations. Its value is better understood and focussed by data mining operations, thus resulting in a data-oriented culture in the organization which enhances interest and motivation for related activities such as development of data warehousing activities, etc. (Piatetsky et al., 1996).

3.4 DATA MINING ALGORITHMS

The data mining algorithms offer different techniques for identifying previously unknown characteristics, patterns and associations in the data which may be mined. These techniques are generally identified with data mining operations as database segmentation, predictive modelling, link analysis and deviation detection.

We shall try to present a summary of the operation itself and also the technique used for it.

3.4.1 Database Segmentation

The goal of this mining operation is to segment or partition a database into segments of similar data (or records of data). Data records that share similar properties are considered as homogeneous. Segmentation and clustering are often used interchangeably. In data mining terminology, a 'cluster' or 'segment' is a group of data records resulting from a mining operation. On the other hand, the word 'segmentation' or 'clustering' refers to the actual operation leading to formation of 'segments' or 'clusters'. The segments (or clusters) have high internal homogeneity and high external heterogeneity, i.e. data elements within the same segment (or cluster) have much more similarity than with those in other segments or clusters.

Figure 3.2 illustrates three segments of data relating to income and age. A segmentation algorithm will segregate the data as indicated above. In this process, there are two different methods possible: *demographic* clustering and *neural* clustering. These two methods are distinguished by: data-types accepted,

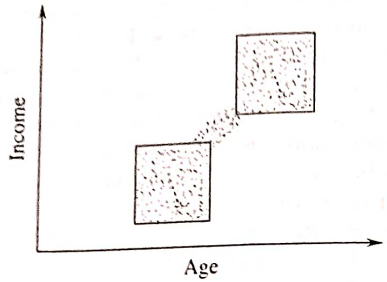


Fig. 3.2 Example of data clustering or database segmentation.

the methodology of calculation of inter-record distances and the way the segments are organized.

Demographic clustering operates on records with categorical orders and the distances are measured using a voting principle called *condorset*. No hierarchy is followed for organizing the resulting segments (or clusters). On the other hand, neural clustering methods are based on neural networks. They accept only numeric quantitative inputs (categorical inputs can be converted into numerical input form).

The distance measurement technique is based on Euclidean distances and the resulting segments are arranged in a hierarchy, where the most similar segments are placed closest together and the least similar segments are placed with largest distance.

Segmentation (or clustering) is used in business applications as customer profiling or target marketing, etc. It has interdisciplinary applications, cutting across all sectors of business.

3.4.2 Predictive Modelling

Just as human beings learn from observation and experience and then be able to predict, it is possible for an algorithm to draw certain general rules on the behaviour of the data and predict the future behaviour.

In data mining operations, predictive modelling analyses the existing database to determine some essential characteristics about the data. It is, however, essential that the data include complete, valid observations from which the model can reach a conclusion on how to make accurate predictions. The model needs to be 'trained' with already known data observations for prediction purpose. This approach is called 'supervised learning'. The model itself can be physically very simple.

It can be first a few 'if–then–else' clauses, for example. Figure 3.3 shows the behaviour pattern of employees of an Indian government organization. From this

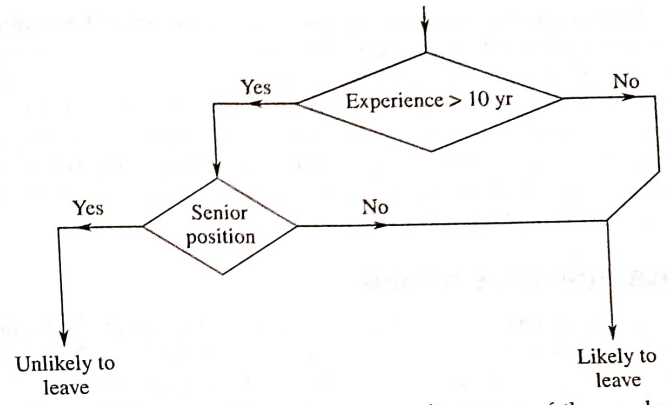


Fig. 3.3 Predictive modelling—behaviour of turnover of the employees.

figure we can see that the resignation data has been analysed and found to be following a particular pattern, i.e. employees who are not in very senior positions and have worked less than 10 years in the organization are displaying a general tendency to leave the organization. This is a predictive model developed on training by mining the data given. Once the model is defined clearly it can be used for prediction purposes.

Both training and testing of the model need to be performed. Training requires large data whereas testing is done on small data.

Predictive modelling has extensive applications across the industry sectors. Customer or employee selection management, credit rating, cross-selling and target marketing are some of the applications of this technique.

3.4.3 Link Analysis

In this technique, the links between individual data records or sets of data records are sought to be identified (as against segmentation or clustering of data records). Link analysis comprises *association* discovery, *sequential pattern* discovery and *similar time sequence* discovery.

Associations discovery refers to identifying previously unknown associations between two or more data items, such as two or more items of purchase in a single group purchase transaction. This type of analysis is also known as market-based analysis or product affinity analysis.

Sequential pattern discovery is used to identify associations across related purchase transactions over time that reveal the sequence in which the customer purchases goods. It aims at longer or mid-term customer buying behaviour and can be made use for timely product promotions.

Similar time sequence discovery is comparison between two time sequence patterns (as above) when two or more time sequence patterns are compared and contrasted, the similarity and deviations can be analysed. Deviation detection using statistics and visualization techniques can be useful in identifying interesting patterns of customer behaviour at different time sequences and thereby drawing conclusions for product promotion campaigns. (For more detailed description of these techniques see Adriaans et al., 1996, Berry and Linoff, 1996, and Bort, 1996.)

3.5 TOOLS FOR DATA MINING

In recent years, there has been a sharp rise in the number of data mining tools available in the market, and this trend may continue further. The tools for data mining are so trendy that many vendors of other software (as for decision-support functions) are attempting to add data mining capabilities. The emerging generic title for these tools is 'siftware', mainly because they 'sift' through a large amount of data. All the tools for data mining in the market can be classified into three categories: *commercial* tools, *public domain* tools and *research prototypes.*

Commercial tools

Commercial tools can be identified as the following products and usually are associated with the consulting activity by the same company:

1. 'Intelligent Miner' from IBM,
2. 'SAS' System from SAS Institute,
3. 'Thought' from Right Information Systems,
4. 'Data Mining Suite' from Information Discovery Inc.,
5. MineSet from Silicon Graphics Inc.,
6. 'Data Mining Marksman' from HNC Software,
7. 'Data Crusher' from DataMind Corp.,
8. 'Cross/z' from Cross/z International,
9. 'Clementine' from Integral Solution Ltd.

The Web sites giving details of these products are given below:

- IBM : http://direct.boulder.ibm.com/bi
  http://www.software.ibm.com/data/products
- SAS : http://www.sas.com/feature/4qpdm/intro.html
- Silicon Graphics : http://www.sgi.com
- Integral Solutions : http://www.isl.co.uk
- HNC : http:/www.hnc.com/prodsmkt.htm
- DataMind : http://www.datamindcorp.com

Public domain tools

These are largely freeware with just registration fees: 'Brute' from University of Washington, Seattle, Washington, 'MCL++' from Stanford University, Stanford, California.

'Knowledge Explorer', Pentaho's open source standards compliant data mining and BI product is well integrated with Pentaho's data integration, reporting, analysis dashboard and also workflow modules. It supports a wide range of data mining algorithms (which can be either applied directly or called from Java code) thereby providing a choice of specific algorithm for a given problem requirement situation. It also permits the inclusion of additional data into the algorithm as and when required. The Knowledge Explorer provides a comprehensive set of machine learning algorithms (from 'Weka' project) including clustering, segmentation, decision trees, principal component analysis and also neural networks. The output can be graphically presented, interacted programmatically or used for reporting and further analysis. Filters are provided for discretization, normalization, resampling, attribute selection and transformation, combining attributes for predicting normal or numeric quantities. Learning schemes such as decision trees, lists, instance-based classifiers, support vector machines, multilayer perceptions, logistic regression and other advanced techniques are also available. However, it enables the development of new machine learning techniques. It is possible to control input and output programmatically. Graphical design tools and administrative tools are integrated with the work bench and are delivered inside Eclipse. All user interaction can be through Graphical User Interface (GUI) for all steps of data mining with adequate security provisions.

Research prototypes

Some of the research products may find their way into commercial market: 'DB Miner' from Simon Fraser University, British Columbia, 'Mining Kernel System' from University of Ulster, North Ireland.

All the above tools can be broadly classified as generic single-task tools and generic multi-task tools. The generic single-task tools generally use neural networks or decision trees. They cover only the data mining part and require extensive pre-processing and post-processing steps. Early examples in this direction are: '4 thought' and early IBM's Neural Network utility. Generic multi-task tools are a step forward. They offer modules for pre-processing and post-processing stages and also offer a broad selection of several popular data mining algorithms as clustering (or partitioning), deviation detection, etc. and also support some statistical processing. This enables the users to solve a broad range of problems with the same tools. Examples of this category of generic

multitasking tools are: the latest IBM's as 'Intelligent Miner' (IBM, 1996) and 'Clementine' and 'Data Mining Suite' from Integral Solutions Ltd. and Information Discovery Inc. respectively.

Technical considerations for evaluation and eventual purchase of the tool will be based on scalability, performance and extendibility through application development. However the support provided by the tool for various steps in the data mining process is also an essential consideration. This may require detailed evaluation of tools for a given mining application.

CONCLUSION

In this chapter, we have surveyed the data mining concepts, techniques and tools. The need, importance and applications of data mining have also been described along with various data mining techniques and algorithms. A few tools for data mining have also been surveyed.

CHAPTER 4

Developing a Data Warehouse

4.1 WHY AND HOW TO BUILD A DATA WAREHOUSE?

Changes are taking place in the world continuously in all forms of activity. From a business perspective, the quest for survival in a globally competitive environment has become extremely demanding for all.

As discussed in Chapter 1, in this scenario, business strategy requires answers to questions in business policy and future strategy. This means that the decisions are required to be taken quickly and correctly using all the available data. As the data size increases continuously, doubling every 18 months, the speed requirements for processing this data so as to comprehend the meaning of this data are also required to be increased significantly. Competition also adds pressure on this situation and herein business intelligence becomes the foundation for successful business strategy (Bracket, 1996).

Here the need for data warehousing technology arises, in terms of the ability to organize, maintain large data and also be able to analyze in a few seconds in the manner and depth as required. (Inmon and Hackathorn, 1994).

Why did the conventional information systems not succeed in meeting these requirements? Actually the conventional information systems and data warehousing tackle two different activity domains—OLTP and OLAP. (as discussed in Chapter 1). These domains are not at all competitive with each other; they deal with two different problem domain requirements. Further, as the technology upgrades, the CPU time and disk space are growing larger and are becoming cheaper with time. Similarly, network bandwidths are increasing and becoming cheaper day by day. Need for more and more heterogeneity and interoperability is increasingly being felt. In this scenario, data warehousing emerges as a promising technology (Anahory and Murray, 1997).

In the following sections, we shall survey all the major issues involved in building a data warehouse, such as approach, architectural strategy, design considerations, data content related issues, meta-data, distribution of data, tools and performance considerations.

4.2 DATA WAREHOUSE ARCHITECTURAL STRATEGIES AND ORGANIZATIONAL ISSUES

A data warehouse can be built either on a top-down or on a bottom-up approach. In other words, one can define a global data warehouse for an entire organization and split it up into individual data marts or sub data warehouses dedicated for individual departments. Alternatively individual data marts can be built for each department and then they all get finally integrated into a central data warehouse (Bischoff and Alexander, 1997).

The bottom-up approach is more realistic but the integration of individual data marts should be made easier with advanced planning and preparation.

All organizations active in information systems area are fully conversant with the issues relating to information systems implementation. But the issues pertaining to data warehousing activity are quite different and divergent from these ones. In building a data warehouse, the organization has to make arrangements for information flow from various internal information systems and databases as well as from internal sources of information. This requires close involvement of the users in identifying the information requirements and identifying the sources of the same from both internal and external data sources and information systems in time (Devlin, 1997).

4.3 DESIGN CONSIDERATIONS

The first and foremost design consideration is to be exhaustive and so to say holistic, i.e. to exhaustively cover all possible data sources for exhaustively designing a data warehouse with all possible components as part of a single complex system so as to effectively meet all possible user requirements. Any failure in this regard may lead to a data warehouse design that is skewed towards a particular requirement (missing other requirements) or a particular data source, missing some other data sources or oriented towards a particular access tool, thereby limiting its usage and accessibility. Web enablement may subsequently enhance the data accessibility and its usage.

There are three major issues that will be faced in data warehouse development: heterogeneity of data sources requiring substantial efforts in data conversion and also in maintaining timeliness and high-quality levels of data integrity, reliability and authenticity. Further, the data may be quite old and historical. While old data of past is essential for a data warehouse, it should be ensured that it will be relevant and useful in the data warehouse form. If any data is too old to be relevant any more it may not be worthwhile to enter into the data warehouse. Another issue is the tendency of the data warehouse to grow very large. Discrete decisions should be made by the designer of the data warehouse in limiting the size of the warehouse by discretely dropping old data and selecting appropriate data to be included into the warehouse.

The data warehouse design is distinctly different from regular OLTP-based information system design in many ways. Data warehouse design is business driven, i.e. it is based on specific business goals to be achieved by the OLAP on the data warehouse. In the case of OLTP systems, the design is driven by the existing information system concerned. Since data warehouse design is business driven, it may never be fully completed as the users' business requirements and also data sources keep changing. Keeping them in view the designer may be cautious in data warehouse design so as to avoid the possible pitfalls.

4.4 DATA CONTENT

In what way does the data content in the warehouse differ from the OLTP system? It is normally misunderstood that data warehouse contains lesser level of detailed data when compared to an OLTP system which may be its source. This is incorrect. The level of detail of data in the data warehouse is normally the same as that of the source OLTP system; only it may be in a different format. Typically, a data warehouse contains detail level data. But the data is cleaned and transferred to fit in the data warehouse model.

The cleaning up process removes the deficiencies and loopholes in the data. In addition, all the aspects related to transaction processing also will be filtered off before putting into the warehouse, as they are not relevant in OLAP applications of the warehouse.

The data model used in the data warehouse reflects the nature, content and structure of the data in the data warehouse. The data model represents the framework of the organization and structure of the data in the data warehouse (we have already discussed this issue in Chapter 1). The data model also identifies how the information is stored in the data warehouse. It identifies major subjects and relationships of the model, including keys, attributes and attribute groupings.

In addition to the data model, the designer should also be able to identify the querying process on the warehouse. Due to their broad scope and powerful nature, queries can also determine the data model. The data model should be optimized so as to meet high level query performance in terms of broad scope and analytical intensity. In addition, data would also have a bearing on the data storage requirements and data loading performance.

As already discussed, the data model for a data warehouse may be quite different from the data model for a data mart depending on the requirements of design of the data marts vis-à-vis the data warehouse. The star schema and snowflake schema are the most commonly used data models in data warehousing (as already discussed in Chapter 1).

4.5 METADATA

Metadata defines the contents and location of the data (or data model) in the data warehouse, relationships between the operational databases and the data warehouse and the business views of the data in the warehouse as accessible to the end-user tools. Metadata is searched by users to find the subject areas and the definitions of the data.

For decision support, the pointers required to data warehouse are provided by the metadata. Therefore, it acts as a logical link between the decision support system application and the data warehouse.

Thus, any data warehouse design should assure that there is a mechanism that populates and maintains the metadata repository and that all access paths to data warehouse have metadata as an entry point. In other words there should be no direct access permitted to the data warehouse data (especially updates) if it does not use metadata definitions to gain the access. Metadata definition can be done by the user in any given data warehousing environment. The software environment as decided by the software tools used will provide a facility for metadata definition in a metadata repository.

4.6 DISTRIBUTION OF DATA

As the data warehouse grows in size, it becomes important to decide which data is to be located where, in which server in the network, etc. Data distribution can be planned so as to be based on subject area, location (or region) or time (e.g. monthly, yearly). In other words, the data can be distributed location-wise, time-wise or subject-wise. The distribution of data should be optimal for the data itself and also for querying purposes. The performance of queries depends upon the location and distribution of the data as the retrieval time is also dependent on the easy and fast accessibility.

4.7 TOOLS FOR DATA WAREHOUSING

For various stages or steps involved in the development of the data warehouse, a number of tools are available from different vendors in the market. Some tools are single vendor based for several stages or quite often multiple vendor sources will be required for different tools in different stages. These tools provide facilities for defining the transformation and clean-up, data movement from operational data sources into the warehouse, end-user querying, reporting and finally for data analysis and data mining. Each tool takes a slightly different approach to data warehousing and often maintains its own version of metadata repository. It is important that the designer of the data warehouse may not sacrifice or dedicate the entire design process so as to fit a specific tool. In the process, loss

of semantics may happen if the tool is weak. Therefore, it is better to go for a very comprehensive tool. It is also very important to ensure that all the related tools are compatible with one another and also with the overall design.

All the tools should also be compatible with the given data warehousing environment and also with one another. This means that all the selected tools are compatible with each other and there can be a common metadata repository. Alternatively, the tools should be able to source the metadata from the warehouse data dictionary (if it is available) or from a CASE tool used to design the database in the data warehouse. Another option is to use metadata gateways that translate one tool's metadata to another tool's format (Kimball, 1996).

If the above guidelines are not meticulously followed, then the resulting data warehouse environment will rapidly become unmanageable since every modification to the warehouse data model may involve some significant and labour-intensive changes to the metadata definitions for every tool in the environment. These changes will also be required to be verified for consistency and integrity (Mattison, 1996).

4.8 PERFORMANCE CONSIDERATIONS

Even though OLAP applications on a data warehouse are not calling for very stringent, real-time responses as in the case of OLTP systems on a database, the interactions of the user with data warehouse should be online and interactive with good enough speed. Rapid query processing is desirable.

The actual performance levels may vary from application to application with different requirements, as the case may be. The requirements of query response time, as defined by a particular business application, are required to be met fully by the data warehouse and its tools. However, it is not possible to predict in advance the performance levels of a data warehouse. As the usage patterns of the data warehousing vary from application to application and are unpredictable, the traditional database design and tuning techniques do not always work in a data warehouse environment.

It is essential therefore to understand and know the specific user requirements in terms of the information and querying before the data warehouse is designed and implemented. Query optimization also can be done if frequently asked queries are well understood. For example, to answer aggregate level queries the warehouse can be (additionally) populated with specified denormalized views containing specific, summarized, derived and aggregated data.

If made correctly available, many end-user requirements and many frequently asked queries can be answered directly and efficiently so that the overall performance levels can be maintained high (Hackney, 1997).

4.9 CRUCIAL DECISIONS IN DESIGNING A DATA WAREHOUSE

The job of designing and implementing a data warehouse is very challenging and difficult one, even though at the same time, there is a lot of focus and importance attached to it. The designer of a data warehouse may be asked by the top management: "Take all enterprise data and build a data warehouse such that the management can get answers to all their questions". This is a daunting task with responsibility being visible and exciting. But how to get started? Where to start? Which data should be put first? Where is that data available? Which queries should be answered? How would you bring down the scope of the project to something smaller and manageable, yet be scalable to gradually upgrade to build a comprehensive data warehouse environment finally?

The recent trend is to build data marts before a real large data warehouse is built. People want something smaller, so as to get manageable results before proceeding to a real data warehouse.

Ralph Kimball identified a nine-step method as follows:

Step 1: Choose the subject matter (one at a time)

Step 2: Decide what the fact table represents

Step 3: Identify and conform the dimensions

Step 4: Choose the facts

Step 5: Store pre-calculations in the fact table

Step 6: Define the dimensions and tables

Step 7: Decide the duration of the database and periodicity of updation

Step 8: Track slowly the changing dimensions

Step 9: Decide the query priorities and the query modes

All the above steps are required before the data warehouse is implemented. The final step or step 10 is to implement a simple data warehouse or a data mart. The approach should be 'from simpler to complex'.

First, only a few data marts are identified, designed and implemented. A data warehouse then will emerge gradually.

Let us discuss the above-mentioned steps in detail. Interaction with the users is essential for obtaining answers to many of the above questions. The users to be interviewed include top management, middle management, executives as also operational users, in addition to salesforce and marketing teams. A clear picture emerges from the entire project on data warehousing as to what are their problems and how they can possibly be solved with the help of the data warehousing.

The priorities of the business issues can also be found. Similarly, interviewing the DBAs in the organization will also give a clear picture as what are the data sources with clean data, valid data and consistent data with assured flow for several years.

Besides the above two issues, no other issue will be suggested by the users. For example, how the design should be made can be decided only by the designer and not by any user, end-users or DBAs.

Let us examine a few steps identified above. Choosing the subject matter (step 1) is the most crucial decision. This will emerge after the above-mentioned user interaction and interviews. The most cost-effective highly demanded subject should be taken up first. The subject should be such that it can answer the user's business questions, with the data sources being available. 'Hot subjects' should be given high priority. For example, any business organization will be interested in customers invoices or monthly statements built into a data mart for analysis and mining. For step 2, the 'fact table' has to be designed.

A *fact table* (as discussed in Chapter 2) is a large control table in a dimensional design that has a multi-part key. Each component of the multi-part key is a foreign key to an individual dimensional table. The basic commodity sales figures can be in terms of regions, weeks (or months) and commodity categories as dimensions.

For step 3, 'Dimension table' has to be designed. The dimensions of the data in 'fact table' are designed as dimension table (as already discussed in earlier chapter). The *dimensions* are the platforms for browsing the available constraint values and launching these constraints.

The dimensions are the source of new headers in the users' final reports, they carry the entire vocabulary of the enterprise to the users. For example, sales data can have time dimension (day, week, month), space dimension (regions) and item class dimension (specific commodity classes). Well-architectured set of dimensions makes the data mart understandable, interesting and useful. Dimensions should be chosen with a long range data warehouse usage in mind (Hackney, 1997).

Duplicate dimensions across data marts should be avoided with a long range perspective of a data warehouse in mind. However, if two data marts end up with the same dimensions, they should be 'conforming' with each other. If the two dimensions are conformed, then it is easy to send the query across both the data marts and finally sort and merge the resultant data on a common set of new headers for a report. The new headers in the report can be common if they are drawn from a conformed dimension common to both the data marts.

If the first three stages (as discussed above) are correctly and effectively executed, then the remaining steps can be executed very easily. However, we shall now discuss some of the more general issues involved in the remaining stages.

4.9.1 Various Technological Considerations

The following technical or technological issues are required to be considered for designing and implementing a data warehouse:

1. The hardware platforms for data warehouse
2. DBMS for supporting data warehouse

3. Communication and network infrastructure for a data warehouse
4. The system management/operating system platforms
5. The software tools for building, operating and using data warehouse.

These are now discussed.

Hardware platforms

Organizations normally tend to utilize the already existing hardware platforms for data warehouse development. However, the disk storage requirements for a data warehouse will be significantly large, especially in comparison with single applications.

Thus, hardware with substantial data storage capacity is essential for data warehousing as per the estimated size. For example, if the population is in millions of records, the disk storage should be proportional and not just equal. For every data size identified, the disk space provided should be two to three times that of the data to accommodate processing, indexing, etc. Both mainframes and non-mainframes (with UNIX or Windows NT) can be used for hosting data warehouse (normally the mainframes were considered more reliable hardware even though many of them are obsolete now). Among the non-mainframes, high-end servers such as Solaris seem to be more reliable (but more expensive) compared to high-end latest Pentium servers in the market. If reliability is a very crucial factor, then 'Tandem' or any back-to-back redundant multiprocessor system can be considered. The low-end solutions will be the latest Pentium servers with RAID architectures for large disk space. If the data warehouse or data mart is small in data size, normal Pentium server will be probably sufficient with not very high reliability standards. However, for a regular large data warehouse application, the server has to be specialized for the tasks associated with a data warehouse. A mainframe, for example is well suited for this purpose, as a data warehouse server. What are the features required for a successful data warehouse server? Firstly, it should be able to support large data volumes and complex query processing. In addition, it has to be highly scalable, as a data warehouse is never finished—it will go on growing with new additions of data with time, new user requirements, new data sources with new historical data continuously incorporated into the data warehouse. As the user population keeps on growing, the network traffic and access traffic increase significantly. Therefore, the requirement of a data warehouse server is the scalable high performance for data loading and ad hoc query processing as well as the ability to support large databases in a reliable and efficient manner. If the querying is going to be on a large public data network (as Internet) then multiprocessor configuration will be required for parallel query processing. In the case of a complex server of configuration with multiple processors and large I/O

bandwidth, a proper balance needs to be made between I/O and processing power (Winter and Brobst, 1994).

The balancing between processors and I/O in a multiprocessor environment is all the more important in data warehouse applications. As the disk space requirements are three times the raw data size, the required number of online disks will be quite large. The throughput or performance efficiency comes from number of disks and parallelism. To balance this situation, it is important to allocate the correct number of processors to efficiently handle all the disk I/O operations. Otherwise, the hardware will end up becoming CPU bound in its execution thus leading to inefficiency. Various processors have different performance ratings and thus can support a different number of disks per CPU. The hardware selection should be based on efficient calculation and careful analysis of disk I/O rates and processor capabilities to derive efficient system configuration so that adequate number of processors are available for driving all the disks required (Creecy et al., 1992).

Another consideration is related to disk controller. A *disk controller* supports a certain amount of data throughput (e.g. 20 MB/sec). Thus, given the number of disks required for storing data warehouse (three times the raw data size) the number of controllers required can be calculated and provided for.

Balanced design considerations should be ensured to all system components for the sake of better efficiency. The resulting configuration will be able to easily handle the known workloads and provide a balanced and scalable computing platform for future growth of the data warehouse. It is very crucial to fine tune a parallel DBMS with a parallel processor hardware architecture for optimal results in query processing (Winter and Brobst, 1994).

DBMS selection

Next to hardware selection, a factor most critical, is the DBMS selection. This determines the speed performance of the data warehousing environment. The requirements of a DBMS for data warehousing environment are scalability, performance in high volume storage and processing and throughput in traffic.

The majority of established RDBMS vendors have implemented various degrees of parallelism in their products. Even though all the well-known vendors—IBM, Oracle, Sybase—support parallel database processing, some of them have improved their architectures so as to better suit the specialized requirements of a data warehouse. The RDBMS products provide additional modules for OLAP cubes (Yazdani and Shirley, 1997; O'Neil, 1997; Poolet and Reilly, 1997; Hammergren, 1997).

One can use the OLAP features of the same DBMS on which the data resides. However, this may not be adequate meeting certain application requirements. In such cases, OLAP servers from other reputed vendors may be used, independently of DBMS.

In addition to regular or traditional RDBMS products, there exist other database products as Red Brick Warehouse (from Redbrick Systems) that have been specifically optimized for data warehousing.

Redbrick offers relational data warehouse with multi-relational approach. The Redbrick interactive query intensive environment offers a Table Management Utility, a high-performance loading sub-system that loads and indexes operational data. A database server stores and manages the warehouse information. An interactive query tool (RISQL) is also available. Red Brick is built to work in very large databases (i.e. 500 GB+) with high speed indexing (1 GB/hr).

The correct choice of OLAP server DB server (DBMS) and Web server can be made by the designer or user of data warehouse depending on the requirement. (Detailed discussion on the software tools, including the appropriate DBMS is available in Chapter 2.)

Communication and networking infrastructure

Data warehouses can be *Internet (Web) enabled* or *Intranet enabled* as the choice may be. If Web enabled, the networking is taken care by the Internet. If only Intranet based, then the appropriate LAN operational environment should be provided so as to be accessible to all the identified users. Thus, network expansion may be required as per the needs. In Web-enabled data warehouses, issues of security, privacy and accessibility need to be considered carefully. Accordingly, Web-enablement facilities should be ensured in the software tools used for data warehouse development, as already discussed is the tool survey in Chapter 2.

Stages in implementation

A data warehouse cannot be purchased and installed. Its implementation requires the integration of implementation of many products. Following are the steps of the data warehouse implementation:

Step 1: Collect and analyse business requirements

Step 2: Create a data model and physical design for the data warehouse after deciding the appropriate hardware platform

Step 3: Define the data sources

Step 4: Choose the DBMS and software platform for data warehouse

Step 5: Extract the data from operational data sources (databases or otherwise), transfer it, clean-up and load into the data warehouse model or data mart

Step 6: Choose database access and reporting tools

Step 7: Choose database connectivity software

Step 8: Choose the data analysis (OLAP) and presentation (client GUI) software

Step 9: Keep refreshing the data warehouse periodically

Access tools

With the exception of SAS (of SAS Institute), all the data warehouses/OLAP vendors are not currently providing comprehensive single-window software tools capable of handling all aspects of a data warehousing project implementation. SAS alone meets the requirements largely independently as it has its own database engine internally with a capability to import data from any vendor DBMS software. Therefore, one can implement a data warehousing and data mining solution independently with SAS. Normally, in all other platforms, a suite of tools from a variety of vendors are required for implementing data warehousing. (Even though the third-party vendor software products may also be supplied by the DBMS vendor himself, as an option.) Data viewing tools, as discussed in Chapter 3 are to be separately processed.

The best way to choose a group of tools is to understand the capabilities and compatibility of different types of access to the data and reporting by selecting best tool in the market for that kind of access. The types of access and reporting are as follows:

1. Time series analysis (statistical analysis and forecasting)
2. Data visualization, graphing, charting and pivoting
3. Complex textual search (text mining)
4. General statistical analysis
5. Artificial intelligence techniques for hypothesis testing, trends discovery, identification and validation of data clusters and segments (also useful for data mining)
6. Mapping of spatial information into geographic information system
7. Ad hoc user-specific queries
8. Predefined repeatable queries
9. Drilling down interactively
10. Reporting and analysis by drilling down
11. Complex queries with multi-table forces, multilevel sub-queries, sophisticated search criteria

In some applications, the user requirements may exceed the capabilities of the tools. In such cases it is essential that tailor-made or custom-designed software be developed for reporting and analysis purposes to bring out the hidden meaning of data so as to meet the users' requirements.

A number of query tools are available in the market today which enable an ordinary user to build customized reports by easily composing and executing ad hoc queries without any necessity to have the knowledge of the underlying design details or database technology, SQL, or even the data model (as business

objects or 'Impromptu' from Cognos, etc.). On the other hand, there are others (as Andyne's GQL) which provide relatively low-level capabilities for an expert user to develop complex ad hoc queries in a fashion similar to developing SQL queries for relational databases. Business requirements which exceed the above capabilities can be met by any other tools (these tools are already discussed in Chapter 2).

CONCLUSION

In this chapter, the sequential steps involved in the development of a data warehouse have been presented. Starting with the need and motivation for developing a data warehouse, various issues such as content in a data warehouse, other issues such as hardware, networking infrastructure requirements for a data warehouse, the software environment and various tools. The issues involved in multiprocessor architecture and the I/O devices have also been discussed. The range of tools available in the market with their mutual compatibility requirements and their features and capabilities have been discussed. Finally, various stages and steps in data warehouse implementation to guide the reader on how to implement a data warehouse are also described.

CHAPTER 5

Applications of Data Warehousing and Data Mining in Government

5.1 INTRODUCTION

Data warehousing and data mining are the important means of preparing the government to face the challenges of the new millennium.

Data warehousing and data mining technologies have extensive potential applications in the government: in various Central Government sectors such as Agriculture, Rural Development, Health and Energy. These technologies can and should therefore be implemented. Similarly, in State Government activities also, large opportunities exist for applying these techniques. Almost all these opportunities have not yet been exploited.

In this chapter, we shall examine both the Central and the State Government applications in terms of the potential applications and actual case studies of already implemented applications are given in Appendices.

5.2 NATIONAL DATA WAREHOUSES

A large number of national data warehouses can be identified from the existing data resources within the Central Government ministries. Let us examine these potential subject areas on which data warehouses may be developed at present and also in future.

5.2.1 Census Data

The Registrar General and Census Commissioner of India decennially compiles information of all individuals, villages, population groups, etc. This information is wide ranging such as the individual-slip, a compilation of information of individual households, of which a database of 5% sample is maintained for analysis. A data warehouse can be built from this database upon which OLAP techniques can be applied. Data mining also can be performed for analysis and knowledge discovery.

A village-level database was originally developed by National Informatics Centre at Hyderabad under General Information Services Terminal of National Informatics Centre (GISTNIC) for the 1991 Census. This consists of two parts: primary census abstract and village amenities. Subsequently, a data warehouse was also developed for village amenities for Tamil Nadu. This enables multidimensional analysis of the village-level data in such sectors as education, health and infrastructure. The fact data pertains to the individual village data compiled under 1991 Census. (A case study report on this is also presented as in Case Study 1.)

As the Census compilation is performed once in 10 years, the data is quasi-static and, therefore, no refreshing of the warehouse needs to be done on a periodic basis. Only the new data needs to be either appended to the data warehouse or alternatively a new data warehouse can be built.

There exist many other subject areas (e.g. migration tables) within the census purview which may be amenable and appropriate for data warehouse development, OLAP and data mining applications on which work can be taken up in future.

5.2.2 Prices of Essential Commodities

The Ministry of Food and Civil Supplies, Government of India compiles daily data (on weekly basis) for about 300 observation centres in the entire country on the prices of essential commodities such as rice, edible oils, etc. This data is compiled at the district level by the respective State Government agencies and transmitted online to Delhi for aggregation and storage. A data warehouse can be built for this data and OLAP techniques can be applied for its analysis. A data mining and forecasting technique can be applied for advance forecasting of the actual prices of these essential commodities. The forecasting model can be strengthened for more accurate forecasting by taking into account the external factors such as rainfall, growth rate of population and inflation.

A limited exercise in this direction was already executed at a state level (in Tamil Nadu). Case Study 1 studies the essential commodity prices for Tamil Nadu, using SAS tools.

5.3 OTHER AREAS FOR DATA WAREHOUSING AND DATA MINING

Other possible areas for data warehousing and data mining in Central Government sectors are discussed in detail in the following sections.

Agriculture

The Agricultural Census performed by the Ministry of Agriculture, Government of India, compiles a large number of agricultural parameters at the national level. District-wise agricultural production, area and yield of crops is compiled; this can be built into a data warehouse for analysis, mining and forecasting. Statistics on consumption of fertilizers also can be turned into a data mart.

Data on agricultural inputs such as seeds and fertilizers can also be effectively analysed in a data warehouse. Data from livestock census can be turned into a data warehouse. Land-use pattern statistics can also be analysed in a warehousing environment. Other data such as watershed details and also agricultural credit data can be effectively used for analysis by applying the technologies of OLAP and data mining.

Thus there is substantial scope for application of data warehousing and data mining techniques in agricultural sector.

Rural development

Data on individuals below poverty line (BPL survey) can be built into a data warehouse. Drinking water census data (from Drinking Water Mission) can be effectively utilized by OLAP and data mining technologies. Monitoring and analysis of progress made on implementation of rural development programmes can also be made using OLAP and data mining techniques.

Health

Community needs assessment data, immunization data, data from national programmes on controlling blindness, leprosy, malaria can all be used for data warehousing implementation, OLAP and data mining applications.

Planning

At the Planning Commission, data warehouses can be built for state plan data on all sectors: labour, energy, education, trade and industry, five year plan, etc.

Education

The Sixth All India Educational Survey data has been converted into a data warehouse (with about 3 GB of data). Various types of analytical queries and reports can be answered.

Commerce and trade

Data bank on trade (imports and exports) can be analysed and converted into a data warehouse*. World price monitoring system can be made to perform better

*This data is available with the Director General of Foreign Trade, Ministry of Commerce.

by using data warehousing and data mining technologies. Provisional estimates of import and export also be made more accurate using forecasting techniques.

Other sectors

In addition to the above-mentioned important applications, there exist a number of other potential application areas for data warehousing and data mining, as follows:

Tourism. Tourist arrival behaviour and preferences; tourism products data; foreign exchange earnings data; and Hotels, Travel and Transportation data.

Programme implementation. Central projects data (for monitoring).

Revenue. Customs data; central excise data; and commercial taxes data (state government).

Economic affairs. Budget and expenditure data; and annual economic survey.

Audit and accounts. Government accounts data.

All government departments or organizations are deeply involved in generating and processing a large amount of data. Conventionally, the government departments have largely been satisfied with developing single management information systems (MIS), or in limited cases, a few databases which were used online for limited reporting purposes. Much of the analysis work was done manually by the Department of Statistics in the Central Government or in any State Government. The techniques used for analysis were conventional statistical techniques on largely batch-mode processing. Prior to the advent of data warehousing and data mining technologies nobody was aware of any better techniques for this activity. Now with the advent and prominence of the data warehousing and data mining technology, there is a paradigm shift. Data warehousing and data mining technologies could lead to the most significant advancements in the government functioning, if properly applied and used in the government activities.

This paradigm shift may finally result in improved governance and better planning by better utilization of data. Instead of the officials wasting their time in processing data, they can rely on data warehousing and data mining technologies for their day-to-day decision-making and concentrate more on the practical implementation of the decisions so taken for better performance of developmental activities.

Further, even though the various departments in the government (State or Central) are functionally interlinked, the data is presently generated, maintained and used independently in each department. This leads to poor (independent) decision-making and isolated planning. Herein lies the importance of data

warehousing technology. Different data marts for separate departments, if built, can be integrated into one data warehouse for the government. This is true for State Government and Central Government. Thus data warehouses can be built at Central level, State level and also at District level.

CONCLUSION

In the government, the individual data marts are required to be maintained by the individual departments (or public sector organizations) and a central data warehouse is required to be maintained by the ministry concerned for the concerned sector. A generic inter-sectoral data warehouse is required to be maintained by a central body (as Planning Commission). Similarly, at the State level, a generic inter-departmental data warehouse can be built and maintained by a nodal agency, and detailed data warehouses can also be built and maintained at the district level by an appropriate agency. National Informatics Centre may possibly play the role of the nodal agency at Central, State and District levels for developing and maintaining data warehouses in various sectors.

CASE STUDIES

CASE STUDY 1

Data Warehousing in the Tamil Nadu Government

C1.1 GISTNIC DATA WAREHOUSE

General Information Service Terminal of National Informatics Centre (GISTNIC) Data Warehouse* is an initiative taken by National Informatics Centre (NIC) to provide a *comprehensive information database* by the government on national issues ranging across diverse subjects like food and agriculture to trends in the economy and latest updates on science and technology. This information base was collated to fulfil the information needs of bureaucrats, politicians, economists and, most important of all, the citizens.

The GISTNIC data warehouse is a *Web-enabled* SAS software solution. The data warehouse aims at providing online information to key decision-makers in the government sector enabling them to make better strategic decisions with regard to administrative policies and investments. The Government of Tamil Nadu is the first one to perceive the need and importance of converting *data into valuable information for better decision-making*. The GISTNIC Web site currently has an online data warehouse which includes data marts on village amenities, rainfall, agricultural census data, essential commodity prices, malaria statistics, Indian economy statics and school health (see Fig. C1.1). This data warehouse was developed during 1998–99.

Information technology pioneers from several public and private sectors appreciate the GISTNIC data warehouse as under:

> The GISTNIC data warehouse is a web-enabled solution, which intends to provide easy access with point and click interfaces to key decision makers across various levels in the government. The effectiveness is evident in the form of simultaneous availability of information and the speed at which it is delivered.
>
> *D. Prakash, Secretary, Information Technology, Government of Tamil Nadu*

> The GISTNIC data warehouse endorses the beginning of a strategic business alliance between SAS Institute and the government sectors. SAS Institute is committed the world over to government organizations and intends to

*This GISTNIC data warehouse for Tamil Nadu was implemented during 1998.

pursue this in India as well. We shall endeavor to extend full support at all times.

Rohini Midha, Managing Director, SAS Institute India Pvt. Ltd.

We look at this data warehouse as a means of preparing the government to face the challenges of the next millennium. The Tamil Nadu Government will be India's first State Government which will ride on the technology wave to streamline processes and drive better decision making on the basis of communication of information in whatever form it may take, unconstrained by distance, time and volume.

C.S.R. Prabhu, Senior Technical Director and State Informatics Officer, National Informatics Centre, Tamil Nadu

The GISTNIC data warehouse for Tamil Nadu indicates the commitment of SAS Institute in working together with the government to enable them to move towards their being IT-savvy.

Gourish Hosangady, National Sales and Technical Manager, SAS Institute India Pvt. Ltd.

C1.2 OBJECTIVES OF THE WEB-ENABLED DATA WAREHOUSE

- To provide powerful decision-making tools in the hands of the end-users in order to facilitate prompt decision-making
- To reduce the amount of resources—time and manpower—spent on managing the volumes and variety of database handled by NIC.

C1.3 DATA ANALYSIS

Data marts of various sectors and their applications are given in Table C1.1 as follows:

Table C1.1 Data Marts of various sectors and their applications

Data mart	Applications
Village amenities. This data mart contains the 1991 census data of village amenities in all the villages in Tamil Nadu. It contains information on availability for amenities like education, health, drinking water, transportation, communication and irrigation (see Figs. C1.2–C1.5).	• Village amenities analysis • Irrigation analysis • Top/bottom analysis • Range analysis—amenities • More amenities—analysis
Rainfall statistics. This data mart has information on daily levels of rainfall across various weather stations in Tamil Nadu. This will help them to plan the water supply to various districts in Tamil Nadu and using various models to forecast rainfall levels (see Fig. C1.6).	• Time-based rainfall analysis • Geography/time-based rainfall analysis

(*contd.*)

Table C1.1 Data Marts of various sectors and their applications (*contd.*)

Data mart	Applications
Agricultural census. This data mart has information on land-holding patterns across the villages in Tamil Nadu. It can be used to analyse information about land-holding amongst individuals, institutions, males, females, scheduled castes and scheduled tribes, etc. (see Figs. C1.7–C1.8).	• Land-holding analysis • Land-holding analysis–multidimensional • Top/bottom analysis • Medium-holding analysis
Essential commodities. To provide updated information on various essential commodity prices, NIC collects the retail/wholesale prices of various essential commodities like vegetables, sugar, rice, oil, cereals, etc. Using the GISTNIC data warehouse, end-users now have the updated information about trends in price change and will thus be able to closely monitor the prices more effectively (see Fig. C1.9).	• Commodity price analysis • Time-wise commodity • Qtr3–4 analysis • Rice analysis • Forecast—rice prices
Malaria statistics. This data mart has information on various health camps conducted across Tamil Nadu to detect and cure malaria patients. This has vital information like number of people suffering from malaria, deaths caused due to malaria, source of malaria infection, demographic information of malaria patients, etc. Using the data warehouse, the end-users will be able to plan various precautionary measures to reduce the number of people suffering from malaria in Tamil Nadu (see Fig. C1.10).	• MDR census on samples collected and tested • MDR source-of malarial parasites • MDR age- and sex-wise malarial census • Graph- and sex-wise malarial census
Indian economy. This data mart has information about statistics on the telecom sector, stock exchange (NSE and BSE) and India's foreign trade. This data is collected on monthly basis from CMIE, Mumbai.	• Capital market analysis • Capital market analysis—MDDB report • Basic telecom analysis—overview • Basic telecom analysis—state-wise • External trade analysis (1997–1998) • Combine report

(*contd.*)

Table C1.1 Data Marts of various sectors and their applications (*contd.*)

Data mart	Applications
School health This data mart has information about various health check-up camps conducted in various schools across Tamil Nadu. It has information about students suffering from various diseases, defects, immunization programmes, etc. (see Fig. C1.11)	• Disease analysis—MDDB report • Disease analysis—graph • Immunization analysis—MDDB • Immunization analysis—graph

Fig. C1.1 The first Web-enabled data warehouse for Tamil Nadu.

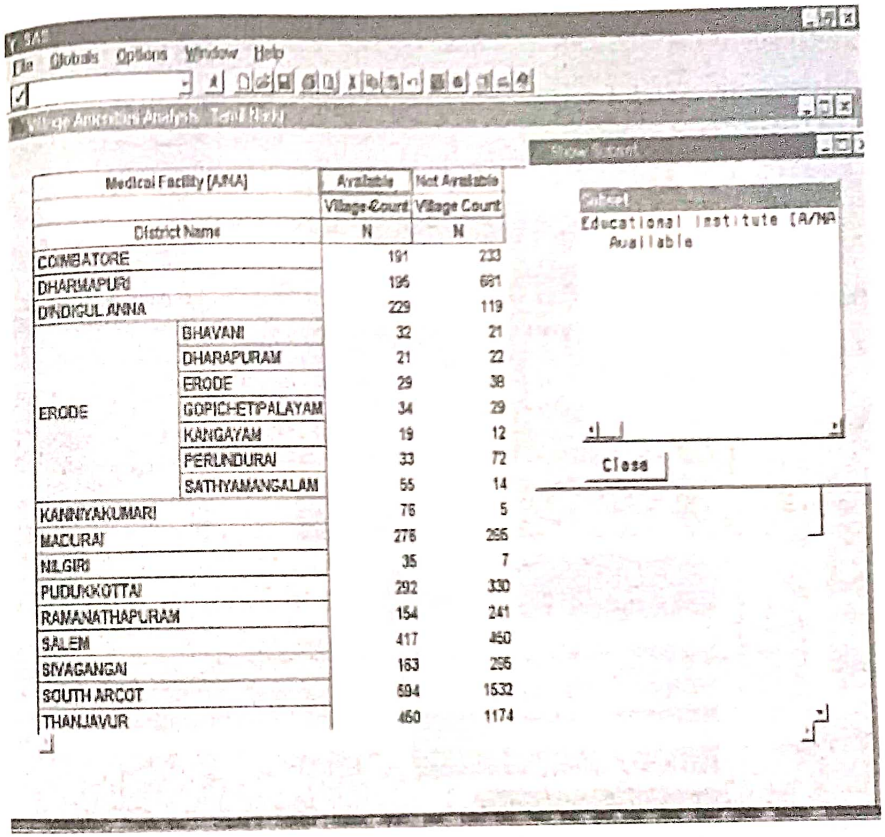


Fig. C1.2 SAS/multidimensional report on the village amenities data. This report is used to analyse villages in Tamil Nadu based on availability of various amenities.

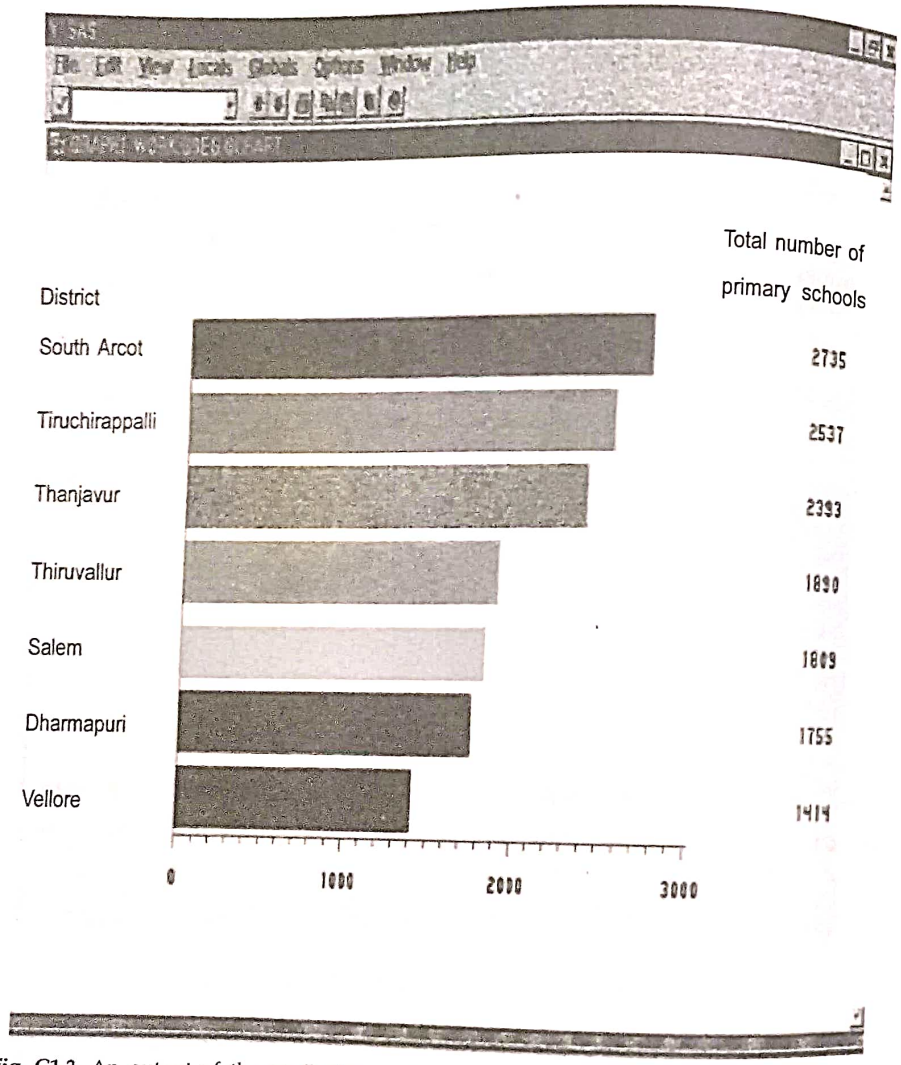


Fig. C1.3 An output of the top/bottom analysis on the village amenities data. This screen displays the number of primary schools in top seven districts in Tamil Nadu.

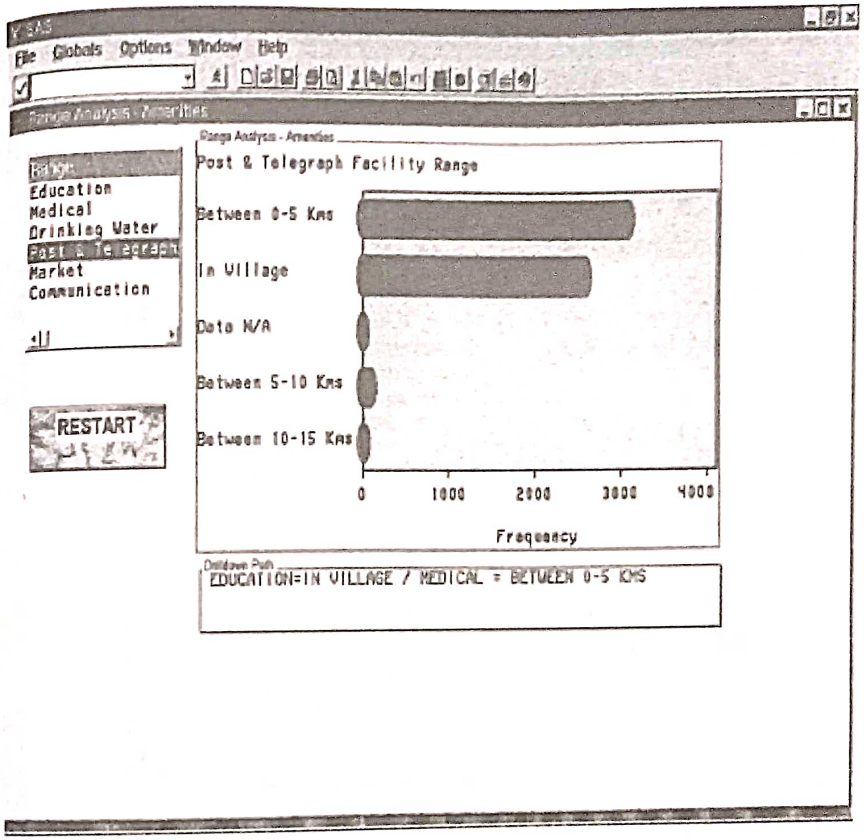


Fig. C1.4 An amenities range analysis screen. It helps analyse if a particular amenity (like education) is not available in the village, then how far (0–5 km, 5–10 km, 10–15 km) is it available.

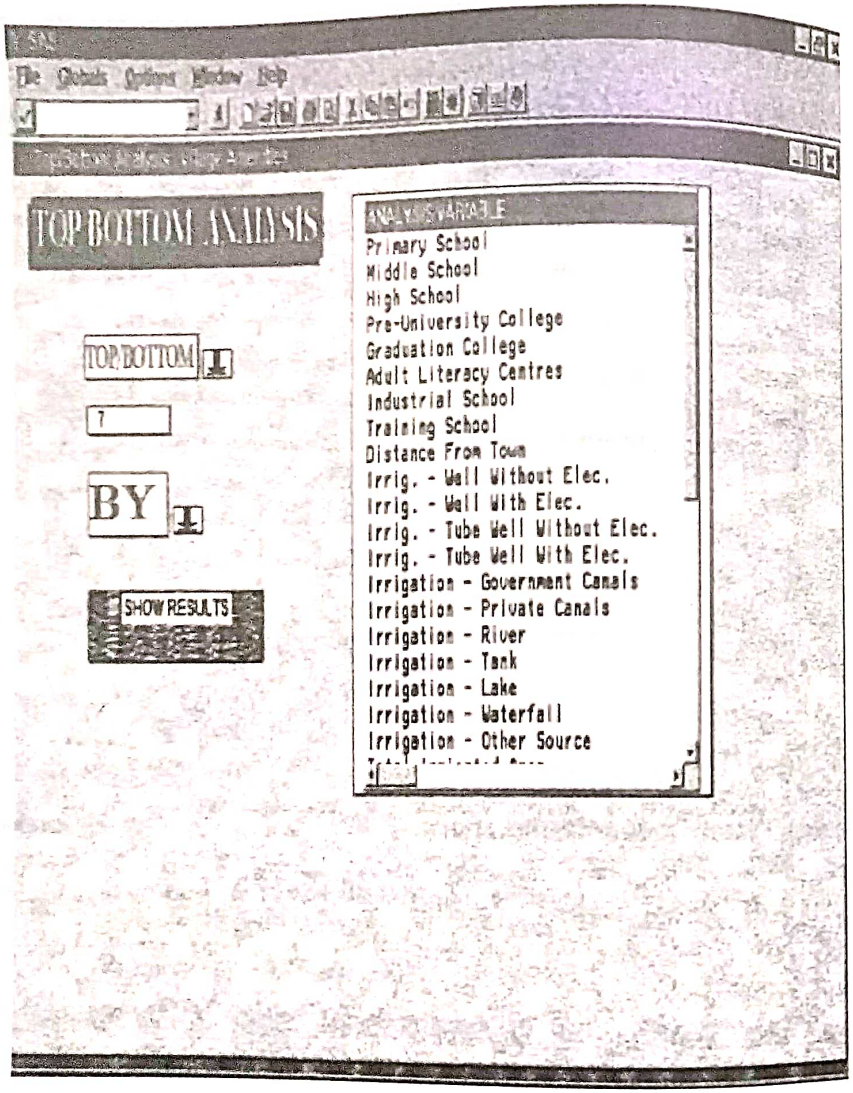


Fig. C1.5 A top/bottom analysis screen which helps in listing names of districts/talukas/villages based on various parameters mentioned in the list box.

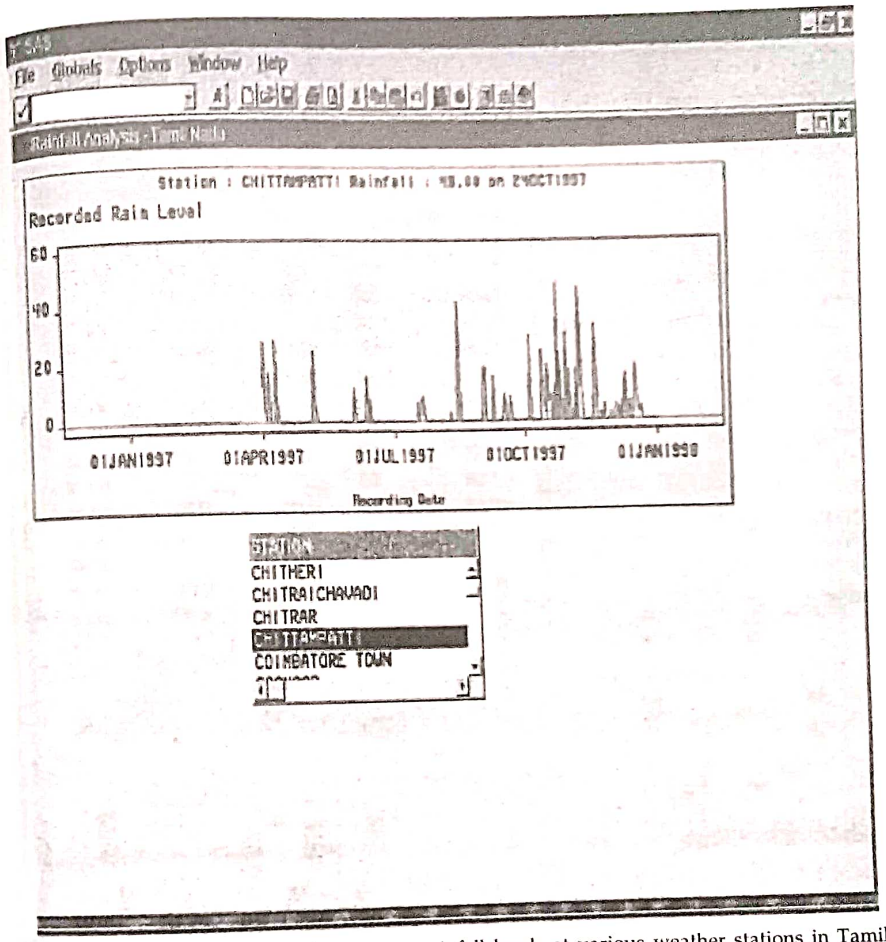


Fig. C1.6 A line plot screen for tracking rainfall levels at various weather stations in Tamil Nadu. On selecting one of the weather stations in the list box on the screen, the line plot changes to reflect the rainfall level for the selected weather station. On clicking any point on the line plot the graph displays the data and rainfall level for the data point.

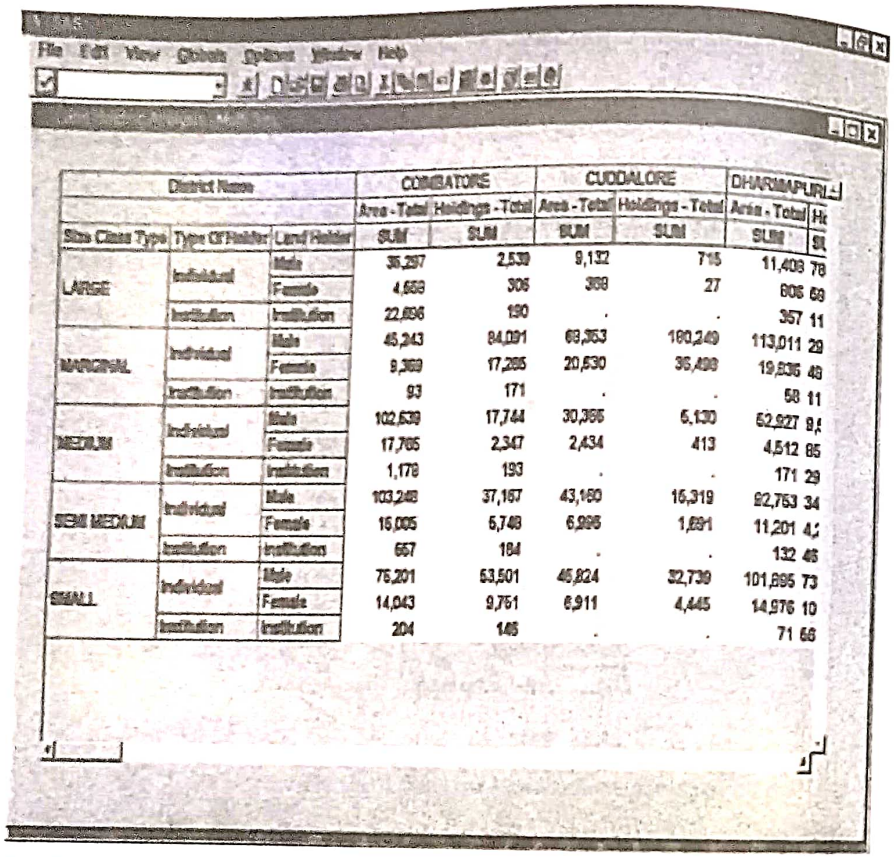


Fig. C1.7 A SAS/EIS multidimensional report on the agricultural census data. The report is displaying area under holding/number of holdings based on size, type of holder, sex of the holder, name of the district, etc.

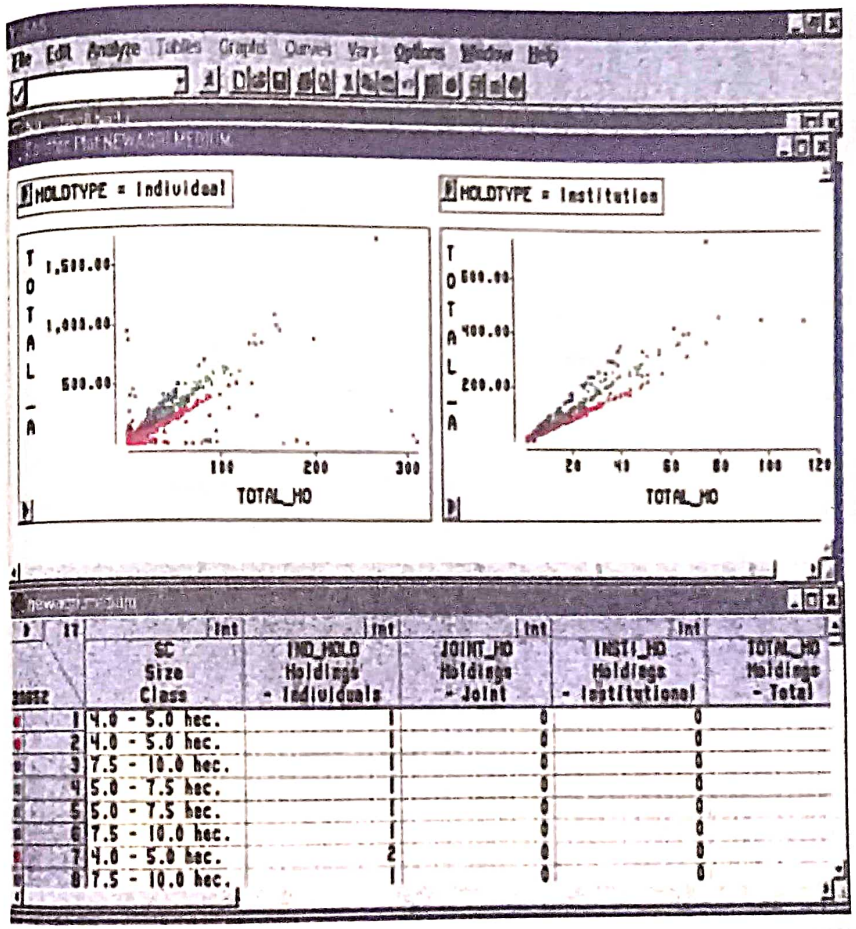


Fig. C1.8 A data visualization screen. The screen helps analysing the area under holdings for individuals and institutions. The various colour patterns in the data reflect the various sizes of the holdings.

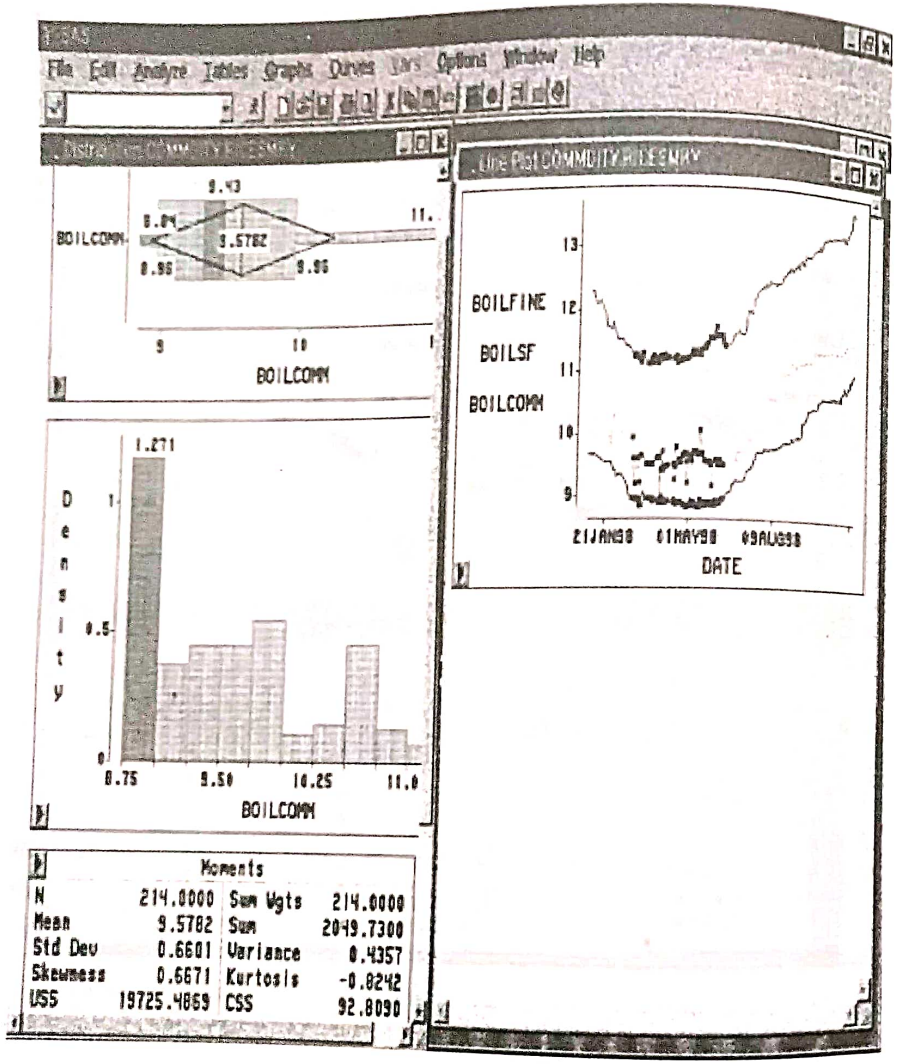


Fig. C1.9 An output of analysis on the retail price of rice in 1998. It displays a line plot, box plot, histogram and certain descriptive statistics on rice.

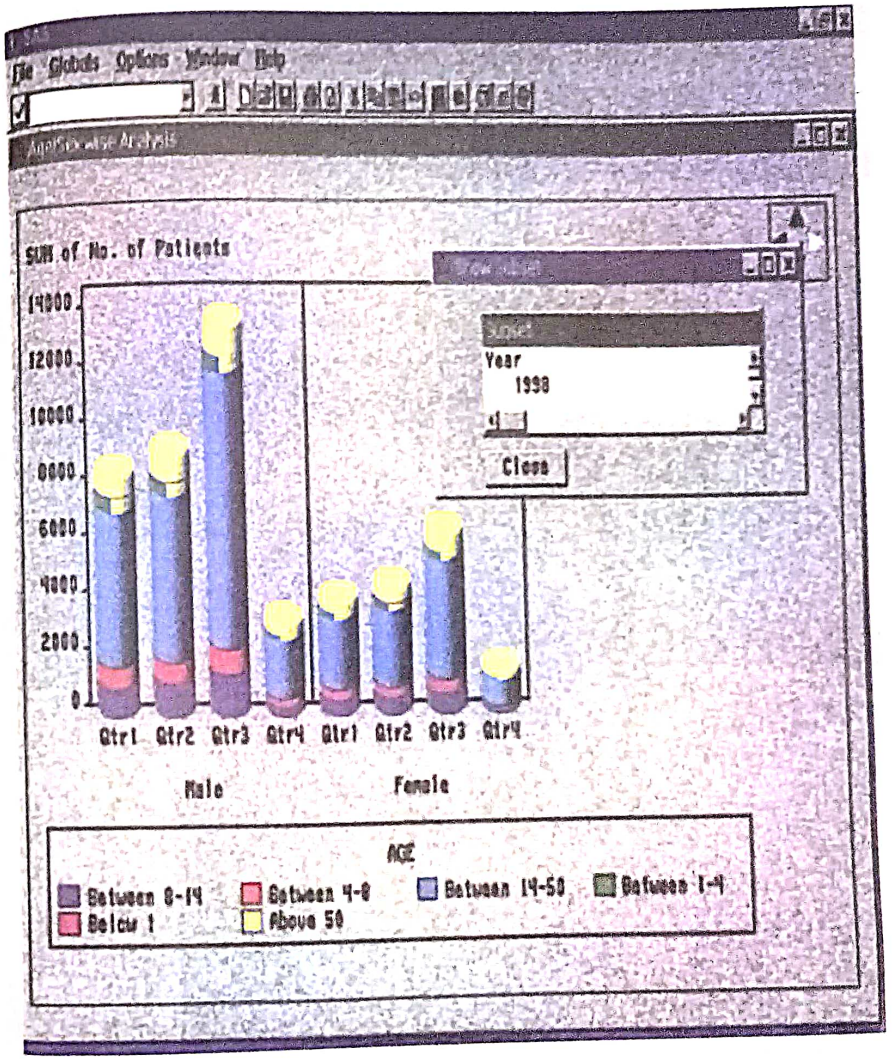


Fig. C1.10 Number of the male/female malaria patients in four quarters of 1998.

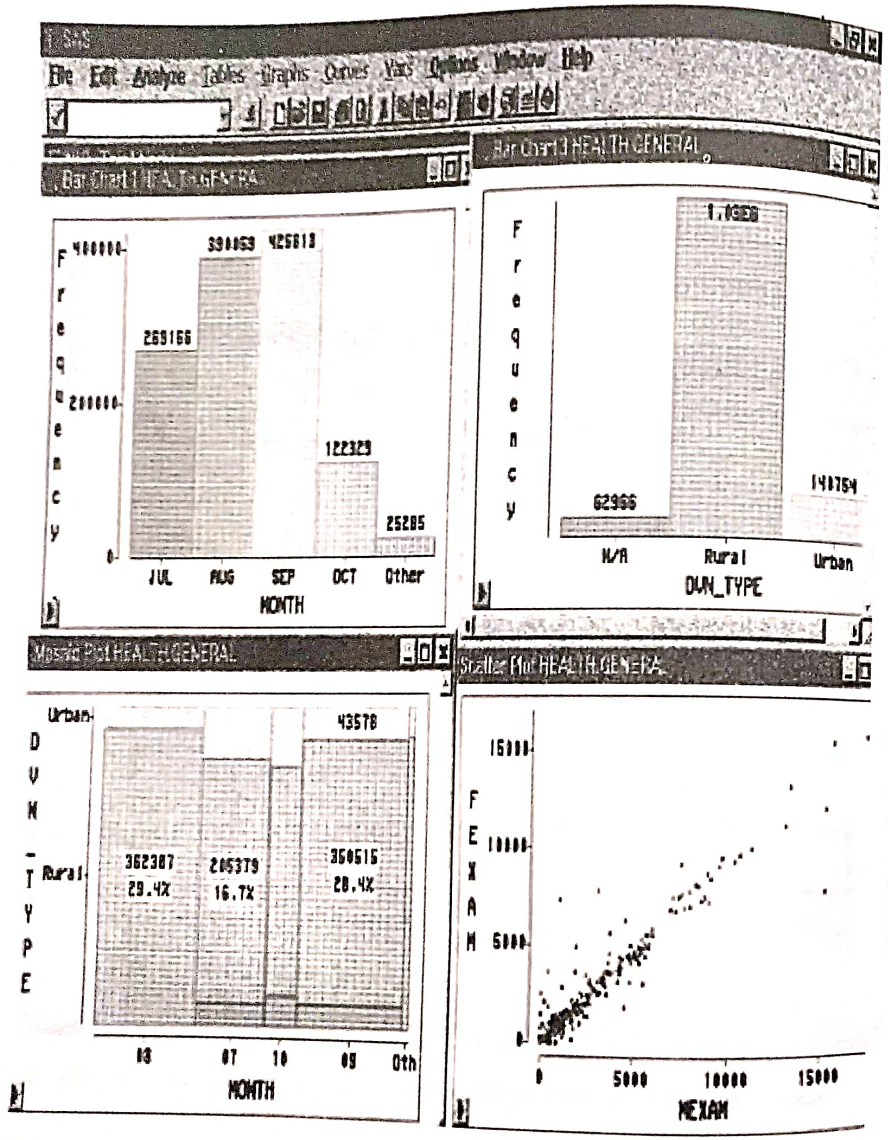


Fig. C1.11 This screen displays the number of male students examined in the school health camps.

CASE STUDY

2 Data Warehouse for the Ministry of Commerce

C2.1 INTRODUCTION

After the liberalization of the economy in 1991, foreign companies have shown keen interest in investing in India. Foreign investment approvals have gone up from US $235 million in 1991 to US $16281 million in 1995 and inflow of foreign direct investment has gone up from US $155 million in 1991 to US $4055 million in 1996 (till December). The investment boom has been injecting a high tide of competitiveness in Indian industries. In the wake of liberalization, many export-friendly schemes have been introduced. To give a further boost to exports, EXIM policies have been drastically liberalized and the procedural bottlenecks have been removed a great deal over the years. Amongst various export promotion schemes, export oriented unit (EOU) and export processing zone (EPZ) schemes occupy an important and distinct place in the country's export effort. The scheme offers an attractive package of incentives and facilities to investors.

The Ministry of Commerce has set up the following seven export processing zones at various locations as illustrated in Fig. C2.1.

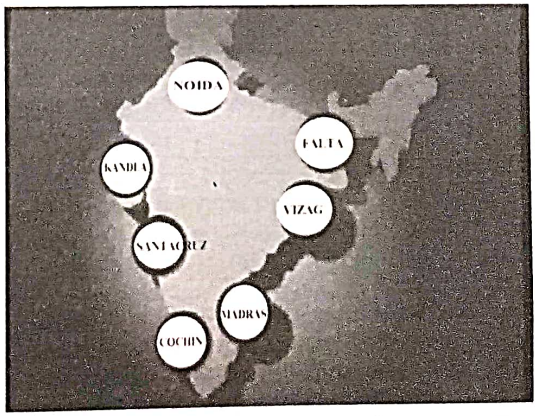


Fig. C2.1 The IP advantage VSATS in all the seven export processing zones in the country.

1. Kandla Free Trade Zone, Gandhidham
2. Santacruz Electronics Export Processing Zone, Bombay
3. Falta Export Processing Zone, Falta (West Bengal)
4. Madras Export Processing Zone, Chennai
5. Cochin Export Processing Zone, Cochin
6. Noida Export Processing Zone, Noida (UP)
7. Visakhapatnam Export Processing Zone, Visakhapatnam

The EPZs are set up to provide an internationally competitive duty-free environment for export production at low cost. The EPZs are responsible for the administration of free trade zones in setting up 100% export-oriented multi-product industrial units. Each export processing zone is headed by a development commissioner. The office of the development commissioner is responsible for the management of the zone and overall development of export from the zone. The development commissioner is the licensing authority in respect of the zone units.

This case study report presents how the data warehouse can be effectively built for the Ministry of Commerce, which is in the phase-II of the computerization programme. In terms of infrastructure, data management and OLAP decision support tools are used to achieve the following objectives:

1. Globalization of India's foreign trade
2. Attracting foreign investment
3. Scaling down tariff barriers
4. Encouraging high and internationally acceptable standards of quality
5. Simplification and streamlining of procedures governing imports and exports

C2.2 WHAT IS DATA WAREHOUSE?

Data warehouse is subject-oriented, integrated, time-varient and non-volatile collection of data in support of management's decision-making process.

The *first* feature of the data warehouse is that it is oriented around the major subjects of the enterprise. The data-driven, subject orientation is in contrast to the more classical process/functional orientation of applications around which most older operational systems are organized. The operational world is designed around applications and functions such as shipping bill filing, custom gate operations, permission, etc. The data warehouse world is organized around major subjects such as zone, exporter, product, activity, etc. The application world is concerned both with database design and with process design. The data

warehouse focussess on data modelling and database design exclusively. Operational data maintains an ongoing relationship between two or more tables based on policy that is in effect. The data from data warehouse spans a spectrum of time, and relationships found in the data warehouse are of many types. Various rules are represented in a data warehouse between two or more tables.

The data found in data warehouse is *integrated*. The integration in the data warehouse is shown in many ways—in consistent naming conventions, in consistent measurement of variables, in consistent encoding structures, in consistent physical attributes of data, and so forth.

All data in the data warehouse is accurate for some moment of time. This basic characteristic of data in the warehouse is very different from the data found in the operational environment. In the operational environment, the data is accurate during accessing. The data in the warehouse represents data over a long time horizon, that is 5–10 years. The time horizon in the operational environment is much shorter, 60–90 days. Every key structure in the data warehouse contains implicitly or explicitly an element of time, such as day, week, month, etc. Once correctly recorded with time the data cannot be updated in a data warehouse unlike the operational systems.

There are only two types of operations in a data warehouse: *initial loading of data* and *access of data*. There is no update of data, in general, in the data warehouse as a normal part of processing.

C2.3 OBJECTIVES OF DATA WAREHOUSE FOR THE MINISTRY OF COMMERCE

The Ministry of Commerce (MoC) has been regularly reviewing the data warehouse in its board meetings. The ministry is equipped with all analysis variables, a reporting form and the zone performance, and the progress of exports are reviewed in each zone, and so of the whole country. Any major setback in exports, defunct units, employees' problems in a unit set-up within the zone, etc. are analysed with the tabulated information generated from the operational system and with very little analytical processing. The analysis also becomes difficult with information in different formats. And drilling down to the problem level is not possible to detect the exact month or day the event occurred to trigger an activity, which improves export performance and the other things.

The data warehouse includes all analysis variables into consideration from all the zones with an option to drill down to the daily data. The aggregation by week, month, quarter, year are possible in association with all other analysis variables. This, when implemented, will take the planner to the problem area with a few clicks of a mouse button. Further, data mining will help them to discover many facts, which will help the planners in the way of revising the EXIM policy in favour of the country's progress. This data warehouse was implemented during 1999.

C2.4 DATA WAREHOUSE IMPLEMENTATION STRATEGY FOR EXPORT PROCESSING ZONES

C2.4.1 Infrastructure

The basic infrastructure required for building the warehouse for the Ministry of Commerce is based on the communication infrastructure, the hardware/software/tools, and manpower.

The communication infrastructure

This includes all the tasks necessary to provide the technical basis for the warehouse environment. This includes the connectivity between the legacy environment and the new warehouse environment on a network as well as on a database level, and the implementation of a population subsystem, an administrative subsystem, and a management subsystem.

All the export processing zones push the data through VSAT into 'data warehouse' service in Delhi. This server is a part of Internet in Udyog Bhavan under the Ministry of Commerce. This is depicted in Fig. C2.2.

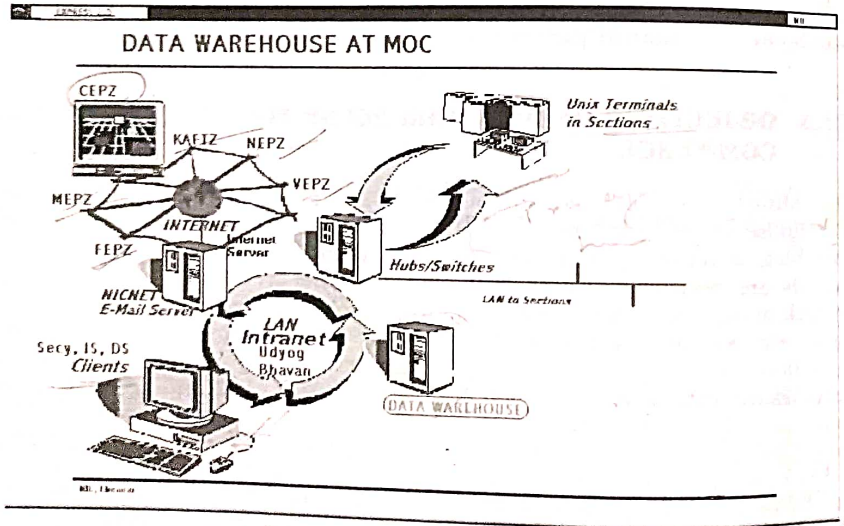


Fig. C2.2 Data warehouse at the Ministry of Commerce.

The hardware and software tools

The necessary hardware for the warehouse server with a backup server is placed in the existing WAN of NICNET (NICNET is the National Informatics Centre

Network connecting all Districts and State Capitals with New Delhi and also providing a gateway to Internet). This connects all the existing zones and also the clients in the Ministry of Commerce used by the senior officials at the ministry. The infrastructure at the Ministry of Commerce will be depending on the cost, performance and need and the tools for building the data warehouse, OLAP and mining can be decided.

Manpower requirements

The senior officials in the Ministry of Commerce sponsored the whole warehouse implementation, and played active role as EXIM policy and business architect for data warehouse, and also as subject area specialists. National Informatics Centre provided the technical know-how for managers, administrators, platform specialists and tools specialists.

C2.4.2 Key Areas for Analysis

The data topic deals with the data access, mapping, derivation, transformation, and aggregation according to the requirement of the ministry. The various key areas of decision-making, which will be used for analytical processing and for data mining at the Ministry of Commerce are listed as follows:

1. Unit-wise, sector-wise, and country-wise imports and exports
2. Direction of imports and exports
3. Sector-wise, country-wise and zone-wise trends imports and exports
4. Comparative country-wise export and import for each sector
5. DTA sales
6. Claims of reimbursement of central sales tax of zones
7. Deemed export benefits
8. Employment generation
9. Investments in the zone
10. Deployment of infrastructure
11. Growth of industrial units
12. Occupancy details

C2.5 IMPLEMENTATION OF DATA WAREHOUSE

Operational data systems

The operational data system (ODS) keeps the organization running by supporting daily transactions such as import and export bills submitted to customs

department in each zone, daily transactions of permissions, allotments, etc. in the estate management within the zone, permissions for all kinds of transactions within the zone, etc.

Future directions, strategies, analysis and faster, better and intelligent decisions based on analysis are not provided by the ODS. Some queries during the analytical processing are as follows:

- What is the export in the textile products during 1997?
- What is the impact in export pattern when the depreciation norms for capital equipment including computer hardware for electronic units raised to overall limit of 70%?
- What is the industry-wise export pattern in each zone?
- Why is there a sudden increase in exports of 'computer software' in MEPZ and SEEPZ during March to June 1998 while other zones remained idle?
- What EXIM policy changes would increase the exports by at least 10% in the next financial year?

The first three queries can be directly answered by SQL and reporting tools. The last two queries require complex data processing. Hence, the analytical processing is needed.

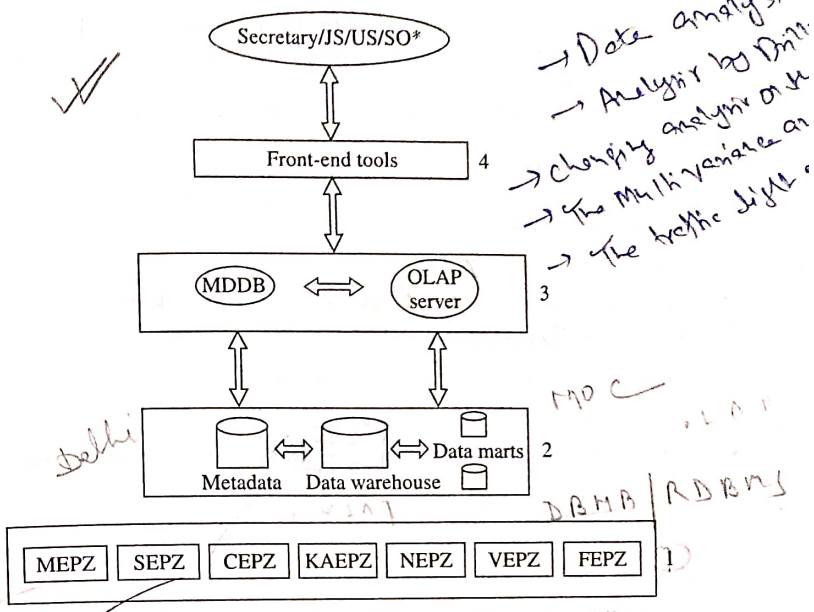
Architecture

Figure C2.3 represents the architecture of the OLAP implementation. All the seven zones have DBMS/RDBMS data for internal management of zone activities and have been forwarding the MIS reports to the MoC, New Delhi. These DBMS/RDBMS data constitute the first level in the architecture of the warehouse implementation.

The second layer is located at the MoC, New Delhi with large metadata repository and data warehouse. All the analysis variables are identified and the data related to these variables are extracted from level 1 of the architecture. The extracted data is tested for integrity, completeness of the data. Then the final data for all analysis/decisions are pushed into the metadata repository located at MoC, New Delhi from all the zones through the VSAT-based communication line. This procedure can be automated for regular update with definite periodicity.

The metadata warehouse tools and OLAP server handling and maintaining the same are focussed in level-3 of the architecture, where the necessary multidimensional databases (MDDB) are built, if the response time from the warehouse is not adequate.

Various front-end tools which are available today including the spreadsheet at the lowest level can be used for all analysis by the decision-makers at the Ministry of Commerce.



* JS—Joint Secretary, US—Under Secretary, SO—Section Officer

Fig. C2.3 Architecture of OLAP implementation in the Ministry of Commerce.

C2.6 DESIGN OF ANALYSIS/CATEGORY VARIABLES

To prepare a data model, the entire data availability and data requirement are analysed and the analysis variables for building the multidimensional cube are listed. The detail processes require general information on the number of approvals, current status of each unit (stages), production details and the unit details.

These details give the following information:

- Employment generation managerial/skilled/unskilled classification with zone/unit/industry break-up.
- Investments in the zone.
- Performance of units and close monitoring during production.
- Deployment of infrastructure (estate management system).

Related tables. ep_mast, ep_stmst, dist_mast, states, indu_mast; eo_mast, eo_stmst.

Analysis variables. EPZ or EOU; zone; type of approval (auto or board); type of industry; state, district, year of approval; month of approval; day of approval;

year of production commencement; month of production commencement; day of production commencement; current status; date of current status; net foreign exchange percentage; stipulated NFE; number of approvals; number of units.

Any of the above parameters can be selected as a dimension except the last two variables—number of approvals and number of units—which are directly generated using the statistical functions.

Exports and imports values with zone, sector, country, port, year, month and day break-ups deal with the following performances:

- Zone-wise performance
- Industry-wise performance overall with zone break-up
- Country-wise performance

This will indicate the following direction of exports:

- Country-wise performance overall and with zone break-up
- Port-wise performance which will decide the examination of infrastructure in these ports
- Export performance during different time periods and the analysis of the same
- Trend over years/quarters/months for different country, sector, and zone
- Comparative country-wise import/export for each sector

Similarly, the analysis variables on various related subjects are listed as follows:

Related tables. Shipbill, country, industry, currency, shipment_mode, export, bill entry.

Export analysis variables. Zone; sector/industry; export type; year of shipping bill; month of shipping bill; day of shipping bill; country; currency; mode of shipment; destination port; value of export (in Rs.); number of units.

Import analysis variables. Auto/approval; zone; import type; import year; import month; import day; type of goods (CG/RM/others); import value (in Rs.); duty foregone; import country; mode of shipment.

DTA sales

Analysis variables. Zone; unit name; VA stipulated; VA achieved; product of export; quantity of export; FOB of export; DTA sales quantity; DTA sales value; year/quarter/month.

Claims of reimbursement of central sales tax

Analysis variables. Zone; claims received; claims disbursed; year/quarter/month.

Deemed export benefits

Analysis variables. Zone; sector; claims received; amount disbursed; year/quarter/month.

Employment generation

Analysis variables. Zone; male/female; managerial/skilled/unskilled; number of employees.

Investments in the zone

Analysis variables. Zone; unit; NRI; foreign investment; Indian investment; remittances received; approved value.

C2.7 EXECUTIVE INFORMATION SYSTEM

The analyses carried out by the executive information system application using the SAS for Windows Release 6.12 are discussed in the following sections.

C2.7.1 Data Analysis

In this case, the data imported into the SAS EIS application and the multi-dimensional analysis are carried out. A MDDB is built with various zones, sectors and time periods in year as the three axes of a cube (see Fig. C2.4).

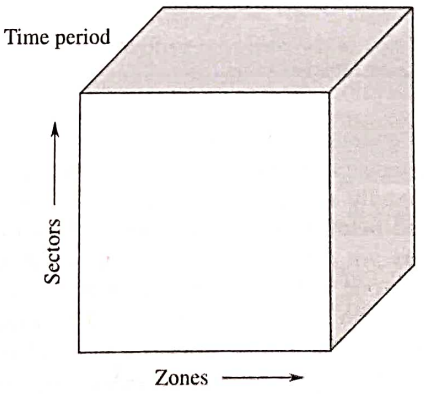


Fig. C2.4 Three-dimensional data cube having zone, sector and time.

The cube with added dimension of country can be analyzed as two different graphs, the association of one category variable with the other (see Fig. C2.5).

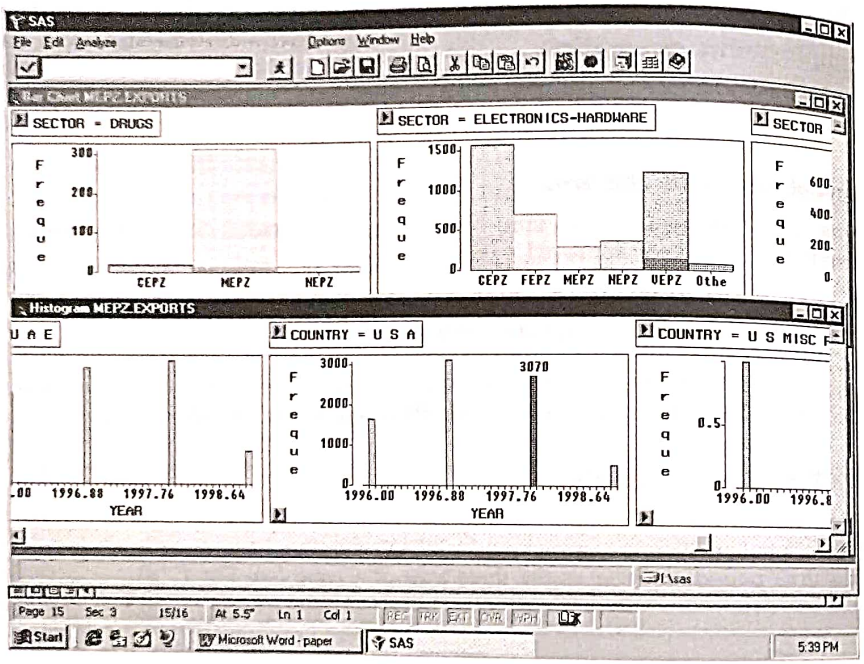


Fig. C2.5 Bar diagrams. Trend of the zone-wise performance (above). Trend of the country-wise performance in different years (below).

The two bar graphs generated are given here. The first one (above) indicates the trend of the zone-wise performance in each sector in terms of number of exports. We can horizontally move to cover the entire spectrum of sectors. The second bar graph (placed below) indicates the trend of country-wise performance in different years. Here, we can also horizontally scroll to view the trend in different countries. Any zone not performing in a sector or the year which has no exports for the country can be obviously known. The direction of exports from India will be known by looking at the number of export country-wise.

Suppose, we want to know the impact of sector-wise, zone-wise exports for different countries, in different years. We would like to know the exports both sector-wise and zone-wise (in Association). First scroll to the country desired and

then click on the bar of the year required, the upper graph changes automatically. Here for the country USA, the year 1996 is selected. The upper graph indicates the sector-wise contribution in exports, whereas the dark colour is zone-wise. The dark-shaded area is the contribution of the selected country and the year of exports. In Fig. C2.6, the year 1996 is the selected and the upper part of the screen shows the Cochin export processing zone's (CEPZ) in exports to the USA in 1996. The whole bar indicates the exports by CEPZ to all the countries all the years.

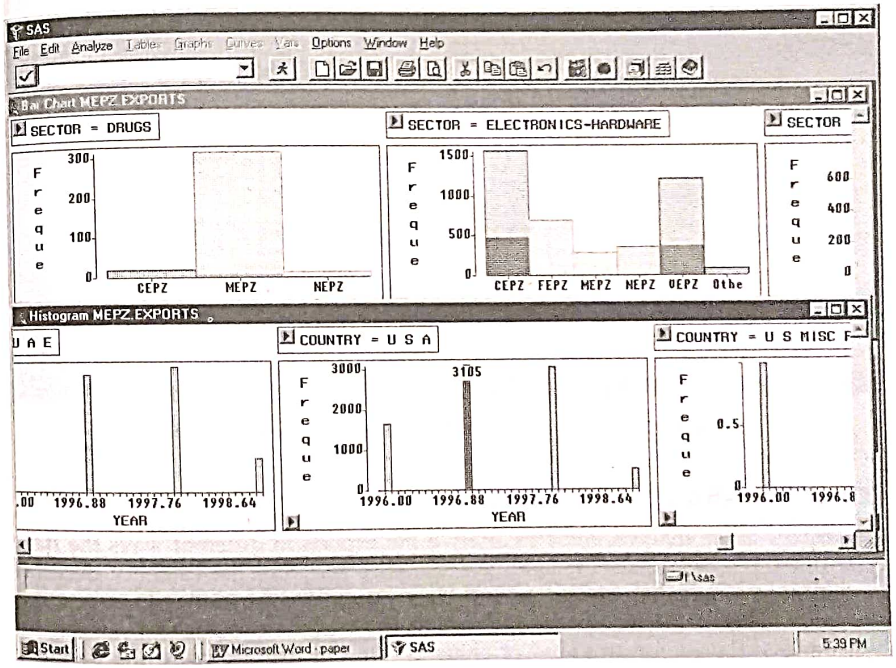


Fig. C2.6 Analysis of exports: sector-wise and country-wise.

Suppose, we want to analyse the exports by CEPZ other-way by selecting a zone (any zone can be selected just by clicking on the bar indicated against that zone in the upper graph) and like to know the country-wise exports in each year for that particular zone against the total contribution. If we want to know what export CEPZ has made for each country and in every year, just click on the CEPZ bar and the graph in the bottom of the screen instantly changes showing the contribution of CEPZ for each country for every year.

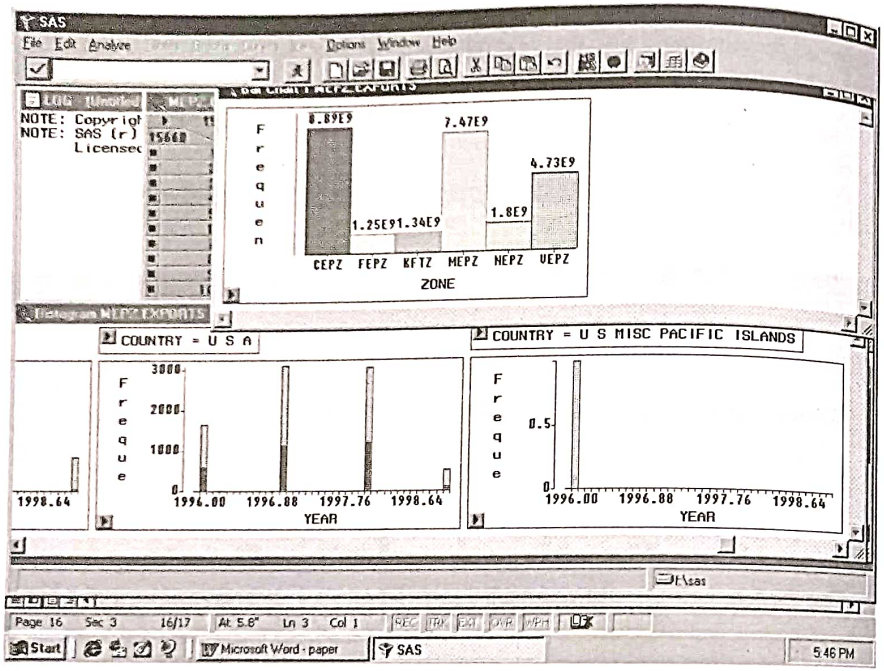


Fig. C2.7 Analysis of exports: zone-wise and country-wise.

C2.7.2 Analysis by Drilling Down

The *drilling-down* analysis helps us analyse the exports in different ways the data cube does. We can rotate the cube, we can slice and dice the category decision variables; any kind of hierarchy and drill down can be defined as per the hierarchy. The hierarchy can be defined in the metadata repository or can be dynamically altered for any kind of analysis. We can drill down to the problem level and find out the cause why exports have gone down during a particular time scale and not in other periods, etc. The analysis is very flexible in nature and there are numerous ways we can do the analysis. The example below uses the hierarchy

Zone-Sector-Country-Year-Quarter-Unit

The zone-wise exports are better shown as 3-D pie graph. Although, this can be represented in various types of graphs and tables. Here, CEPZ is chosen for the drilling down (see Fig. C2.8).

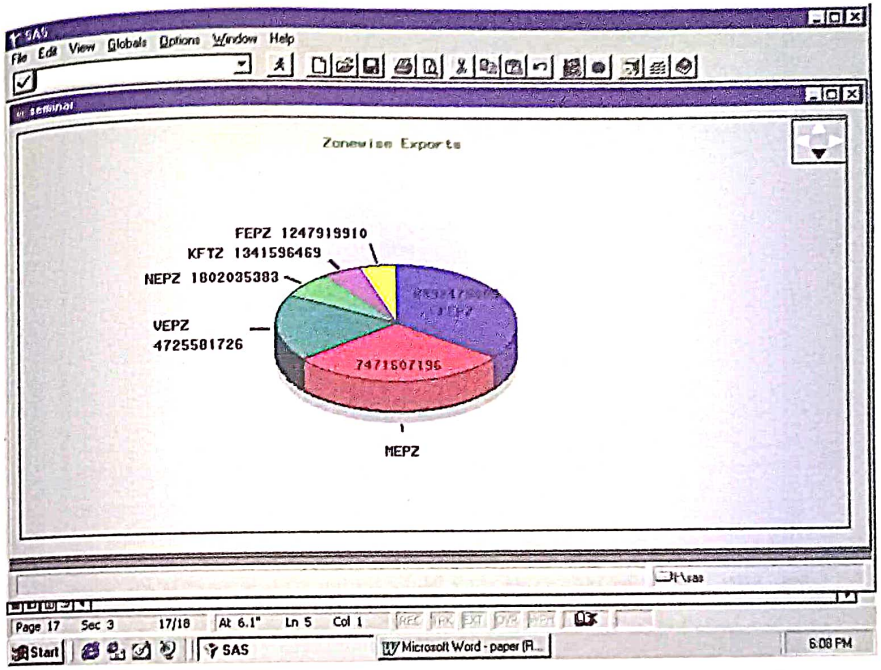


Fig. C2.8 A pie graph for zone-wise exports.

It has been drilled down, for example, for the electronics sector. In this sector, country is chosen as the next category variable for drilling down. It should be noted that at any stage, the order can be changed and data set can be selected for a criteria. The country USA has been chosen as the country for drilling down. (see Fig. C2.9)

Further, the country USA has been chosen as the country for unit-wise and year-wise drilling down (see Figs. C2.10 and C2.11).

This 'drilling down' provides greater flexibility to the planner to get any details with whatever analysis variable he/she would like to monitor and review the performance each time. Similarly, the impact of the new EXIM policy on the exports and imports can be analysed. This can be achieved by monitoring the performance on key decision parameters in various time scales before and after the announcement of the EXIM policy.

Figure C2.12 gives the tabular way of drilling down and locating the desired details to see all the related data at the bottom line.

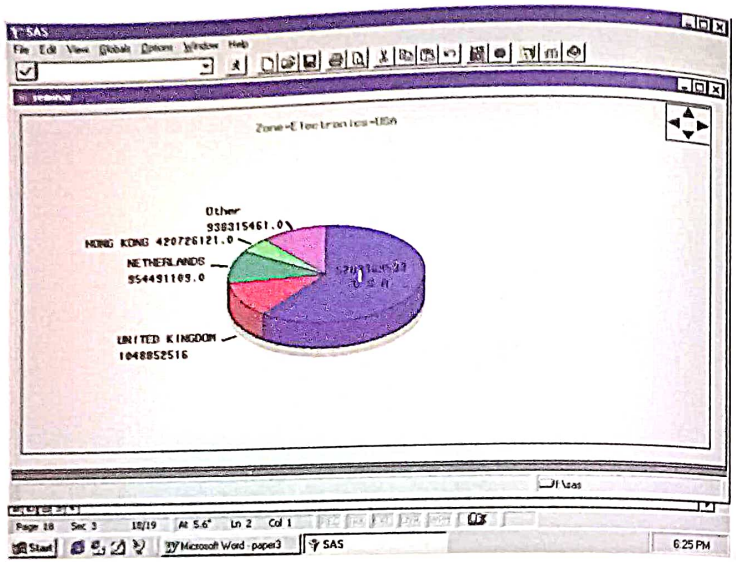


Fig. C2.9 Country-wise exports from MEPZ to the USA in electronics sector.

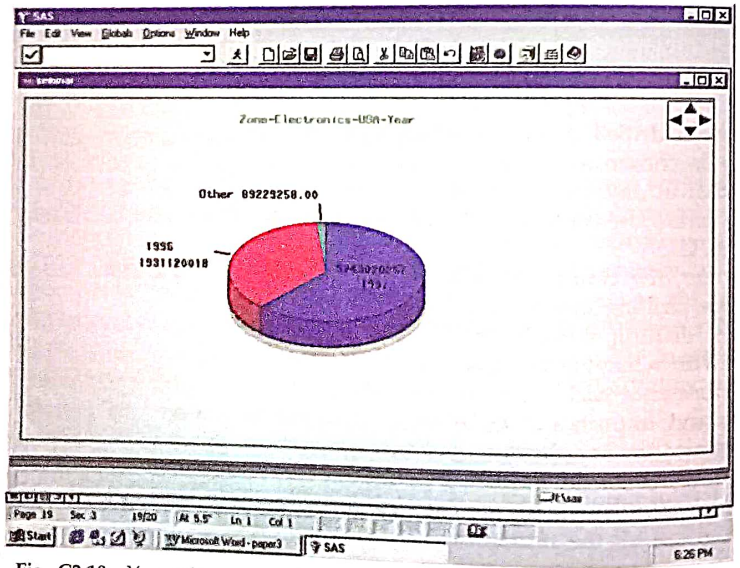


Fig. C2.10 Year-wise exports from MEPZ to the USA in electronics sector.

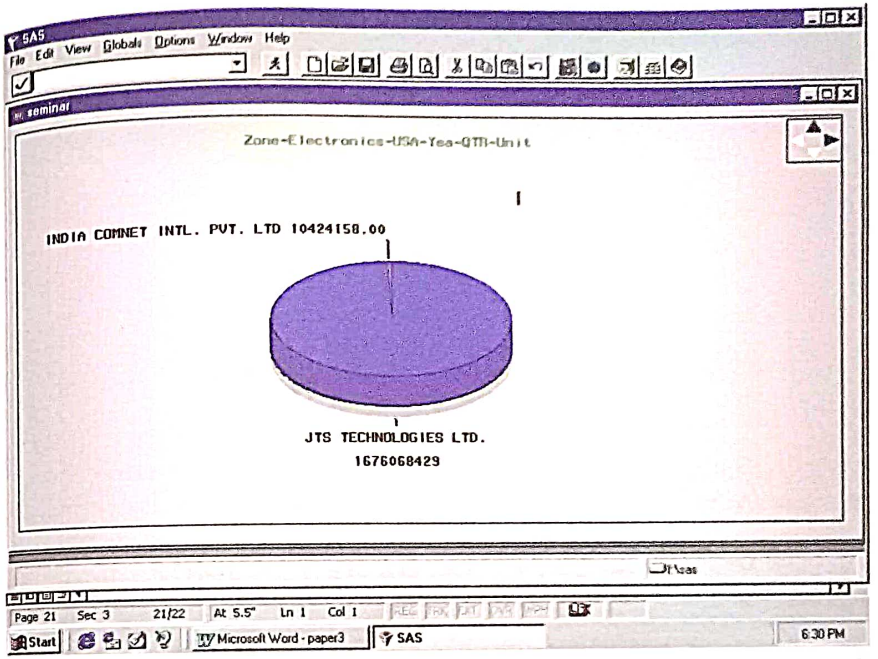


Fig. C2.11 Drill-down to units (companies) contribution.

The two shipping bills marked indicate that there were some exports from the company, Ambassador Garments. All the details of these exports can be studied and obtained just by requesting for the detailed view of data as in Fig. C2.13.

The current cube can be rotated so that the unit-wise exports can be seen from the left and further if we collapse the columns we can have more data for analysis on one screen. Refer to Fig. C2.14.

The attributes like sorting on the category variables, selecting particular range of data for different use, colour distinction for columns, title for the analysis, etc. can be changed dynamically at run time. The various options of attributes are shown in Fig. C2.15.

Changing analysis on the fly

The hierarchy, the category variables across and the statistics, range or any attributes can be changed on the fly during analysis. The options are shown in Fig. C2.16.

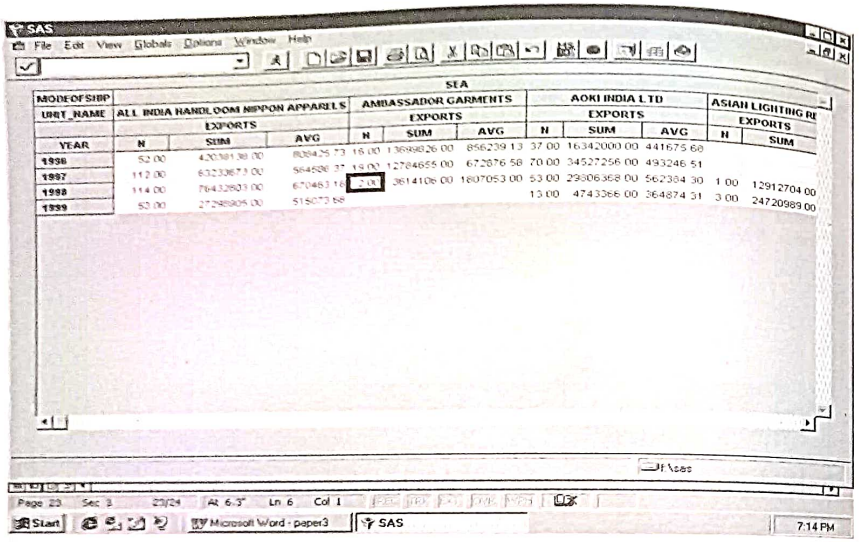


Fig. C2.12 Dimension analysis: year, mode of exports, unit, no. of exports, total exports and average exports.

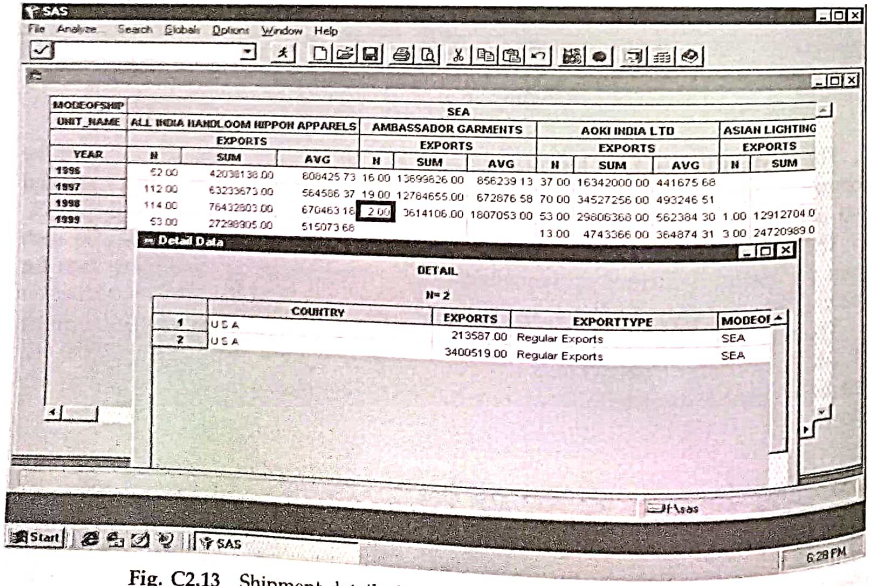


Fig. C2.13 Shipment details for the selected year, mode and unit.

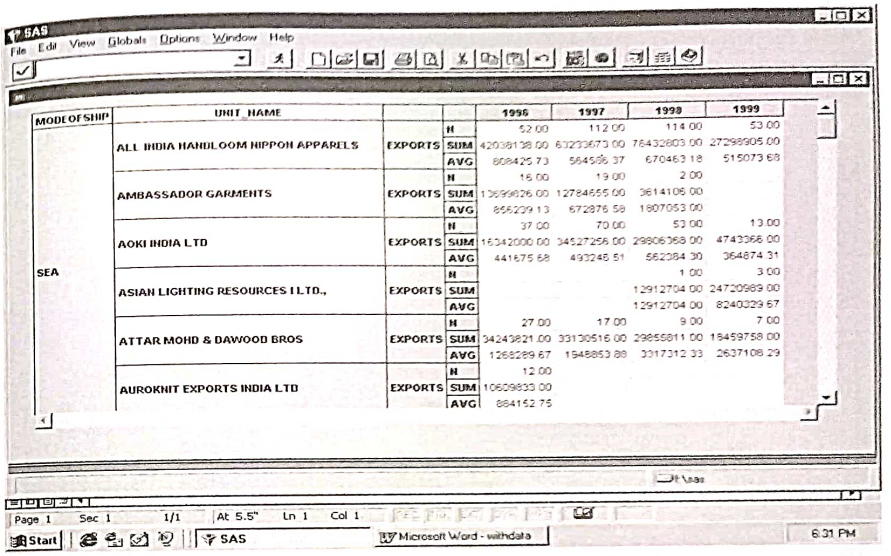


Fig. C2.14 Rotated cube with x and y axes changing.

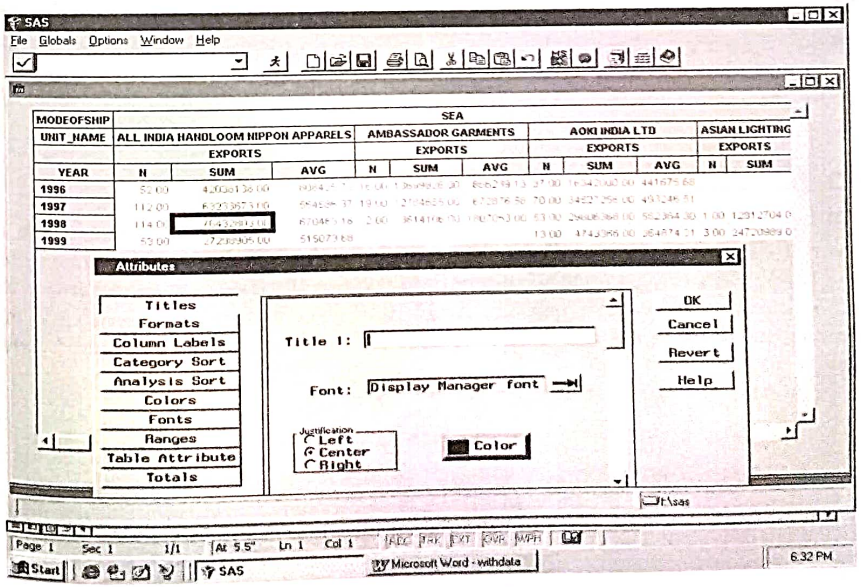


Fig. C2.15 Attribute definition.

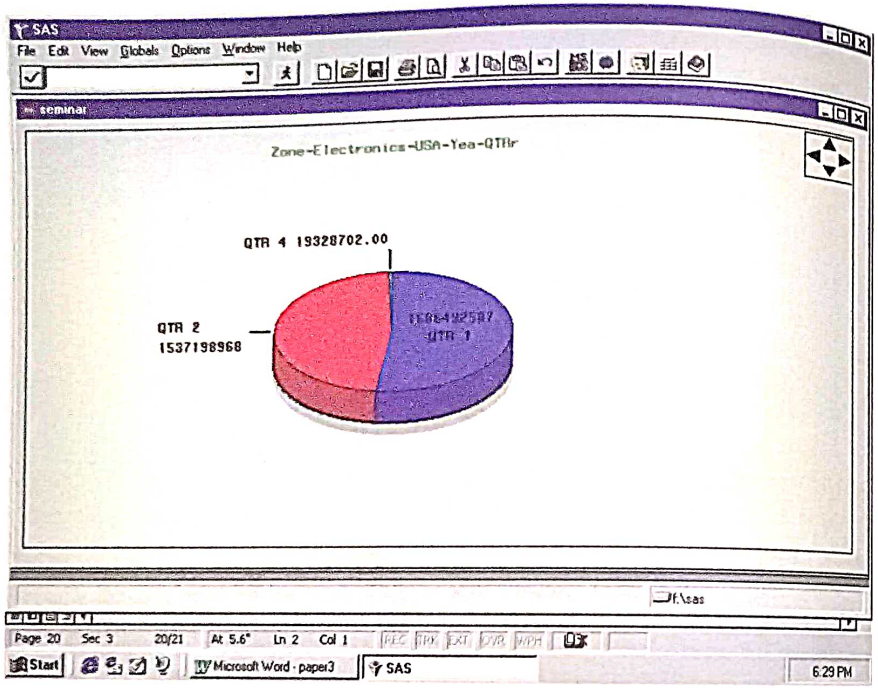


Fig. C2.16 A pie graph of quarter-wise exports in electronics sector from MEPZ to USA.

C2.7.3 The Multi-variance Analysis

All the analysis techniques offered by the tools cannot be listed here; some of them are listed in Fig. C2.17 for the purpose of the warehouse implementation.

The multi-variance analysis filters data for specific data requirement and the subset can be used for analysis. The details of the actual data can be seen and this quickly narrows down our search to fish out the data for analysis and detail study.

Consider the data set below with the hierarchy

Zone-Sector-Country-Year-Quarter-Exports

Also, refer to Fig. C2.18.

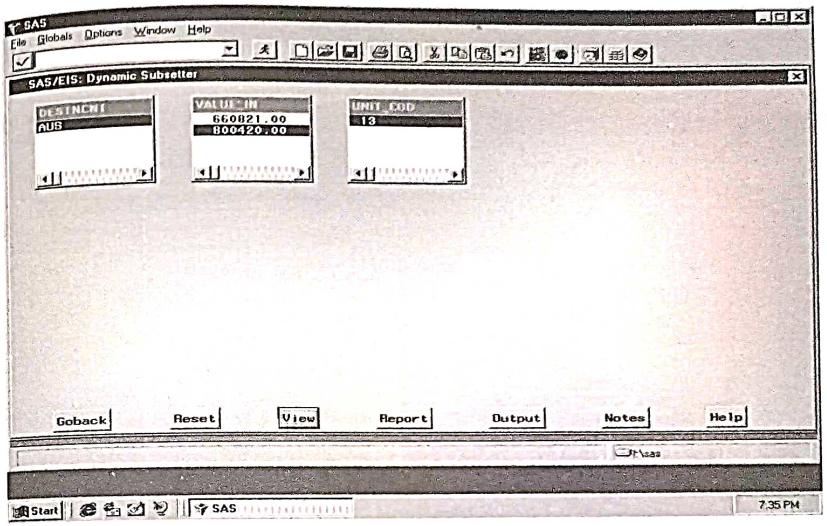


Fig. C2.17 Multi-variance analysis.

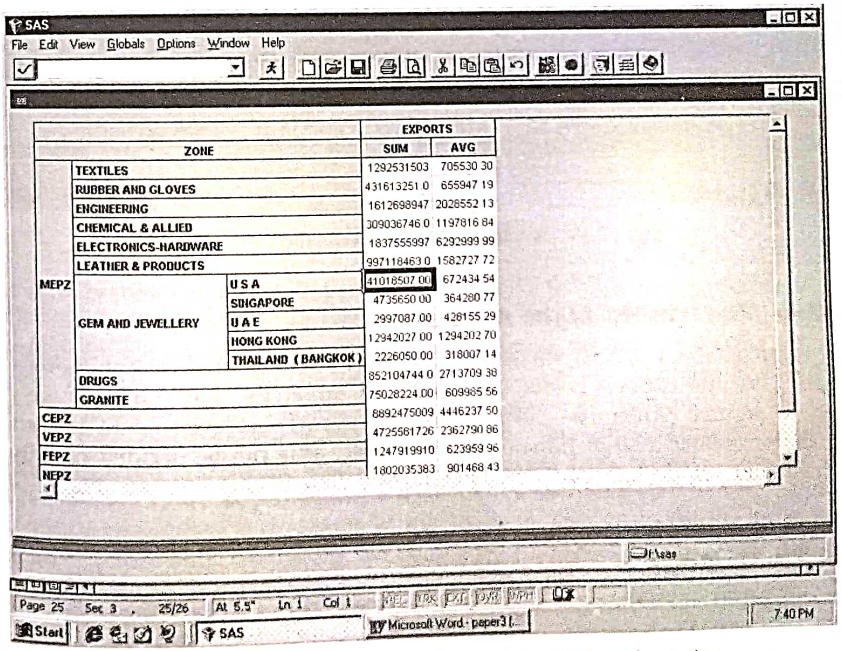


Fig. C2.18 Drill-down feature for zone, product and country.

The quarter-wise actual data with all other details can be seen at any point of time as in Fig. C2.19.

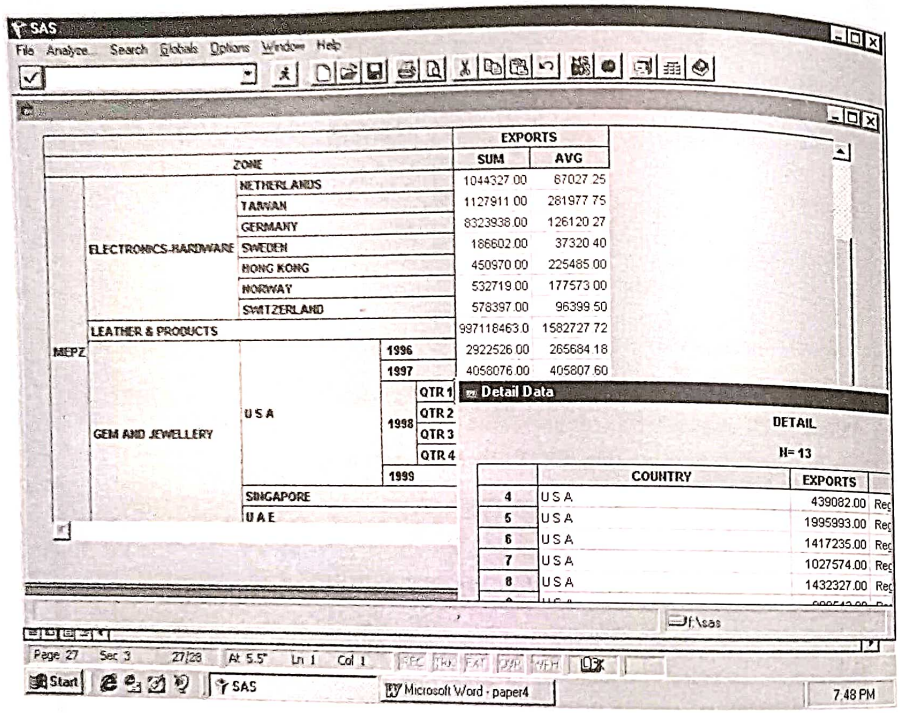


Fig. C2.19 Shipments details for detailed analysis.

C2.7.4 The Traffic Light Analysis

This is an analysis where we define different colours for different themes or ranges. Like the exports below the stipulated target for that company we can mark the figures (Australia—5070.00) and the exports achieved up to 15% more than the stipulated one in (Japan—20300.00), etc. This can be decided on the fly and changed for different subsets of data or different data-sets (see Fig. C2.20).

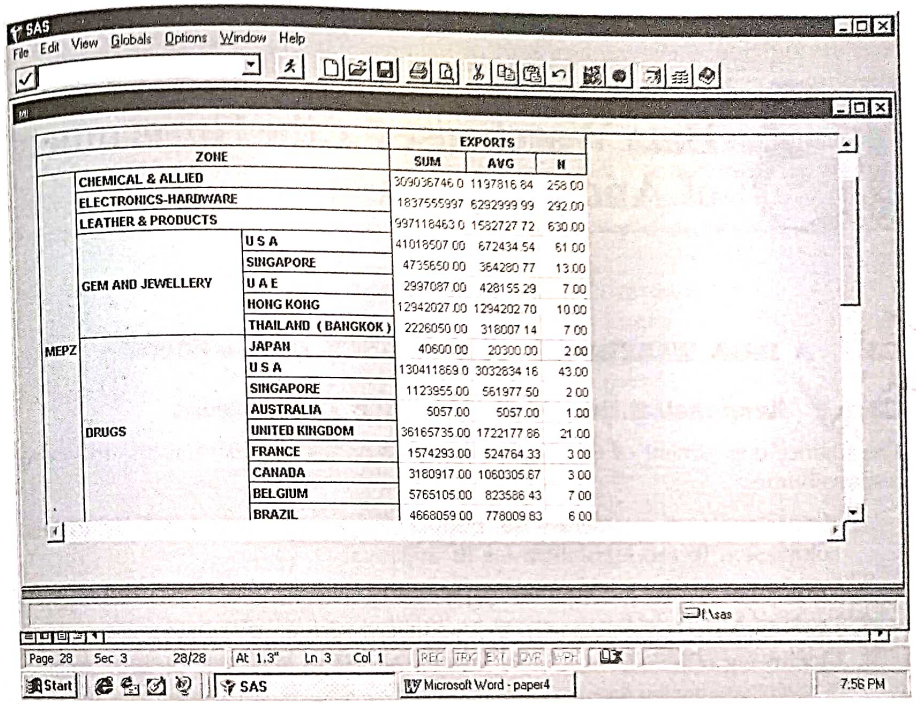


Fig. C2.20 Traffic signal analysis.

CONCLUSION

The data warehouse for EPZs provides the ability to scale large volume of data with seamless presentation of historical, projected and derived data. It helps the planners in what-if analysis and planning without depending on the zones to supply the data. The time lag between the zone and the Ministry is saved and then the analysis can be carried out with the speed of thought analysis.

The data warehouse for the Ministry of Commerce can add more dimensions to the proposed warehouse with the data collected from other offices/agencies (i.e. customs, DGFT, etc.) to evolve a data warehouse model for better analysis of promotion of imports and exports in the country. This will provide an excellent executive information system to the Secretary, Joint Secretaries of the Ministry.

CASE STUDY

3 Data Warehouse for the Government of Andhra Pradesh

C3.1 A DATA WAREHOUSE FOR FINANCE DEPARTMENT

C3.1.1 Responsibilities of the Finance Department

The finance department of the Government of Andhra Pradesh has the following responsibilities:

1. Preparing a department-wise budget up to the sub-detail head and submission to the legislature for its approval.
2. Watching out the government expenditure and revenue department-wise.
3. Looking after development activities under various plan schemes.
4. Monitoring other administrative matters related to all heads of departments.

C3.1.2 Treasuries in Andhra Pradesh

Money standing in the government account are kept either in treasuries or in banks. Money deposited in the banks shall be considered as general fund held in the books of the banks on behalf of the State. Treasuries have the following officers:

Director of Treasuries. Treasuries are under the General Control of the Director of Treasuries and Accounts.

District Treasury Officers (DTO). The immediate executive control of treasuries in a district vests in the District Treasury Officer who is subordinate to the Director of Treasuries and Accounts. The District Treasury Officer is responsible for the proper observance of the procedures prescribed by the rules. He also monitors the punctual submission of all returns required by the treasury by the State Government, the Accountant General and the Reserve Bank of India.

Sub-treasuries

If the requirements of the public business make necessary the establishment of one or more sub-treasuries under a district treasury, the arrangement for the same is made by the Finance Minister after consultation with the Accountant General.

The accounts of receipts and payments at a sub-treasury must be included monthly in the accounts of the district treasury.

Treasuries handle all the government receipts and payments, receipts are paid through 'challans'. (A challan is a form which is used for all types of receipts.) Payments against challans are based on the submission of different types of bills by the drawing officers. These bills may be: paybills, abstract contingent bills, advance GPF, travelling allowances, fully vouchered contingent bills, deposit repayments, pensions, grants-in-aid, refunds, scholarships and loans. Every transaction in the government is made through related departments.

Description of the account head	Length of the code (digit)
Major	4
Sub-major	2
Minor	3
Group sub-head	1
Sub-head	2
Detail head	3
Sub-detail head	3

Altogether it is an 18-digit code.

Each department (or scheme or deposit account) is identified by major head. Other levels are related to respective sub-levels of the department. Major heads are broadly classified as five categories:

1. Less than 2000 : Receipts
2. 2000 < 4000 : Service major heads
3. 4000 < 6000 : Capital outlay
4. 6000 < 8000 : Loans
5. More than 8000 : Deposits

Release of budget to all heads of departments of different government schemes are passed by the legislature, this is known as *voted transaction under plan.*

A project for building a data warehouses (DW) for online analytical processing (OLAP) is implemented by NIC for the department of treasuries. The concept of building DW in the department of treasuries has been established for providing easy access to integrated up-to-date data related to various aspects of the departments functions (revenue, expenditure, economic, financial, etc.) In the Treasury Department, DW technology is used to develop analytical tools designed to provide support for decision-making at all levels of the department.

The information accumulated in most of the treasuries cannot be efficiently used in the framework of traditional Information Systems. Traditional Information Systems implemented in the Department of Treasuries are based on transactional databases (OLTP) which are not designed for providing fast and efficient access to information critical for decision-making. One of the main limitations inherent in such Information Systems is that the managers responsible for decision-making are not provided with direct access to historical business data accumulated over a significant period of time.

Data required for analysis are typically distributed among a number of isolated Information Systems meeting the needs of different sub-treasuries. In the case of complicated queries on a database, the system response time is excessively high. Data models used are not suitable for decision support systems since they are specifically designed to handle only short transactions.

Thus the information stored in traditional systems cannot be efficiently retrieved by managers and analysts involved in decision-making.

Data warehouse technology provided to the Department of Treasuries by National Informatics Centre eliminates these problems by storing current and historical data from disparate Information Systems. These information are required by business decision-makers in a single, consolidated system. The DW technology is based on utilizing multidimensional data modelling which represents a conceptual model of treasury business processes as a collection of facts, each characterizing just one of the features of such processes. This approach makes data readily accessible to the people who need it without interrupting online operational workloads.

Data warehouses provide efficient analysis and monitoring of financial data of treasuries, its structural subdivisions and subsidiaries. It also evaluates the internal and external business factors related to operational, economic and financial conditions of treasuries budget utilization.

The most important problems which can be addressed using DW and OLAP technologies within the Department of Treasuries include the following:

- Analysis of expenditure and revenue of heads of accounts by departments, drawing officers by regions, districts and sub-treasuries
- Operational and financial analyses of the Department of Treasuries and
- Evaluation and monitoring of the financial positions.

National Informatics Centre (NIC) assisted the Department of Treasuries with services in building data warehouses and systems for online analytical processing. NIC officers work closely with the clients, thereby transferring state-of-the-art technologies for developing, updating and maintaining the DW. Some software tools for building DW and OLAP applications are SQL Server 7.0 with OLAP services and ASP 3.0.

C3.1.3 Dimensions Covered Under Finance Data Warehouse

Following are the different *dimensions* taken for drill-down approach against two measures—payments and receipts:

- Department
- District Treasury Office
- Sub-Treasury Office
- Drawing and Disbursing Officer
- Time (year/month/week/day)
- Bank-wise
- Based on different forms (bills)

Refer to Figs. C3.1–C3.14.

C3.1.4 COGNOS Graphic User Interface For Treasuries Data Warehouse

The features of COGNOS PowerPlay version 6.0 used in this category are discussed as follows:

Impromptu. *Impromptu* is used for generating various kind of reports like simple, crosstab, etc. Once an Impromptu report has been published on HTML, one can view the report using the Web browser, Here a Web browser is required but not Impromptu to view HTML reports.

One can view an HTML report:

- on Internet or Intranet Web page;
- on a network; and
- that has been sent via e-mail.

Transformer. *Transformer* model objects may contain definitions of queries, dimensions, measures, dimension views, user classes and related authentication information, as well as objects for one or more cubes that Transformer creates for viewing and reporting in PowerPlay. Transformer stores models as files with the extensions.

Once the model is ready, creating one or more cubes or cube groups based on the model contents is possible. By creating several cube objects and basing different cubes on those objects, one can create subset cubes that serve the needs of distinct user groups. One can also create cubes that provide drill-through access to other cubes.

PowerPlay. COGNOS PowerPlay is used to populate reports with drill-down facility. Popular reports are as follows (also refer to Figs. C3.8–C3.14):

- Dynamic rank position
- Financial report
- Business trend
- Comparative performance
- Foreign currency report

Scheduler. *Scheduler* coordinates the execution of automated processes, called tasks, on a set date and time, or at recurring intervals. Scheduler supports tasks that run once and tasks that run repeatedly. Through Scheduler, Impromptu users can submit Impromptu report requests to be executed either locally, or by an Impromptu Request Server.

Authenticator. *Authenticator* is a user class management system. It provides COGNOS client applications with the ability to create and show data based on user-authenticated access. It also serves as a repository for log-in information, thus providing client applications with auto-access to data sources and servers.

Following are a few sample reports generated by using the above methods (Figs. C3.1–C3.14):

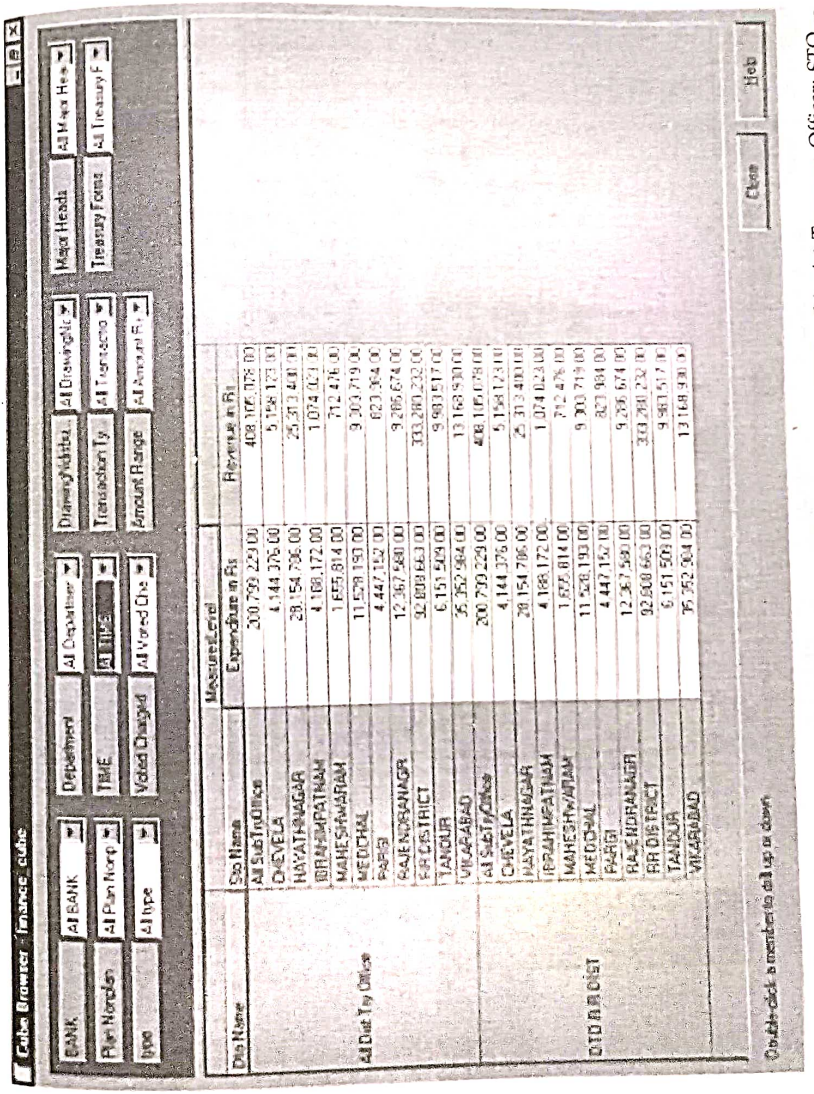


Fig. C3.1 DTO and STO wise breakup of Income and Expenditure. (DTO—District Treasury Officer; STO—Sub Treasury Officer.)

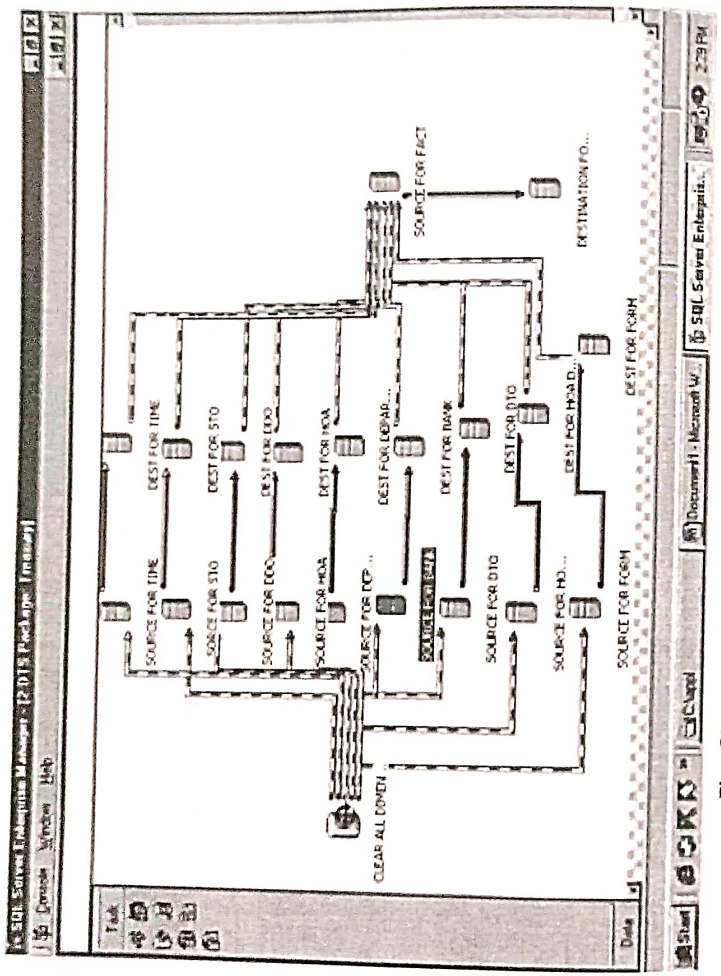


Fig. C3.2 Data transformations services of Finance data cube.

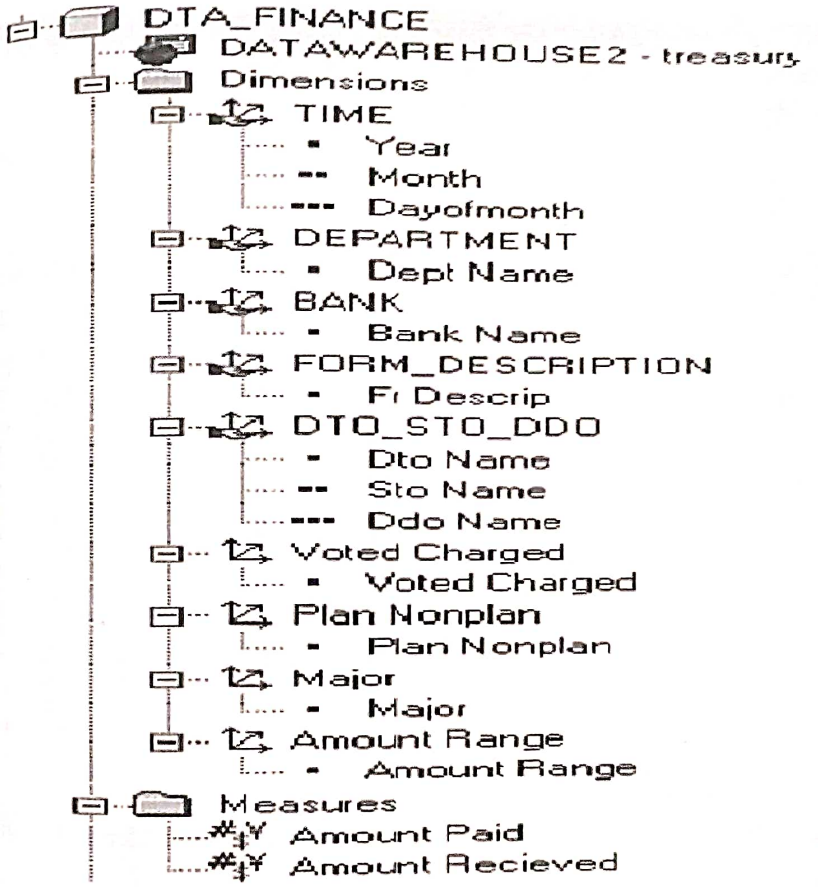


Fig. C3.3 Public, Private dimensions and measures of Finance cube.

dta_finance

Amount Range	BANK	DEPARTMENT	FORM_DESCRIPTION	Major	Plan Nonplan	Voted Charged
All Amount Range	All BANK	All DEPARTMENT	All FORM_DESCRIPTION	All Major	Plan	Voted

			Year	Month Dayofmonth		
			2000		Grand Total	
Dto Name	Sto Name	Ddo Name	Amount Recieved	Amount Paid	Amount Recieved	Amoun
DTO R.R	CHEVELA		0	292193	0	2
	HAYATHNAGAR		0	325729	0	3
	IBRAHIMPATNAM	PROJECT OFICER NFE IBRPATAN	0	200053	0	2
		CDPO ICDS IPATAN	11335	174176	11335	1
		A.D.A.SC.RR DIST	0	151000	0	1
		M.O.PHC MANCHAL	0	78862	0	
		M.O.PHC DANDU MAILARAM	0	70540	0	
		ASST.SM.RFWPC YACHARAM	0	23384	0	
		EE PR(RWS)DIV.RR DIST	0	9121	0	
		GAZETTED ADMN.OFFICER DEO RR	0	6664	0	
		C.A.S.PP UNIT IPATAN	0	4000	0	
		Total	11335	717800	11335	7
	MAHESHWARAM		0	201060	0	2
	MEDCHAL		0	479306	0	4
	PARGI		0	378698	0	3
	RAJENDRANAGR		0	143634	0	1

Fig. C3.4 Drill-down information of DTO/STO/DDO Revenue and Expenditure details. (DTO—District Treasury Officer; STO—Sub Treasury Officer; DDO—Drawing and Disbursement Officer.)

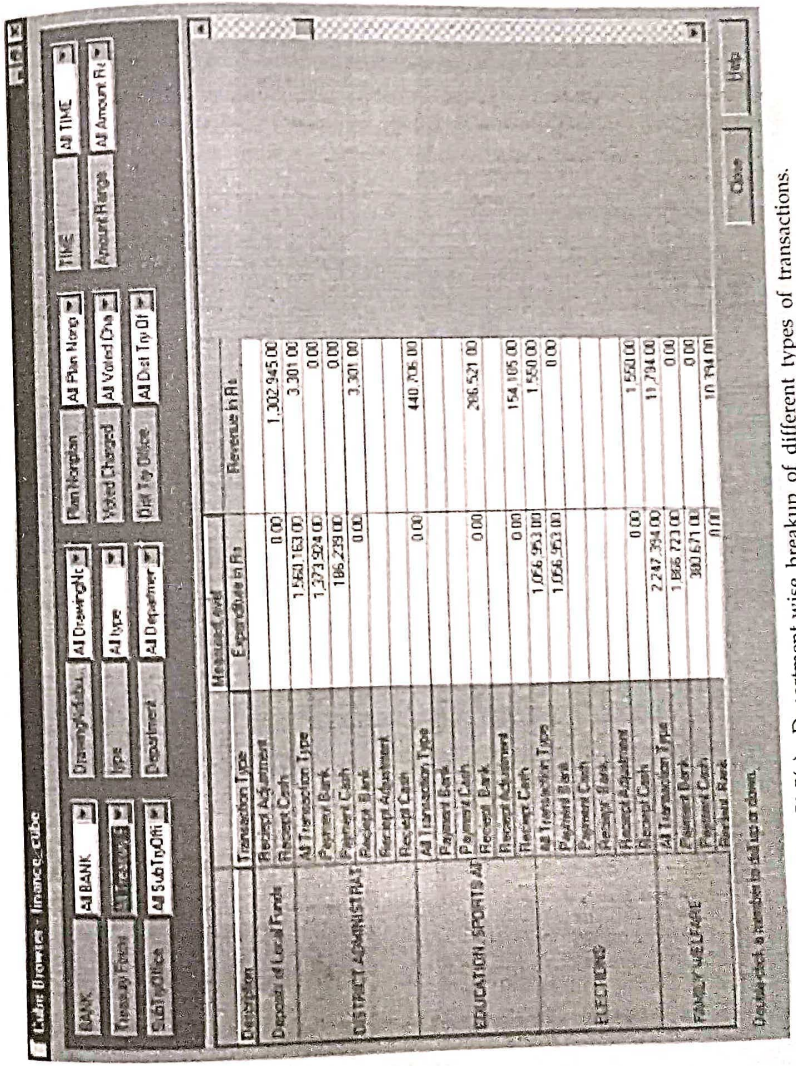


Fig. C3.5(a) Department-wise breakup of different types of transactions.

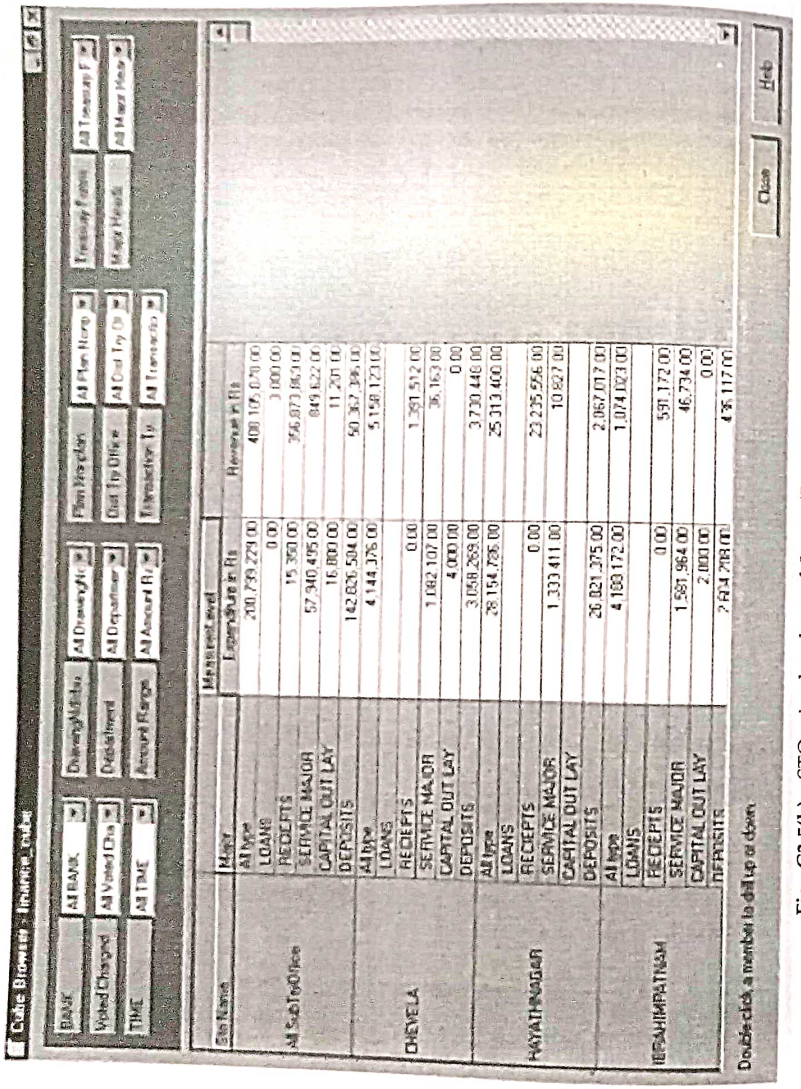


Fig. C3.5(b) STO-wise breakup of Loans/Receipts/Deposits/Capital outlay.

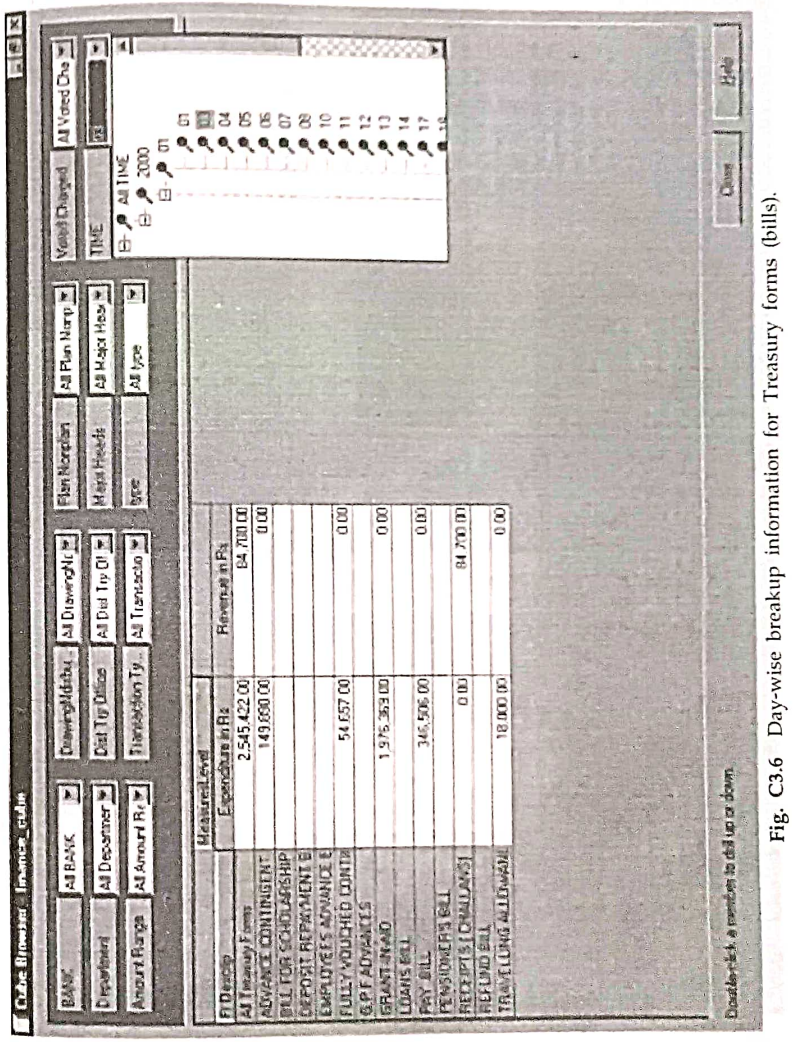


Fig. C3.6 Day-wise breakup information for Treasury forms (bills).

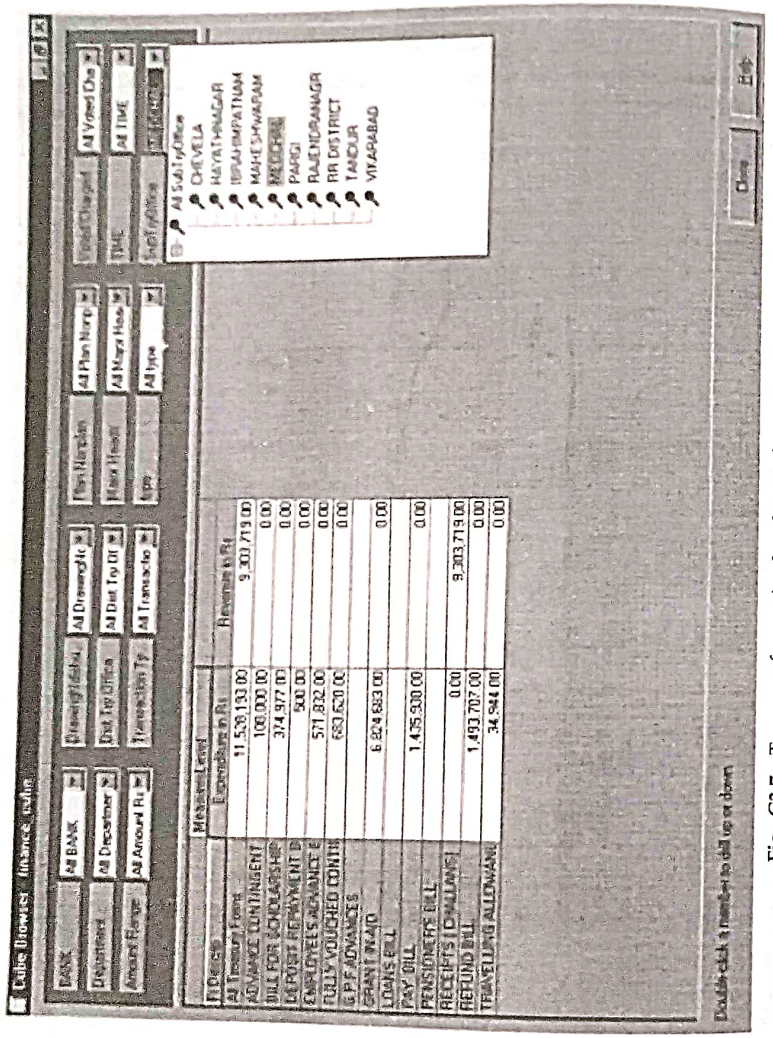


Fig. C3.7 Treasury form-wise breakup information for particular STO.

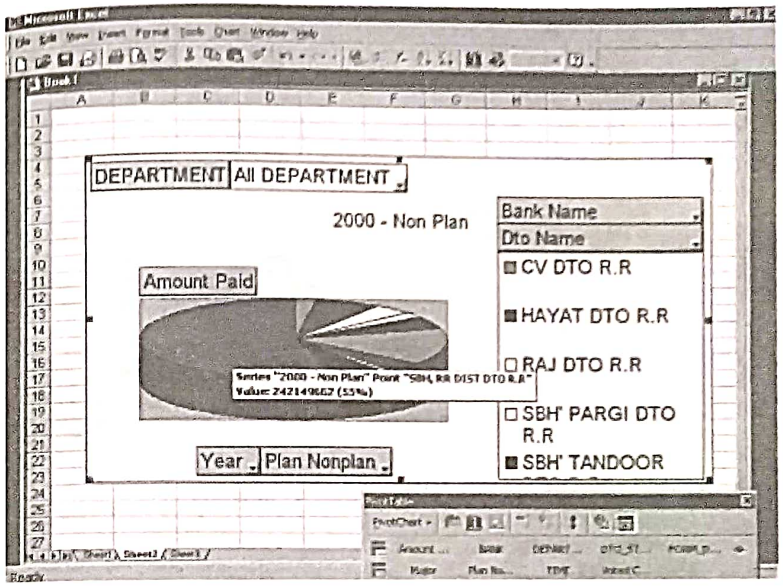


Fig. C3.8 Pivot table (graphical representation) for all departments, plan and non-plan-wise.

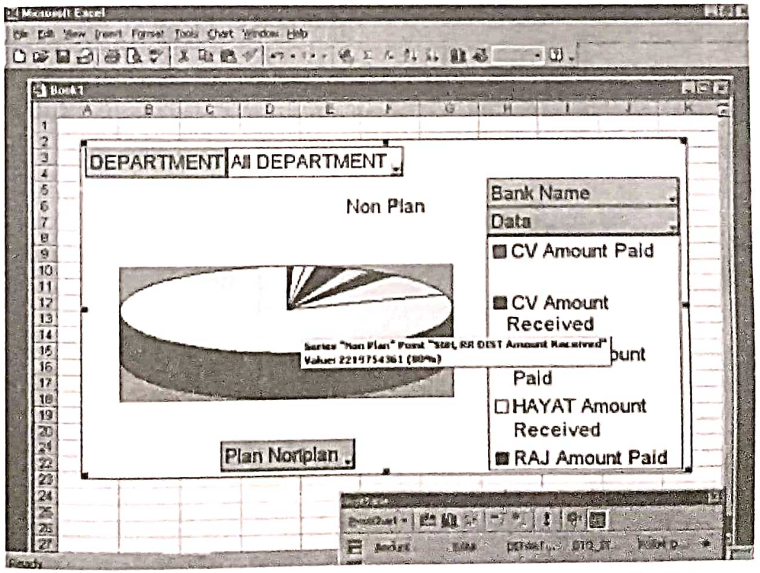


Fig. C3.9 Graphical representation of all departments, bank-wise.

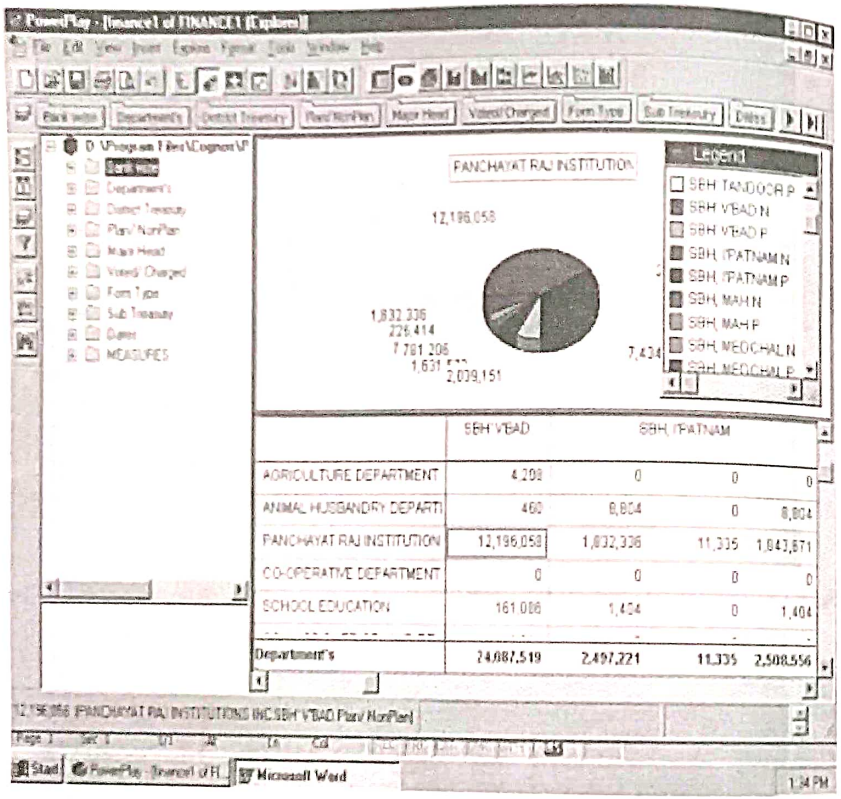


Fig. C3.10 COGNOS PowerPlay report on department/bank-wise expenditure.

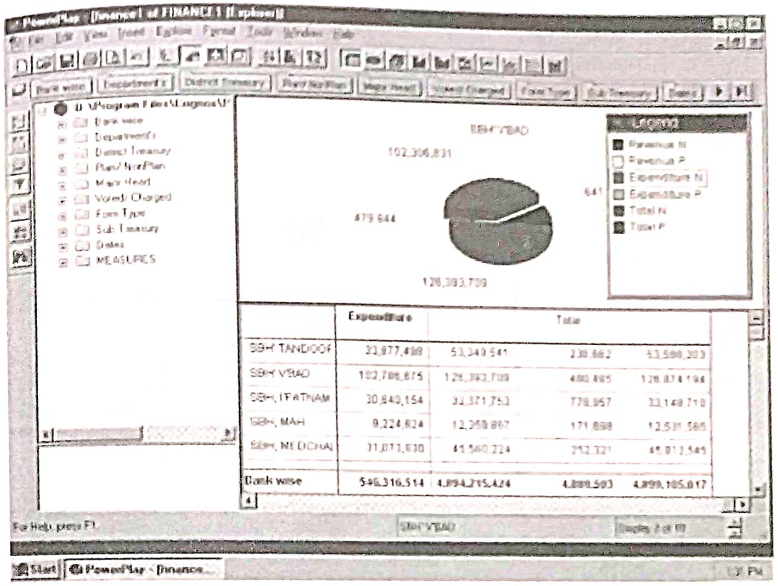


Fig. C3.11 Bank-wise expenditure statement.

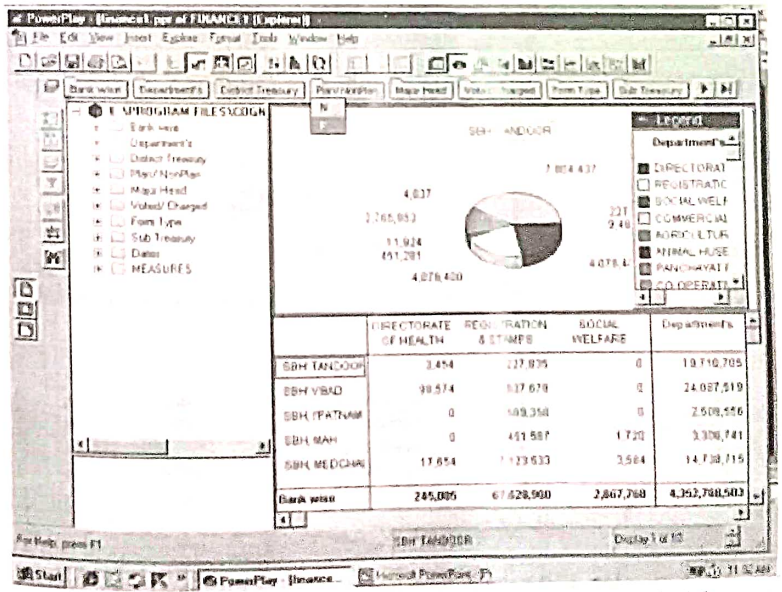


Fig. C3.12 Bank-wise/department-wise revenue under plan schedule.

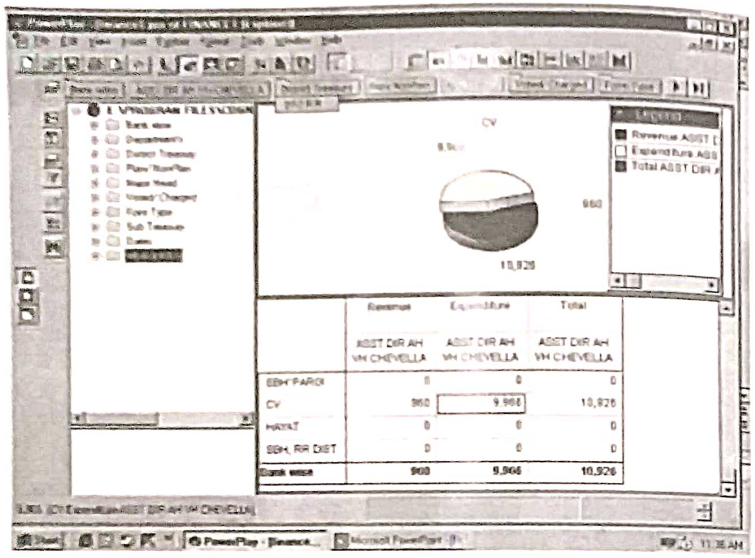


Fig. C3.13 Multi-dimensional analysis using COGNOS tools.

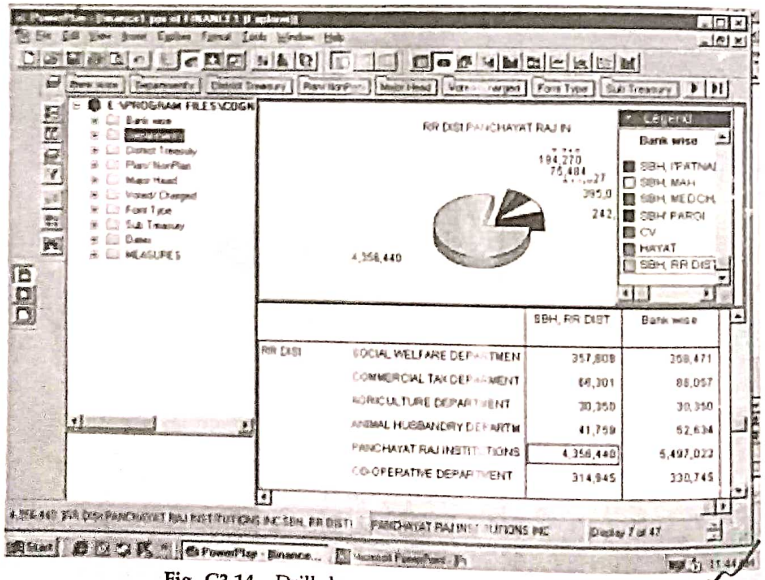
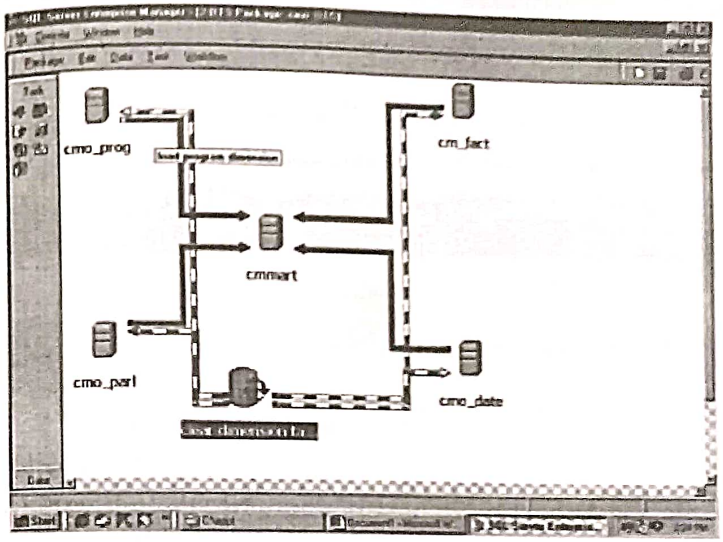


Fig. C3.14 Drill-down report, using COGNOS.

C3.2 A DATA WAREHOUSE FOR JANMABHOOMI WORKS MONITORING



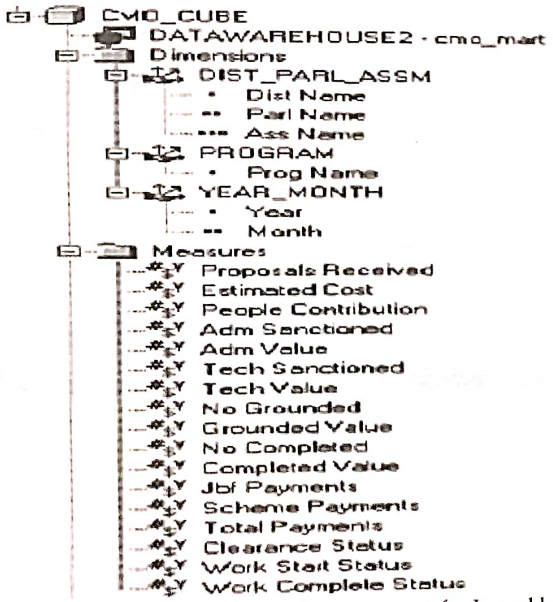


Fig. C3.15 Data transformation map for Janmabhoomi data.

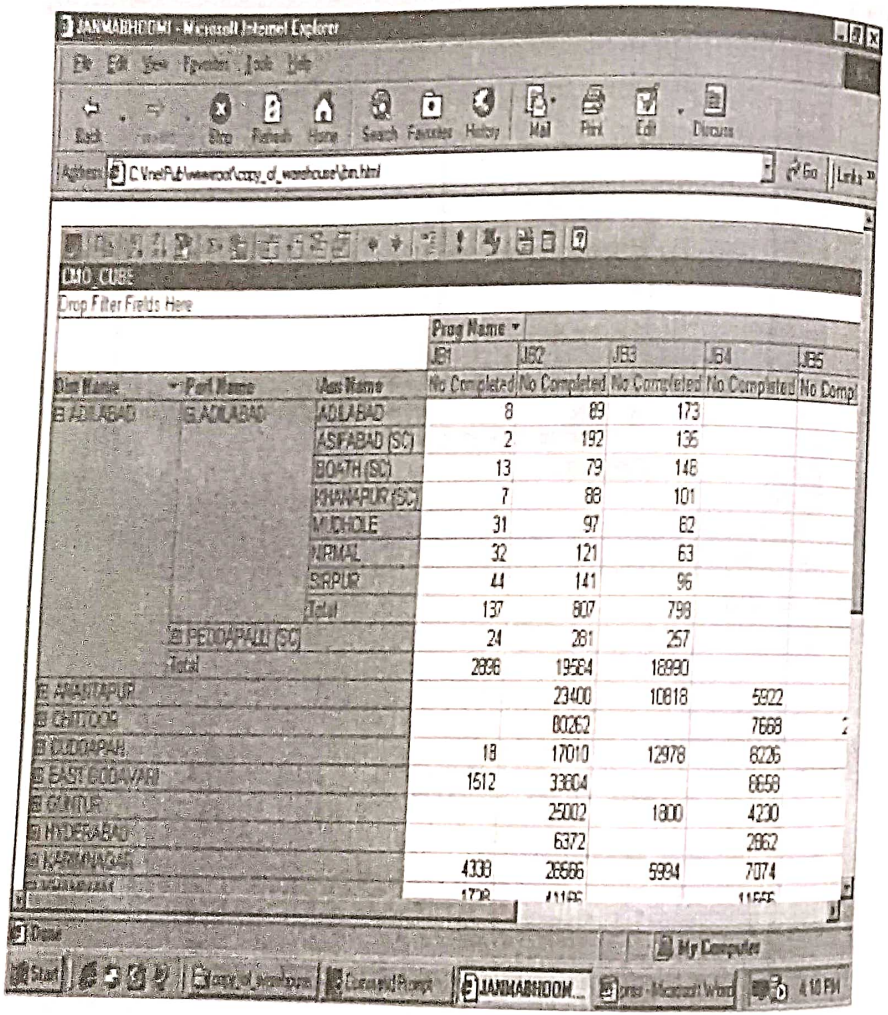
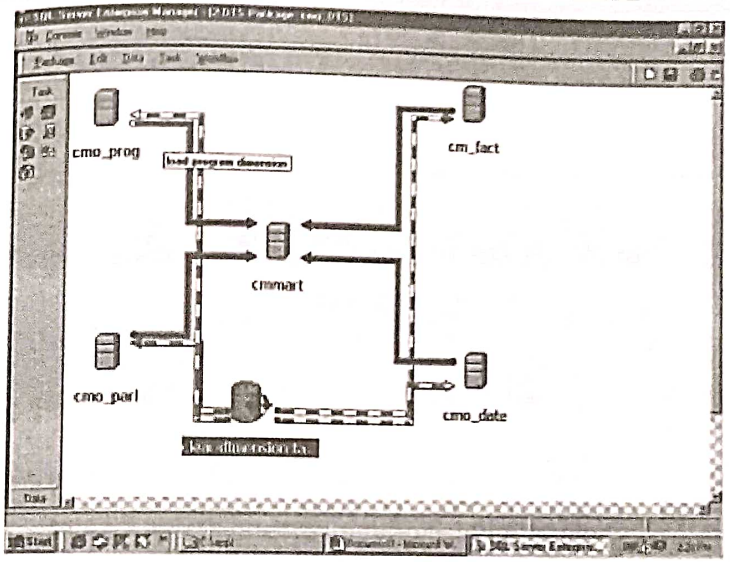


Fig. C3.16 Sample data drill-down picture for Janmabhoomi data.

C3.3 A DATA WAREHOUSE FOR CROPS MONITORING



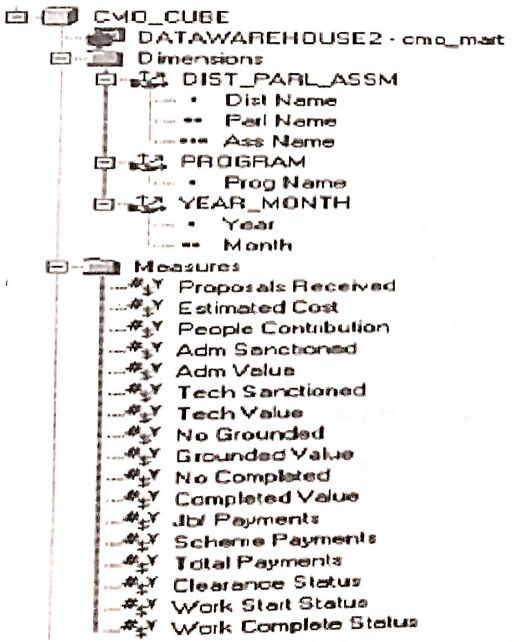


Fig. C3.17 Data transformation map for crop data.

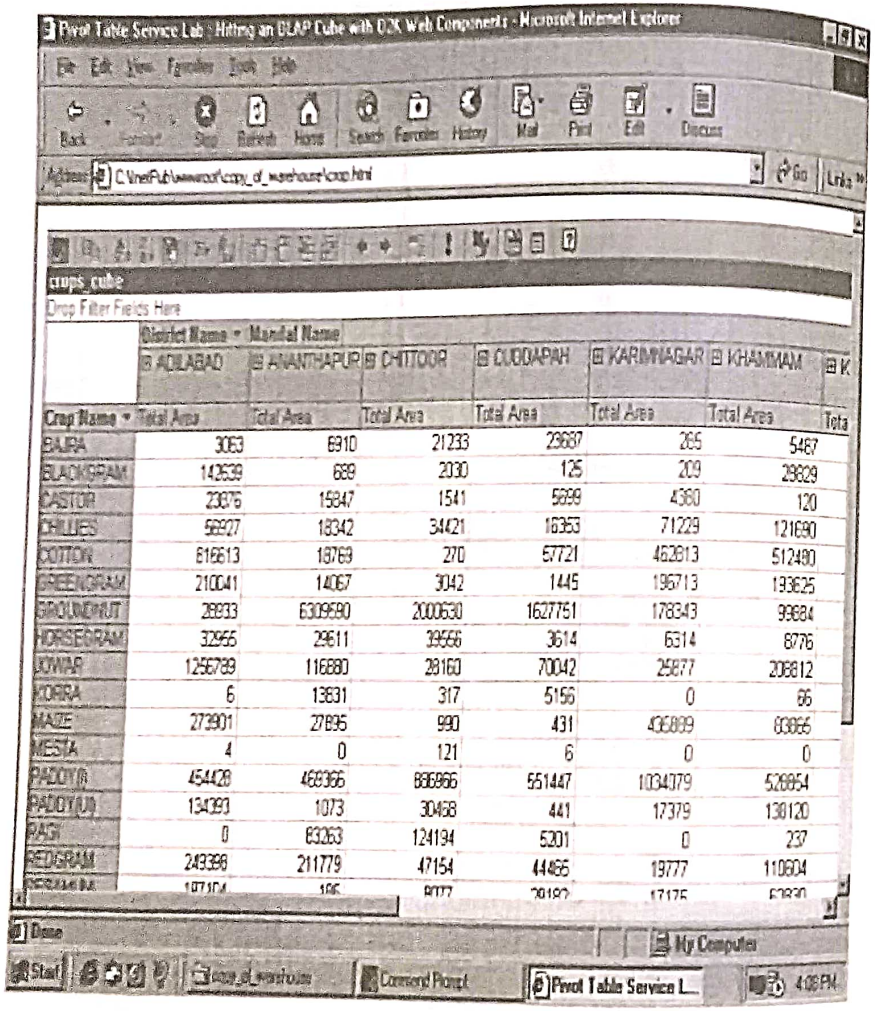


Fig. C3.18 Sample data drill-down picture for crop data.

CASE STUDY

4 Data Warehousing in Hewlett-Packard*

With an OLAP system based on Microsoft® SQL Server™ 7.0 and Knosys ProClarity™, HP can easily access and quickly analyse enormous volumes of sell-through data to help its reseller customers improve the efficiency and profitability of their businesses.

C4.1 INTRODUCTION

With an installed base of more than 55 million printers, Hewlett-Packard is a worldwide market leader in the $18 billion inkjet industry. The company's Consumer Products Group develops and manufactures a full range of imaging products using HP-pioneered thermal-inkjet technology. Responsible for creating more new product categories for HP over the last 10 years than any other part of the company, the Consumer Products Group introduced the world's first desktop inkjet printer in 1984 and pioneered large-format inkjet printers in 1993, all-in-one (printer/fax/scanner/copier) devices in 1994, PC photography in 1997 and colour copiers in 1997.

In addition to its Home Business, Office Business, and Inkjet Supplies Business units, the Consumer Products Group includes HP's Consumer Products Business Organization (CPBO), responsible for worldwide retail sales and distribution of all of HP's consumer-targetted products. While the market for these products continues to grow, it is not experiencing the explosive, double-digit growth rates that characterized the past decade. With technical barriers to entry falling, more and more competitors are bringing products to market, giving each competitor a smaller piece of a shrinking pie.

With approximately 80 per cent of HP's consumer products volume sold directly to top national accounts such as CompUSA, Best Buy, and Office Depot, these resellers are as important a 'customer' as the end-user who plugs in the

*This case study is provided by Microsoft Corporation and therefore the views expressed in the case study belong to Microsoft Corporation.

printer. In the past, HP's brand recognition and reputation for reliability were enough to ensure that a reseller would carry HP products. Today, while those factors are still important, in an increasingly competitive landscape, they are simply not enough.

"To be successful in our business, we have to come up with solutions for the reseller", explains Greg Stanley, Manager of HP CPBO's Business Analysis Group, which is responsible for aggregating and analysing the market data—including information on advertisement spending, market share, pricing, demographics, econometrics, and channel spending—that CPBO uses to run its business. "Resellers have specific needs and wants just as individual users do. To maximize their profitability, they're focussing on the expense and inventory investment side of the profit equation. We have to share information that will help them maximize their efficiency and profitability. If we don't, the competition will walk in and do it for us".

C4.2 ACCESS TO INFORMATION NEEDED USING DATA WAREHOUSING TECHNOLOGY

As a technology company, HP has done a superb job of capturing and storing the information its resellers need, both from primary research and from third parties. But the repositories of information that exist in huge pipelines or reservoirs do not communicate with one another. "When it comes to connecting all the data in the reservoirs to the people who need to use it, they can not get to it", Stanley comments. "It gets so difficult to extract information from the system and put it in a meaningful context that a lot of our users just are not willing to do it".

The Business Analysis Group decided they needed a system that would provide market metric data to help field sales force managers or account teams make brand and channel management decisions. And, at a higher level, the group wanted a unified analytical environment that would allow HP decision-makers to clearly see the complex trends, patterns, and relationships that impact their business—including such data as how advertising affects sales and who is capturing market share and why.

Due to its smaller size and limited IT expertise, the group needed help evaluating potential solutions. "We wanted a system that required low cost, low maintenance, and as simple to administer as possible", Stanley explains. So the group turned to Knosys Inc., a Boise, Idaho-based software company that has developed a business analysis/online analytical processing (OLAP) package called ProClarity™. Having built its software from the ground up for the Microsoft SQL Server 7.0 data mart/data warehouse environment, Knosys recommended that HP adopt the SQL Server 7.0 relational database and ProClarity solution.

At first, Stanley's group was opposed to creating yet another data store. "But Knosys showed us that this solution would enable us to move the data so quickly and at such a low cost of maintenance and ownership that it would solve our problems", he says. "We would be duplicating data, but the new system would bridge the gap between users and data. This was very compelling". But the group also wanted to evaluate other potential solutions, so they brought in a leading OLAP consulting company called Symmetry Inc. With over a decade of OLAP consulting experience, Symmetry offered the product and industry knowledge to effectively evaluate the Knosys and Microsoft approach.

With the endorsement of symmetry, the Business Analysis Group decided to move ahead with the SQL Server 7.0 and ProClarity solution, which they are currently rolling out. Knosys has helped the group build the data flow algorithms with the Microsoft Visual Basic® development system and SQL Server 7.0's Data Transformation Services. Some of the original data was in Microsoft Excel spreadsheets, other flat files, and OLAP exports from other vendors. HP needed some common ground to establish the data flow procedures. Visual Basic for Applications (VBA) and Microsoft Access 97 provided that common ground, plus made the data easier to maintain. From Access, the data goes into SQL Server 7.0.

The OLAP Services include a number of additional features that are attractive to HP. According to Clay Young, Vice President of Marketing for Knosys Inc., the Business Analysis Group was particularly impressed with SQL Server 7.0's hybrid OLAP (HOLAP) capabilities.

"Because of HP's enormous sell-through data volumes, it would take too long to build analytical models with a pure, multidimensional OLAP solution", explains Young. "And pure relational OLAP solutions don't meet the query performance requirements of HP decision makers. The hybrid architecture of SQL Server 7.0's OLAP Services will enable HP to deal with high data volumes and still deliver fast query response".

HP used SQL Server 7.0's virtual cubes and cube partitioning capabilities. Cubes are databases with multiple dimensions: HP has at least eight OLAP cubes, each of which will support a particular group of decision-makers. Virtual cube capabilities also allow decision-makers to cross-analyse data from all these OLAP sources simultaneously. Cube partitioning will allow HP to more effectively manage a large number of OLAP cubes and to easily manage differing levels of aggregation and views of time. Additionally, the virtual cube capabilities allow HP to easily derive new business views from the existing cubes making it considerably easier for the Business Analysis Group to manage its business views. "Basically", Young remarks, "it will give them information about market share, econometrics, demographics, pricing, sell through, and channel activity—all in one view if they desire".

"Knosys ProClarity provides HP decision-makers with the key to analyzing masses of data", explains Young. "It gives them previously unavailable data

visualization techniques as well as easy to use, full Web-enabled analysis capabilities". ProClarity is fully integrated with Microsoft products, and its PC-based client is modelled after Internet Explorer 4.0. "This is because analysing business data is a lot like trying to find information on the Web", he adds.

ProClarity's powerful analytical features, which take full advantage of the robust capabilities found in SQL Server 7.0's OLAP Services, help knowledge workers understand complex data. The software's state-of-the-art data visualization tools enable workers to quickly see patterns, trends, and exception—even in complex environments such as HP's, where they must analyse hundreds of products across thousands of reseller partners.

Given HP's past experience with proprietary, monolithic solutions, Stanley's group wanted an open, highly flexible, analytical application that could be deployed in a variety of ways, including as a PC client, a Web-based client for HP's intranet, and a custom Executive Information System. ProClarity's ActiveX/COM architecture and OLE DB for OLAP connectivity fulfil this need.

CONCLUSION

When the new SQL Server 7.0 and ProClarity system is fully operational, Stanley (the manager of Business Analysis Group of Hewlett-Packard) expects it to provide significant benefits to account representatives in the field, business analysts in his and other groups, and HP's reseller customers: "I expect that account reps who are calling on a major account will log onto a Web page on Monday morning before going to call on that account and pull up last week's sales and inventory levels all the way down to the store level. They can produce a report that shows inventory problems in particular stores—for example, the new printer we just introduced three weeks ago is selling really well but it looks like we are having stock outs in particular stores. Then they can take our in-store audit data, which might show a problem with product placement in these stores. They can suggest that the problem may be that the product is displayed in the computer aisle, and people may not be shopping the computer aisle for a printer. Then, as analysts, we can study these trends over time and say, 'These stores are continually out of stock. These other stores are continually over stock and don't have as high a sell-through rate. Let's move inventory from these stores where it just sits on the shelves for two weeks to these stores where it sells out in half a week'".

"The bottom line", concludes Stanley, "is that this new system, through more accurate, detailed, and timely data, will make our business more efficient so we, in turn, can help our resellers make their businesses more efficient".

CASE STUDY

5 Data Warehousing in Levi Strauss*

C5.1 INTRODUCTION

In 1998, ArsDigita Corporation built a Web service as a front-end to an experimental custom clothing factory operated by Levi Strauss. Users would visit their site to choose a style of khaki pants, enter their waist, inseam, height, weight, and shoe size, and finally check out with their credit card. Their server would attempt to authorize a charge on the credit card through CyberCash. The factory IT system would poll their server's Oracle database periodically so that it could start cutting pants within 10 minutes of a successfully authorized order.

The whole purpose of the factory and Web service was to test and analyse consumer reaction to this method of buying clothing. Therefore, a data warehouse was built into the project almost from the start.

They did not buy any additional hardware or software to support the data warehouse. The public Web site was supported by a mid-range Hewlett-Packard Unix server that had ample leftover capacity to run the data warehouse. They created a new 'dw' Oracle user, GRANTed SELECT on the OLTP tables to the 'dw' user, and wrote procedures to copy all the data from the OLTP system into a star schema of tables owned by the 'dw' user. For queries, they added an IP address to the machine and ran a Web server program bound to that second IP address.

Here is how they explained their engineering decisions to their customer. They employed a standard star join schema for the following reasons:

- Many relational database management systems, including Oracle8.1, were heavily optimized to execute queries against these schemata.
- This kind of schema had been proven to scale to the world's largest data warehouses.
- If a data warehousing nerd was hired off the street, he or she would have no trouble understanding our schema.

*This case study is taken from Internet Web site, which is in public domain. We thank the author of the site.

In a star join schema, there is one fact table ("we sold a pair of khakis at 1:23 pm to Joe Smith") that references a bunch of dimension tables. As a general rule, if they are going to narrow our interest based on a column, it should be in the dimension table. That is, if they are only looking at sales of grey dressy fabric khakis, they should expect to accomplish that with WHERE clauses on columns of a product dimension table. By contrast, if they are going to be aggregating information with a SUM or AVG command, these data should be stored in the columns of the fact table. For example, the dollar amount of the sale should be stored within the fact table. Since they have so few prices (essentially only one), you might think that this should go in a dimension. However, by keeping it in the fact table they are more consistent with traditional data warehouses.

The following dimension tables are designed after some discussions with Levi's executives:

- *Time:* For queries comparing sales by season, quarter, or holiday.
- *Product:* For queries comparing sales by colour or style.
- *Ship to:* For queries comparing sales by region or state.
- *Promotion:* For queries aimed at determining the relationship between discounts and sales.
- *Consumer:* For queries comparing sales by first-time and repeat buyers.
- *User experience:* For queries looking at returned versus exchanged versus accepted items (most useful when combined with other dimensions, e.g. was a particular colour more likely to lead to an exchange request).

These dimensions allow them to answer questions such as:

- In what regions of the country are pleated pants most popular? (Fact table joined with the product and ship-to dimensions.)
- What percentage of pants were bought with coupons and how has that varied from quarter to quarter? (Fact table joined with the promotion and time dimensions.)
- How many pants were sold on holidays versus non-holidays? (Fact table joined with the time dimension.)

C5.2 THE DIMENSION TABLES

The time_dimension table is identical to the example given above.

```
create table time_dimension (
    time_key integer primary key,
    -- just to make it a little easier to work with; this is
    -- midnight (TRUNC) of the date in question
    oracle_date date not null,
    day_of_week varchar(9) not null, -- 'Monday', 'Tuesday'...
    day_number_in_month integer not null, -- 1 to 31
    day_number_overall integer not null, -- days from the epoch (first day is 1)
    week_number_in_year integer not null, -- 1 to 52
    week_number_overall integer not null, -- weeks start on Sunday
    month integer not null, -- 1 to 12
    month_number_overall integer not null,
    quarter integer not null, -- 1 to 4
    fiscal_period varchar(10),
    holiday_flag char(1) default 'f' check (holiday_flag in ('t', 'f')),
    weekday_flag char(1) default 'f' check (weekday_flag in ('t', 'f')),
    season varchar(50),
    event varchar(50)
);
```


The time_dimension table was populated with a single INSERT statement. The core work is done by Oracle date formatting functions. A helper table, integers, is used to supply a series of numbers to add to a starting date (1 July, 1998).

```
-- Uses the integers table to drive the insertion, which just contains
-- a set of integers, from 0 to n.
-- The 'epoch' is hardcoded here as July 1, 1998.

-- d below is the Oracle date of the day we're inserting.
insert into time_dimension
time_key, oracle_date, day_of_week, day_number_in_month,
day_number_overall, week_number_in_year, week_number_overall,
month, month_number_overall, quarter, weekday_flag)
select n, d, rtrim(to_char(d, 'Day')), to_char(d, 'DD'), n + 1,
    to_char(d, 'WW'),
    trunc((n + 3) / 7), -- July 1, 1998 was a Wednesday, so +3 to get the week
    numbers to line up with the week
    to_char(d, 'MM'), trunc(months_between(d, '1998-07-01') + 1),
    to_char(d, 'Q'), decode(to_char(d, 'D'), '1', 'f', '7', 'f', 't')
from (select n, to_date('1998-07-01', 'YYYY-MM-DD') + n as d
    from integers);
```

A bit of Oracle minutia is helpful in understanding this transaction. If you add a number to an Oracle date, you get another Oracle date. So adding 3 to '1998-07-01' will yield '1998-07-04'.

There are several fields left to be populated that cannot be derived using Oracle date functions: season, fiscal period, holiday flag, season, event. Fiscal period depended on Levi's choice of fiscal year. The event column was set aside for arbitrary blocks of time that were particularly interesting to the Levi's marketing team, e.g. a sale period. In practice, it was not used.

To update the holiday_flag field, two helper tables were used, one for 'fixed' holidays (those which occur on the same day each year), and one for 'floating' holidays (those which move around).

```
create table fixed_holidays (
    month               integer not null check (month >= 1 and month <= 12),
    day                 integer not null check (day >= 1 and day <= 31),
    name                varchar(100) not null,
    primary key (month, day)
);

-- Specifies holidays that fall on the Nth DAY_OF_WEEK in MONTH.
-- Negative means count backwards from the end.
create table floating_holidays (
    month integer not null check (month >= 1 and month <= 12),
    day_of_week varchar(9) not null,
    nth integer not null,
    name varchar(100) not null,
    primary key (month, day_of_week, nth)
);
```

Some example holidays are:

```
insert into fixed_holidays (name, month, day)
    values ('New Year''s Day', 1, 1);
insert into fixed_holidays (name, month, day)
    values ('Christmas', 12, 25);
insert into fixed_holidays (name, month, day)
    values ('Veteran''s Day', 11, 11);
insert into fixed_holidays (name, month, day)
    values ('Independence Day', 7, 4);

insert into floating_holidays (month, day_of_week, nth, name)
    values (1, 'Monday', 3, 'Martin Luther King Day');
insert into floating_holidays (month, day_of_week, nth, name)
    values (10, 'Monday', 2, 'Columbus Day');
insert into floating_holidays (month, day_of_week, nth, name)
    values (11, 'Thursday', 4, 'Thanksgiving');
insert into floating_holidays (month, day_of_week, nth, name)
    values (2, 'Monday', 3, 'President''s Day');
insert into floating_holidays (month, day_of_week, nth, name)
    values (9, 'Monday', 1, 'Labor Day');
insert into floating_holidays (month, day_of_week, nth, name)
    values (5, 'Monday', -1, 'Memorial Day');
```

(An extremely clever person who would recently read SQL for Smarties would probably be able to come up with an SQL statement to update the holiday_flag in the time_dimension rows.) Oracle includes two procedural languages, Java and PL/SQL. We can simply implement the following pseudocode in the procedural language of choice:


```
foreach row in "select name, month, day from fixed_holidays"
    update time_dimension
        set holiday_flag = 't'
        where month = row.month and day_number_in_month = row.day;
end foreach

foreach row in "select month, day_of_week, nth, name from floating_holidays"
    if row.nth > 0 then
        # If nth is positive, put together a date range constraint
        # to pick out the right week.
        ending_day_of_month := row.nth * 7
        starting_day_of_month := ending_day_of_month - 6

        update time_dimension
            set holiday_flag = 't'
                where month = row.month
                    and day_of_week = row.day_of_week
                    and starting_day_of_month <= day_number_in_month
                    and day_number_in_month <= ending_day_of_month;
else
    # If it is negative, get all the available dates
    # and get the nth one from the end.
    i := 0;
    foreach row2 in "select day_number_in_month from time_dimension
                        where month = row.month
                            and day_of_week = row.day_of_week
                        order by day_number_in_month desc"
        i := i - 1;
        if i = row.nth then
            update time_dimension
                set holiday_flag = 't'
                where month = row.month
                    and day_number_in_month = row2.day_number_in_month
                break;
            end if
        end foreach
    end if
end foreach
```

The product dimension

The product dimension contains one row for each unique combination of colour, style, cuffs, pleats, etc.

```
create table product_dimension (
    product_key integer primary key,
    -- right now this will always be "ikhakis"
    product_type varchar(20) not null,
    -- could be "men", "women", "kids", "unisex adults"
    expected_consumers varchar(20),
    color varchar(20),
    -- "dressy" or "casual"
    fabric varchar(20),
    -- "cuffed" or "hemmed" for pants
    -- null for stuff where it doesn't matter
    cuff_state varchar(20),
    -- "pleated" or "plain front" for pants
    pleat_state varchar(20)
);
```

To populate this dimension, a one-column table was created for each field in the dimension table and use a multi-table join without a WHERE clause. This generates the cartesian product of all the possible values for each field:

```
create table t1 (expected_consumers varchar(20));
create table t2 (color varchar(20));
create table t3 (fabric varchar(20));
create table t4 (cuff_state varchar(20));
create table t5 (pleat_state varchar(20));

insert into t1 values ('men');
insert into t1 values ('women');
insert into t1 values ('kids');
insert into t1 values ('unisex');
insert into t1 values ('adults');
[etc.]

insert into product_dimension
(product_key, product_type, expected_consumers,
color, fabric, cuff_state, pleat_state)
select
    product_key_sequence.nextval,
    'ikhakis',
    t1.expected_consumers,
    t2.color,
    t3.fabric,
    t4.cuff_state,
    t5.pleat_state
from t1,t2,t3,t4,t5;
```

Notice that an Oracle sequence, `product_key_sequence`, is used to generate unique integer keys for each row as it is inserted into the dimension.

The promotion dimension

The art of building the promotion dimension is dividing the world of coupons into a broad categories, e.g. 'between 10 and 20 dollars'. This categorization depended on the learning that the marketing executives did not care about the difference between a $3.50 and a $3.75 coupon.

```
create table promotion_dimension (
    promotion_key integer primary key,
    -- can be "coupon" or "no coupon"
    coupon_state varchar(20),
    -- a text string such as "under $10"
    coupon_range varchar(20)
);
```

The separate `coupon_state` and `coupon_range` columns allow for reporting of sales figures broken down into fullprice/discounted or into a bunch of rows, one for each range of coupon size.

The consumer dimension

The consumer dimension is extremely simple. It is used to record whether or not a sale in the fact table was to a new or a repeat customer.

```
create table consumer_dimension (
consumer_key integer primary key,
-- 'new customer' or 'repeat customer'
repeat_class varchar(20)
);
```

The user experience dimension

If we are interested in building a report of the average amount of time spent contemplating a purchase versus whether the purchase was ultimately kept, the `user_experience_dimension` table will help.

```
create table user_experience_dimension (
    user_experience_key integer primary key,
    -- 'shipped on time', 'shipped late'
    on_time_status varchar(20),
    -- 'kept', 'returned for exchange', 'returned for refund'
    returned_status varchar(30)
);
```

The ship-to dimension

Classically one of the most powerful dimensions in a data warehouse, our ship_to_dimension table allows us to group sales by region or state.

```
create table ship_to_dimension (
    ship_to_key integer primary key,
    -- e.g., Northeast
    ship_to_region varchar(30) not null,
    ship_to_state char(2) not null
);

create table state_regions (
    state char(2) not null primary key,
    region varchar(50) not null
);

-- to populate:
insert into ship_to_dimension
(ship_to_key, ship_to_region, ship_to_state)
select ship_to_key_sequence.nextval, region, state
from state_regions;
```

Had this been a full-scale product for Levi Strauss, they would probably have wanted at least extra columns for country, city, and zip code. These columns would allow a regional sales manager to look at sales within a state. Much of detail is left out.

(In a data warehouse for a manufacturing wholesaler, the ship-to dimension would contain columns for the customer's company name, the division of the customer's company that received the items, the sales district of the salesperson who sold the order, etc.)

C5.3 THE FACT TABLE

The granularity of the fact table is one order. It was decided that we could afford this because the conventional wisdom in the data warehousing business in 1998 was that up to billion-row fact tables were manageable. Our retail price was $40 and it was tough to foresee a time when the factory could make more than 1,000 pants per day. So it did not seem extravagant to budget one row per order.

```
create table sales_fact (
    -- keys over to the OLTP production database
    order_id                    integer primary key,
    consumer_id                 integer not null,
    time_key                    not null references time_dimension,
    product_key                 not null references product_dimension,
```



```
    promotion_key               not null references promotion_dimension,
    consumer_key                not null references consumer_dimension,
    user_experience_key         not null references user_experience_dimension,
    ship_to_key                 not null references ship_to_dimension,
    -- time stuff
    minutes_login_to_order          number,
    days_first_invite_to_order      number,
    days_order_to_shipment          number,
    -- this will be NULL normally (unless order was returned)
    days_shipment_to_intent         number,
    pants_id                    integer,
    price_charged               number,
    tax_charged                 number,
    shipping_charged            number
);
```

After defining the fact table, it was populated with a single insert statement:

```
-- find_product, find_promotion, find_consumer, and find_user_experience
-- are PL/SQL procedures that return the appropriate key from the dimension
-- tables for a given set of parameters

insert into sales_fact
    select o.order_id, o.consumer_id, td.time_key,
    find_product(o.color, o.casual_p, o.cuff_p, o.pleat_p),
    find_promotion(o.coupon_id),
    find_consumer(o.pants_id),
    find_user_experience(o.order_state, o.confirmed_date, o.shipped_date),
    std.ship_to_key,
    minutes_login_to_order(o.order_id, usom.user_session_id),
    decode(sign(o.confirmed_date - gt.issue_date), -1, null,
    round(o.confirmed_date - gt.issue_date, 6)),
    round(o.shipped_date - o.confirmed_date, 6),
    round(o.intent_date - o.shipped_date, 6),
    o.pants_id, o.price_charged, o.tax_charged, o.shipping_charged
from khaki.reportable_orders o, ship_to_dimension std,
    khaki.user_session_order_map usom, time_dimension td,
    khaki.addresses a, khaki.golden_tickets gt
where o.shipping = a.address_id
    and std.ship_to_state = a.usps_abbrev
    and o.order_id = usom.order_id(+)
    and trunc(o.confirmed_date) = td.oracle_date
    and o.consumer_id = gt.consumer_id;
```

As noted in the comment at top, most of the work here is done by PL/SQL procedures such as find_product that dig up the right row in a dimension table for this particular order.

The preceding insert will load an empty data warehouse from the on-line transaction processing system's tables. Keeping the data warehouse up-to-date with what is happening in OLTP land requires a similar INSERT with an extra restriction of WHERE clause limiting orders to only those order ID is larger than the maximum of the order IDs currently in the warehouse. This is a safe transaction to execute as many times per day as necessary—even two simultaneous INSERTs would not corrupt the data warehouse with duplicate rows because of the primary key constraint on `order_id`. A daily update is traditional in the data warehousing world so we scheduled one every 24 hours using the Oracle `dbms_job` package.

C5.4 SAMPLE QUERIES

The following were completed: (a) defined a star schema, (b) populated the dimension tables, (c) loaded the fact table, and (d) arranged for periodic updating of the fact table. Now we can proceed to the interesting part of our data warehouse: retrieving information.

Using only the `sales_fact` table, we can ask for:

- the total number of orders, total revenue to date, tax paid, shipping costs to date, the average price paid for each item sold, and the average number of days to ship:

```
select count(*) as n_orders,
   round(sum(price_charged)) as total_revenue,
   round(sum(tax_charged)) as total_tax,
   round(sum(shipping_charged)) as total_shipping,
   round(avg(price_charged), 2) as avg_price,
   round(avg(days_order_to_shipment), 2) as avg_days_to_ship
from sales_fact;
```

- the average number of minutes from login to order (we exclude user sessions longer than 30 minutes to avoid skewing the results from people who inter-rupted their shopping session to go out to lunch or sleep for a few hours):

```
select round(avg(minutes_login_to_order), 2)
from sales_fact
where minutes_login_to_order < 30
```

- the average number of days from first being invited to the site by email to the first order (excluding periods longer than 2 weeks to remove outliers):

```
select round(avg(days_first_invite_to_order), 2)
from sales_fact
where days_first_invite_to_order < 14
```


Joining against the `ship_to_dimension` table let us ask how many pants were shipped to each region of the United States:

```
select ship_to_region, count(*) as n_pants
from sales_fact f, ship_to_dimension s
where f.ship_to_key = s.ship_to_key
group by ship_to_region
order by n_pants desc
```

Region	Pants Sold
New England Region	612
NY and NJ Region	321
Mid Atlantic Region	318
Western Region	288
Southeast Region	282
Southern Region	193
Great Lakes Region	177
Northwestern Region	159
Central Region	134
North Central Region	121

Note: These data are based on a random subset of orders from the Levi's site and we have also made manual changes to the report values. The numbers are here to give you an idea of what these queries do, not to provide insight into the Levi's custom clothing business.

Joining against the `time_dimension`, we can ask how many pants were sold for each day of the week:

```
select day_of_week, count(*) as n_pants
from sales_fact f, time_dimension t
where f.time_key = t.time_key
group by day_of_week
order by n_pants desc
```

Day of Week	Pants Sold
Thursday	3428
Wednesday	2823
Tuesday	2780
Monday	2571
Friday	2499
Saturday	1165
Sunday	814

It is possible to make pants with either a 'dressy' or 'casual' fabric. Joining against the `product_dimension` table can tell us how popular each option was as a function of colour:

```
select color, count(*) as n_pants, sum(decode(fabric,'dressy',1,0)) as n_dressy
from sales_fact f, product_dimension p
where f.product_key = p.product_key
group by color
order by n_pants desc
```

Color	Pants Sold	% Dressy
Dark tan	486	100
Light tan	305	49
Dark grey	243	100
Black	225	97
Navy blue	218	61
Medium tan	209	0
Olive green	179	63

Note: 100% and 0% indicate that those colours were available only in one fabric.

Here is a good case of how the data warehouse may lead to a practical result. If these were the real numbers from the Levi's warehouse, what would pop out at the manufacturing guys is that 97% of the black pants sold were in one fabric style. It might not make sense to keep an inventory of casual black fabric if there is so little consumer demand for it.

C5.4.1 Query Generation: The Commercial Closed-Source Route

The promise of a data warehouse is not fulfilled if all users must learn SQL syntax and how to run SQL*PLUS. It was suggested to Levi Strauss that they use Seagate Crystal Reports and Crystal Info to analyse their data. These packaged tools, however, ended up not fitting very well with what Levi's wanted to accomplish. First, constructing queries was not semantically simpler than coding SQL. The Crystal Reports consultant said that most of his clients ended up having a programmer set up the report queries and the business people would simply run the report every day against new data. If professional programmers had to construct queries, it seemed just as easy just to write more admin pages using our standard Web development tools, which required about 15 minutes per page. Second, it was impossible to ensure availability of data warehouse queries to authorized users anywhere on the Internet. Finally there were security and social issues associated with allowing a SQL*Net connection from a Windows machine running Crystal Reports out through the Levi's firewall to our Oracle data warehouse on the Web.

Not knowing if any other commercial product would work better and not wanting to disappoint our customer, the ArsDigita Community System was extended with a data warehouse query module that runs as a Web-only tool. This is a free open-source system and comes with the standard ACS package that can be downloaded.

C5.4.2 Query Generation: The Open-Source ACS Route

The 'dw' module in the ArsDigita Community System is designed with the following goals:

1. Naive users can build simple queries by themselves.
2. Professional programmers can step in to help out the naive users.
3. A user with no skill can re-execute a saved query.

It is possible to keep one row per query in the queries table as follows:

```
create table queries (
    query_id integer primary key,
    query_name varchar(100) not null,
    query_owner not null references users,
    definition_time date not null,
    -- if this is non-null, we just forget about all the query_columns
    -- stuff; the user has hand-edited the SQL
    query_sql varchar(4000)
);
```

Unless the `query_sql` column is populated with a hand-edited query, the query will be built up by looking at several rows in the `query_columns` table:

```
-- this specifies the columns we will be using in a query and
-- what to do with each one, e.g., "select_and_group_by" or
-- "select_and_aggregate"

-- "restrict_by" is tricky; value1 contains the restriction value, e.g., '40'
-- or 'MA' and value2 contains the SQL comparion operator, e.g., "=" or ">"
create table query_columns (
    query_id not null references queries,
    column_name varchar(30),
    pretty_name varchar(50),
    what_to_do varchar(30),
    -- meaning depends on value of what_to_do
    value1 varchar(4000),
    value2 varchar(4000)
);

create index query_columns_idx on query_columns(query_id);
```

The `query_columns` definition appears strange at first. It specifies the name of a column but not a table. This module is predicated on the simplifying assumption that we have one enormous view, `ad_hoc_query_view`, that contains all the dimension tables' columns alongside the fact table's columns.

Here is how the view for the Levi's data warehouse was created:

```
create or replace view ad_hoc_query_view
as
select minutes_login_to_order, days_first_invite_to_order,
    days_order_to_shipment, days_shipment_to_intent, pants_id,
    price_charged, tax_charged, shipping_charged,
    oracle_date, day_of_week,
    day_number_in_month, week_number_in_year, week_number_overall,
    month, month_number_overall, quarter, fiscal_period,
    holiday_flag, weekday_flag, season, color, fabric, cuff_state,
    pleat_state, coupon_state, coupon_range, repeat_class,
    on_time_status, returned_status, ship_to_region, ship_to_state
from sales_fact f, time_dimension t, product_dimension p,
    promotion_dimension pr, consumer_dimension c,
    user_experience_dimension u, ship_to_dimension s
where f.time_key = t.time_key
and f.product_key = p.product_key
and f.promotion_key = pr.promotion_key
and f.consumer_key = c.consumer_key
and f.user_experience_key = u.user_experience_key
and f.ship_to_key = s.ship_to_key;
```

At first glance, this looks like sluggish Oracle performance. We shall be doing a seven-way JOIN for every data warehouse query, regardless of whether we need information from some of the dimension tables or not.

We can test this assumption as follows:

```
-- tell SQL*Plus to turn on query tracing
set autotrace on

-- let's look at how many pants of each color
-- were sold in each region

SELECT ship_to_region, color, count(pants_id)
FROM ad_hoc_query_view
GROUP BY ship_to_region, color;
```


Oracle will return the query results first and then explain how those results were obtained:

ship_to_region	color	count(pants_id)
Central Region	black	46
Central Region	dark grey	23
Central Region	dark tan	39
..		
Western Region	medium tan	223
Western Region	navy blue	245
Western Region	olive green	212

```
Execution Plan
  0       SELECT STATEMENT Optimizer=CHOOSE (Cost=181 Card=15 Bytes=2430)
  1   0     SORT (GROUP BY) (Cost=181 Card=15 Bytes=2430)
  2   1       NESTED LOOPS (Cost=12 Card=2894 Bytes=468828)
  3   2         HASH JOIN (Cost=12 Card=885 Bytes=131865)
  4   3           TABLE ACCESS (FULL) OF 'PRODUCT_DIMENSION' (Cost=1 Card=336
                  Bytes=8400)
  5   3           HASH JOIN (Cost=6 Card=885 Bytes=109740)
  6   5             TABLE ACCESS (FULL) OF 'SHIP_TO_DIMENSION' (Cost=1
                    Card=55 Bytes=1485)
  7   5             NESTED LOOPS (Cost=3 Card=885 Bytes=85845)
  8   7               NESTED LOOPS (Cost=3 Card=1079 Bytes=90636)
  9   8                 NESTED LOOPS (Cost=3 Card=1316 Bytes=93436)
 10   9                   TABLE ACCESS (FULL) OF 'SALES_FACT' (Cost=3
                          Card=1605 Bytes=93090)
 11   9                   INDEX (UNIQUE SCAN) OF 'SYS_C0016416' (UNIQUE)
 12   8                 INDEX (UNIQUE SCAN) OF 'SYS_C0016394' (UNIQUE)
 13   7               INDEX (UNIQUE SCAN) OF 'SYS_C0016450' (UNIQUE)
 14   2             INDEX (UNIQUE SCAN) OF 'SYS_C0016447' (UNIQUE)
```

As can be seen from the table names in bold face, Oracle was smart enough to examine only tables relevant to our query: `product_dimension`, because we asked about colour; `ship_to_dimension`, because we asked about region; `sales_fact`, because we asked for a count of pants sold. Bottom line: Oracle did a three-way JOIN instead of the seven-way JOIN specified by the view.

To generate an SQL query into `ad_hoc_query_view` from the information stored in `query_columns` is most easily done with a function in a procedural language such as Java, PL/SQL, PERL, or Tcl (here is pseudocode):

```
proc generate_sql_for_query(a_query_id)
    select_list_items list;
    group_by_items list;
    order_clauses list;

    foreach row in "select column_name, pretty_name
           from query_columns
           where query_id = a_query_id
              and what_to_do = 'select_and_group_by'"]
    if row.pretty_name is null then
       append_to_list(group_by_items, row.column_name)
    else
       append_to_list(group_by_items, row.column_name || 'as"' ||
    row.pretty_name || '"'
    end if
end foreach

foreach row in "select column_name, pretty_name, value1
       from query_columns
       where query_id = a_query_id
          and what_to_do = 'select_and_aggregate'"
   if row.pretty_name is null then
      append_to_list(select_list_items, row.value1 || row.column_name)
   else
      append_to_list(select_list_items, row.value1 || row.column_name ||
   ' as "' || row.pretty_name || '"'
   end if
end foreach

foreach row in "select column_name, value1, value2
       from query_columns
       where query_id = a_query_id
          and what_to_do = 'restrict_by'"
      append_to_list(where_clauses, row.column_name || ' ' || row.value2 ||
   ' ' || row.value1)
   end foreach

   foreach row in "select column_name
          from query_columns
          where query_id = a_query_id
             and what_to_do = 'order_by'"]
      append_to_list(order_clauses, row.column_name)
   end foreach

   sql := "SELECT " || join(select_list_items, ', ') ||
      " FROM ad_hoc_query_view"
   if list_length(where_clauses) > 0 then
      append(sql, 'WHERE' || join(where_clauses, 'AND'))
   end if
```



```
   if list_length(group_by_items) > 0 then
      append(sql, 'GROUP BY' || join(group_by_items, ','))
   end if

   if list_length(order_clauses) > 0 then
      append(sql, ' ORDER BY ' || join(order_clauses, ','))
   end if

   return sql
end proc
```

How well does this work in practice? Suppose we were going to run regional advertisements. Should the models be pictured with pleated or plain front pants? We need to look at recent sales by region. With the ACS query tool, a user can use HTML forms to specify the following:

```
pants_id : select and aggregate using count
ship_to_region : select and group by
pleat_state : select and group by
```

The preceding pseudocode turns that into

```
SELECT ship_to_region, pleat_state, count(pants_id)
FROM ad_hoc_query_view
GROUP BY ship_to_region, pleat_state
```

which is going to report sales going back to the dawn of time. If we were not clever enough to anticipate the need for time windowing in our forms-based interface, the 'hand edit the SQL' option will save us. A professional programmer can be grabbed for a few minutes to add

```
SELECT ship_to_region, pleat_state, count(pants_id)
FROM ad_hoc_query_view
WHERE oracle_date > sysdate - 45
GROUP BY ship_to_region, pleat_state
```

Now we are limiting results to the last 45 days (Table C5.1):

Table C5.1 Limiting results of sales for the last 45 days

ship_to_region	pleat_state	count(pants_id)
Central Region	plain front	8
Central Region	pleated	26
Great Lakes Region	plain front	14
Great Lakes Region	pleated	63
Mid Atlantic Region	plain front	56
Mid Atlantic Region	pleated	162
NY and NJ Region	plain front	62
NY and NJ Region	pleated	159
New England Region	plain front	173
New England Region	pleated	339
North Central Region	plain front	7
North Central Region	pleated	14
Northwestern Region	plain front	20
Northwestern Region	pleated	39
Southeast Region	plain front	51
Southeast Region	pleated	131
Southern Region	plain front	13
Southern Region	pleated	80
Western Region	plain front	68
Western Region	pleated	120

We can see above that plain front pants are very unpopular in the Great Lakes and South but more popular in New England and the West. It would be nicer to see percentages within region, but standard SQL does not make it possible to combine results to values in surrounding rows. We will need to refer to the "SQL for Analysis" chapter in the Oracle data warehousing documents to read up on extensions to SQL that makes this possible:

```
SELECT
    ship_to_region,
    pleat_state,
    count(pants_id),
    ratio_to_report(count(pants_id))
       over (partition by ship_to_region) as percent_in_region
FROM ad_hoc_query_view
WHERE oracle_date > sysdate - 45
GROUP BY ship_to_region, pleat_state
```

When asked Oracle to window the results ('partition by ship_to_region') and compare the number of pants in each row to the sum across all the rows within a regional group. Here is the result (Table C5.2):

Table C5.2 Sum across all the rows with in a regional group

ship_to_region	pleat_state	count(pants_id)	percent_in_region
...			
Great Lakes Region	plain front	14	.181818182
Great Lakes Region	pleated	63	.818181818
...			
New England Region	plain front	173	.337890625
New England Region	pleated	339	.662109375
...			

This is not quite what we want. The 'percents' are fractions of 1 and reported with far too much precision. We tried inserting the Oracle built-in round function in various places of this SQL statement but all we got for our troubles was "ERROR at line 5: ORA-30484: missing window specification for this function". We had to add an extra layer of SELECT, a view-on-the-fly, to get the report that we wanted:

```
select ship_to_region, pleat_state, n_pants, round(percent_in_region*100)
from
(SELECT
    ship_to_region,
    pleat_state,
    count(pants_id) as n_pants,
    ratio_to_report(count(pants_id))
       over (partition by ship_to_region) as percent_in_region
FROM ad_hoc_query_view
WHERE oracle_date > sysdate - 45
GROUP BY ship_to_region, pleat_state)
```

returns

Table C5.3

ship_to_region	pleat_state	count(pants_id)	percent_in_region
...			
Great Lakes Region	plain front	14	18
Great Lakes Region	pleated	63	82
...			
New England Region	plain front	173	34
New England Region	pleated	339	66
...			

CONCLUSION

This case study brings out in a very single, lucid and popular style how to build and use single data warehouses using Oracle platform.

CASE STUDY

6 Data Warehousing in the World Bank

C6.1 INTRODUCTION

The World Bank collects and maintains huge data of economic and developmental parameters for all the third world countries across the globe. Originally the Bank performed analysis on this huge data manually and later with limited tools for analysis. Through the 'Live Database' and Country Economic Time Series, World Bank aimed to provide the economists world over with greater power of analysis of the data.

For the purpose of monitoring the effectiveness of the various World Bank assisted projects in the third world countries, the Bank started collecting and analysing macroeconomic financial statistics and also information on parameters such as poverty, health, education, environment and public sector. For more than one hundred developing countries, several thousands of economic indicators were being captured. It was a great challenge to refresh this data continuously (periodically) and make it accessible to various levels of Governments all over the world and also to economists, educators, academicians and any member of the general public. This broad accessibility was expected to contribute in many ways positively towards the growth of these developing economies.

To quote Ronnic Hammad, an economist at the World Bank, "Bringing the Live Databases to the clients will allow policy makers, civil society, researchers and others to have access to the latest economic information so as to influence decisions that will help in planning for better future. Such Live Databases in every Government, Ministry of Finance, every Central Bank and every Statistics Office, all made accessible will mean a lot for the economic planning".

C6.2 THE 'LIVE DATABASE' (LDB) DATA WAREHOUSE

In 1995 the 'Live Database' of the World Bank was developed by World Bank East Africa team. This was developed on SQL Server 2000 platform for the database. The OLAP cubes were defined for this database using OLAP server module of SQL Server 2000. Universal access was provided for this data

warehouse which was called 'Live Database'. This was so popular and so successful that the American Productivity Council called it the 'number one example of the transfer of best practice within an organization'. This product was given *meritorious award* and also resulted in much cost savings. The African region reported a saving of $12 million after the introduction to LDB data warehouse.

C6.3 BENEFITS OF THE SECOND-GENERATION LDB DATA WAREHOUSE

The first-generation LDB data warehouse had certain limitations—only a programmer could modify or add the data to it. Therefore, the second-generation LDB data warehouse was built using SQL Server 2000 Analysis Server along with 'Proclarity' (GUI tool) of Knosys Corporation. This package offered direct user functionality which was otherwise requiring technical intervention by a programmer. Proclarity also provided Web enablement, thereby ensuring universal accessibility. The highlight of this second-generation LDB data warehouse was that economists in the World Bank and all over the world were able to perform analysis using complex queries using Multi Dimensional extension (MDX) on OLAP cubes in SQL Server 2000 platform over the Web.

This exercise has resulted in significant cost savings by reducing the time and effort required to prepare a large variety of reports to suit varying needs of the economists and other governmental decision-makers to aid effective and better economic planning.

CASE STUDY

7 HARBOR, A Highly Available Data Warehouse

C7.1 INTRODUCTION

Today's data warehouses depend critically on various computational elements and networking elements. The computational elements are: processors, disk units, etc. The networking elements include switches, hubs, modems and lines. If any of these elements fail then it will result in the failure of the data warehousing and OLAP services offered. In other words a highly available data warehouse is needed to ensure a high-level user satisfaction of the services.

Any highly available database system or data warehousing system will use data replication to ensure that the data access continues with no or very few interruptions, the latter may arise if some computational system or component fails. Each database object, e.g. a tuple, a partition or a table, will have to be replicated at least once and distributed among many sites. Normally, the approaches for high availability include identical sites, identical replicas and identical ways of storing the replicas and identical mechanisms for distribution. The system under consideration for a highly available data warehouse, called HARBOR (Highly Available Replication-Based Online Recovery) is more flexible and does not insist on all these requirements of identical copies, as long as they represent logically the same data. With this flexibility, it is possible to store data redundantly in different sort orders in various data compression formats. The different updatable materialized views are also possible so that a wider variety of queries can be answered by the query optimizer and query processor (than is possible with the uniform formats of copies stored). A system called **C-Store** achieves an order of magnitude higher performance than is usual by storing the data in different sort orders and different compression formats. Data redundancy can, therefore, provide higher performance and also higher availability in HARBOR.

Normally, data warehouses cater specifically to large ad hoc query workloads over large read data sets intermingled with a small number of OLTP transactions which may touch a few records only, but affect the most recent data. Under such conditions the conventional locking techniques cause poor performance due to

considerable lock contention. A better solution is to use 'snapshot isolation' in which the read-only transactions read data without locks and updated transactions modify a copy of the data. In this the conflicts are resolved by an optimistic concurrency protocol with locks. In the case of HARBOR, a time travel mechanism is used, similar to snapshot isolation that involves explicit versioned and time stamped representation of data to isolate read-only transactions. Historical queries are as of some earlier T read time slices of the data that are guaranteed to remain unaffected by the subsequent transactions and, hence, proceed without acquiring read locks. Updated transactions and read-only transactions that wish to read the most up-to-date data can use the conventional read and write locks for isolation (as in two-phase locking).

C7.2 FAULT/FAILURE TOLERANCE

Any highly available database system will deploy some form of replication for fault or failure tolerance. A fault/failure tolerant system is defined to provide K-safety if upto K-sites fail and the system will still continue to provide services for any query or transaction. The minimum number of sites required for K-safety is K+1, where the K+1 workers store the same replicated data. In the approach taken in HARBOR it is assumed that the database designer has replicated the data and structured the database in such a way as to provide K-safety. The high availability in HARBOR guarantees that K simultaneous failures can be tolerated and still bring the failed sites online. If more than K-sites fail simultaneously, then this approach will not be applicable and the recovery can be achieved by other time tested solutions such as restoring from archival copy or rolling back changes to restore some globally consistent state. However, the database design can choose an appropriate value for K to reduce the probability of K simultaneous failures down to some acceptable value (for a specific application), as this recovery approach can be applied to bring sites online as they fail.

The approach assumes reliable network transfers via TCP/IP. It also assumes 'fail stop failures' (this means that when a failure occurs it is assumed to be a complete stop and not partial or incompletely written disk pages and network particulars) and does not deal with corrupted data, incompletely written disk pages and network partitions.

C7.3 HISTORICAL QUERY PROCESSING

In HARBOR, the historical queries are processed in a unique manner. By definition, a historical query, as of some past time, is a read-only query that returns a result, as if the query had been executed on the database at time T, i.e. a historical query at time T sees not committed or uncommitted updates after time T. This time travel feature enables the inspection of past states of the database.

Such historical queries are supported in HARBOR using 'versioned' representation of data in which time stamps are associated with each tuple. 'Insertion time' and 'deletion time' are added on to the tuple as insertion time, deletion time, a_1, a_2, a_3 ... a_N instead of the usual tuple a_1, a_2, ..., a_N.

The time stamp values are assigned at commit time as part of the commit protocol. Upon insertion, the insertion time stamp is inserted with a '0' value for deletion time stamp and vice versa upon deletion. If there is an update transaction which updates the tuple rather than updating the tuple in place, such an update is represented as a deletion of the old tuple and an insertion of the new tuple.

On doing this, the information necessary to answer historical queries is preserved. To answer a historical query on a tuple at time 'T', it has to be confirmed whether the tuple was inserted at or before time T and deleted T, or after T.

As historical queries view an old time slice of the database and all subsequent update transactions use later time stamps hence no (current) update transactions can ever affect the output of a historical query for some past time. Moreover, no locks are required for historical queries. The amount of history maintained in a system can be confirmed by the user by deleting the tuples before a specific time.

C7.4 RECOVERY ALGORITHM

The algorithm for recovery consists of three phases and uses time stamps associated with tuples to answer time-based range queries for the tuples inserted or deleted during a specified time range. In the first phase, a checkpointing mechanism is used to determine the most recent time T such that it is guaranteed that all the updates upto and including time T have been flushed to the disk. The crashed site then uses the time stamps available for historical queries to run local update transactions to restore itself to the time of its last checkpoint. In order to record checkpoints, the buffer pool will have to maintain a *duty pages table* with the identity of all *in memory* pages and a flag for each page indicating whether it contains any changes not yet flushed to the disk. During the normal processing, the database periodically writes a checkpoint for some past time T by first taking a snapshot of the dirty pages table at time T+1. For each page that is dirty in the snapshot, the system obtains a write latch for the page, flushes the page to the disk, and then releases the latch. After all the dirty pages are flushed, T is recorded onto the same well-known location on the disk and the checkpoint is completed. The checkpoint will be 0 on initial condition or when it got corrupted.

In the second phase, the site executes historical queries on other live sites that contain replicated copies of its data in order to catch up with the changes made between the last checkpoint and sometime closer to the present. It is to be noted

as a significant feature of the algorithm that read locks are not required to run historical queries—this ensures that the system is not quitened or slowed down while large amounts of data are copied over the network—otherwise this approach will not be viable (if read locks are required to be obtained for recovery query).

In the third (or final) phase, the site executes the standard non-historical queries with read locks to catch up with any committed changes between the start of the second phase and the current time. The coordinator then forwards any relevant update requests of ongoing transaction to the site to enable the crashed site to join any pending transaction and come online—this is possible only if the coordinator maintains a queue of update reports for each ongoing transaction.

C7.5 PERFORMANCE

Performance evaluation had shown that the recovery approach described in the foregoing section works well for data warehouses and similar environments, where the updated work loads consist primarily of insertions with relatively few updates to historical data.

CONCLUSION

In this case study we have presented an advanced and effective methodology for ensuring high availability of a data warehouse, HARBOR. Similar techniques can be used for ensuring high availability for any data warehouse.

CASE STUDY

8 A Typical Business Data Warehouse for a Trading Company

Let us consider a typical trading company. This hypothetical company sells products around the world and records data into its sample database that is created on a MS SQL Server 2000. The organization's businessowners would like to have analytical views, including graphs and charts that display the company's performance in terms of customer, employee, supplier, and product. Having such a tool would help the stakeholders promote the products that fall short of being sold in hot trading areas.

The example considered in this case study is very simple, as it is hypothetical and therefore, quite easy to build. The reasons are as follows:

First, we have a set of reports in mind to be supported by the warehouse. Usually, business owners know they need a data warehouse but are unable to give a concrete idea of the warehouse. It might take at least a few interviews to figure out and identify exactly what type of reports and analytical views the users would like to see. But in our present case study we know very clearly what type of reports are required.

As the data is already in a MS SQL Server database, which has a fairly simple structure, the first few steps of a typical warehouse project are already ready for us. In reality, however we may not always be so lucky: The data warehouse architect usually has to identify the multiple data sources that will be used to populate the warehouse. The organization's data could be stored in various relational database management systems (Oracle, MS SQL Server, DB2, and MS Access being the most common), spreadsheets, e-mail systems, and even in paper format. Once we identify all the data sources, we need to create data extraction routines using ETL (Extract Transform and Load) or DTS (Data Transformation Services) as the case may be, to transfer the data from its source to a SQL Server database. Furthermore, depending on the sources, we may not be in a position to manipulate the data until it is in MS SQL Server format. The data undergoes cleansing and validation as defined by the user in the ETL/DTS software (whichever may be used).

The main database has very clear object names; for example, the Orders table contains information on customer orders, Employees table has data about the

employees, and Order Details table has details of each order. Again, in the real world this might not always be the case—so simple and straightforward. We may have to figure out what some of the cryptic object names mean and exactly which data elements we require. The data warehouse architect often needs to create a list of data mappings and clean the data as it is loaded into the warehouse. For example, the customer names might not always be stored in the same format in the various data sources.

Once we have decided which data we need, we can create and populate a staging database and then the dimensional data model. Depending on the project, we may or may not have to have a staging database. If we have multiple data sources and need to correlate the data from these sources prior to populating a dimensional data structure, then having a staging database is convenient. Furthermore, a staging database will be a handy tool for testing. We can compare a number of records in the original data source with the number of records in the staging tables to ensure that the ETL routines work correctly. In this case the database already has all the data we need in easily accessible format; therefore, we need not create a staging database.

Now let us perform the dimensional modelling for this data warehouse.

Dimensional modelling is different from its relational counterpart, the relational database modelling. The dimensional models consist of fact tables and dimension tables. The typical fact tables contain numerous foreign keys referencing the dimension tables. The dimension tables, on the other hand, usually contain very few columns—dimension key, value, create, and update date, and perhaps an obsolete date. The fact tables record occurrences of a measurable fact, such as customer orders. The dimension tables provide a way to slice business data across various diagonals of company's operations; for example, we can examine orders by customer or by product.

We can use the `obsolete_date` column within the dimension tables to track the history of the values that change over time. This concept is known as *slowly changing dimension*. For example, consumers of the products may change their last names for any personal reason. Similarly, multiple departments within the organization can be combined into one, or one department can be divided into many. In some cases, we may keep only the current value. If so, we can simply override the existing value with the new value in the dimension table. In other cases, we must keep track of the old value, and the new value. This is when we use the record obsolete date to track the timeframe during which the record was valid.

In the case of our present data warehouse, the dimensional model will be very simple, consisting of the following dimension tables and one fact table. Since this is just a static database and there is no new data to populate it regularly, we need not add the `obsolete_date` column to the dimensions. We can create fact and dimension tables using the following script (in MS SQL Server):

```
CREATE TABLE dbo.dim_supplier(
    supplier_ident INT IDENTITY(1, 1),
    supplier_id INT NOT NULL,
    supplier_name VARCHAR(255) NOT NULL,
    supplier_city VARCHAR (255) NULL,
    country VARCHAR(255) NULL
)

CREATE TABLE dbo.dim_product (
    product_ident INT IDENTITY(1, 1),
    product_id INT NOT NULL,
    product_name VARCHAR(255) NOT NULL,
    discontinued BIT NOT NULL
)

CREATE TABLE dbo.dim_customer (
    customer_ident INT IDENTITY(1, 1),
    customer_id VARCHAR(20) NOT NULL,
    customer_name VARCHAR(255) NOT NULL,
    customer_city VARCHAR(255) NULL,
    customer_country VARCHAR(255) NULL
)

CREATE TABLE dbo.dim_employee (
    employee_ident INT IDENTITY(1, 1),
    employee_id INT NOT NULL,
    employee_name VARCHAR(85) NOT NULL,
    employee_city VARCHAR(255) NULL,
    employee_country VARCHAR(255) NULL
)

CREATE TABLE dbo.dim_time (
    time_member_key INT NOT NULL ,
    calendar_date_dt DATETIME NOT NULL ,
    calendar_day_of_week_num INT NOT NULL ,
    calendar_day_of_week_name VARCHAR(15) NOT NULL ,
    calendar_day_of_month_num INT NOT NULL ,
    calendar_day_of_year_num INT NOT NULL ,
    calendar_week_num INT NOT NULL ,
    calendar_month_num INT NOT NULL ,
    calendar_month_name VARCHAR (15) NOT NULL ,
    calendar_quarter_num INT NOT NULL ,
    calendar_year_num INT NOT NULL
)

CREATE TABLE fact_sales (
    customer_ident INT NOT NULL,
    product_ident INT NOT NULL,
    employee_ident INT NOT NULL,
    supplier_ident INT NOT NULL,
```



```
    total_sale SMALLMONEY NOT NULL,
    time_member_key INT NOT NULL
)
```

Next let us populate these tables using the following queries:

```
-supplier dimension:
INSERT dim_supplier (
    supplier_id ,
    supplier_name ,
    supplier_city ,
    country )
SELECT
    supplierid,
    companyname,
    city,
    country
FROM suppliers

-product dimension:
INSERT dim_product (
    product_id,
    product_name,
    discontinued)
SELECT
    productid,
    productname,
    discontinued
FROM products

-customer dimension:
INSERT dim_customer (
    customer_id,
    customer_name,
    customer_city,
    customer_country)
SELECT
    customerid,
    companyname,
    city,
    country
FROM customers

-employee dimension:
INSERT dim_employee (
    employee_id,
    employee_name,
    employee_city,
    employee_country)
SELECT
    employeeid,
```

```
    TItleOfCourtesy + ' ' + FirstName + ' ' + LastName AS employee_name,
    city,
    country
FROM employees
```

Note that dim_time is a special dimension. It is not populated by data that is already in the warehouse. Instead we populate it with calendar dates and date parts (day, month, quarter, year, and so forth) so that we can aggregate the warehouse data as needed. You can come up with a routine that populates your own time dimension; here is a sample store procedure that one can use to populate the time dimension:

```
CREATE PROCEDURE load_dim_time (
    @dim_table_name VARCHAR(255),
    @start_date_dt SMALLDATETIME,
    @end_date_dt SMALLDATETIME
)
AS
SET NOCOUNT ON
DECLARE
    @sql_string NVARCHAR(1024)
    , @time_member_key INT
    , @calendar_date_dt SMALLDATETIME
    , @calendar_day_of_week_num INT
    , @calendar_day_of_week_name VARCHAR(10)
    , @calendar_day_of_month_num INT
    , @calendar_day_of_year_num INT
    , @calendar_week_num INT
    , @calendar_month_num INT
    , @calendar_month_name VARCHAR(10)
    , @calendar_quarter_num INT
    , @calendar_year_num INT

SET @calendar_date_dt = @start_date_dt
WHILE (@calendar_date_dt <= @end_date_dt)
    BEGIN
    IF NOT EXISTS
    (
        SELECT time_member_key
        FROM dim_time
        WHERE calendar_date_dt = @calendar_date_dt
    )
BEGIN
    SELECT
    @calendar_day_of_week_num = DATEPART(dw, @calendar_date_dt)
    , @calendar_day_of_week_name = DATENAME(WEEKDAY, @calendar_date_dt)
    , @calendar_day_of_month_num = DATEPART(DD, @calendar_date_dt)
    , @calendar_day_of_year_num = DATEPART(DY, @calendar_date_dt)
    , @calendar_week_num = DATEPART(WK, @calendar_date_dt)
```



```
    , @calendar_month_num = DATEPART(M, @calendar_date_dt)
    , @calendar_month_name = DATENAME(MONTH, @calendar_date_dt)
    , @calendar_quarter_num = DATEPART(QQ, @calendar_date_dt)
    , @calendar_year_num = DATEPART(YYYY, @calendar_date_dt)
    , @time_member_key =
    CAST(
    CAST(@calendar_year_num AS VARCHAR) +
    RIGHT('00' + CAST(@calendar_day_of_year_num AS VARCHAR), 3)
    AS INT)
    SELECT @sql_string =
    'INSERT INTO ' + @dim_table_name +
    ' (' +
    'time_member_key, ' +
    'calendar_date_dt, ' +
    'calendar_day_of_week_num,' +
    'calendar_day_of_week_name,' +
    'calendar_day_of_month_num,' +
    'calendar_day_of_year_num,' +
    'calendar_week_num,' +
    'calendar_month_num,' +
    'calendar_month_name,' +
    'calendar_quarter_num, ' +
    'calendar_year_num' +
    ') ' +
    'VALUES ' +
    '(' +
    CHAR(39) + CAST(@time_member_key AS VARCHAR) + CHAR(39) + ',' +
    CHAR(39) + CAST(@calendar_date_dt AS VARCHAR) + CHAR(39) + ',' +
    CAST(@calendar_day_of_week_num AS VARCHAR) + ',' +
    CHAR(39) + @calendar_day_of_week_name + CHAR(39) + ',' +
    CAST(@calendar_day_of_month_num AS VARCHAR) + ',' +
    CAST(@calendar_day_of_year_num AS VARCHAR) + ',' +
    CAST(@calendar_week_num AS VARCHAR) + ',' +
    CAST(@calendar_month_num AS VARCHAR) + ',' +
    CHAR(39) + @calendar_month_name + CHAR(39) + ',' +
    CAST(@calendar_quarter_num AS VARCHAR) + ',' +
    CAST(@calendar_year_num AS VARCHAR) + ')'

    EXEC sp_executesql @sql_string
END

    SET @calendar_date_dt = @calendar_date_dt + 1
END
    SET @calendar_date_dt = @calendar_date_dt + 1
END

/* now use load_dim_time procedure to populate dim_time table with needed dates
*/

EXEC load_dim_time dim_time, '1/1/96', '1/1/99'
```

MS SQL Server has an ETL tool—DTS—which can be used to execute and schedule the data warehouse population routines. A typical DTS package determines which data rows need to be extracted from their source and inserts such rows into the appropriate dimension and fact tables. As this is a sample application, there is no need to create any DTS packages, but in the real world ETL routines may become considerably complicated and might take several weeks or more to develop.

CONCLUSION

In this case study, we have introduced the steps involved in building a typical data warehouse. We showed how to model, create and populate a dimensional data model which will be a cornerstone of the warehouse and analytical reports, using MS SQL Server 2000 for a typical trading company.

Note: The above case study is derived from the open websites—we thankfully acknowledge the original author of the case on the web.

CASE STUDY

9 Customer Data Warehouse of the World's First and Largest Online Bank in the United Kingdom

Egg PLC provides banking, insurance, investments and mortgages to more than 3 million customers through its Internet site and other distribution channels. Established in 1998, Egg PLC pioneered online banking not only in the United Kingdom but also throughout the world. Its technical infrastructure and support was provided by Sun Microsystems.

C9.1 BACKGROUND AND MOTIVATION FOR A DATA WAREHOUSE

Egg PLC, with more than 2.5 million transactions per day, required highly scalable but reliable IT and Internet infrastructure, as it supported all its customer services through the Internet. This high traffic Internet banking application was supported by high-end Sun V880 servers running the Solaris operating system. Sun Java System Application Server software was the web server utilized. Egg PLC supported 85% of its total transactions on the Internet through its site www.egg.com. Until 2001 only online transactions were provided. Originally, Egg was able to obtain customer information (for serving it online) by outsourcing the data warehousing activity to a company by the name Experian. However, the delay or latency between outsourcing and the provision of data became a serious issue for Egg, and it was forced to build and maintain its own data warehouse using Oracle and SAS, in addition to the existing Sun software and hardware infrastructure. The goal of the data warehouse was to provide a single source of truth to Egg regarding the details of the customers. Thus, a new data warehouse was designed, built and tested.

C9.2 THE CUSTOMER DATA WAREHOUSE: ENVIRONMENT, DATA FLOW

The first version of the Customer Data Warehouse (CDW) of Egg was built on Sun Fire 6800 Server and later, as it was expanded and called for greater computing resources, on Sun Fire 15K.

Finally, Egg's data warehouse resided on Sun Fire 15K with 16 CPUs and 10 GB for the core system. All the incoming data feeds for the data warehouse were fed onto this system. The storage application is an EMC's SAN (Storage Area Network). The system works on Solaris 9 operating system, with Oracle 9i as the DBMS pending migration to Oracle 10g. Virtual domaining, a special feature supported by Solaris 10 is utilized by Egg's data warehouse. Oracle 10g supports RAC (Real Application Cluster) for better performance by load sharing along with failover facility within the cluster of 2 or more servers.

C9.3 USER INTERFACE

While the data comes out from Oracle, Egg's internal users (staff of the Bank) can use any technology of their choice for analysis, as per their requirements. They use SAS (of the SAS Institute) to extract, join and mine the data for applications such as credit decisions or campaign management. Comms Builder, a tool written in SAS, and other modules of SAS such as SAS BASE, SAS Connect, SAS Share, SAS SPDS and SAS Start are also used. Oracle's Discoverer is also utilized for data mining.

Egg's CDW is about 2 TB in size, with about 10 GB added each month. Customers also use this data warehouse. They can get not only the regular transactions (OLTP) on the banking applications but also can use data mining services along with OLAP and data warehousing services.

C9.4 SOURCES OF DATA

For the CDW, the data is sourced from a variety of input customer data channels, both internal and external. These include credit cards, loans and insurance services, etc. Therefore, if a customer has all the three, viz. credit card, loan and insurance with Egg, that customer is seen as one entity having three Egg products. Hence, by CDW it is possible to see all the possible ways in which the customer is actually engaged with Egg. This, in turn, will enable Egg to determine the propensity and potentiality of the customer to buy additional products and also make the right choice of such products for the customer.

C9.5 BENEFITS

Egg has availed great benefits out of CDW. Sales and Marketing campaigns, up to 120 per month, could be developed using CDW. It also made possible daily credit decisions with due daily risk assessment and risk management. For example, if a customer has a credit card, the Egg management is actually able to

make a decision, using the data in the data warehouses, on a transactional basis to determine whether that customer's account requires collection, is just slightly overdue, or conversely, that the customer is a good credit risk and should be offered a pre-approved loan.

Campaigns developed on the basis of CDW have resulted in greater sales.

C9.6 SECURITY AND VERSION MANAGEMENT

Egg has implemented many security controls to ensure that access to CDW conforms to the internal policies and also external regulatory requirements. The data warehousing team comprising DBAs, meta data analysts and Oracle developers are responsible for the data security. Only they can make changes to the code. The data warehouse team decides and implements the changes that are required from time to time. External Regulatory Acts such as the UK's Data Protection Act of 1998, Financial Services Authority within the UK, Banking Act in the UK provide the regulatory framework for which the CDW has to be made fully compliant.

C9.7 REFRESH POLICY AND DATA MARTS

The data in CDW is refreshed in real time. For example, when a customer applies for a credit card, by the time he gets his credit card, his details has already been entered in the CDW.

The data is cleansed, matched and used to populate the core data warehouse. Then, data marts are published from CDW for individual departments of Egg. The back-end credit decisioning data mart is refreshed everyday. The materialized views for financing are refreshed everyday for financial reporting and counting of customer numbers. The marketing data mart is refreshed three times every week.

However, not all data inflows are in real time. Some data refreshes may come in batch mode from the internal sources themselves. Service Level Agreements (SLAs) may be required, but often they may be delayed or late in sending the refreshes. This may result in delays, or latency problems which may affect the accuracy and updateness of data in the data warehouse.

C9.8 CUSTOMER'S BENEFITS

Without the CDW Egg customers may be severely handicapped. The CDW plays a strong and crucial role in providing information, regarding finances necessary for the customers. All the services of Egg critically depend on CDW and thus, the customers are critically dependent on it.

C9.9 RELIABILITY

The architecture and design of IT infrastructure is required to be reliable. Hot plugins are provided to cover any failure in hardware components. Multiple fail-safes are made available with SAN. A disaster recovery system is also provided with the help of the original Sun Fire 6800 server. 90% uptime is maintained with a scheduled 10% downtime. SUN systems are found to be reliable and never has had any failure that was beyond the scope of maintenance engineers on a 24/7 basis.

CONCLUSION

In this case study we have seen how a large online bank, Egg of the UK deploys data warehousing for their core functions using reliable IT infrastructure. We have also seen how such deployment has resulted in enhanced success in banking service delivery to the customers and was helped the internal staff in better planning of marketing decisions.

Note: This case study is derived from Egg's websites on its data warehouse. We thankfully acknowledge Egg for this.

CASE STUDY

10 A German Supermarket EDEKA's Data Warehouse

EDEKA Gmbh is a German wholesale and retail food chain. The top management of EDEKA felt a need to be able to forecast specific product demands by identifying emerging market trends—very crucial for its business growth.

EDEKA engaged Melsungen, a company based in Germany, to implement the data warehouse applications based on DB2 of IBM.

C10.1 OBJECTIVES OF THE DATA WAREHOUSE

In an environment of fierce competition with the local retail industry in Germany, EDEKA felt a data warehouse could give it an advantage by providing the ability to conduct analyses on information such as sales turnover and inventory levels. The super market wanted to acquire the ability to adapt more quickly to the changing business conditions and requirements, with 50% faster responses to the queries. It aimed at obtaining higher availability and richer marketing data, improved end-user productivity with more reliable application, reduced administration costs and be able to leverage the existing skill base.

C10.2 SOFTWARE ENVIRONMENT

EDEKA used IBM's DB2 Universal Database DBMS for iSeries and DB2 Connect softwares on IBM hardware environment.

C10.3 HARDWARE ENVIRONMENT

EDEKA used the following hardware environment for its data warehouse

1. IBM @ Server iSeries 850 (processor)
2. IBM TotalStorage® Enterprise Storage Server™ (disk storage)
3. IBM TotalStorage Enterprise Tape System 3590 (tape storage)

C10.4 BUSINESS GROWTH AND THE DATA WAREHOUSE

EDEKA has expanded its business at a rapid pace. In the last two decades, it set up 60 wholly owned super markets and 1000 retailers in Hessen and Thuringian regions of Germany. Today, the company has about 7000 employees. The initial data warehouse was valuable but it could not keep track with EDEKA's business growth. As the business grew, the number of users of the data warehouse also increased proportionately. With increased usage, the demand on the system to provide greater number of analyses went up. This made the system slow. A highly scalable hardware and software architecture alone could withstand this situation successfully.

By upgrading the data warehouse to DB2 for iSeries, EDEKA was able to leverage on scalability and performance. EDEKA gained a business intelligence infrastructure that delivered on a wide variety of valuable marketing information, faster and more efficiently. Now, analyses could be done for various business divisions of EDEKA, such as the central purchasing department, financial control department and logistics department—all these depended on the data from the sales at the wholesale level that were supplied from the data warehouse.

Moreover, analysts could assess the volume of returned goods, find the reasons for changing shopping patterns and subsequently propose corrective measures. In the end, the system resulted in improving the employee productivity, reduced operating costs and optimized the utilization of IT and business resources.

C10.5 PERFORMANCE

With the help of iSeries servers the transaction processing was made 50% faster. As the data warehouse was run on the IBM DB2 Connect, it contributed to a better system performance. 'Business Objects' product from IBM's business partner also helped in to better access the sales information and track, understand and manage the wealth of information stored in the data warehouse.

C10.6 UPDATES AND REFRESHING

The information in the data warehouse is updated within an hour the concerned sales activity is completed, ensuring that the data is fresh always. By using the timely business analysis, EDEKA is able to anticipate market patterns more accurately.

C10.7 FORECASTING

Based on latest data EDEKA performed the analysis were able and to anticipate market patterns and purchase patterns in the immediate and near future. This allowed them for example, to forecast demand growth and suggest corrective measures in the event of a sudden surge in returned goods. Therefore, the data warehouse helped the company in responding quickly to changing business conditions and competitive pressures and enabled it to differentiate and distinguish itself in the open market place from other super markets in the region.

C10.8 EXPANSION

EDEKA plans to include additional data storage capacity. In future, the data warehouse will play a key role in addressing EDEKA's ongoing efforts to understand better consumer shopping patterns. This will help its executives keep the right products in the stores at the right time, based on accurate and up-to-date sales data from the data warehouse.

C10.9 PERFORMANCE

On the whole, EDEKA feels immensely satisfied with the present data warehouse as it "empowers Edeka to make much more logical business decisions which in turn result in improving profits and business objectives significantly".

CONCLUSION

In this case study, we have examined how a German super market chain, EDEKA leverages the data warehousing technology with reliable hardware to enhance its business objectives and profits, while faring better against its competition.

Note: This case study is derived from EDEKA's web sites. We thankfully acknowledge the same.